A Principal Component Algorithm for Feedforward Active Noise and Vibration Control

Randolph H. Cabell

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Mechanical Engineering

April, 1998
Blacksburg, Virginia

Keywords: Active Control, Aircraft Interior Noise, Principal Component Analysis

A Principal Component Algorithm for Feedforward Active Noise and Vibration Control

Randolph H. Cabell

(ABSTRACT)

A principal component least mean square (PC-LMS) adaptive algorithm is described that has considerable benefits for large control systems used to implement feedforward control of single frequency disturbances. The algorithm is a transform domain version of the multichannel filtered-x LMS algorithm. The transformation corresponds to the principal components (PCs) of the transfer function matrix between the sensors and actuators in a control system at a single frequency. The method is similar to other transform domain LMS algorithms because the transformation can be used to accelerate convergence when the control system is ill-conditioned. This ill-conditioning is due to actuator and sensor placement on a continuous structure. The principal component transformation rotates the control filter coefficient axes to a more convenient coordinate system where (1) independent convergence factors can be used on each coordinate to accelerate convergence, (2) insignificant control coordinates can be eliminated from the controller, and (3) coordinates that require excessive control effort can be eliminated from the controller. The resulting transform domain algorithm has lower computational requirements than the filtered-x LMS algorithm. The formulation of the algorithm given here applies only to single frequency control problems, and computation of the decoupling transforms requires an estimate of the transfer function matrix between control actuators and error sensors at the frequency of interest. The robustness of the PC-LMS algorithm to modeling errors is shown to be identical to the filtered-x LMS algorithm, and the statistical properties of the principal components are used to select a subset of the PCs to control, depending on the control application. The variation of the PCs with frequency is investigated, and an online identification procedure is described to compute the transfer function matrix while the controller is operating. The feasibility of the PC-LMS method was demonstrated in real-time noise control experiments involving 48 microphones and 12 control actuators mounted on a closed cylindrical shell. Convergence of the PC-LMS algorithm was more stable than the filtered-x LMS algorithm. In addition, the PC-LMS controller produced more noise reduction with less control effort than the filtered-x LMS controller in several tests.

This work was supported by a grant from the Structural Acoustics Branch at NASA Langley Research Center.

Acknowledgements

There are many persons to whom I owe thanks, for both technical and personal reasons, for helping me during my tenure as a graduate student. I would like to thank my chairman, Dr. Chris Fuller, for supporting me during my many years working for him as a Research Associate. I appreciate the freedom he gave me and his trust during those years, and I thank him for challenging me when necessary to produce something better than I might have otherwise.

I also appreciate the financial and technical support provided by Dr. Rich Silcox of the Structural Acoustics Branch at NASA Langley Research Center. His interest and enthusiasm for my work have been sources of encouragement to me for several years, and I look forward to continuing the relationship as an NRC Research fellow at NASA Langley. I feel fortunate to have worked on such an interesting topic in the stimulating environment at NASA Langley, and I thank Rich for giving me the opportunity to evaluate the topic of my dissertation in a flight test.

I would like to thank all of my committe members for their attention and input to my dissertation. I especially appreciate the patience of Dr. Baumann, Dr. Burdisso, and Dr. Wicks in Blacksburg, for putting up with the inconvenience of a student 250 miles away in Hampton, Virginia.

I owe a heartfelt thanks to Sharon Padula of the Multidisciplinary Optimization Branch, and Kevin Dutton of the Guidance and Control Branch at NASA Langley, for their valuable assistance proofreading my dissertation.

I would like to thank several people at NASA Langley for their technical assistance, often provided in hallway chats or informal blackboard musings, including: Dr. Gary Gibbs and Dan Palumbo of the Structural Acoustics Branch; Dave Cox of the Guidance and Control Branch; Dr. Feri Farassat of the Aeroacoustics Branch; and Don Brown of Lockheed-Martin. I was fortunate to be near so many creative, intelligent people who could provide feedback on my work. I hope I can repay the favor in years to come.

On the personal side, I would like to thank my parents and sisters for providing inspiration to me all of these years, each in their own distinct way. I also owe a big thanks to numerous friends, some of whom are mentioned above, for supporting me during rough times and listening to my complaints.

And last but not least, I thank Karen Fischer, for standing by me all of these years and encouraging me through thick and thin. I learned many things while in school, but you provided me with some of my most valuable and treasured lessons.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The work described in this thesis is motivated by the desire of the aircraft industry to improve passenger comfort by reducing interior cabin noise levels. There are many sources of noise inside an aircraft, however the present work addresses only periodic sources such as propeller blade passage noise or engine noise. Traditional methods for reducing cabin noise involve adding sound dampening materials inside the fuselage. While these methods are useful for high frequency noise, they require excessive weight to be effective at low frequencies. Therefore a more practical approach is needed.

An alternative approach for noise reduction that has gained attention in the past 15 years is based on active control (Nelson and Elliott 1992; Fuller et al. 1996). Active elements such as loudspeakers or inertial force actuators are used to alter the sound transmission path from the source to the cabin interior to make the interior sound field more pleasant for passengers. Periodic noise sources are often the easiest to control because simple methods can be used to actively reduce the noise generated by these sources. The control approach discussed here is known as feedforward active control. This approach assumes a signal is available which is highly correlated with the periodic noise source. This signal can be operated on by the control system to derive the optimal signals to drive the control actuators, whether they are loudspeakers or force actuators (Fuller et al. 1996; Widrow and Stearns 1985; Elliott et al. 1992). In an airplane, a once-per-revolution signal from a propeller shaft is an example of a signal that is related to the sound field created by the propeller. This shaft revolution signal can be used as an input to the control system to control noise at the harmonics of the blade passage frequency of the propeller.

In order for an active control system to be effective in a complex environment such as an aircraft cabin, a large number of actuators and performance monitoring sensors are needed. The sensors provide feedback to the control algorithm that is used to update the signals which are being sent to the control actuators. Assuming the noise source is constant, the control inputs could be hardwired to fixed values, but in general performance is improved when feedback from the sensors is used to adapt the control inputs.

The goal of this thesis is to describe a control algorithm that simplifies the implementation and improves the performance of feedforward active control on a complex structure where many sensors and actuators are needed to achieve acceptable control performance. Before presenting the details of the algorithm, it will be useful to review relevant concepts of adaptive filtering and active noise control.

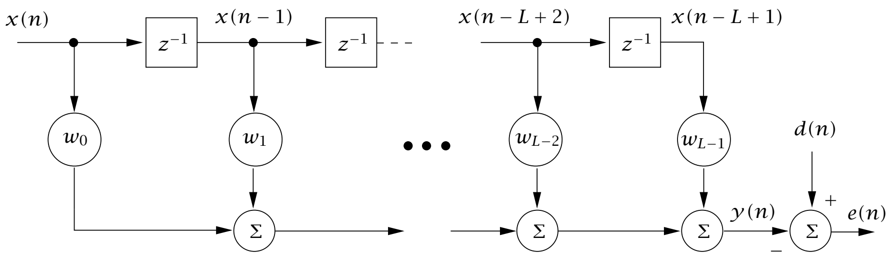

Figure 1.1: General transversal filtering problem

1.1 Overview: Adaptive Filtering

Feedforward control algorithms are closely related to adaptive filtering algorithms. An adaptive filter, in the context of the present work, refers to an iterative procedure for computing the optimal solution to the Wiener filtering problem when a finite impulse response (FIR) filter is used shortcitehaykin-aa. A schematic of a simple FIR filtering problem is shown in Fig. 1.1, where $x(n), x(n-1), \ldots,$ are consecutive values of an input time series, $w_0, w_1, \ldots,$ are filter coefficients, $y(n)$ is the filter output, and $d(n)$ is the desired filter output at time n. The Wiener filtering problem specifies the optimal filter coefficients, $w_0, w_1, \ldots,$ that minimize the mean square value of $e(n)$ for all time. A well known adaptive solution to this problem is a recursive technique called the least mean square (LMS) algorithm (Widrow and M. E. Hoff 1960). The LMS algorithm is a stochastic gradient descent method which is popular because it is easy to understand, program, and debug (Widrow and Stearns 1985). The algorithm is used in many applications, such as echo cancellation, adaptive line enhancement, adaptive beamforming, and adaptive noise cancellation (Haykin 1996; Morgan 1980; Widrow et al. 1975; Widrow and Stearns 1985).

The filtering problem depicted in Fig. 1.1 can be easily related to a single channel active control problem. An early application of the LMS algorithm to an acoustic control problem was described in Burgess (1981), in which an adaptive control system was used to compute the input to a speaker mounted in the wall of a duct in order to control unwanted sound propagating through the duct. The input time sequence (or reference signal) to the adaptive filter was provided by a microphone mounted upstream in the duct; the desired time sequence (or error signal) was provided by a second microphone placed at a point downstream where the sound was to be controlled. The filter output was used to drive the loudspeaker mounted in the wall of the duct. If sensor and duct dynamics are ignored for simplicity, the optimal speaker output is a delayed version of the output of the upstream microphone. The LMS algorithm must therefore determine the filter coefficients that produce the optimal delay. The physical distance between the speaker output and the downstream microphone introduces a delay that complicates the adaptive algorithm. To compensate for this delay, a variant of the LMS algorithm known as the filtered-x LMS algorithm was developed (Widrow and Stearns 1985; Morgan 1980). The filtered-x LMS algorithm uses a model of the dynamics of the path from the actuator input to the microphone output to pre-filter the incoming signal. Because the incoming signal is usually labeled $x(n)$, as in Fig. 1.1, the method was called the "filtered-x" LMS algorithm.

The filtered-x LMS algorithm concerns single input/single output systems, but extension of the concepts to multichannel systems is straightforward, and the resulting algorithm is called the multiple error LMS algorithm (Elliott et al. 1987). The multiple error LMS algorithm, which is sometimes referred to as the multichannel filtered-x LMS algorithm, can be used with multiple reference inputs, $x(n)$, multiple control filters, and multiple error sensors. The multiple error LMS algorithm is used in numerous applications for controlling periodic and random disturbances on mechanical systems;

example applications include aircraft noise control, sound reduction in automobiles, vibration control, and others that can be found in the references (e.g., see Nelson and Elliott (1992), Fuller et al. (1996), Kuo and Morgan (1996)). The multiple error LMS algorithm forms the basis for the theoretical developments described in the current work.

1.2 Overview: Transform Domain Adaptive Filtering

The LMS algorithm, in either its single channel or multiple channel forms, is easy to implement, but the algorithm does have limitations that have motivated a great deal of research. One limitation concerns the sensitivity of convergence to the statistics of the input signal. An examination of this limitation, and the methods that have been devised to deal with it, provide background material for the adaptive algorithm developed in this thesis.

In the context of the single input/single output (SISO) case, the speed at which the filter coefficients converge to the optimum coefficients is dependent on the statistics of the input time series, $x(n)$. When this time series is spectrally colored, meaning certain spectral components are stronger than other components, the convergence time of the LMS algorithm is increased. The amount of spectral coloration in the input time series is quantified by the condition number of the correlation matrix of x (Widrow and Stearns 1985; Haykin 1996). A high condition number indicates slow convergence, and a low condition number indicates fast convergence.

Because convergence speed is critical in certain filtering applications, methods have been devised to accelerate the convergence of the filter coefficients when the correlation matrix of the input time sequence is ill-conditioned (i.e. has a very high condition number). One such method is the self-orthogonalizing adaptive filter (Panda et al. 1986; Haykin 1996), which transforms the input time series to an orthogonal coordinate system, normalizes each coordinate by the inverse of its power, and then updates the FIR coefficients in the coordinates of orthogonal system. The transformation and scaling by the inverse power effectively reduces the condition number of the input correlation matrix to unity, in which case the convergence speed of the LMS algorithm is optimal. The transformation to orthogonal coordinates relies on the eigenvectors of the correlation matrix of the input time sequence, and hence the method has two drawbacks: the correlation matrix must be known, and its eigenvectors must be computed. These are unreasonable requirements in most SISO filtering applications where fast convergence is essential.

A related technique for accelerating the convergence of the adaptive filter is to use Newton's method to adaptively determine the optimum filter coefficients. Newton's method can achieve the fastest possible convergence of an adaptive algorithm, and hence it is sometimes used as a benchmark against which other algorithms are compared (Widrow and Stearns 1985). The method is only a benchmark, however, because it assumes the inverse of the correlation matrix of the input time series is known. As with the computation of the eigenvectors, it is often impractical to compute the correlation matrix and its inverse when fast convergence is critical. The recursive least squares (RLS) algorithm, which has the benefits of the fast convergence of Newton's method, requires no *a priori* knowledge of the inverse of the input correlation matrix (Haykin 1996). Instead, the RLS algorithm recursively computes an estimate of the inverse of the input correlation matrix from past values of the input time series. The drawback of the RLS algorithm is that its computational requirements can be very high, especially when the adaptive filter contains many coefficients. Reduced computation versions of RLS exist, but these algorithms often suffer from numerical conditioning problems due to a build up of finite precision errors in the recursive variables of the algorithm (Marshall and Jenkins 1992). Yet another method based on Newton's method is the quasi-newton approach (Marshall and Jenkins

1992). This method computes an estimate of the inverse correlation matrix, but does so using blocks of input data, so its computational requirements are much lower than RLS, and it avoids the numerical conditioning problems as well.

A class of algorithms that are related to the self-orthogonalizing adaptive filter, having much lower computational requirements than RLS, are known as transform domain algorithms. The self-orthogonalizing adaptive filter is actually one kind of transform domain algorithm, but in general the transform used is either a discrete Fourier transform (DFT) or a discrete cosine transform (DCT) (S. Shankar Narayan 1983; Beaufays 1995; Lee and Un 1986; Haykin 1996). The approach is to transform the input series to the frequency domain, and compute filter coefficients for each spectral component separately. Because the frequency bins are theoretically orthogonal with one another, the effects of spectral coloration can be compensated and the overall rate of convergence accelerated. An important benefit of this approach is that the coefficients which transform a time series to the frequency domain are fixed, for a fixed number of points in the time series. This means the DFT and DCT are independent of the time series, and does not need to be recomputed each time the statistics of the time series change. In reality, adjacent frequency bins are slightly correlated because only a finite number of time series points are transformed, so the transformation does not completely decouple the input time sequence. Nonetheless, the transformation is almost always useful for accelerating the convergence of the LMS algorithm when the input time series is spectrally colored.

Convergence of the filter coefficients in the multiple error LMS algorithm is also sensitive to the conditioning of the correlation matrix of the input signals, but there are additional considerations for the multichannel problem. For example, the dynamics of the path between the control actuator and the error sensor will color the input signal (Morgan 1980; Haykin 1996). Thus even if the original input time series is not spectrally colored, the dynamics of the control to error sensor path could add coloration to the signal. In addition, the spatial arrangement of control actuators and error sensors on a reverberant structure will affect the convergence of the algorithm. The spatial arrangement determines how the actuators and sensors couple into the modes of the structure, hence two actuators mounted at physically distinct positions could have the same coupling factors into the modes of the structure. This would create ill-conditioning in the control system, since multiple coordinates of the controller would be indistinguishable in terms of control system performance. Ill-conditioning in the context of a multichannel controller is sometimes referred to as multicollinearity (Ruckman 1994; Swanson 1993), because it is produced by constant linear relationships between the outputs of multiple actuators at the error sensors. Diagnostics for detecting multicollinearity in a controller have been described by Swanson (1993) and Ruckman (1994). The methods are based on examining correlations between the columns of the transfer function matrix between actuators and sensors at a single frequency.

Note that the discrete cosine or Fourier transforms cannot be used to decouple spatial correlations between sensors and actuators on a distributed control system. Ill-conditioning in a multichannel control system will cause the actuators to have very high control inputs, which means large control forces will be input to the structure. In most applications this is to be avoided, therefore limiting the effects of ill-conditioning in a multichannel control system is important for reasons other than accelerating convergence.

Thus, for the multichannel control problem, a method is needed to deal with spatial correlations between components of the control system at a single frequency. Newton's algorithm, and related methods such as RLS and quasi-Newton approaches, can be used to make the convergence insensitive to spatial correlations in the control system. For example, a stochastic Newton algorithm has been described (Nelson and Elliott 1992), that requires estimating the input correlation matrix, which includes the effects of temporal and spatial correlations, before control is applied. The inverse of this matrix is then used in the adaptive algorithm to make convergence insensitive to the conditioning of the input correlation matrix. The recursive least squares method can be used in a similar fashion, and it has been applied to the multichannel control problem (Bronzel and Fuller 1995), however as with Newton's algorithm, the RLS algorithm only makes convergence insensitive to the conditioning of the input correlation matrix. These methods do not provide a means to analyze the effect of spatial correlations on control effort, nor do they provide a simple way to reduce the effects of spatial correlations on the control solution. The computational complexity of these approaches is another limitation.

Simpler methods, such as the transform domain approaches using the DCT, have been applied to multichannel controllers (Paillard et al. 1995; Bouchard and Paillard 1996). These transformations are useful for accelerating the convergence of the multichannel LMS algorithm due to spectral coloration of the input signal, but they do not reduce the effects of spatial correlations between actuators. In fact, in Bouchard and Paillard (1996), after the DCT removes temporal correlation in the input signal, spatial correlation is removed with a quasi-Newton approach similar to the stochastic newton approach described in Nelson and Elliott (1992).

A few other transform-like approaches have been applied to the multichannel control problem, but the purpose has been to simplify the controller, and not explicitly to limit the effects of ill-conditioning. Clark (1995) describes independent modal space control (Meirovitch 1990) in the context of the multichannel feedforward control problem. The technique is to transform the control actuator inputs and sensor outputs such that individual modal responses of the structure can be observed and controlled. Because control is implemented in terms of the modes of the structure, the coordinates of the controller are decoupled from one another and can be controlled independently. There are two drawbacks of this approach: firstly, the modal characteristics of the structure must be known so that modal transformation filters can be designed for the sensor outputs and actuator inputs. Secondly, it can be extremely difficult to design accurate transformations that take the inputs or outputs of discrete elements, such as point force actuators or accelerometers, into individual modal responses (Meirovitch 1990). Morgan (1991) addressed the second of these problems by formulating an optimization problem to determine the best possible transformations from discrete elements to modal responses. This technique is more practical, but it still assumes knowledge of the modal characteristics of the structure, which can be unreasonable for a very complex system such as an aircraft fuselage. The general principle has also been applied to radiation from a panel (Snyder and Burgemeister 1996), whereby microphone responses in the far-field are transformed to responses of orthogonal radiation patterns. Control is then implemented by reducing the outputs of the important radiation patterns at a given frequency. A related approach that implicitly considers the spatial actuator arrangement was described by Fuller and Carneal (1993), and is based on a biological control approach. The input to a single actuator, called the "master," is first optimized using a conventional optimization approach such as the LMS algorithm. Once the input to the master actuator is determined, the inputs to multiple slave actuators are independently determined using very simple learning rules. The spatial arrangement of the actuators will naturally determine the optimal input configuration for the actuator array.

The approach taken here, which has not been considered elsewhere, is to use a signal dependent orthogonalizing transformation to decouple a multichannel feedforward control system at a single frequency. The idea is similar to the self-orthogonalizing adaptive filter (Panda et al. 1986; Haykin 1996), except it is applied to a multichannel controller on a reverberant system. The approach uses fixed transformation vectors to transform sensor responses into responses of the principal components of the control system. Likewise, a different set of transformation vectors are used to transform control inputs, computed in terms of the principal components of the control system, into inputs to individual actuators. Spatial correlations in the control system are decoupled when control is implemented in terms of the principal components, so problems of convergence speed and control effort can be addressed more easily than when the controller coordinates are coupled together. Because the control algorithm is essentially a transform domain version of the multiple error LMS algorithm, it is called principal component LMS, or PC-LMS.

The transformation vectors to and from principal components are computed from a singular value decomposition (SVD) of the transfer function matrix between sensors and actuators at the frequency of interest. This approach thus suffers from the same drawback as the self-orthogonalizing adaptive filter: the orthogonalizing transform depends on the singular vectors of a matrix that can change. However, this is not as great a drawback when control is implemented on a reverberant structure, because we can assume the transfer function matrix is known anyway. This matrix is needed to implement control using the multiple error LMS algorithm, so the only additional requirement for decoupling the controller is to compute the SVD of this matrix. Thus, an added restriction for this control approach is that the transfer function matrix between sensors and actuators must be changing slowly enough to allow for its SVD to be recomputed periodically.

Each principal component specifies a mathematical grouping of either the actuators or the sensors that changes as the transfer function matrix changes. If the SVD is recomputed as the transfer functions change, the actuator/sensor groups are essentially reconfigured online. This method has the advantage of always producing an optimal actuator/sensor grouping, in contrast to the approach whereby the groups are fixed for all operating conditions. The development of a dynamically reconfigurable controller was one of the goals of the work described here.

The SVD has previously been applied to the feedforward active control problem. Gibbs et al. (1997) used the SVD to analyze vibration responses of an aircraft structure, in order to find dominant vibration patterns in the response. Control actuators were placed on the structure to recreate these dominant patterns. Rosenthal (1992) has also investigated the SVD for improving the conditioning of a feedforward control system by solving the control problem in the frequency domain with a pseudo-inverse approach (e.g., see Strang (1988)). The idea is related to the work described here, however Rosenthal (1992) did not discuss the SVD approach in detail, and did not address numerous practical considerations involving the implementation of a principal component control algorithm based on the SVD. Rossetti et al. (1996) used the SVD of the transfer function matrix of a feedforward control system to implement a sophisticated effort penalty. The idea was similar to that of Rosenthal (1992), in that a threshold was applied to the singular values of the control system, and an effort penalty was applied to singular values that fell below the threshold.

The convergence behavior of the multiple error LMS algorithm is usually analyzed in terms of its principal components (Elliott et al. 1992; Nelson and Elliott 1992; Fuller et al. 1996; Haykin 1996; Widrow and Stearns 1985). In this respect, the principal components of a feedforward control system are well understood. The control effort and noise reduction associated with individual PCs have been investigated (Elliott et al. 1992), but have not been proposed as criteria for deciding which PCs to control. Thus the work here extends the principal components from an analysis tool to an implementation tool, in order to improve the understanding and ease the application of the multiple error LMS algorithm.

The orthogonality property of the principal components is very important, because once the control problem has been transformed into its principal components, it is simple to selectively control or not control individual coordinates of the controller. Selection of which PCs to control can be based on control effort, reduction in the error term, or the statistical properties of the PCs. A direct correspondence between individual control coordinates and performance metrics such as noise reduction and control effort is not possible when the control coordinates are fully coupled to one another. The application of a statistical selection metric is based on previous work in applying statistics to the feedforward control problem (Ruckman and Fuller 1995; Snyder and Hansen 1991). Statistics have been used for actuator selection, such as in Snyder and Hansen (1991), and for optimizing actuator and sensor locations, and Ruckman and Fuller (1995), where multicollinearity diagnostics were used to decide which actuators to remove to minimize ill-conditioning in the control system.

1.3 Overview: Active Control of Aircraft Interior Noise

The second part of the thesis describes the results of an experiment using the principal component control algorithm to reduce sound inside a cylindrical shell. The shell contained structural elements similar to an aircraft fuselage, including ribs, stringers, and an interior floor, and it has been used in the past for evaluating control strategies for controlling noise in aircraft cabins (Silcox et al. 1989; Palumbo et al. 1996). The control system consisted of 48 microphones arranged uniformly inside the shell and 12 inertial force actuators mounted on the ring frames of the shell. The experiment relied on the concept of active structural acoustic control (Fuller and Jones 1987; Fuller 1987), or ASAC, in which force inputs to a structure are used to control sound radiation from the structure.

The active control of aircraft interior noise has been the subject of much research for the past decade (Fuller and Jones 1987; Nelson et al. 1987). Numerous references can be found on the general subject of the active control of enclosed sound fields, including cylindrical shells (Thomas et al. 1993; Lester and Silcox 1992; Lester and Lefebvre 1993; Houston et al. 1996; Grosveld and Lester 1995; Bullmore et al. 1987; Bullmore et al. 1990; Fuller et al. 1996; Nelson and Elliott 1992).

Some early experiments on active control of aircraft interior noise used loudspeakers inside the cabin to create a control sound field (Elliott et al. 1989; Dorling et al. 1989). The approach was based on creating a sound field to interfere destructively with the sound field created by the propellers. Elliott et al. (1989) conducted tests on a BAe 748 twin turboprop for actively controlling the fundamental and first two harmonics of the blade passage frequency of the propeller. The control sound field was produced by 16 loudspeakers in the cabin, and 32 microphones were used as error sensors. The multiple error LMS algorithm was used to adaptively compute the loudspeaker outputs. Reductions of 13 dB at the fundamental frequency, and up to 12 dB at the first and second harmonics were obtained. A similar experiment on the same aircraft and reported at the same time was conducted by Dorling et al. (1989). This second experiment was conducted using 24 loudspeakers and 72 microphones in the control system and a proprietary control algorithm. The results were similar to those obtained by Elliott et al. (1990); the fundamental of the blade passage frequency was reduced by 9 dB, and the two harmonics were reduced by up to 13 dB.

Active control of noise in cylinders using the ASAC approach has also been experimentally demonstrated, although fewer flight tests of this approach have been reported. Fuller and Jones (1987) conducted experiments using vibrational inputs to a cylinder to reduce sound transmission through the cylinder. An important property of sound transmission for the enclosed cylindrical shell problem is the selective coupling between motions in the cylindrical structure and the response of the enclosed acoustic space (Lester and Fuller 1986; Silcox et al. 1987). This means only a few structural vibration patterns in the shell couple efficiently into the acoustic response of the enclosed cavity at a given frequency. As a result, only a few force actuators are needed on the structure in order to control the structural motions responsible for transmitting sound. The resulting control system should therefore be simpler than a control system based on loudspeakers inside the shell (Fuller and Jones 1987). A recent flight test demonstrated a related approach, in which inertial force shakers were used to quiet the cabin of a de Havilland DASH-8 turboprop (Ross and Purver 1997). The shakers were mounted at intersections of ring frames and stringers in the plane of the propeller, and 52 microphones were arranged in the cabin as error sensors. Reductions of 10.5 dB, 7.6 dB, 4.4 dB, and 3.0 dB, were obtained at the fundamental and next three harmonics of the blade passage frequency.

A drawback of the ASAC approach is the tendency of small force actuators to couple into high order structural motions that couple poorly with the interior acoustic response (Thomas et al. 1993; Silcox et al. 1987). This is not necessarily a problem for interior sound control, but it may have serious implications for the fatigue life of the structure because the noise control system causes an increase in vibration levels in the fuselage. This behavior has motivated research into intelligent control systems that can drive multiple distributed actuators with the same control input, when possible (Lester and Silcox 1992; Fuller and Carneal 1993; Cabell et al. 1993). The resulting control force is thereby distributed over a larger area of the structure, reducing the tendency to excite high order structural motions. Because the principal component controller drives groups of actuators in groups specified by the principal component transformation matrices, the PC-LMS algorithm should provide some of the same benefits of the grouping controllers.

1.4 Scope and Objectives

The purpose of this dissertation is to describe a modified version of the multichannel filtered-x LMS algorithm that addresses practical issues that arise when control is implemented on a complex structure with a large control system. The approach is based on a transform domain version of the filtered-x LMS algorithm, where the transformation consists of the principal components of the control system. The resulting control algorithm can be used to address ill-conditioning due to the spatial arrangement of the components of the control system, including such effects as slow convergence, excessive control effort requirements, and excessive computational burden for a large control system. The transform domain algorithm, called the PC-LMS algorithm, decouples the control system, making it easy to selectively control individual independent coordinates of the controller. This ability is beneficial when certain controller coordinates are associated with ill-conditioning, or have little utility for reducing the controller cost function. The statistical properties of the control coordinates are also studied, and are related to control effort and noise reduction properties.

The practical issues involved in a real-time implementation of the algorithm are also addressed, including: computational requirements of the algorithm relative to other feedforward control algorithms; sensitivity to random noise in the error sensors; and convergence sensitivity to modeling errors. In addition, a procedure is described for identifying the transfer function matrix between sensors and actuators while the controller is operating.

The practicality of the PC-LMS algorithm is demonstrated in a real-time noise experiment conducted on a closed cylindrical shell modeling an aircraft fuselage. The PC-LMS algorithm is coded on a digital signal processor and used to derive inputs to 12 control actuators based on the outputs of 48 error microphones inside the cylindrical shell. The experiments illustrate the benefits of the PC-LMS algorithm relative to the filtered-x LMS algorithm for implementing feedforward control on a large control system.

1.5 Organization

This thesis is divided roughly into three sections; the first section, consisting of Chapters 2 and 3, contains background material on adaptive filter theory for single input/single output, and multiple input/multiple output systems. The convergence sensitivity of the adaptive algorithms, and how to reduce this sensitivity, are emphasized because the solutions are related to the PC-LMS algorithm.

The second section consists of Chapters 4 and 5, both of which contain new material describing the principal component LMS algorithm. The bulk of the new material can be found in Chapter 4, which develops the PC-LMS algorithm, and describes the control effort and noise reduction properties of each principal component, and the robustness of the PC-LMS algorithm to modeling error. Chapter 5 describes statistical material that is useful for understanding the properties of the principal components of a control system. This chapter also describes statistical methods for selecting which principal components to control.

The third section, consisting of Chapters 6-8, describes implementation issues of the PC-LMS algorithm. Chapter 6 discusses how the principal components vary with frequency, using numerical models of three simple control systems: a simply supported vibrating plate; a plate radiating sound into a half space; and a cylindrical shell. Because the principal components are shown to vary with frequency in Chapter 6, Chapter 7 discusses a method for identifying the transfer function matrix online during operation of the controller. This does not constitute a new type of identification algorithm, but is included for completeness.

The experimental results are described in Chapter 8. The performance of the PC-LMS algorithm is compared with the filtered-x LMS algorithm for controlling sound inside a cylindrical shell. The results of control tests using 12 force actuators and 48 microphones are discussed at two test frequencies. A simple experimental comparison of the robustness of the two control algorithms to errors in the transfer function matrix is described at the end of Chapter 8. Conclusions and recommendations for future work are given in Chapter 9.

Chapter 2

Single Channel Adaptive Filters

The goal of this chapter is to establish a framework for the principal component adaptive filtering algorithm starting with the simple and familiar case of a single input/single output (SISO) adaptive transversal filter. The material here has been discussed in detail elsewhere (e.g, see Widrow and Stearns (1985), Nelson and Elliott (1992), Haykin (1996), and Fuller et al. (1996)), but certain topics are emphasized for their relevance to principal component filtering. The steepest descent and least mean square (LMS) algorithms are discussed, and their convergence behavior is shown to depend on the statistics of the input signal being filtered. Methods to reduce this sensitivity are discussed, including Newton's method, recursive least squares, and a self-orthogonalizing adaptive algorithm. In addition, transform domain methods for accelerating convergence by transforming the input signal to an orthogonal or near orthogonal coordinate system are discussed. The principal component algorithm is closely related to these transform domain algorithms.

There are some important assumptions made here concerning the types of signals and systems being studied. Firstly, the time domain signals are assumed to be generated by random processes that are wide-sense stationary and ergodic, hence their mean, autocorrelation, and autocovariance are constant with time (Bendat and Piersol 1986). This assumption will be relaxed slightly to allow for the possibility that the statistics of an input signal slowly vary with time, but in general the stationarity assumption will hold. Secondly, since the goal of this work is to develop algorithms that can be implemented in digital signal processors (DSPs), the signals are assumed to be real-valued and discrete. The time domain signals are also assumed to be electrical signals, even though some of the applications discussed in subsequent chapters concern adaptive filtering applied to acoustic fields in an enclosure. This means the signals being adaptively filtered have already been filtered by the dynamic response of a transducer, such as a pressure sensor. The dynamics of any transducers are thus external to this analysis, which greatly simplifies the discussion.

2.1 Optimal Transversal Filter

An ideal filtering problem is discussed here to introduce notation, properties, and analysis techniques that can be applied to adaptive transversal filters. The discussion closely follows that given in Haykin (1996). The filtering problem of interest is that of a transversal filter whose output is a weighted summation of delayed values of an input time series, shown schematically in Fig. 2.1. This schematic

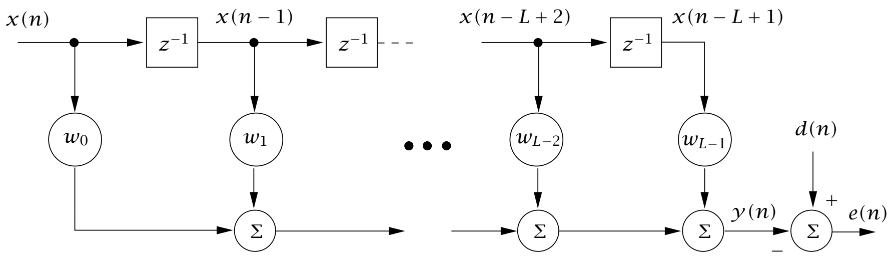

Figure 2.1: General transversal filtering problem

representation, where z^{-1} is a unit delay operator, is a well-accepted form for representing a finite impulse response (FIR) filter (Widrow and Stearns 1985; Haykin 1996). The input signal to the filter consists of L tap-delay, or consecutively sampled values of a time series x, denoted as the $(L \times 1)$ vector $\mathbf{x}(n) = [x(n), x(n-1), \ldots, x(n-L+1)]'$ where n denotes the time index and ' denotes transpose (for clarity, matrices, vectors, and scalars will be represented by upper-case bold, lower-case bold, and regular face letters, respectively). The tap inputs $\mathbf{x}(n)$ are weighted by a time-invariant, $(L \times 1)$ vector of tap weights, $\mathbf{w} = [w_0, w_1, \ldots, w_{L-1}]'$. The time invariance of $\mathbf{w}$ will be relaxed when adaptive algorithms are discussed. The filter output at time n is denoted as $y(n)$ and is the sum of the weighted input values. A desired filter output, $d(n)$, is also known at each time step, and the error between the desired and actual filter outputs at time n is defined as

$$e(n) = d(n) - y(n) \tag{2.1}$$

$$= d(n) - \mathbf{w}'\mathbf{x}(n) \tag{2.2}$$

where the second equation is written using vector notation.

An infinite impulse response (IIR) filter could also be used to filter $\mathbf{x}(n)$ and produce an output $y(n)$ for comparison with the desired, $d(n)$. However, the FIR filter in the figure is preferred due to its inherent stability (Widrow and Stearns 1985; Haykin 1996), which is especially important when an adaptive algorithm is used to modify the coefficients in the weight vector $\mathbf{w}$.

The filtering problem is to determine the tap weight vector $\mathbf{w}$ that minimizes the difference in a mean square sense between the actual and desired filter outputs. The mean square criterion is used here because the resulting filtering problem requires solution of a quadratic function of the weight vector $\mathbf{w}$, and a quadratic performance criterion is smoothly varying and has a unique global minimum. Using E[] to denote statistical expectation, the mean square error criterion can be written as a function of the tap weight vector as

$$J = \mathrm{E}\left[e(n)^2\right] \tag{2.3}$$

$$= \mathrm{E}\left[(d(n) - \mathbf{w}'\mathbf{x}(n))^2\right] \tag{2.4}$$

$$= \mathrm{E}\left[d^2(n)\right] - 2\mathbf{w}'\mathrm{E}\left[\mathbf{x}(n)d(n)\right] + \mathbf{w}'\mathrm{E}\left[\mathbf{x}'(n)\mathbf{x}(n)\right]\mathbf{w} \tag{2.5}$$

$$= \mathrm{E}\left[d^2(n)\right] - 2\mathbf{w}'\mathbf{p} + \mathbf{w}'\mathbf{R}\mathbf{w} \tag{2.6}$$

where $\mathbf{p} = \mathrm{E}[\mathbf{x}(n)d(n)]$ is the $(L \times 1)$ cross correlation vector between the input and desired time series, and $\mathbf{R} = \mathrm{E}[\mathbf{x}'(n)\mathbf{x}(n)]$ is the $(L \times L)$ correlation matrix of the input time series. The stationarity assumption means $\mathbf{R}$ and $\mathbf{p}$ are constant with time. The symbol J will be used here to represent the function to be optimized, which is usually called a cost function. The last term in Eq. 2.5 depends on the square of $\mathbf{w}$, hence the cost function varies quadratically with $\mathbf{w}$. Assuming the correlation matrix $\mathbf{R}$ is positive definite, there is a weight vector that corresponds to a minimum cost; this weight

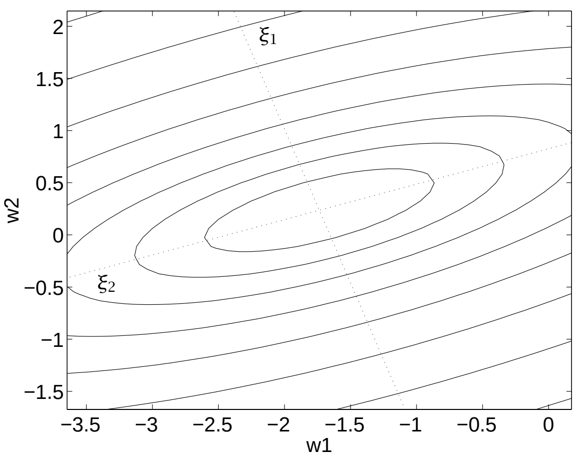

Figure 2.2: Contour plot of error surface

vector will be denoted $\mathbf{w}_{opt}$. If the input time series contains at least $(L/2)$ frequency components, the correlations between the tap-delayed values in $\mathbf{x}$ will be significant, and the resulting correlation matrix $\mathbf{R}$ will be full rank.

The expectations used to determine $\mathbf{R}$ and $\mathbf{p}$ are taken over many ensembles of data from the random processes x and d, hence $\mathbf{R}$ and $\mathbf{p}$ are called ensemble average properties. The optimal weight vector defined in Eq. 2.8 is therefore an ensemble average optimum. In almost any practical application of the techniques described here the ensemble average values of $\mathbf{R}$ and $\mathbf{p}$ will be unknown. For this reason the solution in Eq. 2.8 represents an ideal solution that will be used to measure the effectiveness of more practical solution algorithms.

It is helpful to envision the cost function as a multidimensional bowl-shaped surface in a space defined by the elements of the tap weight vector $\mathbf{w}$. A plot of a hypothetical cost function, J, defined by two tap weights, w_1 and w_2, is shown in Fig. 2.2. The elliptical curves define contours of equal cost that increase away from the minimum cost at the center of the plot, and the principal axes of the ellipses are shown as dotted lines labeled ξ_1 and ξ_2. Assuming the correlation matrix $\mathbf{R}$ is positive definite, the opening of the bowl-shaped function points upward. The coordinates defining the center of the plot in the figure, which corresponds to the minimum of the cost surface, define the elements of the optimal tap weight vector, $\mathbf{w}_{opt}$. The principal axes and eccentricity of the elliptical contours are functions of the eigenvectors and eigenvalues, respectively, of $\mathbf{R}$ (Widrow and Stearns 1985; Haykin 1996; Nelson and Elliott 1992). The steepness of the error surface in the direction of the ith principal axis is proportional to the ith eigenvalue of $\mathbf{R}$, and therefore the elongation of the bowl-shape can be quantified by the condition number, or ratio of the largest to the smallest eigenvalue, of $\mathbf{R}$. The condition number is greater than or equal to one, and if it equals one, the contours of equal cost are circles. For the surface shown in the plot, the condition number of $\mathbf{R}$ was 10.0, hence the equal cost contours have an elliptical shape.

The tap weight vector corresponding to the minimum cost can be found by differentiating J with respect to $\mathbf{w}$ and setting the derivative to zero since this stationary point is known to be a minimum. Differentiating J yields

$$\nabla \mathbf{J} = -2(\mathbf{p} - \mathbf{R}\mathbf{w}) \tag{2.7}$$

If $\mathbf{R}$ is positive definite and thus invertible, the tap weight vector that minimizes J is given by

$$\mathbf{w}_{opt} = \mathbf{R}^{-1}\mathbf{p} \tag{2.8}$$

Therefore knowing the correlation matrix $\mathbf{R}$ of the input signal and the cross correlation $\mathbf{p}$ between the input and desired signals, a unique optimal weight vector can be computed from Eq. 2.8. Although the times series are assumed to be stationary, in many practical applications the statistics could change slowly with time, hence the optimal tap weight vector would also change with time. For these reasons, adaptive algorithms are often used to determine the optimal tap weight vector.

2.2 Steepest Descent Algorithm

The adaptive algorithms of interest here are based on gradient descent techniques, of which the steepest descent algorithm is one example. The steepest descent algorithm relies on a recursive equation that moves an arbitrary tap weight vector towards the optimum tap weight vector using the negative of the gradient of the cost function. The steepest descent algorithm is ideal in the sense that the exact gradient vector is assumed to be known at each iteration. The behavior of more practical algorithms can be easily related to the behavior of the steepest descent algorithm, hence the properties of this algorithm will be discussed in some detail. In this discussion, time invariance of the tap weights is no longer assumed, and the computations described here represent iterations computed at each time step.

The weight update recursion for the steepest descent algorithm specifies that the tap weight vector at iteration $(n+1)$ is computed from the weight vector at the previous iteration plus a correction based on the negative of the gradient of the cost function. This recursion is written

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\mu}{2}[-\nabla J(n)] \tag{2.9}$$

where $\nabla J(n)$ denotes the vector gradient of the cost function with respect to each element of the tap weight vector. The symbol μ denotes a step size parameter that controls how much the weight vector is changed at each iteration. The algorithm essentially relies on the idea that the tap weight vector is slowly adjusted toward its optimal value that minimizes the cost function. This approach can be thought of as starting from an arbitrary point on the cost surface and sliding down the surface in its steepest direction to the minimum. The convergence of $\mathbf{w}$ to the optimum $\mathbf{w}_{opt}$ is assured as long as the step size parameter μ satisfies a few simple constraints (Widrow and Stearns 1985; Haykin 1996) discussed later.

For the filtering problem, the gradient of the cost function was given in Eq. 2.7, and is rewritten here as a function of the tap weight vector at the nth time step,

$$\nabla J(n) = -2(\mathbf{p} - \mathbf{R}\mathbf{w}(n)) \tag{2.10}$$

Substituting this result into Eq. 2.9 yields

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(\mathbf{p} - \mathbf{R}\mathbf{w}(n)) \tag{2.11}$$

The path in L-dimensional space taken by the tap weight vector during convergence is of particular interest. Because the steepest descent algorithm changes the weight vector according to the gradient of the cost surface, it is reasonable to assume the shape of that surface will determine the convergence behavior of $\mathbf{w}$. This hypothesis is confirmed with a few manipulations of the weight update relation in Eq. 2.11, and using the fact that $\mathbf{p} = \mathbf{R}\mathbf{w}_{opt}$ (Eq. 2.8) to produce

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(\mathbf{R}\mathbf{w}_{opt} - \mathbf{R}\mathbf{w}(n)) \quad (2.12)$$
$$= \mathbf{w}(n) + \mu\mathbf{R}(\mathbf{w}_{opt} - \mathbf{w}(n)) \quad (2.13)$$

A weight error vector is defined as $\boldsymbol{\epsilon}(n) = \mathbf{w}(n) - \mathbf{w}_{opt}$, which describes the deviation of $\mathbf{w}(n)$ from the optimal vector at iteration n. Subtracting the optimal weight vector from both sides of Eq. 2.13 and substituting $\boldsymbol{\epsilon}$ yields

$$\mathbf{w}(n+1) - \mathbf{w}_{opt} = \mathbf{w}(n) - \mathbf{w}_{opt} + \mu\mathbf{R}(\mathbf{w}_{opt} - \mathbf{w}(n)) \quad (2.14)$$
$$\boldsymbol{\epsilon}(n+1) = (\mathbf{I} - \mu\mathbf{R})\boldsymbol{\epsilon}(n) \quad (2.15)$$

where $\mathbf{I}$ is the identity matrix. This is a difference equation describing the behavior of the weight error vector as a function of the iterative step number, n. Further simplification is possible by expressing the correlation matrix $\mathbf{R}$ in terms of its eigenvectors and eigenvalues. Specifically, a non-defective (i.e. distinct eigenvalues) square matrix can be diagonalized with a similarity transformation $\mathbf{R} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}'$ (Strang 1988), where the columns of $\mathbf{V}$ contain the eigenvectors of $\mathbf{R}$, and $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues of $\mathbf{R}$. The ith eigenvalue is denoted λ_i. Substituting for $\mathbf{R}$ in Eq. 2.15 produces

$$\boldsymbol{\epsilon}(n+1) = (\mathbf{I} - \mu\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}')\boldsymbol{\epsilon}(n) \quad (2.16)$$
$$= \mathbf{V}(\mathbf{I} - \mu\boldsymbol{\Lambda})\mathbf{V}'\boldsymbol{\epsilon}(n) \quad (2.17)$$

At this point it is helpful to introduce a new symbol $\boldsymbol{\xi}(n) = \mathbf{V}'\boldsymbol{\epsilon}(n)$, so the expression can be written

$$\boldsymbol{\xi}(n+1) = (\mathbf{I} - \mu\boldsymbol{\Lambda})\boldsymbol{\xi}(n) \quad (2.18)$$

The variable $\boldsymbol{\xi}$ represents a transformation of the weight error vector, $\boldsymbol{\epsilon}$, to the principal coordinates, or principal components, of the adaptive algorithm (Widrow and Stearns 1985; Nelson and Elliott 1992; Haykin 1996). The transformation of the control problem to its principal coordinates is often used to analyze the convergence behavior of an adaptive algorithm. This is a useful form for studying the convergence because the matrix $\boldsymbol{\Lambda}$ is diagonal, hence the principal components, or modes, of the control system are decoupled and converge independently of one another. A recursion relation can therefore be written to describe the behavior of the ith mode as

$$\xi_i(n+1) = (1 - \mu\lambda_i)\xi_i(n) \quad (2.19)$$

Thus the convergence behavior of the weight vector in the steepest descent algorithm is determined by the eigenproperties of the matrix $\mathbf{R}$.

Equation 2.19 can be used to bound the step size parameter, μ. Consider the convergence behavior of the ith principal coordinate ξ_i, whose initial value at time $n = 0$ is denoted by $\xi(0)$. Solving Eq. 2.19 for $\xi_i(n)$ gives the solution (Nelson and Elliott 1992; Haykin 1996)

$$\xi_i(n) = (1 - \mu\lambda_i)^n\xi_i(0) \quad (2.20)$$

The algorithm will converge if each transformed error term ξ_i goes to zero as n goes to infinity, which will occur as long as $|1 - \mu\lambda_i| < 1$ for all eigenvalues. Because the matrix $\mathbf{R}$ is positive definite, its eigenvalues are all greater than zero, hence the constraint on μ is satisfied only if

$$0 < \mu < \frac{2}{\lambda_i} \quad \text{for all } i \quad (2.21)$$

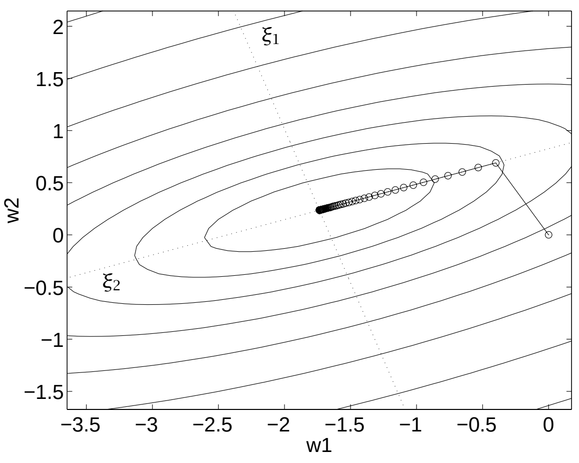

Figure 2.3: Convergence behavior of weight vector

The upper bound on μ is given by $\mu < (2/\lambda_{max})$, where λ_{max} is the largest eigenvalue of **R**.

Assuming μ is bounded by the largest eigenvalue, ξ_i will decrease with time and will be contained by an exponential envelope. The time constant for the decay of the ith mode is (Haykin 1996)

$$\tau_i = \frac{-1}{\ln(1-\mu\lambda_i)} \tag{2.22}$$

The time constant decreases if either μ or λ_i increases. Therefore the principal coordinates with large eigenvalues will converge more quickly than principal coordinates with small eigenvalues. This gives rise to the terminology “fast” and “slow” modes of convergence to describe the convergence behavior of the tap weight vector **w** in the steepest descent algorithm. The convergence speed of the slow modes can be sped up by increasing the step size μ; however, the maximum step size is constrained by the maximum eigenvalue as shown above. If the ratio of the largest to the smallest eigenvalue of **R** is very large, the convergence speed of the algorithm could be severely limited by the convergence of the slow modes. On the other hand, if the condition number of **R** is unity, ideal convergence behavior can be obtained because there are no slow modes of convergence.

The convergence of a tap weight vector that is adjusted using the steepest descent algorithm is shown on a hypothetical error surface in Fig. 2.3. This is the same error surface that was shown earlier in Fig. 2.2, and the correlation matrix used to generate the error surface had a condition number of 10.0. The value of the weight vector at each iteration during convergence is indicated by a circle, and a line connecting the circles defines the convergence path of **w**. The path starts at the right-hand side of the plot, where $w_1 = w_2 = 0$, and the final value of the weight vector is at the center of the plot where the cost function has its minimum. For this example, the step size μ was set to one half of its maximum theoretical value, or $\mu = 1/\lambda_{max}$. The path of the weight vector initially moves in the steepest direction, nearly parallel to the first principal axis, ξ_1. Once the cost function is minimized in this direction, the weight vector begins to move in the direction of the second principal axis, ξ_2, eventually reaching the cost function minimum. If μ had been set to a larger value, the convergence path would have resembled a damped sine wave with oscillations about the path shown in the plot. If μ had been set to a smaller value, convergence would have taken longer and the path in the figure would be smoother. The convergence behavior of **w** can thus be described as being underdamped if $1/\lambda_{max} < \mu < 2/\lambda_{max}$, critically damped if $\mu = 1/\lambda_{max}$ as shown in the figure, or overdamped if $0 < \mu < 1/\lambda_{max}$.

This discussion illustrates a fundamental weakness of the steepest descent algorithm that carries over to related algorithms: the convergence speed is dependent on the eigenvalue spread of the input correlation matrix **R**. The maximum value of the step size parameter μ is determined by λ_{max}, but the convergence time is almost always limited by λ_{min}, so when **R** has a large condition number the convergence time can be excessively long. The only time the slowest mode will not limit the convergence speed is if the tap weight vector happens to be initialized on a principal axis of one of the fast modes of **R**, in which case the convergence path would always be orthogonal to the principal axis of a slow mode. Because **R** describes the statistics of the input signal, the convergence is said to be dependent on the statistics of the input signal.

For the SISO system pictured in Fig. 2.1, **R** describes the correlation between the tap inputs. If the input data is white, the tap inputs are uncorrelated, and **R** is diagonal with a condition number of 1. The convergence of the steepest descent algorithm will be optimal in this case, since all modes converge at the same rate. As the dynamic range of the input signal increases and certain spectral components have more power than others, the condition number of **R** will increase, causing a corresponding decrease in the convergence speed of a steepest descent algorithm. In fact, the minimum and maximum eigenvalues of the correlation matrix of a discrete time stochastic process are bounded by the minimum and maximum values of the power spectral density of the process (Haykin 1996). An input signal with a large dynamic range could present problems for a steepest descent-type of algorithm. Later in the chapter a few methods will be described that seek to improve the convergence speed of gradient descent algorithms by pre-whitening a non-white input signal.

2.3 LMS algorithm

The steepest descent algorithm is a convenient way to solve for the optimal tap weight vector, but it requires knowledge of the exact gradient at each time step. A more practical algorithm is the least mean square (LMS) algorithm, which uses the instantaneous estimate of the gradient at each time step (Widrow and M. E. Hoff 1960). Many practical implementations of adaptive filters use the LMS algorithm, in part because it is easy to program, debug, and understand (Widrow and Stearns 1985). Principal component control is derived from the LMS algorithm, hence it is worth describing some of the details of the algorithm here.

The steepest descent algorithm is called a deterministic gradient descent algorithm because the true gradient is known at each time step, whereas the LMS algorithm is called a stochastic gradient descent algorithm because it relies on an estimate of the gradient (Haykin 1996; Cowan and Grant 1985). The LMS algorithm can be easily derived from the weight update equation for steepest descent, given earlier as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(\mathbf{p} - \mathbf{R}\mathbf{w}(n)) \tag{2.23}$$

where $(\mathbf{p} - \mathbf{R}\mathbf{w}(n))$ is the deterministic gradient of the cost function. An equivalent form of this equation is obtained by rearranging the expected value of Eq. 2.1 as follows:

$$\begin{aligned} \mathrm{E}\left[e(n)\mathbf{x}'(n)\right] &= \mathrm{E}\left[d(n)\mathbf{x}'(n)\right] - \mathrm{E}\left[\mathbf{w}'\mathbf{x}(n)\mathbf{x}'(n)\right] && (2.24)\\ &= \mathbf{p}' - \mathbf{w}'\mathbf{R} && (2.25)\\ \mathrm{E}\left[\mathbf{x}(n)e(n)\right] &= \mathbf{p} - \mathbf{R}\mathbf{w} && (2.26) \end{aligned}$$

The instataneous approximation to the left hand side is $(\mathbf{x}(n)e(n))$. Substituting this estimate for the gradient into the weight update equation yields

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{x}(n)e(n) \tag{2.27}$$

which is the weight update equation for the LMS algorithm (Widrow and M. E. Hoff 1960). The simplicity of this algorithm is apparent, since at each time step the weight update is dependent only on the current input vector $\mathbf{x}(n)$ multiplied by the current error, $e(n)$.

The stochastic gradient makes the LMS algorithm very easy to implement but unfortunately it complicates an analysis of the convergence behavior of the algorithm. Various discussions of the convergence behavior of the LMS algorithm, given with varying degrees of mathematical rigor, can be found in the references (Cowan and Grant 1985; Haykin 1996; Widrow and Stearns 1985). Fortunately, because the LMS algorithm is closely related to the steepest descent algorithm, an understanding of the general behavior of the LMS algorithm can be obtained by comparison with steepest descent. In particular, the average behavior of the weight error vector in the LMS algorithm resembles the behavior of the weight error vector in the steepest descent algorithm (Cowan and Grant 1985), which was shown earlier in Eq. 2.15. The two algorithms are similar only on an average basis, because the convergence path taken by a single realization of the LMS algorithm will look like a noisy version of the convergence path for steepest descent. If one were to average many such convergence paths for the LMS algorithm the result would resemble the path taken by the steepest descent algorithm. Just as the convergence of the weight vector for the steepest descent algorithm was shown to be given by a combination of decaying exponential terms each with its own time constant of decay, the convergence of the LMS algorithm is given by a summation of noisy decaying exponential terms (Haykin 1996). This means the convergence behavior of the LMS algorithm depends on the statistics of the input signal, just as was true for the steepest descent algorithm, and thus one can speak of "fast" and "slow" modes of convergence. Each mode corresponds to a different frequency component of the tap-delay input vector, $\mathbf{x}$.

To maintain stable convergence of the LMS algorithm, the step size parameter must be bounded by constraints similar to those derived for the steepest descent algorithm. The references given different values for the precise bounds on μ, which is perhaps an indication of the difficulty in rigorously analyzing the convergence behavior of a stochastic gradient algorithm. Haykin (1996) states that the step size parameter must be bounded just as for steepest descent,

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{2.28}$$

where λ_{max} is the maximum eigenvalue of the correlation matrix $\mathbf{R}$. A slightly more restrictive bound on μ that is three times smaller than in Eq. 2.28 is given in Cowan and Grant (1985). A more practical approach, based on the reasoning that λ_{max} is not generally known, is to bound μ by (Elliott and Nelson 1993)

$$0 < \mu < \frac{2}{\mathrm{tr}(\mathbf{R})} \tag{2.29}$$

where $\mathrm{tr}(\mathbf{R})$ is the trace of $\mathbf{R}$, i.e. the sum of the eigenvalues of $\mathbf{R}$. The trace of $\mathbf{R}$ is approximately equal to the power of the input signal x times the number of filter coefficients (Widrow and Stearns 1985).

The noisy gradient in the LMS algorithm prevents the final value of the tap weight vector from converging to the optimal weight vector, and instead $\mathbf{w}$ oscillates about the optimum value. The magnitude of the oscillations is called the convergence misadjustment (Widrow and Stearns 1985; Cowan and Grant 1985; Haykin 1996), and is proportional to the step size parameter μ. A high value of μ produces fast convergence but higher misadjustment once converged, while a small μ produces less final misadjustment at the expense of increased convergence time.

2.4 Convergence Improvements

There are numerous ways to accelerate the convergence of the LMS and steepest descent algorithms when the condition number of the input correlation matrix is high. Methods discussed in this section include transformations of the input signal to decouple the filtering problem, and other techniques to pre-whiten the input signal. These approaches are useful with the principal component control algorithm.

2.4.1 Newton's Method

The first modification discussed is the classical optimization technique called Newton's method. The basic approach is similar to steepest descent in that gradient information is used to adjust the weight vector, but Newton's method also uses information about the shape of the error surface to accelerate convergence (Haftka and Gurdal 1992). For a quadratic cost function this additional information can produce convergence to the optimum in a single step, which is similar to the convergence of the steepest descent algorithm when the condition number of **R** is unity.

In the context of the adaptive filtering problem, the weight update equation using Newton's method is written

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{R}^{-1}(\mathbf{p} - \mathbf{R}\mathbf{w}) \tag{2.30}$$

where for this discussion the deterministic gradient $(\mathbf{p} - \mathbf{R}\mathbf{w})$ is used, and $\mathbf{R}^{-1}$ denotes the inverse of the input correlation matrix. The weight vector will converge from an arbitrary starting position to the optimal weight vector in a single step if the step size parameter $\mu = 1$. Otherwise, μ must be bounded between 0 and 2 for the algorithm to converge (Haykin 1996; Haftka and Gurdal 1992).

Applying the same type of analysis of the convergence behavior of Newton's algorithm as was used for steepest descent reveals that the convergence does not depend on the eigenvalue spread of **R**. Substituting $\mathbf{p} = \mathbf{R}\mathbf{w}_{opt}$ into Eq. 2.30 gives

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{R}^{-1}(\mathbf{R}\mathbf{w}_{opt} - \mathbf{R}\mathbf{w}) \tag{2.31}$$

Subtracting the optimal weight vector from both sides, and substituting the weight error vector, $\boldsymbol{\epsilon}(n) = \mathbf{w}(n) - \mathbf{w}_{opt}$, yields (Haykin 1996)

$$\mathbf{w}(n+1) - \mathbf{w}_{opt} = \mathbf{w}(n) - \mathbf{w}_{opt} + \mu\mathbf{R}^{-1}(\mathbf{R}\mathbf{w}_{opt} - \mathbf{R}\mathbf{w}(n)) \tag{2.32}$$

$$\boldsymbol{\epsilon}(n+1) = \boldsymbol{\epsilon}(n) - \mu\mathbf{R}^{-1}\mathbf{R}(\mathbf{w}(n) - \mathbf{w}_{opt}) \tag{2.33}$$

$$= (1-\mu)\boldsymbol{\epsilon}(n) \tag{2.34}$$

the solution of which, as a function of the initial value of the weight error vector $\boldsymbol{\epsilon}(0)$, is given by the difference equation

$$\boldsymbol{\epsilon}(n) = (1-\mu)^n\boldsymbol{\epsilon}(0) \tag{2.35}$$

Therefore convergence of $\boldsymbol{\epsilon}$ is determined only by the step size parameter μ, and not by the eigenvalue spread of the correlation matrix **R**, in contrast to the steepest descent algorithm. One interpretation of this result is that for a quadratic cost function the weight update vector always points in the direction of the optimum weight vector, which usually is not in the same direction as the negative of the gradient. In terms of the error surface, Newton's method causes the weight vector to converge as if the condition number of **R** were unity.

Unfortunately Eq. 2.30 is impractical for most filtering applications because both the deterministic gradient and the inverse of the correlation matrix must be known. A slightly more practical algorithm, callled the LMS/Newton algorithm (Widrow and Stearns 1985) is obtained by substituting the stochastic gradient into the weight update relation to obtain

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \mathbf{R}^{-1}\mathbf{x}(n)e(n) \tag{2.36}$$

This algorithm represents the ideal scenario where the correlation matrix **R** is known *a priori* and thus its inverse can be computed. A further approximation to this algorithm is obtained by using an estimate of the input correlation matrix **R** that is computed from sampled data, resulting in the Quasi-Newton adaptive filtering algorithm (Marshall and Jenkins 1992). This algorithm has the fast convergence behavior of Newton's algorithm but does not require *a priori* information about the statistics of the input signal. The ability of this algorithm to track changes in the statistics of the input signal depends on how fast the estimate of $\mathbf{R}^{-1}$ can be updated.

2.4.2 Recursive Least Squares

The recursive least squares (RLS) algorithm is similar to the Quasi-Newton algorithm because an estimate of the inverse correlation matrix is computed from sampled data. The estimate is computed recursively, hence the name of the approach. The RLS algorithm is computationally intensive, but its convergence performance approaches that of the 'ideal' LMS/Newton algorithm, i.e. independent of the statistics of the input signal. The RLS algorithm does have some serious drawbacks, including its high computational burden and susceptibility to numerical ill-conditioning (Haykin 1996; Marshall and Jenkins 1992), both of which make the algorithm impractical for many applications. The algorithm is summarized very briefly here, only to illustrate an interpretation of RLS in the context of Newton's algorithm. A much more thorough treatment of the complexities and power of the RLS algorithm can be found in Haykin (1996, Chapter 13).

The recursive least squares algorithm is based on the concept of sampled data properties instead of ensemble average properties (Haykin 1996). The RLS algorithm optimizes a least squares performance metric based on a finite duration sampling of data, with no assumptions about the ensemble average properties of the data. For the SISO filtering application, where the error $e(n)$ is defined to be the difference between the actual and desired filter outputs at time n, a cost function for the exponentially weighted recursive least squares algorithm defined over a sampling of i points is written (Haykin 1996)

$$\mathcal{J} = \sum_{n=1}^{i} \alpha^{i-n} |e(n)^2| \tag{2.37}$$

The symbol $\mathcal{J}$ distinguishes this cost function from the cost function defined for the steepest descent and LMS algorithms. The term α^{i-n}, where α is a real scalar that is close to but less than 1, defines an exponential weighting of the squared errors that emphasizes more recent errors relative to older errors. This weighting term, in conjunction with the finite length of data over which the cost function is defined, provide the RLS algorithm with the ability to track changes in the statistics of the input signal (Haykin 1996). The solution to this cost function will vary with i, the number of points over which the summation in Eq. 2.37 is defined.

In contrast to this new cost function, the cost function for the steepest descent and LMS algorithms is defined as the expected value of the filter error, which is a function of the ensemble average quantities **R** and **p**. If the input and desired time series are assumed to be wide-sense stationary, the distinction between the two cost functions is relatively minor as far as an implementation of either algorithm is concerned. The distinction will have greater meaning, however, for understanding how each algorithm tracks signal statistics that are slowly varying with time. In theory the RLS algorithm should outperform the LMS algorithm if the statistics of the data are time varying. However, recent work has suggested that this may not be true, and that LMS may be as good as or better than RLS in tracking time-varying properties (Haykin 1996; Nelson and Elliott 1992).

The derivation of the RLS algorithm is quite involved (e.g., see Haykin (1996)), but the algorithm can be summarized by the four equations given below:

$$[\mathbf{\Phi}^{-1}(n)\mathbf{x}(n)] = \frac{\alpha^{-1}\mathbf{P}(n-1)\mathbf{x}(n)}{1+\alpha^{-1}\mathbf{x}'(n)\mathbf{P}(n-1)\mathbf{x}(n)} \tag{2.38}$$

$$\xi(n) = d(n) - \mathbf{w}'(n-1)\mathbf{x}(n) \tag{2.39}$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + [\mathbf{\Phi}^{-1}(n)\mathbf{x}(n)]\xi(n) \tag{2.40}$$

$$\mathbf{P}(n) = \alpha^{-1}\mathbf{P}(n-1) - \alpha^{-1}[\mathbf{\Phi}^{-1}(n)\mathbf{x}(n)]\mathbf{x}'(n)\mathbf{P}(n-1) \tag{2.41}$$

where $\mathbf{\Phi}$ is the recursively computed estimate of the input correlation matrix, **R**, α is an exponential weighting factor from the definition of the cost function (Eq. 2.37), **P** is an $(L \times L)$ matrix used to recursively update $\mathbf{\Phi}^{-1}$, and $\zeta(n)$ is the prediction error, similar to the filtering error, $e(n)$. The input signal $\mathbf{x}(n)$ and the weight vector $\mathbf{w}(n)$ are as defined previously. The algorithm is initialized by setting the matrix $\mathbf{P} = \delta\mathbf{I}$, where δ is a small positive constant, and by setting **w** equal to zero. At each time step, a new input vector $\mathbf{x}(n)$ and desired output $d(n)$ are measured or given.

Equation 2.38 results from the application of the matrix inversion lemma (e.g., see (Haykin 1996)) to recursively compute $[\mathbf{\Phi}^{-1}(n)\mathbf{x}(n)]$, the product of the inverse sample correlation matrix and the input vector. Equation 2.39 computes the filtering error used in the weight update. This error is slightly different from the error computed in the steepest descent and LMS algorithms, since $\xi(n)$ is a function of $\mathbf{w}(n-1)$, not $\mathbf{w}(n)$. Equation 2.40 is the weight update recursion. Comparing this equation with the update used for the LMS/Newton algorithm in Eq. 2.36, reveals that the main difference between the two is that $\mathbf{\Phi}^{-1}$ is used in the RLS update, whereas $\mathbf{R}^{-1}$ is used in the LMS/Newton algorithm. Equation 2.41 updates **P**, in preparation for next recursive update of $(\mathbf{\Phi}^{-1}(n)\mathbf{x}(n))$.

The attraction of the algorithm is due in part to the fact that the inverse of $\mathbf{\Phi}$ is computed by a recursion instead of direct inversion. The principal drawback of the RLS algorithm is the high computational burden of Eqs. 2.38–2.41, which have to be solved at each time step. The algorithm is also subject to numerical ill-conditioning, due to the accumulation of finite-precision errors in the **P** and $(\mathbf{\Phi}^{-1}(n)\mathbf{x}(n))$ terms (Marshall and Jenkins 1992).

2.4.3 Self-Orthogonalizing Adaptive Filters

A different interpretation of the LMS/Newton algorithm can be used to derive a self-orthogonalizing adaptive filtering algorithm (Panda et al. 1986; Haykin 1996). A rearrangement of the LMS/Newton update equation reveals an equivalent adaptive approach based on transforming the input vector, $\mathbf{x}(n)$, to an orthogonal coordinate system, and then computing the filter outputs and weight updates in terms of the transformed coordinates. By virtue of the transformation, each orthogonal coordinate has its own step size parameter, and therefore the convergence speed of each coordinate can be set such that the overall convergence speed is independent of the eigenvalue spread of the input correlation matrix. The principal component algorithm is equivalent to the application of this self-orthogonalizing concept to a multichannel, single frequency, filtering problem.

The self-orthogonalizing adaptive filter is obtained by expressing the correlation matrix **R** in the LMS/Newton weight update equation in terms of its eigenvectors and eigenvalues via a similarity transformation. Earlier, it was shown that the square matrix **R** could be expressed as the product $\mathbf{R} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}'$, where the columns of **V** contain the eigenvectors of **R**, and $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues of **R**. The orthonormal eigenvectors in **V** can be used to rotate the input vector **x** to an alternate coordinate system. Let the vector **q** denote the transformed input vector **x**, such that the ith element of **q** is given by

$$q_i(n) = \mathbf{v}_i'\mathbf{x}(n) \tag{2.42}$$

where $\mathbf{v}_i$ is the ith eigenvector of **R**. Using the ortho-normality property of the eigenvectors, the correlation between the q_i is

$$\mathrm{E}\left[q_i(n)q_j(n)\right] = \begin{cases} \lambda_i & i = j \\ 0 & i \neq j \end{cases} \tag{2.43}$$

where λ_i is the eigenvalue corresponding to the ith eigenvector. This demonstrates that the q_i are mutually orthogonal, and their correlation matrix is given by the diagonal matrix of eigenvalues, $\boldsymbol{\Lambda}$.

Substituting the matrices **V** and $\boldsymbol{\Lambda}$ for **R** in the LMS/Newton weight update equation and rearranging yields

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{V}\boldsymbol{\Lambda}^{-1}\mathbf{V}'\mathbf{x}(n)e(n) \tag{2.44}$$

$$\mathbf{V}'\mathbf{w}(n+1) = \mathbf{V}'\mathbf{w}(n) + \mu\boldsymbol{\Lambda}^{-1}\mathbf{V}'\mathbf{x}(n)e(n) \tag{2.45}$$

$$\boldsymbol{\nu}(n+1) = \boldsymbol{\nu}(n) + \mu\boldsymbol{\Lambda}^{-1}\mathbf{q}(n)e(n) \tag{2.46}$$

where the new variable $\boldsymbol{\nu}(n) = \mathbf{V}'\mathbf{w}(n)$ represents a transformation of the tap weight vector to orthogonal coordinates. The update relation for the ith coordinate is written (Haykin 1996)

$$\nu_i(n+1) = \nu_i(n) + \frac{\mu}{\lambda_i}q_i(n)e(n) \tag{2.47}$$

The significance of this transformation is that each coordinate q_i has a different step size, (μ/λ_i), that is inversely proportional to the ith eigenvalue. Therefore, as the eigenvalue decreases the step size increases, effectively compensating for the slower convergence of the slow modes of the algorithm. Because Eq. 2.47 was obtained merely by transforming the weight update equation for the LMS/Newton algorithm, it demonstrates how the convergence behavior of LMS/Newton can be interpreted in terms of individual step sizes applied to the principal components of the algorithm. Scaling each coordinate q_i by the inverse of the ith eigenvalue is equivalent to changing the eigenvalue spread of the input correlation matrix to unity, making the input signal appear as if it were white. In terms of the cost surface, the transformation and scaling converts the hyper-ellipsoidal contours to hyper-spheroidal contours (Marshall et al. 1989), so for example the ellipses of the error surface shown in Fig. 2.3 would appear as circles in terms of the transformed and scaled coordinates.

The weight update relation in Eq. 2.47 suggests a method for accelerating the convergence of the LMS algorithm:

1. transform the input data to a set of orthogonal coordinates
2. scale each coordinate to make the eigenvalue spread in terms of the transformed coordinates equal to unity
3. adapt the tap weights in the transformed coordinate system

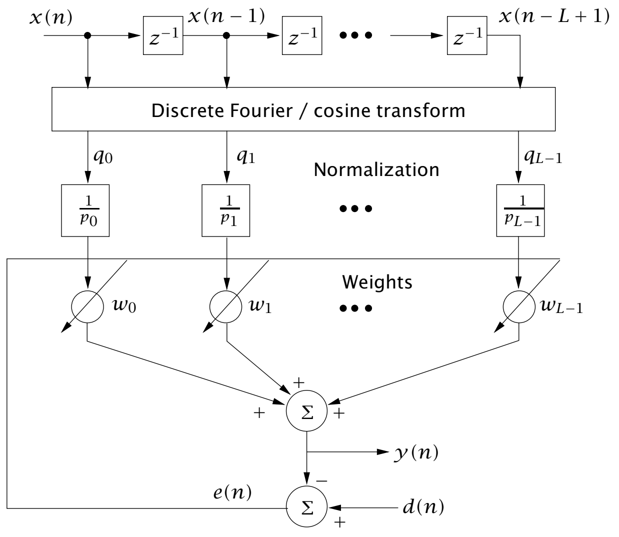

Figure 2.4: Schematic of transform domain LMS (Beaufays 1995)

The main drawback of this approach is that the eigenvectors of **R** must be known in order to orthogonalize the input signal. This orthogonalizing transformation is thus signal dependent, changing whenever the input signal statistics change. This is a problem when the statistics of the input signal are not known *a priori* but fast convergence is desired nonetheless. A signal independent transformation, that nearly orthogonalizes the input signal, is discussed in the next section.

2.4.4 Transform Domain Adaptive Filters

A signal independent transformation that almost orthogonalizes many input signals is described here. The approach is based on transforming the input time domain signal to the frequency domain via either a discrete Fourier transform (DFT) or discrete cosine transform (DCT). The result is an adaptive filter that operates in the frequency domain on the orthogonal individual frequency components. The transformation is said to be "signal dependent" because the computations to transform an arbitrary time domain signal to the frequency domain, or cosine frequency domain, depend only on the length of the input time sequence being transformed, and not on the statistical properties of the time sequence.

Frequency domain adaptive filters have been used to reduce the computational requirements of the LMS algorithm when the input vector contains many tap values (Ferrara 1980), as well as to accelerate the convergence of the LMS algorithm (S. Shankar Narayan 1983; Marshall et al. 1989; Beaufays 1995; Lee and Un 1986). The DFT and DCT are used in the same manner as the eigenvector transformation that was used in the self-orthogonalizing adaptive filter. However, for a fixed number of input data points, the DFT and DCT transforms are constant, hence the transformation is no longer signal dependent. The drawback of this approach is that there will be leakage due to the finite length of the transform window, resulting in some correlation between adjacent frequency bins. This correlation means neither the DFT nor the DCT can be used to reduce the eigenvalue spread of an input signal to unity, although they can still be used to greatly reduce the eigenvalue spread and thereby accelerate convergence of the LMS algorithm. Because the DFT and DCT are applied to blocks of data, the resulting adaptive algorithm is called a block adaptive algorithm (Panda et al. 1986; Haykin 1996).

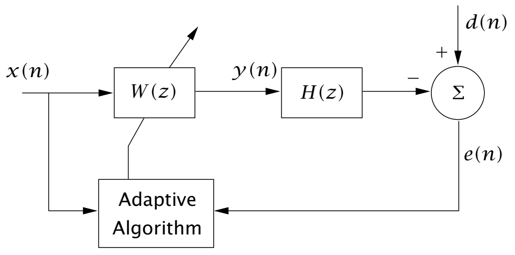

Figure 2.5: Transfer function in error path

The basic operation of a transform domain adaptive filter is illustrated in Fig. 2.4 (Beaufays 1995). The input time series is shown along the top of the plot, and it consists of L consecutively sampled values of the time series x. These L values are transformed using either the DFT or DCT to produce L frequency components, denoted $q_0, q_1, \ldots, q_{L-1}$. In the context of the self-orthogonalizing adaptive filter, this step corresponds to transforming the input vector by the eigenvectors of the correlation matrix $\mathbf{R}$, shown in Eq. 2.42. Each frequency component q_i is then scaled by the square root of its power (Beaufays 1995), denoted p_i in the figure. This step is equivalent to applying a whitening normalization to the input signal, and corresponds to scaling by the inverse eigenvalue for the self-orthogonalizing algorithm in Eq. 2.47. The filter output $y(n)$ is then computed in the transform domain, by multiplying each component q_i by a weight w_i and summing the products.

The normalization factors, p_i, represent the power per frequency bin and can be estimated using a low-pass filter or exponentially decaying window (Lee and Un 1986; Beaufays 1995). Although either the DFT or DCT could be used to transform the input signal, the DCT is generally preferred because the operations are all real, and the DCT is a better approximation of the optimal eigenvector transformation than the DFT (Beaufays 1995).

2.5 Filtered-x LMS algorithm

The last topic in this discussion of SISO filters concerns the effect of a transfer function between the filter output and the error computation. Until now it was assumed that the filtering error, $e(n)$, was the difference between the desired signal and the filter output, but in many applications there are digital-to-analog converters, analog filters, power amplifiers, or other physical components that operate on the filter output before the error is computed. These components introduce a transfer function between the control filter output and the error sensor input, which is called the error path of the filter, which must be accounted for in the weight update equation. The transfer function also modifies the eigenvalue spread of the input to the adaptive algorithm, and therefore the error path transfer function must be considered when optimizing the convergence characteristics of the LMS algorithm. The analysis here leads to the well known filtered-x LMS algorithm (Morgan 1980; Burgess 1981; Widrow and Stearns 1985).

A schematic of an adaptive filter with a transfer function in the error path is shown in Fig. 2.5. The filter input $x(n)$ is multiplied by the tap weight vector $W(z)$ to produce the output $y(n)$. This output is then filtered by the transfer function $H(z)$ such that the error signal for this configuration is written

$$\begin{aligned} e(n) &= d(n) - h(n) * y(n) && (2.48)\\ &= d(n) - h(n) * [\mathbf{w}'(n)\mathbf{x}(n)] && (2.49) \end{aligned}$$

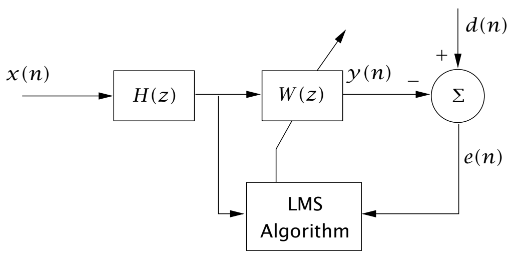

Figure 2.6: Commutation of weighting and filtering operations

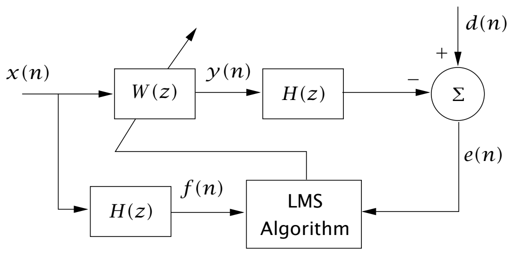

Figure 2.7: Implementation of filtered-x LMS algorithm

(Widrow and Stearns 1985)

where $h(n)$ is the impulse response of $H(z)$ at time n and $*$ denotes convolution. If the tap weight vector changes slowly and the transfer function $H(z)$ is constant, an equivalent representation is obtained by commuting the order of the filtering and weighting operations (Morgan 1980; Widrow and Stearns 1985). A block diagram with the reordered operations is shown in Fig. 2.6. This representation illustrates that the input to the adaptive algorithm is a filtered version of the signal **x**, hence the LMS algorithm based on the commutability of the filtering and weighting operations is called the "filtered-x" LMS algorithm (Widrow and Stearns 1985). In the actual implementation of the algorithm, the order of the filtering and weighting operations is shown in Fig. 2.7. Rearranging Eq. 2.49 to reflect the commuted operations produces

$$\begin{aligned} e(n) &= d(n) - \mathbf{w}'(n)[h(n) * \mathbf{x}(n)] \qquad (2.50)\\ &= d(n) - \mathbf{w}'(n)\mathbf{f}(n) \qquad (2.51)\end{aligned}$$

where $\mathbf{f}(n) = h(n) * \mathbf{x}(n)$ is the input filtered by the impulse response of the transfer function $H(z)$. Commuting the filtering and weighting operations can introduce significant errors when the time series are not slowly varying. In this case, the error due to the reordered operations can be estimated and removed as described in Bronzel and Fuller (1995).

A steepest descent algorithm that includes the effect of the error path transfer function can be derived from Eq. 2.51. The cost function to be minimized is defined as the mean square of the filtering error,

$$\begin{aligned} J &= \mathrm{E}\left[e^2(n)\right] \qquad (2.52)\\ &= \mathrm{E}\left[(d(n) - \mathbf{w}'(n)\mathbf{f}(n))^2\right] \qquad (2.53)\\ &= \mathrm{E}\left[d^2(n)\right] - 2\mathbf{w}'(n)\mathrm{E}\left[\mathbf{f}(n)d(n)\right] + \mathbf{w}'(n)\mathrm{E}\left[\mathbf{f}(n)\mathbf{f}'(n)\right]\mathbf{w}(n) \qquad (2.54)\\ &= \mathrm{E}\left[d^2(n)\right] - 2\mathbf{w}'(n)\tilde{\mathbf{p}} + \mathbf{w}'(n)\tilde{\mathbf{R}}\mathbf{w} \qquad (2.55)\end{aligned}$$

where $\tilde{\mathbf{p}} = \mathrm{E}[\mathbf{f}(n)d(n)]$ is substituted for the cross correlation vector between the desired and the filtered input signal, and $\tilde{\mathbf{R}} = \mathrm{E}[\mathbf{f}(n)\mathbf{f}'(n)]$ is substituted for the correlation matrix of the filtered input. Differentiating the cost with respect to $\mathbf{w}$ produces

$$\nabla J = -2\tilde{\mathbf{p}} + \tilde{\mathbf{R}}\mathbf{w} \tag{2.56}$$

Substituting this result into the recursion relation for the weight vector yields

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(\tilde{\mathbf{p}} - \tilde{\mathbf{R}}\mathbf{w}) \tag{2.57}$$

which is the steepest descent algorithm with a transfer function in the error path. This result is nearly identical to the weight update equation for steepest descent without an error path transfer function, given in Eq. 2.11. The only difference between the two is the fact that $\tilde{\mathbf{R}}$ and vector $\tilde{\mathbf{p}}$ include the effects of the transfer function $H(z)$. Substitution of the stochastic gradient $(\mathbf{f}(n)e(n))$ into Eq. 2.57 gives the filtered-x LMS algorithm (Widrow and Stearns 1985; Kuo and Morgan 1996),

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{f}(n)e(n) \tag{2.58}$$

By analogy with the steepest descent algorithm and the convergence behavior of that algorithm as shown in Eqs. 2.13–2.19, the convergence of the filtered-x algorithm depends on the eigenvalue spread of $\tilde{\mathbf{R}}$. The eigenvalue spread of $\tilde{\mathbf{R}}$ is determined by the statistics of the input signal $\mathbf{x}$ and the frequency response of the error path transfer function $H(z)$. Even if the input signal $\mathbf{x}$ were white, the transfer function $H(z)$ could introduce spectral coloration, depending on the dynamics of the transfer function, which would slow down the convergence of the LMS algorithm. Thus any discussion of the convergence behavior of the filtered-x algorithm must account for the effect of the error path transfer function, and how it colors the input signal.

For the filtered-x LMS algorithm shown in Eq. 2.58, the bounds on μ are tighter than for the regular LMS algorithm. Elliott and Nelson (1993) state that a practical bound on μ is

$$0 < \mu < \frac{2}{P^2(L+\Delta)} \tag{2.59}$$

where P^2 is the power of the filtered input signal $f(n)$, L is the number of adaptive filter coefficients, and Δ is the sample delay of the error path transfer function. By comparison with the bounds for the LMS algorithm without the transfer function given in Eq. 2.29, the transport delay of $H(z)$ is seen to limit the maximum step size parameter.

In a practical implementation of the filtered-x LMS algorithm, the error path transfer function is almost never known exactly and is therefore modeled with an electrical filter. An FIR filter is sometimes used for this purpose; therefore, before control is applied, the coefficients of the FIR are adjusted to accurately represent the impulse response of the error path transfer function. If $\hat{h}(n)$ represents the model of the true impulse response $h(n)$, the filtered reference signal is given by

$$\mathbf{f}(n) = \hat{h}(n) * \mathbf{x}(n) \tag{2.60}$$

Errors in the model $\hat{h}(n)$ will affect the stability of the filtered-x LMS algorithm. Fortunately, it has been shown that the filtered-x LMS algorithm is remarkably robust to errors in $\hat{h}(n)$, and will converge with phase errors of as much as 90° between the estimated and true transfer functions (Morgan 1980; Widrow and Stearns 1985; Elliott et al. 1987). Phase errors will slow down convergence, however, and Morgan (1980) demonstrated that the convergence time goes to infinity as the phase error approaches 90°. Other references confirm the robustness of the filtered-x LMS algorithm to modeling errors, and conclude that the most important attribute of $\hat{H}(z)$ is that its impulse response have at least as great a transport delay as $H(z)$ (Widrow and Stearns 1985; Feintuch et al. 1993).

Chapter 3

Multichannel Adaptive Filters

The goal of this chapter is to extend SISO adaptive filtering concepts to the multichannel case, and introduce additional background material that will be useful for describing the principal component (PC) control algorithm in the next chapter. To simplify the discussion here, a frequency domain model of a multichannel filter will be used to explore the convergence behavior and robustness to modeling error of adaptive algorithms operating at a single frequency. The results here are therefore restricted to filtering problems involving deterministic signals that can be analyzed in terms of their component frequencies.

The multichannel filter will be discussed in the context of a control problem, in which control actuators are used to cancel an unwanted disturbance from multiple error sensors. In this context, the input signal is assumed to be coherent with the disturbance at the error sensors, and because this input signal is fed through the control filter weights, the controller is characterized as *feedforward*. A control problem of particular interest that motivated the development of the principal component controller is the active control of a periodic disturbance in a reverberant system, such as vibrations on a plate or an acoustic field in an enclosure. The convergence speed of an adaptive algorithm is of some importance in this application, but a more important consideration is the amount of control effort required to reduce the disturbance at the sensors. To address this problem, a control effort penalty is added to the cost function of the controller. The analysis here will illustrate that control effort can be related to the properties of the input correlation matrix, and hence methods from Chapter 2 to accelerate convergence can be used on the control effort problem for the multichannel controller. Newton's algorithm, recursive least squares, and transform domain algorithms will be briefly discussed in the context of the multichannel control problem.

As in Chapter 2, the systems being analyzed are assumed to be linear, and the time domain signals are assumed to be electrical signals.

3.1 Introduction

The feedforward multichannel control problem is depicted in the block diagram in Fig. 3.1. A double-width arrow denotes a vector of signals, and a single width arrow denotes a scalar time series. The schematic is a multichannel version of the filtered-x LMS algorithm that was shown in Fig. 2.7, where a single input signal is denoted by $x(n)$, a vector of desired or disturbance filter outputs by $\mathbf{d}(n)$, and a matrix of control filter weights by $\mathbf{W}(z)$. It is helpful to relate the elements in the figure to an acoustic control problem, such as the active control of sound in an aircraft cabin. The input signal, $x(n)$, is assumed to be coherent with a sound source that produces an acoustic field inside the aircraft cabin, and because $x(n)$ is also an input to the controller, it is usually called the reference input signal. For example, the input signal could be the sinusoidally varying output of a shaft rotation sensor on a propeller that is coherent with propeller noise at the blade passage frequency. The sound field coherent with the reference input is called the primary or disturbance sound field, and is denoted by $\mathbf{d}(n)$ in the figure. The linear relationship between $x(n)$ and the primary sound field is indicated by the unknown transfer function $\mathbf{P}(z)$ operating on $x(n)$ to produce $\mathbf{d}(n)$. The goal of the control system will be to reduce this primary sound field. In the context of the control problem, the desired output of the control filter is the opposite of the disturbance sound field, $\mathbf{d}(n)$. The terms "desired" and "disturbance" will be used interchangeably here to refer to $\mathbf{d}(n)$, since the result of applying control depends on whether the control and disturbance fields are summed or differenced at the error sensors.

An essential component of an adaptive filter is the error sensors, the outputs of which are minimized in some sense by the adaptive filter. Microphones are a convenient error sensor for the active control of sound in an enclosure, although other possibilities exist such as acoustic energy density sensors (Sommerfeldt et al. 1995). The type of sensor will be one factor in determining the physical significance of the cost function, but the discussion here is meant to be general in nature, hence the sensor type is not of particular importance. For simplicity, it will be assumed that microphones are used as error sensors, and their responses to the primary sound field are denoted by the vector $\mathbf{d}(n)$ in the figure. The microphones also respond to the outputs of the control filters, $\mathbf{y}(n)$, filtered through the error path transfer functions, $\mathbf{H}(z)$. The outputs of the control filters will be assumed to drive sound producing actuators, such as loudspeakers inside the cabin, to create a control or secondary sound field that can be adjusted to sum destructively with the primary sound field to produce a net reduction in sound at the microphones. For the acoustic control problem, the primary and secondary sound fields sum at the error sensors, as indicated by the + signs at the summation symbol in Fig. 3.1. Because the error signal for the SISO adaptive algorithms in Chapter 2 was defined as the difference between two signals, the update equations given here will differ slightly from those given earlier. The error path transfer functions represent transfer functions between the control filter outputs and the microphones, including the acoustic transport delay and the influence of the modes of the enclosure.

The physical significance of the cost function depends on the number, type, and location of the error sensors. Minimizing the outputs of many microphones scattered throughout an enclosure would approximately correspond to minimizing the acoustic potential energy in the enclosure (Nelson and Elliott 1992; Fuller et al. 1996). Alternatively, it may be more practical to minimize microphone responses that are concentrated at certain locations in the enclosure, such as near the ears of passengers or crew.

This simplistic description of the active control of sound in an aircraft cabin is sufficient for describing the components and operation of a multichannel adaptive controller. However, the actual problem of actively reducing sound in an enclosure warrants a much more thorough treatment such as that given in Nelson and Elliott (1992, Chapters 10 & 11) or Fuller et al. (1996, Chapter 9).

There are two important differences between the control problem depicted in Fig. 3.1 and the usual application of SISO adaptive filters. Much of the technology development for SISO adaptive filters was directed towards processing information bearing signals such as radar returns or communication signals for the purposes of echo cancellation, noise cancellation, and line enhancement (Morgan 1980; Widrow et al. 1975; Haykin 1996). For these problems, convergence speed is very important and very little is known *a priori* about the statistics of the signal being filtered. These considerations motivated

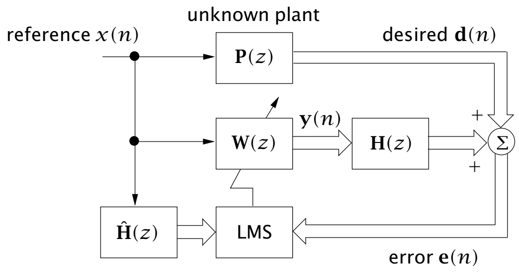

Figure 3.1: Multichannel LMS control with transfer functions in the error path

the development of fast convergence techniques that do not rely on signal dependent information (S. Shankar Narayan 1983; Beaufays 1995; Haykin 1996; Marshall et al. 1989).

In contrast, it will be assumed here that the reference input signal, and hence the primary excitation, is persistent and the transfer functions $\mathbf{P}(z)$ and $\mathbf{H}(z)$ are stationary or very slowly changing. Any changes that do occur are assumed to happen on a time scale longer than the convergence time of an adaptive controller. This assumption means there is usually sufficient time to compute a reasonably accurate estimate of the input signal statistics, including the effects of the error path transfer functions, $\mathbf{H}(z)$. In spite of these assumptions, an adaptive algorithm is still preferred over a non-adaptive one to produce optimal performance in the presence of small errors or noise in the estimates of the transfer functions and the statistics of the input signal.

A second difference between the typical SISO problem and the multichannel control problem is the importance of the control effort required to minimize the outputs of the error sensors. The outputs of the control filters drive physical actuators such as loudspeakers that require a power source, and the cost of providing this power provides some motivation for finding an efficient solution to the control problem. Limiting the control input to the actuators can also be used as a partial solution to the problem of control spillover. Control spillover refers to the situation whereby the control actuators are driven to excessively high levels to reduce the primary field at the sensors, but the summation of the primary and control fields at points away from the sensors increases compared to the primary field alone. While this effect may occur to some degree in many applications, limiting the control effort can sometimes limit the severity of the spillover (Thomas et al. 1993; Silcox et al. 1987). The control effort can also be an important consideration when the control actuators are mounted on the enclosure walls (Jones and Fuller 1987; Lester and Silcox 1992), and excessive control inputs might produce high localized structural vibrations that might not affect the acoustic field in the enclosure but do affect the fatigue life of the structure (Thomas et al. 1993; Lester and Lefebvre 1993; Cabell et al. 1992). Most of the discussion in this chapter will include consideration of a control effort penalty in the cost function.

3.2 Frequency domain description of multichannel control

The weight update algorithm for the multichannel controller is given here using a frequency domain representation. A frequency domain analysis of the multichannel controller is much simpler than a time domain analysis if the reference input is deterministic (Fuller et al. 1996; Nelson and Elliott 1992; Elliott et al. 1992) and hence can be split into its component frequencies and analyzed at each frequency. The response of the system at all frequencies can then be determined by a superposition of contributions from each frequency.

The behavior of the multichannel controller at a frequency ω will be discussed. For simplicity and without loss of generality, the reference input $x(n)$ is assumed to be a unit amplitude, zero phase complex exponential, $e^{-j\omega t}$. With this convention, the outputs of the control filters are simply the elements of **w**, a vector of complex filter weights that specifies amplitude and phase changes for each control output relative to the reference input. At a single frequency, each column of the transfer function matrix **H** represents the response of the error sensors to a unit amplitude input to a single actuator acting alone. Thus the element $H(i,j)$ is a single complex value that describes the amplitude and phase shift of the error path transfer function between the jth control filter output and the ith error sensor at the frequency ω. The desired signal, **d**, contains the complex microphone responses due to the primary excitation. The size of the control system here and in subsequent analyses is defined by the variables:

r	= number of control actuators (and control filters)
m	= number of error sensors

Using the notation discussed above, the output of the error sensors, **e**, due to the combined effects of the primary and secondary excitations is

$$\mathbf{e} = \mathbf{H}\mathbf{w} + \mathbf{d} \tag{3.1}$$

where the dependence of each term on frequency ω is implicit. The vectors **e** and **d** are each of length $(m \times 1)$, **w** is of length $(r \times 1)$, and **H** has dimensions $(m \times r)$. This frequency domain model represents the control system operating in steady state with no transients. Because the reference signal is assumed to consist of a single frequency, causality is not an issue; in addition this model cannot be used to analyze the effects of delays in the error path transfer functions, $\mathbf{H}(z)$ (Nelson and Elliott 1992; Widrow and Stearns 1985).

To simplify the ensuing discussion, we only consider the case where there are more error sensors than control filters, i.e. $m > r$, which is a common control arrangement for enclosure noise control (Nelson and Elliott 1992; Fuller et al. 1996). The resulting control problem corresponds to an over-determined system of equations with fewer unknowns (control filter weights) than equations (desired responses at the error sensors).

In Chapter 2, a cost function was defined as the expected value of the squared filtering error, and its multichannel equivalent is simply the sum of the expected values of the squared error sensors response. The frequency domain model assumes steady state conditions, so the cost function can be written

$$J = \mathbf{e}^H\mathbf{e} \tag{3.2}$$

where $()^H$ denotes the complex conjugate transpose. A more useful cost function that considers the control effort used to reduce the error is written (Elliott et al. 1987)

$$J = \mathbf{e}^H\mathbf{e} + \mathbf{w}^H\mathbf{B}\mathbf{w} \tag{3.3}$$

where the matrix **B** determines how the control filter outputs are weighted in the effort penalty. A commonly used penalty is $\mathbf{B} = \beta\mathbf{I}$, where β is a real scalar and **I** is the identity matrix. This penalty is proportional to the norm of the filter weight vector, $\mathbf{w}^H\mathbf{w}$, with equal consideration of each filter weight in the norm. Substituting Eq. 3.1 into the cost function and multiplying terms yields

$$J = \mathbf{d}^H\mathbf{d} + \mathbf{w}^H\mathbf{H}^H\mathbf{d} + \mathbf{d}^H\mathbf{H}\mathbf{w} + \mathbf{w}^H\left[\mathbf{H}^H\mathbf{H} + \beta\mathbf{I}\right]\mathbf{w} \tag{3.4}$$

This is a Hermitian quadratic form with a unique minimum if $[\mathbf{H}^H\mathbf{H}+\beta\mathbf{I}]$ is positive definite. Positive definiteness is assured at a single frequency if there are at least as many error sensors as control actuators, and no two columns of $\mathbf{H}$ are identical (Fuller et al. 1996; Nelson and Elliott 1992). The vector of complex filter coefficients that corresponds to the minimum of the cost function, J, is found by differentiating the real and imaginary parts of J with respect to $\mathbf{w}$, and setting the derivatives to zero. The resulting optimum complex weight vector is (Nelson and Elliott 1992)

$$\mathbf{w}_{opt} = -\left[\mathbf{H}^H\mathbf{H}+\beta\mathbf{I}\right]^{-1}\mathbf{H}^H\mathbf{d} \tag{3.5}$$

Proceeding as in Chapter 2, a steepest descent algorithm can be written to recursively compute the optimum weight vector from an arbitrary initial value. Using the derivative of J with respect to $\mathbf{w}$, the weight update relation is (Elliott et al. 1992; Fuller et al. 1996)

$$\mathbf{w}(n+1) = (1-\mu\beta)\mathbf{w}(n) - \mu\mathbf{H}^H\mathbf{e}(n) \tag{3.6}$$

which includes consideration of the effort penalty. The step size parameter μ controls how much the weight vector is changed at each iteration. Interpretation of this weight update equation should be done with some care (Nelson and Elliott 1992; Fuller et al. 1996), since the frequency domain representation from which this update was derived, implicitly assumes steady state behavior, but changing the weight vector at each time step violates this assumption. Nonetheless, this representation is useful for understanding the general behavior of the multichannel LMS algorithm (Elliott et al. 1987).

3.2.1 Convergence behavior

The analysis of the convergence behavior of Eq. 3.6 can be done in a similar fashion to the convergence analysis of the SISO steepest descent algorithm. A weight error vector is defined as $\boldsymbol{\epsilon}(n) = \mathbf{w}(n)-\mathbf{w}_{opt}$. Here, $\boldsymbol{\epsilon}$ is an $(r\times 1)$ vector of complex values, whereas in Chapter 2 it was a vector of real values whose length was determined by the number of tap inputs. Subtracting $\mathbf{w}_{opt}$ from both sides of Eq. 3.6 and substituting $\boldsymbol{\epsilon}$ produces

$$\boldsymbol{\epsilon}(n+1) = \left(\mathbf{I}-\mu[\mathbf{H}^H\mathbf{H}+\beta\mathbf{I}]\right)\boldsymbol{\epsilon}(n) \tag{3.7}$$

which is a difference equation for the weight error vector. This can be further simplified by expanding the bracketed term as the product of its eigenvalues and eigenvectors,

$$\left[\mathbf{H}^H\mathbf{H}+\beta\mathbf{I}\right] = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^H \tag{3.8}$$

where the columns of $\mathbf{V}$ contain the eigenvectors, and $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues, $\lambda_1,\ldots,\lambda_r$. The eigenvalues are all real because the bracketed term is a Hermitian matrix. Substituting this result into Eq. 3.7 yields

$$\begin{aligned}\boldsymbol{\epsilon}(n+1) &= \left(\mathbf{I}-\mu\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^H\right)\boldsymbol{\epsilon}(n) & (3.9)\\ &= \mathbf{V}\left(\mathbf{I}-\mu\boldsymbol{\Lambda}\right)\mathbf{V}^H\boldsymbol{\epsilon}(n) & (3.10)\\ \mathbf{V}^H\boldsymbol{\epsilon}(n+1) &= \left(\mathbf{I}-\mu\boldsymbol{\Lambda}\right)\mathbf{V}^H\boldsymbol{\epsilon}(n) & (3.11)\end{aligned}$$

The vector $(\mathbf{V}^H\boldsymbol{\epsilon})$ is the mapping of the weight error vector onto the principal coordinates of the controller. As in the SISO case, a transformed error vector is defined as $\boldsymbol{\xi}(n) = \mathbf{V}^H\boldsymbol{\epsilon}(n)$. The recursion relation for $\boldsymbol{\xi}$ is given by

$$\boldsymbol{\xi}(n+1) = \left(\mathbf{I}-\mu\boldsymbol{\Lambda}\right)\boldsymbol{\xi}(n) \tag{3.12}$$

Solving for the transformed weight error vector at time n as a function of the initial value, $\boldsymbol{\xi}(0)$, yields (Elliott et al. 1987)

$$\boldsymbol{\xi}(n) = (\mathbf{I} - \mu\boldsymbol{\Lambda})^n\boldsymbol{\xi}(0) \tag{3.13}$$

The difference equation for an individual coordinate, ξ_i, is

$$\xi_i(n) = (1 - \mu\lambda_i)^n\xi_i(0) \tag{3.14}$$

which will converge to zero as n gets large if $(0 < \mu < 2/\lambda_i)$. The maximum value of the step size parameter is given by $(0 < \mu < 2/\lambda_{max})$ where λ_{max} is the maximum eigenvalue of $[\mathbf{H}^H\mathbf{H} + \beta\mathbf{I}]$.

Equation 3.13 indicates that the convergence of the steepest descent multichannel algorithm can be described in terms of independent modes of the algorithm (Elliott et al. 1992), as for the SISO filter. The convergence of the original, untransformed weight vector $\mathbf{w}$ is the summation of contributions from the individual modes. The time constant for the decay rate of a particular mode, assuming stable convergence, is inversely proportional to the eigenvalue of that mode (Elliott et al. 1992). As in the case of the SISO filter, the largest allowable step size is determined by the maximum eigenvalue, but the slowest converging mode corresponds to the smallest eigenvalue. Thus the convergence time can be excessively long when the condition number of $[\mathbf{H}^H\mathbf{H} + \beta\mathbf{I}]$ is large, because the step size parameter cannot be increased enough to speed up the convergence of the slow modes without causing the faster modes to go unstable.

The convergence of the steepest descent algorithm in the multichannel case is therefore dependent on the properties of the matrix $(\mathbf{H}^H\mathbf{H})$. In contrast to the SISO case, where the eigenvalue spread of the input correlation matrix was determined by the spectral coloration of the input signal, the eigenvalue spread of $(\mathbf{H}^H\mathbf{H})$ is determined by the spatial arrangement of the control actuators and error sensors on the system being controlled. For the sound control problem, the acoustic response inside a rigid-walled enclosure can be described in terms of contributions from individual modes of the enclosure. At a particular frequency, the eigenvalue spread will depend on the number of contributing modes in the acoustic response relative to the number of control actuators, and the coupling between each actuator and the dominant modes. Ill-conditioning can occur if the actuators are mounted close to one another, which would mean two columns of the transfer function matrix $\mathbf{H}$ would be nearly identical. Ill-conditioning can also occur when the controlled system has low modal density and is lightly damped, even if the actuators are not physically close to one another.

The control effort penalty β can accelerate the convergence of the steepest descent algorithm when the condition number of $[\mathbf{H}^H\mathbf{H} + \beta\mathbf{I}]$ is large. If $[\lambda_1, \dots, \lambda_r]$ denote the eigenvalues of the Hermitian matrix $(\mathbf{H}^H\mathbf{H})$, the eigenvalues of $[\mathbf{H}^H\mathbf{H} + \beta\mathbf{I}]$ are equal to (Elliott et al. 1987)

$$[(\lambda_1 + \beta), (\lambda_2 + \beta), \dots, (\lambda_r + \beta)] \tag{3.15}$$

Earlier we stated that the time constant for the convergence of an individual mode of the algorithm was inversely proportional to the corresponding eigenvalue of $[\mathbf{H}^H\mathbf{H}+\beta\mathbf{I}]$. Therefore as long as β is greater than zero, the effort penalty will increase the decay rate of each mode of the algorithm. This effect will be most pronounced for modes where β is large relative to the eigenvalue. This improvement in convergence speed is not free, however, since a reduction in control effort means less reduction in the error term as well. The tradeoff between β and error reduction will be discussed later.

3.2.2 Robustness to errors in filtered-x estimate

The error path transfer functions, $\mathbf{H}(z)$, between the control filter outputs and the error sensor inputs are not usually known and hence are often modeled with a digital filter. As with the SISO filtered-x algorithm, the reference signal is pre-filtered with this transfer function model in order to compute the updates to the control filter weights. The robustness of the steepest descent algorithm to errors in the model of **H** are examined here. The analysis here shows that convergence will be stable if the eigenvalues of the product of the transfer function model and the true transfer function matrix have positive real parts. For the SISO case, this simplifies to the requirement that the phase of the model be within $\pm 90^\circ$ of its true value at the frequency of interest. The robustness analysis here will be used in the next chapter to examine the robustness of the principal component algorithm.

The robustness analysis concerns accuracy requirements on a model, denoted $\hat{\mathbf{H}}$, of the error path transfer function matrix **H**. The model can be generated in a number of ways, for example with a finite-element model of the dynamics of the error path from actuators to sensors, or using measured data to characterize the path dynamics at the frequencies of interest. For the single frequency case, $\hat{\mathbf{H}}$ is a $(m \times r)$ matrix of complex values. Each element of the model represents an estimate of the amplitude and phase shift introduced by the dynamics of the path from the control filter output to an error sensor input. The transfer function model is only used to compute the weight updates; the outputs of the error sensors are still determined by the true transfer function matrix as

$$\mathbf{e} = \mathbf{H}\mathbf{w} + \mathbf{d} \tag{3.16}$$

(which is simply a restatement of Eq. 3.1). Substituting the model $\hat{\mathbf{H}}$ into the weight update relation for the steepest descent algorithm (Eq. 3.6) yields

$$\mathbf{w}(n) = (1 - \mu\beta)\mathbf{w}(n-1) - \mu\hat{\mathbf{H}}^H\mathbf{e}(n-1) \tag{3.17}$$

which assumes an effort penalty is included in the cost function and the effort weight matrix $\mathbf{B} = \beta\mathbf{I}$. Substituting the expression for the error sensor outputs from Eq. 3.16 produces

$$\mathbf{w}(n) = (1 - \mu\beta)\mathbf{w}(n-1) - \mu\hat{\mathbf{H}}^H(\mathbf{d} - \mathbf{H}\mathbf{w}(n-1)) \tag{3.18}$$

Assuming stable convergence, the weight vector converges not to the optimum $\mathbf{w}_{opt}$ given in Eq. 3.5 but instead to (Boucher et al. 1991; Omoto and Elliott 1997)

$$\mathbf{w}_\infty = -\left[\hat{\mathbf{H}}^H\mathbf{H} + \beta\mathbf{I}\right]^{-1}\hat{\mathbf{H}}^H\mathbf{d} \tag{3.19}$$

which resembles the optimum given in Eq. 3.5 except for the presence of the model, $\hat{\mathbf{H}}$. Rearranging this equation in terms of $\hat{\mathbf{H}}^H\mathbf{d}$, substituting the result into Eq. 3.18, then subtracting the final weight vector $\mathbf{w}_\infty$ from both sides of the equation yields

$$\mathbf{w}(n) - \mathbf{w}_\infty = \left[\mathbf{I} - \mu\left[\hat{\mathbf{H}}^H\mathbf{H} + \beta\mathbf{I}\right]\right](\mathbf{w}(n-1) - \mathbf{w}_\infty) \tag{3.20}$$

$$\hat{\boldsymbol{\epsilon}}(n) = \left[\mathbf{I} - \mu\left[\hat{\mathbf{H}}^H\mathbf{H} + \beta\mathbf{I}\right]\right]\hat{\boldsymbol{\epsilon}}(n-1) \tag{3.21}$$

where $\hat{\boldsymbol{\epsilon}} = (\mathbf{w}(n) - \mathbf{w}_\infty)$ is a weight error vector. The value of the weight error vector at time n, as a function of its initial value $\hat{\boldsymbol{\epsilon}}(0)$ is given by

$$\hat{\boldsymbol{\epsilon}}(n) = \left[\mathbf{I} - \mu\left(\hat{\mathbf{H}}^H\mathbf{H} + \beta\mathbf{I}\right)\right]^n\hat{\boldsymbol{\epsilon}}(0) \tag{3.22}$$

which is nearly identical to the difference equation for the case with no modeling error, shown in Eq. 3.7.

The stability of convergence of the weight update relation in Eq. 3.18 depends on the eigenvalues of the bracketed matrix, $\left[\mathbf{I} - \mu\left(\hat{\mathbf{H}}^H\mathbf{H} + \beta\mathbf{I}\right)\right]$. The weight error vector $\hat{\boldsymbol{\epsilon}}(n)$ will converge to zero as n gets large if the absolute value of the eigenvalues of the bracketed matrix are less than 1 (Morgan

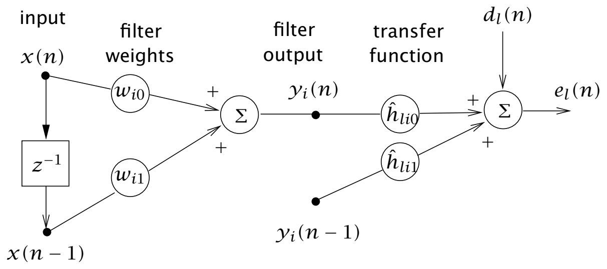

Figure 3.2: Error sensor output due to a control filter at a single frequency

1980; Boucher et al. 1991). If $\hat{\lambda}_i$ denotes the ith eigenvalue of $(\hat{\mathbf{H}}^H\mathbf{H})$, the weight updates will exhibit stable convergence if

$$|1 - \mu(\hat{\lambda}_i + \beta)| < 1 \tag{3.23}$$

The eigenvalues are not necessarily real. Rearranging this stability constraint as bounds on the step size parameter μ produces (Boucher et al. 1991)

$$0 < \mu < \frac{2(\mathrm{Re}(\hat{\lambda}_i) + \beta)}{|\hat{\lambda}_i + \beta|^2} \quad \text{for } i = 1, \ldots, r \tag{3.24}$$

If the effort penalty parameter β is zero, Eq. 3.24 can be satisfied only when the real part of each eigenvalue is positive (Morgan 1980). For the single channel case this requirement implies the phase of the transfer function model at the frequency of interest must be within $\pm 90^\circ$ of its true value. The stability requirement does not simplify as neatly into accuracy bounds on individual transfer function estimates for the multichannel case, however.

A non-zero value of β can stabilize convergence when modeling errors in $\hat{\mathbf{H}}$ are small, such as when small amounts of random measurement noise are present (Boucher et al. 1991). This is due to the fact that the real part of the numerator in Eq. 3.24 must be greater than zero for all eigenvalues, thus if the real part of a particular eigenvalue $\hat{\lambda}_i$ were negative, as long as $\beta > \mathrm{Re}(\hat{\lambda}_i)$, convergence will be stable. This stabilizing effect is most pronounced on slightly ill-conditioned control systems, where one or more eigenvalues of $(\hat{\mathbf{H}}^H\mathbf{H})$ are very small (Rossetti et al. 1996; Boucher et al. 1991).

3.3 Time domain weight update equation

At this point it is useful to give a time domain version of the weight update relation in Eq. 3.6 for a sampled data implementation of the algorithm. The resulting time domain weight update equation is called the multiple error LMS algorithm (Elliott et al. 1987). As in the SISO case, the LMS algorithm is obtained by substituting the instantaneous gradient of the cost function into the weight update equation for the steepest descent algorithm. The derivation here is considerably simplified because the reference signal is assumed to consist of a single frequency.

The time domain filter derived here is based on the notation in Fig. 3.2. The figure shows the response of the lth error sensor, $e_l(n)$, due to the output of the ith control filter, $y_i(n)$. Only two tap-delay values of the reference input signal, $x(n)$ and $x(n-1)$, are needed since the signal contains only a single frequency. The adaptively updated tap weights to the ith control filter are (w_{i0}, w_{i1}), where the first subscript denotes the control filter and the second subscript denotes which tap input is scaled by the weight. The current control filter output, $y_i(n)$, and its output at the previous time step, $y_i(n-1)$, are shown in the figure. The coefficients of an FIR filter model of the transfer function between the control filter output and the error sensor at the frequency ω is denoted $(\hat{h}_{li0}, \hat{h}_{li1})$, where the first subscript corresponds to the error sensor, the second subscript to the control filter, and the third to the time step in the impulse response model. The desired response of the lth error sensor at time n is written $d_l(n)$. An estimate of the error sensor response at time n is given by

$$e_l(n) = d_l(n) + \hat{h}_{li0}\; y_i(n) + \hat{h}_{li1}\; y_i(n-1) \tag{3.25}$$

where the estimated coefficients $(\hat{h}_{li0}, \hat{h}_{li1})$ mean Eq. 3.25 is only an estimate of the true error sensor response. The sensor response due to contributions from r control filters is written

$$\begin{aligned} e_l(n) &= d_l(n) + \sum_{i=1}^{r} (\hat{h}_{li0}\; y_i(n) + \hat{h}_{li1}\; y_i(n-1)) && (3.26) \\ &= d_l(n) + \sum_{i=1}^{r} \left(\hat{h}_{li0}\; (w_{i0}\; x(n) + w_{i1}\; x(n-1)) + \hat{h}_{li1}\; (w_{i0}\; x(n-1) + w_{i1}\; x(n-2))\right) && (3.27) \end{aligned}$$

Assuming the commutability of the weighting and filtering steps, as in the derivation of the filtered-x LMS algorithm in Chapter 2, a filtered version of the reference signal can be defined as

$$\hat{f}_{li}(n) = \hat{h}_{li0}\; x(n) + \hat{h}_{li1}\; x(n-1) \tag{3.28}$$

Substituting the filtered reference into Eq. 3.27 yields

$$e_l(n) = d_l(n) + \sum_{i=1}^{r} \left(w_{i0}\; \hat{f}_{li}(n) + w_{i1}\; \hat{f}_{li}(n-1)\right) \tag{3.29}$$

The cost function in the frequency domain was given in Eq. 3.3; its instantaneous value in the time domain, assuming no effort penalty, is written

$$J(n) = \sum_{j=1}^{m} e_j^2(n) \tag{3.30}$$

Differentiating $J(n)$ with respect to each weight produces the update equations

$$w_{i0}(n+1) = w_{i0}(n) - \mu \sum_{j=1}^{m} e_j(n)\hat{f}_{ji}(n) \tag{3.31}$$

$$w_{i1}(n+1) = w_{i1}(n) - \mu \sum_{j=1}^{m} e_j(n)\hat{f}_{ji}(n-1) \tag{3.32}$$

which represent the weight update relations for the multiple error LMS algorithm (Elliott et al. 1987), for the single frequency case.

3.4 Bias in solution due to effort penalty

This section examines how the effort penalty in the cost function affects the optimal weight vector and reduction of the error term, $\mathbf{e}^H\mathbf{e}$, relative to the case with no effort penalty. The effect of the effort penalty can be expressed in terms of the principal coordinates of the controller. These expressions will be helpful for comparing the principal component control algorithm to be discussed in the next chapter with the multichannel LMS algorithm with an effort penalty. The discussion here follows the treatment given in Elliott et al. (1992).

The singular value decomposition (SVD) of the transfer function matrix **H** will be useful in this analysis. From the theory of the SVD (e.g., see Strang (1988)), a non-square matrix **H** with dimensions $(m \times r)$ can be represented as a product of three matrices,

$$\mathbf{H} = \mathbf{U}\mathbf{S}\mathbf{V}^H \tag{3.33}$$

The matrices **U** and **V** contain the eigenvectors of $\mathbf{H}\mathbf{H}^H$ and $\mathbf{H}^H\mathbf{H}$, respectively. Both **U** and **V** are unitary, which means $\mathbf{U}^H\mathbf{U} = \mathbf{I}$ and $\mathbf{V}^H\mathbf{V} = \mathbf{I}$, where **I** is the identity matrix. The $(m \times r)$ matrix **S** contains the real singular values of **H** on its diagonal, where the singular values are the positive square roots of the eigenvalues of $\mathbf{H}\mathbf{H}^H$ and $\mathbf{H}^H\mathbf{H}$. If s_i denotes the ith singular value, **S** is written

$$\mathbf{S} = \begin{bmatrix} s_1 & 0 & \dots & 0 \\ 0 & s_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & s_r \\ & \multicolumn{2}{c}{\mathbf{0}} & \end{bmatrix} \tag{3.34}$$

assuming $m > r$. The singular values are ordered such that $s_1 > s_2 > \dots > s_r$. The structure of the matrix **S** means the transfer function matrix **H** can be written as the summation

$$\mathbf{H} = s_1\mathbf{u}_1\mathbf{v}_1^H + s_2\mathbf{u}_2\mathbf{v}_2^H + \dots + s_r\mathbf{u}_r\mathbf{v}_r^H \tag{3.35}$$

where $\mathbf{u}_i$ and $\mathbf{v}_i$ are the ith columns of the matrices **U** and **V**, respectively. This equation illustrates the close relationship between a singular value, s_i, and the corresponding columns of the matrices **U** and **V**. Because the singular values have a well-defined ordering, the singular value s_1 and columns $\mathbf{u}_1$ and $\mathbf{v}_1$ are said to be associated with the first principal coordinate of **H**. Likewise, s_r, $\mathbf{u}_r$, and $\mathbf{v}_r$ are associated with the last principal coordinate. This terminology of “first” and “last” principal coordinates of **H** will be used repeatedly throughout the remainder of this thesis.

Earlier, it was shown that the convergence of the steepest descent algorithm could be described in terms of modes of the control system, and modes with small eigenvalues were called slow modes of convergence. The ordering of the singular values from largest to smallest, and the fact that the singular values are simply the square roots of the eigenvalues, means the slow modes of convergence correspond to the last few principal coordinates of **H**, and the fast modes correspond to the first few principal coordinates. The number of principal coordinates considered to be “fast” or “slow” depends of the magnitudes of the singular values, which in turn depends on **H**.

The matrix **V** obtained from an SVD of **H** is identical to the eigenvector matrix of $(\mathbf{H}^H\mathbf{H} + \beta\mathbf{I})$, which was defined earlier as a matrix **V** in Eq. 3.8. These eigenvectors are equal because adding the diagonal effort penalty $\beta\mathbf{I}$ to $(\mathbf{H}^H\mathbf{H})$ only affects the eigenvalues, not the eigenvectors. The matrix **V** was also used earlier to transform the weight error vector $\boldsymbol{\epsilon}$ to the principal coordinates of the algorithm, resulting in a transformed weight error vector, $\boldsymbol{\xi}$. The matrix **U** from the SVD of **H** will be used in a similar fashion to transform an $(m \times 1)$ vector onto the principal coordinates of the algorithm at the error sensors.

Using the SVD of the transfer function matrix **H**, the bias introduced in the minimum cost due to the effort penalty can be investigated. Using the optimum weight vector given in Eq. 3.5 for the cost function with an effort penalty, and assuming the transfer function matrix is known exactly, the minimum cost is written as

$$J_{min} = \mathbf{d}^H \left[\mathbf{I} - \mathbf{H} \left(\mathbf{H}^H \mathbf{H} + \beta \mathbf{I} \right)^{-1} \mathbf{H}^H \right] \mathbf{d} \tag{3.36}$$

Substituting the SVD of **H** and simplifying produces

$$J_{min} = \mathbf{d}^H \mathbf{U} \left[\mathbf{I} - \mathbf{S} \left(\mathbf{S}' \mathbf{S} + \beta \mathbf{I} \right)^{-1} \mathbf{S}' \right] \mathbf{U}^H \mathbf{d} \tag{3.37}$$

Let $\mathbf{p} = \mathbf{U}^H \mathbf{d}$, which describes the mapping of the desired response **d** onto the principal coordinates at the error sensors. Using this definition to simplify Eq. 3.37 yields

$$J_{min} = \mathbf{p}^H \left[\mathbf{I} - \mathbf{S} \left(\mathbf{S}' \mathbf{S} + \beta \mathbf{I} \right)^{-1} \mathbf{S}' \right] \mathbf{p} \tag{3.38}$$

Given the diagonal singular value matrix **S**, the minimum cost due to the ith principal component, J^i_{min}, is

$$J^i_{min} = p_i^* p_i - p_i^* s_i \left(s_i^2 + \beta \right)^{-1} s_i p_i \tag{3.39}$$

where p_i^* is the complex conjugate of the ith element of the vector **p**, and s_i is the ith singular value. Multiplying $(p_i^* p_i)$ by $(s_i^2 + \beta)/(s_i^2 + \beta)$, and assuming there are more error sensors than control filters, the minimum cost can be expressed as (Elliott et al. 1992),

$$J_{min} = \sum_{i=1}^{r} \frac{\beta}{s_i^2 + \beta} |p_i|^2 + \sum_{i=r+1}^{m} |p_i|^2 \tag{3.40}$$

This equation demonstrates that the effort penalty affects only the first r elements of the vector **p**, and only has a significant affect on principal components where the singular value s_i is small relative to the penalty parameter, β. If there is no effort penalty, the minimum cost contains contributions only from the $(r+1)$ through m terms of **p**, since the summation of the first r terms of **p** is zero when β is zero. If β is not zero, the contribution of p_i to the minimum cost depends on the value of s_i relative to β. If the singular value s_i is very large relative to β, the term $\beta/(s_i^2 + \beta)$ will be very small and thus the contribution of $|p_i|^2$ to the minimum cost will also be very small. This means the control system has effectively cancelled p_i, ith principal component at the error sensors, since it is not present in the minimum cost, J_{min}. If the singular value s_i is very small relative to β, the term $\beta/(s_i^2 + \beta)$ is approximately 1, which means p_i contributes to the minimum cost, and is therefore unaffected by the control system.

It is important to note that a singular value, s_i, is independent of the mapping of the primary response onto the corresponding principal component, p_i. This means that if the effort penalty parameter β is specified such that the last several principal components are not controlled (i.e. those whose singular values are small relative to β) there is no guarantee those same principal component will not correlate strongly with the primary response. Not controlling these highly correlated PCs means not reducing the error term of the cost function, $\mathbf{e}^H \mathbf{e}$, which is the purpose of applying control in the first place. This fact should be kept in mind when a control effort term is added to the cost function, since the effort penalty could severely limit the usefulness of the controller in some situations.

The true benefit of the control effort penalty becomes apparent by examining how it affects the norm of the control effort. This norm is simply $(\mathbf{w}^H \mathbf{w})$, and can be put into a more useful form by using the weight error vector $\boldsymbol{\epsilon}$, and its transformed equivalent, $\boldsymbol{\xi}$. Recall that $\boldsymbol{\epsilon}(n) = \mathbf{w}(n) - \mathbf{w}_{opt}$,

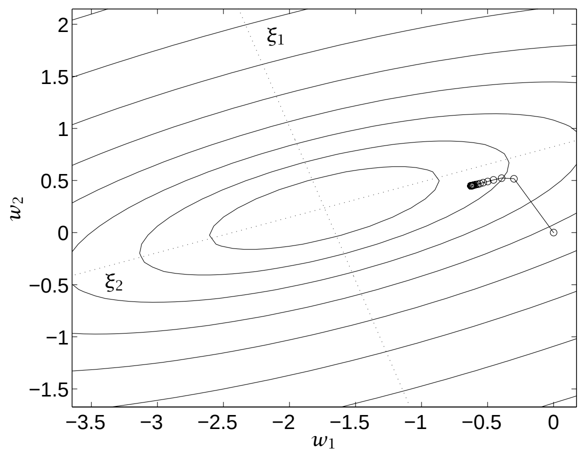

Figure 3.3: Convergence of weight vector with effort penalty

and this vector can be mapped onto the principal components by pre-multiplying by the eigenvector matrix $\mathbf{V}$, such that $\boldsymbol{\xi}(n) = \mathbf{V}^H\boldsymbol{\epsilon}(n)$. Assuming the initial weight vector, $\mathbf{w}(n)$, is the zero vector, the initial value of the transformed weight vector is $\boldsymbol{\xi}(0) = -\mathbf{V}^H\mathbf{w}_{opt}$. Substituting these expressions into the effort norm produces

$$\mathbf{w}^H(n)\mathbf{w}(n) = \left[\mathbf{V}(\boldsymbol{\xi}(n) - \boldsymbol{\xi}(0))\right]^H \left[\mathbf{V}(\boldsymbol{\xi}(n) - \boldsymbol{\xi}(0))\right] \tag{3.41}$$

$$= (\boldsymbol{\xi}(n) - \boldsymbol{\xi}(0))^H(\boldsymbol{\xi}(n) - \boldsymbol{\xi}(0)) \tag{3.42}$$

A recursion relation for the value of $\boldsymbol{\xi}(n)$ as a function of $\boldsymbol{\xi}(0)$ was given earlier in Eq. 3.13. Substituting this relation into the above expression yields

$$\mathbf{w}^H(n)\mathbf{w}(n) = \left[(\mathbf{I} - \mu\boldsymbol{\Lambda})^n\boldsymbol{\xi}(0)\right]^H \left[(\mathbf{I} - \mu\boldsymbol{\Lambda})^n\boldsymbol{\xi}(0)\right] \tag{3.43}$$

where $\boldsymbol{\Lambda}$ is the diagonal matrix containing the eigenvalues of $(\mathbf{H}^H\mathbf{H} + \beta\mathbf{I})$. Multiplying the bracketed terms, substituting for $\boldsymbol{\xi}(0)$, and letting $n \to \infty$, the norm of the optimum control effort is (Elliott et al. 1987)

$$\mathbf{w}_{opt}^H\mathbf{w}_{opt} = \sum_{i=1}^{r} \frac{s_i^2|p_i|^2}{(s_i^2 + \beta)^2} \tag{3.44}$$

which, like the minimum cost in Eq. 3.40, is a function of the singular values s_i, effort penalty parameter β, and the mapping of the desired response onto the principal components, p_i. If the parameter β is set to zero such that there is no effort penalty, the contribution of the ith principal component to the norm of the optimum control effort is equal to $(|p_i|^2/s_i^2)$. Therefore when the singular value s_i^2 is very small and $|p_i|^2$ is not zero, the ith principal component could have a very large contribution to the control effort. As the matrix $(\mathbf{H}^H\mathbf{H})$ becomes ill-conditioned, and the last few singular values become exceedingly small, the control effort for the corresponding last few principal components could be enormous. Any non-zero value of β will reduce the contribution of the ith principal component to the control effort norm, since the denominator in Eq. 3.44, $(s_i^2 + \beta)$, is greater than s_i^2 alone. When a singular value is large relative to the value of β, the contribution of the ith PC to the effort is approximately equal to its contribution with no effort penalty at all, $(|p_i|^2/s_i^2)$.

The effort penalty's effect on the minimum cost and control effort are illustrated in the convergence behavior of a weight vector when an effort penalty is included in the cost function, shown in Fig. 3.3.

The figure contains the same error surface and notation used in Chapter 2 to illustrate the convergence behavior of the steepest descent algorithm. This simple model of the convergence behavior was for a SISO system, but nonetheless is useful for discussing the behavior of a multichannel controller with an effort penalty. The convergence path of the weight vector is indicated by the connected circles, starting from the right-hand side of the plot where both weights are zero. The contours in Fig. 3.3 correspond to the value of only the $\mathbf{e}^H\mathbf{e}$ term as a function of the weight vector, and hence illustrate the bias introduced into the solution by the effort penalty. Specifically, the weight vector shown in the plot minimizes the cost function $J = \mathbf{e}^H\mathbf{e} + \beta\mathbf{w}^H\mathbf{w}$, but is plotted on a surface defined only by the term $(\mathbf{e}^H\mathbf{e})$.

We know from Chapter 2 that the shape of the error surface for the SISO filtering problem is determined by the condition number of the input correlation matrix, **R**, which was 10.0 for the data in the plot. The two singular values of the correlation matrix **R** were $(1.0, 0.32)$, and the step size μ was 0.75, or $3/8$ of its maximum theoretical value of 2.0. The effort penalty parameter β was set to 0.25 to penalize the second principal component more heavily than the first principal component. Equations 3.40 and 3.44 demonstrate that the effect of the effort penalty on a principal component can be related to the value of β relative to the singular value s_i.

The general behavior of the weight vector in Fig. 3.3 resembles the convergence behavior without an effort penalty shown in Fig. 2.3, except for not converging to the minimum of the error term. The weight vector initially converged in a direction nearly parallel to the first principal axis, ξ_1, then converged parallel to the second principal axis, ξ_2. However, the effort penalty kept the weight vector from minimizing the error term in either direction ξ_1 or ξ_2, although the weight vector was clearly constrained the greatest in the direction of the second principal axis, ξ_2. These results illustrate that a uniform effort penalty such as $\mathbf{B} = \beta\mathbf{I}$ in the cost function will have some affect on all of the modes of the control system, although each mode will be affected differently depending on its singular values relative to the parameter β.

A more sophisticated effort penalty can be implemented by using a more complicated weighting matrix in the controller cost function. One such penalty of particular interest here is called a threshold weighting penalty (Rossetti et al. 1996), which only penalizes ill-conditioned principal components of the controller. The selective nature of the penalty is achieved using the SVD of the transfer function matrix **H**, given in Eq. 3.33. The singular values in the matrix **S** are partitioned into a well-conditioned subset, called $\mathbf{S}_W$, and an ill-conditioned subset, $\mathbf{S}_I$, such that $\mathbf{S} = \mathbf{S}_W + \mathbf{S}_I$. The subsets are defined as

$$\mathbf{S}_W = \begin{bmatrix} \mathrm{diag}(s_1, s_2, \ldots, s_\rho, 0, \ldots, 0) \\ \mathbf{0} \end{bmatrix} \tag{3.45}$$

$$\mathbf{S}_I = \begin{bmatrix} \mathrm{diag}(0, \ldots, 0, s_{\rho+1}, \ldots, s_r) \\ \mathbf{0} \end{bmatrix} \tag{3.46}$$

where $\mathbf{S}_W$ contains the first ρ singular values of **S** that satisfy $s_\rho > \varepsilon$, where ε is an appropriate threshold, and $\mathbf{S}_I$ contains the remaining singular values.

The weighting matrix, **B**, for the effort penalty in the cost function is defined as

$$\mathbf{B} = \beta\mathbf{V}\left(\mathbf{I} - \mathbf{S}_W^+\mathbf{S}_W\right)\mathbf{V}^H \tag{3.47}$$

The parameter β is a real scalar as for the uniform effort penalty, and **V** is the eigenvector matrix from the SVD of **H.** The matrix $\mathbf{S}_W^+$ is the pseudo-inverse of $\mathbf{S}_W$, which is obtained by inverting the non-zero diagonal elements of $\mathbf{S}_W$. Substituting **B** defined above into the expression for the optimum weight

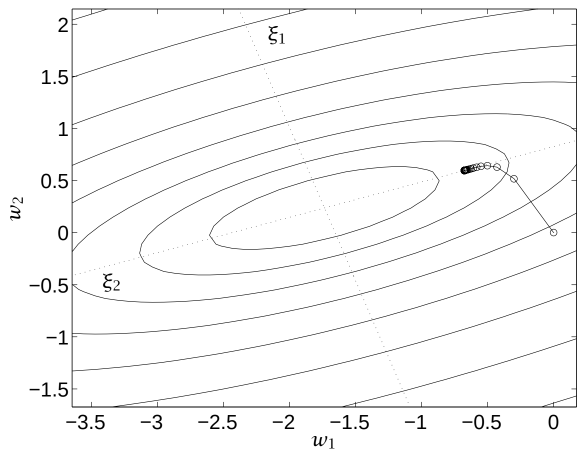

Figure 3.4: Convergence of weight vector with threshold effort penalty

vector given in Eq. 3.5 yields

$$\mathbf{w}_{opt} = -\left[\mathbf{H}^H\mathbf{H} + \beta\mathbf{V}\left(\mathbf{I} - \mathbf{S}_W^+\mathbf{S}_W\right)\mathbf{V}^H\right]^{-1}\mathbf{H}^H\mathbf{d} \tag{3.48}$$

$$= -\left[\mathbf{V}\mathbf{\Lambda}\mathbf{V}^H + \beta\mathbf{V}\left(\mathbf{I} - \mathbf{S}_W^+\mathbf{S}_W\right)\mathbf{V}^H\right]^{-1}\mathbf{H}^H\mathbf{d} \tag{3.49}$$

$$= -\left[\mathbf{V}\left(\mathbf{\Lambda} + \beta\left(\mathbf{I} - \mathbf{S}_W^+\mathbf{S}_W\right)\right)\mathbf{V}^H\right]^{-1}\mathbf{H}^H\mathbf{d} \tag{3.50}$$

The difference $(\mathbf{I} - \mathbf{S}_W^+\mathbf{S}_W)$ is a diagonal matrix whose first ρ diagonal elements are zero, and elements $(\rho + 1)$ through r are unity. Therefore, if $\lambda_1, \ldots, \lambda_r$ are the eigenvalues of $(\mathbf{H}^H\mathbf{H})$, the eigenvalues of the bracketed matrix in Eq. 3.50 are

$$(\lambda_1, \ldots, \lambda_\rho, (\lambda_{\rho+1} + \beta), \ldots (\lambda_r + \beta)) \tag{3.51}$$

This means the first ρ principal components are unaffected by the threshold effort penalty. The advantage of this penalty relative to a uniform effort penalty is that the well-conditioned principal components are minimized completely.

The convergence behavior of a weight vector when the cost function contains a threshold effort penalty is shown in Fig. 3.4. The error surface and adaptation parameters μ and β were identical to those used to generate Fig. 3.3. The threshold, ε, used to partition the singular values into well-conditioned and ill-conditioned sets was set such that the first PC was considered well-conditioned and the second PC was ill-conditioned. The convergence of the weight vector is indicated by the connected circles in the plot, and the weight vector was initialized to $w_1 = w_2 = 0$, at the right hand side of the plot. In contrast with the convergence path for a uniform effort penalty, shown in Fig. 3.3, the first principal component was completely minimized. The penalty only affected the convergence of the weight vector in the direction of the second principal component. This type of penalty would be useful when the singular values could be easily partitioned into well-conditioned and ill-conditioned sets. However, as with the uniform penalty it will still be possible to penalize principal components that are highly correlated with the primary disturbance and thereby reduce the effectiveness of the controller at reducing the error term in the cost function.

A drawback of the threshold effort penalty is the necessity of computing the eigenvector matrix $\mathbf{V}$ of $\mathbf{H}$. The matrix $\mathbf{V}$ will have to be recomputed if the transfer function matrix $\mathbf{H}$ ever changes, which could also require changing the threshold value, ε. The threshold penalty also requires additional computations in the weight update equation. The frequency domain weight update equation with a threshold effort penalty is written

$$\mathbf{w}(n+1) = (\mathbf{I} - \mu\mathbf{B})\mathbf{w}(n) - \mu\mathbf{H}^H\mathbf{e}(n) \tag{3.52}$$

which indicates the update requires multiplying $\mathbf{w}(n)$ by the full matrix $\mathbf{B}$ from Eq. 3.47. In contrast, the uniform effort penalty only required that $\mathbf{w}(n)$ be multiplied by a diagonal matrix. As for the threshold ε, Rossetti et al. (1996) recommend that it be set to penalize principal components whose singular values are less than 0.01 times the maximum singular value.

3.5 Convergence Improvements

Various methods can be used to improve the convergence behavior of the multichannel LMS algorithm in the presence of ill-conditioning. The convergence of the multichannel controller is sensitive to the spectral coloration of the filtered reference signal, just as the SISO filter was, but in addition the spatial arrangement of the sensors and actuators affects the conditioning and hence the convergence behavior. At a single frequency, the convergence is sensitive to the eigenvalues of the cross product of the transfer function matrix between actuators and sensors, $(\mathbf{H}^H\mathbf{H})$, which is determined by spatial effects and transducer responses to those effects. Three general methods for improving the convergence of a multichannel controller are briefly discussed here to demonstrate that the convergence acceleration methods described in Chapter 2 have been applied with some success to the multichannel control problem.

3.5.1 Newton's method

Newton's algorithm can be used to make the convergence of the weight vector for the multichannel control problem insensitive to the eigenvalue spread of $(\mathbf{H}^H\mathbf{H} + \beta\mathbf{I})$. Naturally the approach requires knowledge of the transfer function matrix $\mathbf{H}$, although an estimate could also be used to implement a quasi-Newton type of algorithm.

As with the SISO filter, Newton's algorithm is obtained by pre-multiplying the weight update in the steepest descent algorithm by the inverse of the second derivative of the cost function with respect to $\mathbf{w}$ (Haftka and Gurdal 1992). The second derivative for the multichannel, single frequency controller with a uniform effort penalty is equal to $[\mathbf{H}^H\mathbf{H} + \beta\mathbf{I}]$. Substituting this expression into the weight update relation for the steepest descent algorithm yields (Fuller et al. 1996)

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\left(\mathbf{H}^H\mathbf{H} + \beta\mathbf{I}\right)^{-1}\mathbf{H}^H\mathbf{e}(n) \tag{3.53}$$

The convergence of the weight vector depends only on the value of the step size parameter μ, not the eigenvalue spread of $(\mathbf{H}^H\mathbf{H} + \beta\mathbf{I})$.

One drawback of this approach is the need to precompute the matrix inverse in the weight update equation. Assuming the transfer function matrix $\mathbf{H}$ is constant, the inverse could be computed before control is applied using an estimate of $\mathbf{H}$, which produces the stochastic Newton algorithm (Nelson and Elliott 1992). A second drawback, in addition to the computational burden of computing the inverse of $(\mathbf{H}^H\mathbf{H} + \beta\mathbf{I})$, is the fact that this inverse is a full $(r \times r)$ matrix that must be multiplied with an $(r \times 1)$ vector at each iteration of the algorithm. This introduces a significant computational burden, relative to the multichannel versions of the steepest descent or LMS algorithms, that could be problematic for controllers with large numbers of actuators.

3.5.2 Recursive least squares

The recursive least squares (RLS) algorithm has also been applied to the multichannel feedforward control problem (Bronzel and Fuller 1995). As mentioned in Chapter 2, the weight update equation in RLS is similar to the weight update from Newton's algorithm, where the weight update term is pre-multiplied by the inverse of the input correlation matrix. This makes the convergence of the weight vector insensitive to the statistics of the input signal. For the multichannel control problem with a filter in the error path, the statistics of the input signal depend on both the spectral coloration of the reference, and the spatial correlation between actuators.

Application of the RLS algorithm to a control system with a filter in the error path is more complicated than the filtered-x LMS algorithm due to the slight error introduced by assuming the commutability of the control filter weighting and error path filtering operations. The filtered-x algorithm is relatively insensitive to errors from exchanging the weighting and filtering operations, but the RLS algorithm will fail due to the non-commutability of these operations (Bronzel and Fuller 1995). A remedy is to estimate the error introduced by commuting the two operations at each time step and use that error as a correction in the recursive algorithm (Bronzel and Fuller 1995).

The similarity between RLS and Newton's algorithm means the RLS is useful for accelerating convergence, but is not a systematic method for eliminating ill-conditioning in a control system due to spatial effects. It simply makes the convergence insensitive to that ill-conditioning.

3.5.3 Transform Domain Algorithms

Many different transforms can be applied to a multichannel controller on a reverberant structure. Because the performance of a SISO adaptive filter is limited by the spectral properties of the input signal, transforms to improve the performance of a SISO adaptive filter algorithm are usually based on a time-to-frequency domain transformation followed by power normalization (S. Shankar Narayan 1983; Beaufays 1995; Marshall et al. 1989). A similar approach has been applied to the multichannel LMS algorithm to accelerate convergence (Paillard et al. 1995; Bouchard and Paillard 1996). However, because multichannel LMS is often applied to a reverberant structure, other transforms based on spatial considerations can also be used, such as the modes of the underlying physical structure (Clark 1995; Morgan 1991), or the underlying physical processes such as acoustic radiation from a structure (Snyder and Burgemeister 1996). A slightly different transform approach that does not rely on a fixed transformation is the biological control approach described by Fuller and Carneal (1993).

The discrete cosine transform (DCT) has been applied to both SISO (Paillard et al. 1995) and multichannel control (Bouchard and Paillard 1996) of broadband sound in a duct. The DCT was used to accelerate the convergence of the LMS algorithm, which was slowed by spectral coloration of the reference input signal. The coloration was due to both the spectral properties of the original reference signal and additional coloration introduced by the filtering of the reference by the error path transfer function. The DCT followed by power normalization was used to reduce the effects of the spectral coloration, but the DCT was only used in the weight update portion of the algorithm and not to generate the control filter outputs (Bouchard and Paillard 1996). This eliminated circular convolution problems (Bouchard and Paillard 1996; Haykin 1996; Beaufays 1995) and allowed the update portion of

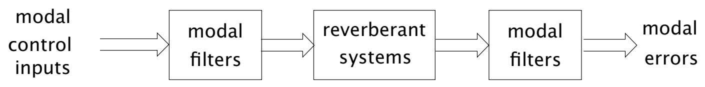

Figure 3.5: Modal space feedforward control (Clark 1995)

the algorithm to proceed on a slower time scale than the control filtering portion. The DCT could not remove correlation at a single cosine “frequency” produced by spatial correlations as a result of actuator placement. To remove the effects of this correlation, Bouchard and Paillard (1996) implemented a Newton algorithm, computing the inverse of the input covariance matrix in the transform domain. They also noted that the DCT approach may not be useful for harmonic control problems involving relatively few tap inputs, since the performance of the DCT improves with increasing tap filter length, and very few tap inputs are needed for harmonic control (Bouchard and Paillard 1996).

Another feedforward control method that can be called a transform domain technique is based on controlling independent modes of the structure (Clark 1995). These are not the same as the modes of the control algorithm discussed in the convergence analysis, but instead refer to the fact that the response of a reverberant structure can be described as a summation of orthogonal modal contributions. The error sensors inputs and control filter outputs are transformed using modal filters so the weight update algorithm operates in terms of decoupled modal coordinates of the structure. This can be used to accelerate the convergence of the algorithm, but the main benefit is a decoupling of the coordinates of the controller.

A schematic of the modal control approach is shown in Fig. 3.5 (Clark 1995). The approach is based on independent modal space control (Meirovitch 1990), and the idea that computation of the optimal control weight vector is considerably simplified if each element of the weight vector is decoupled from every other element. For actuators on a structure, this decoupling can be achieved by applying modal filters to the error sensor outputs that compute the responses of each contributing mode; control can then be applied to individual modes, and the modal control inputs transformed back to physical actuator coordinates via another modal filter. Control can be applied to the modes independent of the characteristics of the reference input signal, as long as it is available as an input to the control algorithm (Clark 1995). Thus the modal control approach works for harmonic and broadband inputs.

The drawback of this approach lies in the difficulty of determining the modal filters at the sensors and actuators. Even if the modal characteristics of the structure were known sufficiently accurately, determination of the modal filters could be very difficult in practice for discrete actuators and sensors (Meirovitch 1990). Any non-ideal characteristics of these modal filters will degrade the performance of the controller (Clark 1995).

A more pragmatic approach to modal control for narrowband excitations is described by Morgan (1991). The idea is similar to that shown in Fig. 3.5. However, the modal filters are not assumed to be ideal, and hence only approximately decouple the controller. For the modal filters at the sensors, an “almost orthonormal” transformation is computed by minimizing the average cross-response between the contributing modes at the error sensors (Morgan 1991). This is a practical way to design the modal filters with discrete sensors and actuators. Once the controller is approximately decoupled, a scaling is applied to the different modes to account for different excitation levels, thus reducing the sensitivity of the convergence behavior to spatial correlation effects. As with the approach described by Clark (1995), the drawback is the requirement to have a modal model of the system being controlled. Similar modal transformations have been used for controlling sound radiation from a panel (Snyder and Burgemeister 1996), although the nature of the problem does not lend itself as well to a broadband modal description (Cunefare 1992; Baumann et al. 1991; Burdisso and Fuller 1994).

The bioligical control approach described by Fuller and Carneal (1993) relies on a transform determined online by optimizing a cost function. The approach is suitable for multichannel harmonic controllers where one actuator can be designated a “master” and the remaining actuators as “slaves.” The control input to the master actuator is optimized using a conventional adaptive technique such as the LMS algorithm. Once the input to the master is optimized, the inputs to the slaves are optimized individually to be either in-phase or 180° out of phase with the master actuator, or completely off, where the appropriate input depends on how the cost function is affected. This approach is not useful for accelerating the convergence of the LMS algorithm since the optimization procedure is considerably more complicated. Nonetheless, the optimization procedure implicitly accounts for the spatial distribution of the actuators, and hence is one approach to account for the effects of this distribution when optimizing the control inputs.

Chapter 4

Principal Component Multichannel Feedforward Control

The purpose of this chapter is to describe the implementation of adaptive feedforward control using the principal components of the control system, and the resulting algorithm is called principal component LMS, or PC-LMS. The algorithm is essentially a transform domain version of the multiple error LMS algorithm, where the transform is signal dependent, and completely orthogonalizes the controller at a single frequency. This means multichannel control can be implemented in terms of independent, non-interacting control coordinates. Slow convergence problems and even a rank deficient control system can be easily addressed in the context of the PC-LMS algorithm. The amount of control effort that is needed to reduce each independent control coordinate can be easily determined, as well as the corresponding reduction in the control cost function that will be obtained. This information is helpful for maximizing the performance of a control system in terms of cost function reduction, while simultaneously minimizing the complexity of the controller.

Principal component control will be discussed in the context of the multichannel feedforward control problem depicted in Fig. 4.1, which was also shown in Chapter 3. The reference input, $x(n)$, is assumed to be deterministic, so the control problem can be analyzed at individual frequencies. Transfer functions between the control actuators and the error sensors at the single frequency of interest are represented by $\mathbf{H}$. It will be shown that the eigenvectors of $(\mathbf{H}^H\mathbf{H})$ and $(\mathbf{H}\mathbf{H}^H)$ can be used to transform the control problem to an orthogonal coordinate system, and control outputs and control filter weight updates can be implemented in this coordinate system. The eigenvector matrices that are used to transform the controller to orthogonal coordinates can be computed from a singular value decomposition (SVD) of the transfer function matrix $\mathbf{H}$, hence much of the discussion in this chapter will concern the SVD of $\mathbf{H}$.

It will be shown that the PC-LMS algorithm is a computationally simple method for optimizing the convergence rate of the control filter coefficients. The transformation to principal components decouples the controller, hence, each decoupled controller coordinate can be assigned an individual adaptation rate. The rates can be scaled so the overall convergence rate of the algorithm is equivalent to that of Newton's algorithm, if fast convergence is important. The sampled data implementation of the PC-LMS algorithm actually requires fewer computations per sample iteration than the multichannel LMS algorithm. PC-LMS is therefore a practical method to accelerate convergence, especially in comparison to the computational complexity of a multichannel recursive least squares algorithm.

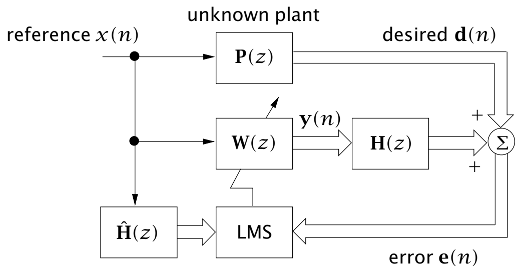

Figure 4.1: Multichannel LMS control with transfer functions in the error path

The PC-LMS approach is similar to the discrete cosine transform method applied to the SISO adaptive filtering problem, which is also related to the self-orthogonalizing adaptive filter algorithm (Beaufays 1995; S. Shankar Narayan 1983; Panda et al. 1986). These SISO methods transformed the adaptive filtering problem to a coordinate system where the controller coordinates were decoupled, or nearly decoupled. The transformed coordinates were then scaled in order to accelerate the convergence of the slow modes of the adaptive algorithm. In contrast to these SISO methods, however, the principal component approach decouples the controller at a single frequency, and therefore decouples spatial correlations between the components of the control system, not frequency components of the reference signal. The eigenvector matrices used in PC-LMS to decouple the control system are properties of the transfer function matrix **H**, and therefore depend on the reference frequency being controlled, and on the plant dynamics. If either of these changes, the eigenvector matrices are likely to change as well, and will have to be recomputed. This limits the application of the PC-LMS algorithm to cases where the reference frequency and the transfer function matrix change slowly. It will be shown that the PC-LMS algorithm has the same robustness to errors in the transfer function matrix as the filtered-x LMS algorithm. This means the on-line methods used to identify the transfer function matrix **H** for the filtered-x LMS algorithm can also be used for the PC-LMS algorithm.

The PC-LMS algorithm also provides a convenient framework for implementing a control effort penalty. Chapter 3 discussed how the amount of control effort required to cancel the ith principal component of a control system was inversely proportional to the square of the ith singular value. Therefore a simple way to implement an effort penalty is to not cancel principal components with small singular values. In PC-LMS, not cancelling a specific PC is as simple as setting the adaptation rate for the corresponding control coordinate to zero. This selective control of the principal components can be used to satisfy other performance criteria, such as cost function reduction, or minimizing controller complexity. Statistical criteria that can be used to select which PCs to control and which PCs to not control are described in the next chapter.

4.1 Frequency domain description of PC-LMS control algorithm

A feedforward control system operating at a single frequency at steady state and with no transients is shown in Fig. 4.1 and is described by the following expression:

$$\mathbf{e} = \mathbf{H}\mathbf{w} + \mathbf{d} \tag{4.1}$$

where the frequency dependence of each variable is implicit. If m and r denote the number of sensors and actuators, respectively, then the $(m \times 1)$ vector **e** contains the responses of the error sensors due to the control and disturbance excitations. The $(m \times r)$ matrix **H** contains the transfer functions between sensors and actuators at the frequency of interest, and the $(r \times 1)$ vector **w** contains the control filter weights. The $(m \times 1)$ vector $\mathbf{d}$ contains the error sensor responses to the primary disturbance. Each element of these matrices and vectors is a complex number. The effect of modeling errors in $\mathbf{H}$ will be considered later, but for now it is assumed to be known exactly.

The SVD of $\mathbf{H}$ produces two matrices that can be used to decouple the control system at the frequency where $\mathbf{H}$ is measured. Using the same notation as in previous chapters, the SVD of $\mathbf{H}$ is written

$$\mathbf{H} = \mathbf{USV}^H \tag{4.2}$$

where $\mathbf{U}$ and $\mathbf{V}$ contain the complex eigenvectors of $\mathbf{HH}^H$ and $\mathbf{H}^H\mathbf{H}$, respectively. From the terminology of the SVD, the matrix $\mathbf{U}$ contains the left singular vectors of $\mathbf{H}$, while $\mathbf{V}$ contains the right singular vectors. Each column of $\mathbf{U}$ is an $(m \times 1)$ vector of complex values that maps a $(m \times 1)$ vector of sensor responses onto the principal components of the control system. For example, the sensor responses due to the primary excitation are given by the $(m \times 1)$ vector $\mathbf{d}$, hence $\mathbf{p} = \mathbf{U}^H\mathbf{d}$ is simply a mapping of the primary response onto the m PCs of the control system. In a similar manner, the columns of $\mathbf{V}$ consist of $(r \times 1)$ vectors that can be used to transform vectors of actuator inputs into vectors of principal component inputs. Since $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices, the inverse transformation, from principal components to either sensors or actuators, can be computed using the complex conjugate transposes, $\mathbf{U}^H$ or $\mathbf{V}^H$. The singular values in the diagonals of $\mathbf{S}$ have a strictly defined ordering, such that $s_1 > s_2 > \ldots > s_r$, where r is the rank of $\mathbf{H}$. This ordering means there is an association between the ith column of $\mathbf{U}$ and the ith column of $\mathbf{V}$. The significance of this association will be made apparent in the following derivations.

Substituting the SVD of $\mathbf{H}$ into the expression for the error sensor responses in Eq. 4.1 yields

$$\mathbf{e} = \mathbf{USV}^H\mathbf{w} + \mathbf{d} \tag{4.3}$$

Premultiplying both sides by $\mathbf{U}^H$ produces

$$\mathbf{U}^H\mathbf{e} = \mathbf{SV}^H\mathbf{w} + \mathbf{U}^H\mathbf{d} \tag{4.4}$$

As in Chapter 3, $\mathbf{p} = \mathbf{U}^H\mathbf{d}$ will be used to denote the mapping of the primary response onto the principal components, and $\boldsymbol{\nu} = \mathbf{V}^H\mathbf{w}$ will be used to denote the filter weights mapped onto the PCs. The symbol $\boldsymbol{\zeta} = \mathbf{U}^H\mathbf{e}$, will be used to denote the total response of the error sensors mapped onto the principal components. With these symbol substitutions, Eq. 4.4 is written

$$\boldsymbol{\zeta} = \mathbf{S}\boldsymbol{\nu} + \mathbf{p} \tag{4.5}$$

This simple equation describes principal component control. It was obtained by transforming the sensor responses and control filter weights into the principal components of the control system using the left and right singular vectors of the transfer function matrix $\mathbf{H}$. These singular vectors rotate the coordinates of the control problem with no loss of information, but the problem is easier to solve in the rotated coordinate system. Equation 4.5 is a frequency domain description of a feedforward control problem, similar to Eq. 4.1. The $(m \times 1)$ vector $\mathbf{p}$ denotes the primary response of the PCs, the $(r \times 1)$ vector $\boldsymbol{\nu}$ contains the control filter weights in terms of the PCs, and the $(m \times r)$ matrix $\mathbf{S}$ contains the transfer functions from PC inputs to PC outputs. The $(m \times 1)$ vector $\boldsymbol{\zeta}$ is the total response, or equivalently the error response, of the principal components due to the combined control and primary excitations.

The transfer function matrix $\mathbf{S}$ is diagonal, meaning the response of the ith PC depends only on the input to the ith PC. Expanding Eq. 4.5 term by term yields

$$\zeta_i = \begin{cases} s_i \nu_i + p_i & \text{for } i = 1, \ldots, r \\ p_i & \text{for } i = r+1, \ldots, m \end{cases} \tag{4.6}$$

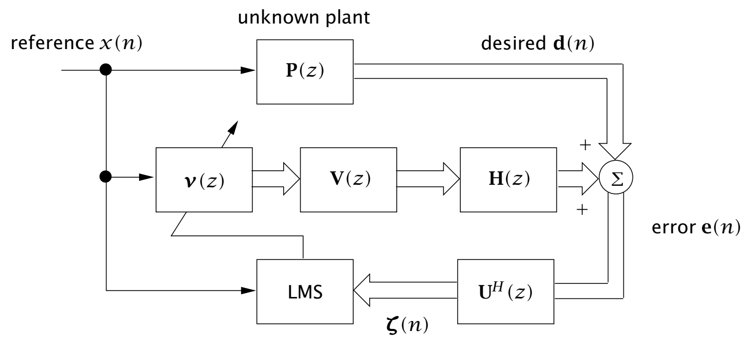

Figure 4.2: Principal component LMS with transfer functions in the error path

The PC filter weights, $\boldsymbol{\nu}$, only appear in the equation for the first r PC error terms, hence the last $(r+1)$ through m elements of $\boldsymbol{\zeta}$ are not controllable. The transformation to principal components also greatly simplifies the update equations for the PC filter weights, $\boldsymbol{\nu}$, which will be discussed momentarily.

Figure 4.2 contains a schematic that illustrates how principal component control is accomplished in a real control system. The reference signal is the input to the control system, as for the multichannel filtered-x LMS controller. However, the control outputs are computed here in terms of the principal components, as represented by the block containing the PC control weights, $\boldsymbol{\nu}(z)$. The control outputs in terms of the PCs then have to be transformed into actuator coordinates using the matrix of right singular vectors. The actuator outputs are then filtered by the error path transfer function, $\mathbf{H}(z)$, and are then summed at the error sensors with the primary response. The vector of sensor responses, $\mathbf{e}(n)$, are then transformed into responses of the principal components, using the matrix of left singular vectors. The PC control weights, $\boldsymbol{\nu}$, are then updated with an adaptive algorithm such as the LMS algorithm. The untransformed control filter weight vector, $\mathbf{w}(z)$, is never actually computed.

The transformations $\mathbf{V}(z)$ and $\mathbf{U}^H(z)$ constitute additional transfer functions between the PC control filter outputs and the PC error sensor inputs. These matrices do not introduce any phase difference between the PC inputs and outputs; instead they make the phase difference zero. However, in a sampled data implementation of the algorithm, the discrete transformations will introduce time delays between the input and output of the PC control system. These delays limit the maximum step size parameter that can be used in the adaptive algorithm for updating the PC control filter weights.

The cost function which determines the optimal principal component weight vector is defined as the norm of the principal component error vector, or $J = \boldsymbol{\zeta}^H\boldsymbol{\zeta}$. This cost function is simply a translation of the norm of the error sensors responses $\mathbf{e}$,

$$J = \mathbf{e}^H\mathbf{e} \tag{4.7}$$

$$= \boldsymbol{\zeta}^H\mathbf{U}^H\mathbf{U}\boldsymbol{\zeta} \tag{4.8}$$

$$= \boldsymbol{\zeta}^H\boldsymbol{\zeta} \tag{4.9}$$

The optimum value of the ith PC weight, or $\nu_{i,opt}$, is obtained by solving for ν_i from the upper half of Eq. 4.6. This equation contains complex values, hence the minimum value of ζ_i occurs when its real and imaginary components are zero. With this condition,

$$\nu_{i,opt} = -\frac{p_i}{s_i} \quad \text{for } i = 1, \dots, r \tag{4.10}$$

and therefore the ith PC control weight is directly proportional to the correlation between the ith PC and the primary response, p_i, and inversely proportional to the ith singular value, s_i. The norm of the total control effort is the summation of the effort norms for the individual PCs given as

$$\boldsymbol{v}_{opt}^H \boldsymbol{v}_{opt} = v_{1,opt}^* v_{1,opt} + v_{2,opt}^* v_{2,opt} + \ldots + v_{r,opt}^* v_{r,opt} \tag{4.11}$$

$$= \frac{|p_1|^2}{s_1^2} + \frac{|p_2|^2}{s_2^2} + \ldots + \frac{|p_r|^2}{s_r^2} \tag{4.12}$$

This expression is identical to one derived in the context of the filtered-x LMS algorithm, in Eq. 3.44, for an effort penalty of zero. An important difference between the two expressions is that here the relationship between the norm of the ith coordinate of the controller and the control effort is a function of s_i and p_i only, and is thus independent of every other controller coordinate. Because the singular values are decreasing, the last few principal components will usually have the greatest contribution to the control effort norm, especially when the control system is slightly ill-conditioned. In contrast, the norm of any particular control filter weight w_i from Eq. 3.44 will in general depend on all r singular values and correlations in $\mathbf{p}$. It is thus more difficult to determine from Eq. 3.44 how a single control filter weight w_i will be affected by ill-conditioning in $(\mathbf{H}^H\mathbf{H})$.

Reformulating the control problem in terms of the principal components also reveals how much each PC reduces the error norm, $\mathbf{e}^H\mathbf{e}$. Equation 4.6 indicates that if the ith principal component is controlled, the contribution of p_i to the error response, ζ_i, is cancelled. The resulting reduction in the norm of the error, expressed in decibels (dB) relative to the error norm with no control is

$$\Delta_{dB}^i = 10 * \log_{10}\left(\frac{|\mathbf{p}|^2 - |p_i|^2}{|\mathbf{p}|^2}\right) \tag{4.13}$$

$$= 10 * \log_{10}\left(1 - \frac{|p_i|^2}{|\mathbf{p}|^2}\right) \tag{4.14}$$

This equation indicates that if the primary response $\mathbf{d}$ is known and its correlation $\mathbf{p}$ with each PC is calculated, the PCs can be ranked according to error norm reduction. The decibel units in Eq. 4.14 can obfuscate the importance of each PC for noise reduction, so an alternative metric is the percent each PC reduces the primary, or

$$\text{PercentReduction} = 100 * \frac{|p_i|^2}{|\mathbf{p}|^2} \tag{4.15}$$

In an actual control system, the reduction of the PC response terms ζ_i will be limited by the numerical precision of the control system and other components such as analog to digital converters.

4.1.1 An adaptive algorithm in PC coordinates

Although the optimal PC control weight, v_i, can be determined from Eq. 4.10, there are many situations where adaptively computing the control weight would be advantageous. An adaptive algorithm can often produce a better estimate of the optimal control weight when noise or errors are present in the elements of Eq. 4.10. Two adaptive algorithms will be given here; the first algorithm is based on Newton's algorithm, and the second is based on the steepest descent algorithm. Both algorithms are derived by transforming the weight update equations in terms of $\mathbf{w}$ to the principal coordinates of the controller.

Assuming a cost function of the form in Eq. 4.7, the weight update equation for Newton's algorithm, in terms of the untransformed weight vector $\mathbf{w}$, is (Nelson and Elliott 1992; Fuller et al. 1996)

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\left(\mathbf{H}^H\mathbf{H}\right)^{-1}\mathbf{H}^H\mathbf{e}(n) \tag{4.16}$$

Substituting the SVD of **H** yields

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu \left(\mathbf{V}\mathbf{S}'\mathbf{U}^H\mathbf{U}\mathbf{S}\mathbf{V}^H\right)^{-1}\mathbf{V}\mathbf{S}'\mathbf{U}^H\mathbf{e}(n) \tag{4.17}$$

$$= \mathbf{w}(n) - \mu\mathbf{V}\left(\mathbf{S}'\mathbf{S}\right)^{-1}\mathbf{S}'\mathbf{U}^H\mathbf{e}(n) \tag{4.18}$$

Premultiplying both sides by $\mathbf{V}^H$, then substituting $\boldsymbol{\nu}$ and $\boldsymbol{\zeta}$ for the transformed error and weight vectors, respectively, produces

$$\boldsymbol{\nu}(n+1) = \boldsymbol{\nu}(n) - \mu\left(\mathbf{S}'\mathbf{S}\right)^{-1}\mathbf{S}'\boldsymbol{\zeta}(n) \tag{4.19}$$

This is the weight update relation for the PC weight vector, based on Newton's algorithm. This equation can be simplified further by expanding the product $(\mathbf{S}'\mathbf{S})^{-1}\mathbf{S}'$ as

$$(\mathbf{S}'\mathbf{S})^{-1}\mathbf{S}' = \left[\begin{array}{cccc|c} 1/s_1 & 0 & \dots & 0 & \mathbf{0} \\ 0 & 1/s_2 & \dots & 0 & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1/s_r & \mathbf{0} \end{array}\right] \tag{4.20}$$

where elements $(r+1)$ through m of each row are zero. Substituting this result into Eq. 4.19, the update recursion for the ith principal component weight, ν_i, is written

$$\nu_i(n+1) = \nu_i(n) - \frac{\mu}{s_i}\zeta_i(n) \tag{4.21}$$

The update for the ith PC weight depends only on the ith PC error term, ζ_i, which in turn depends only on p_i, ν_i, and s_i (see Eq. 4.6). The update for ν_i is therefore independent of all other principal component weights. The weight update for an untransformed weight, w_i, given in Eq. 4.16, depends on a summation of the m error sensor responses, weighted by the transfer function matrix **H**. Because the error sensor responses are functions of every control filter output, the recursions for the untransformed control filter weights depend on the values of every other weight.

The update recursion for the PC control weights in Eq. 4.21 is similar to the update recursions for the self-orthogonalizing adaptive filter described in Chapter 2. In both cases, an orthogonalizing transform is applied to the control inputs, then each orthogonal controller coordinate is scaled by the inverse of the power in the coordinate to normalize the convergence rates of all coordinates. In Equation 4.21, the normalization factor for the PC weight updates is $(1/s_i)$. As a singular value decreases, the normalization factor increases, and the effective step size parameter, μ/s_i, also increases.

A weight update recursion for the PC weight vector can also be derived from the steepest descent algorithm. The steepest descent update recursion in terms of the weight vector **w** is written (Nelson and Elliott 1992)

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\mathbf{H}^H\mathbf{e}(n) \tag{4.22}$$

Substituting the SVD of **H** produces

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\mathbf{V}\mathbf{S}'\mathbf{U}^H\mathbf{e}(n) \tag{4.23}$$

$$\mathbf{V}^H\mathbf{w}(n+1) = \mathbf{V}^H\mathbf{w}(n) - \mu\mathbf{S}'\mathbf{U}^H\mathbf{e}(n) \tag{4.24}$$

$$\boldsymbol{\nu}(n+1) = \boldsymbol{\nu}(n) - \mu\mathbf{S}'\boldsymbol{\zeta}(n) \tag{4.25}$$

This equation can be expanded as a recursion for individual principal components as

$$\nu_i(n+1) = \nu_i(n) - \mu s_i\zeta_i(n) \tag{4.26}$$

The only difference between this update recursion and the recursion derived from Newton's algorithm in Eq. 4.21 is the effective step size parameter, μs_i. Instead of scaling the update for each controller coordinate by the inverse of its singular value, it is scaled by the singular value itself. As the singular values gets smaller, the product μs_i also gets smaller, which explains why the last few principal components with small singular values can take such a long time to converge when the steepest descent algorithm is used.

The similarity between the two recursion relations in Eq. 4.21 and Eq. 4.26 suggests the following general update recursion:

$$\boldsymbol{\nu}(n+1) = \boldsymbol{\nu}(n) - \boldsymbol{\alpha\zeta}(n) \tag{4.27}$$

where $\boldsymbol{\alpha}$ is an $(r \times 1)$ vector of step size parameters that can be varied for each principal component weight. Stability bounds on the α_i can be derived by considering the convergence behavior of the weights in Eq. 4.27. Substituting $\zeta_i = s_i \nu_i + p_i$ into the equation, and using $(p_i = -\nu_{i,opt} s_i)$, the update recursion for a single PC weight is written

$$\nu_i(n) = \nu_i(n-1) - \alpha_i \left(s_i \nu_i(n-1) - \nu_{i,opt} s_i \right) \tag{4.28}$$

Subtracting $\nu_{i,opt}$ from both sides and rearranging produces

$$\nu_i(n) - \nu_{i,opt} = \nu_i(n-1) - \nu_{i,opt} - \alpha_i s_i \left(\nu_i(n-1) - \nu_{i,opt} \right) \tag{4.29}$$

This can be further simplified by introducing the weight error, $\epsilon_i = \nu_i(n) - \nu_{i,opt}$. This substitution yields

$$\epsilon_i(n) = (1 - \alpha_i s_i)\epsilon_i(n-1) \tag{4.30}$$

which can be rewritten as a function of the initial weight error, $\epsilon_i(0)$, as

$$\epsilon_i(n) = (1 - \alpha_i s_i)^n \epsilon_i(0) \tag{4.31}$$

The value ϵ_i will go to zero as n gets large if the constraint $|1 - \alpha_i s_i| < 1$ is satisfied. This means α_i must be bounded as

$$0 < \alpha_i < \frac{2}{s_i} \tag{4.32}$$

If the step size parameters are defined to be proportional to the inverse of the singular values the weights will converge at a rate equivalent to Newton's algorithm. If the α_i are directly proportional to the singular values, s_i, the weight, $\boldsymbol{\nu}$, will converge at a rate equal to the steepest descent algorithm. If all of the α_i are set to the same value, the convergence rate will lie between that of Newton's algorithm and the steepest descent algorithm. The actual bounds on the step size parameters will be slightly lower than those given in Eq. 4.32 due to physical delays in the control system.

4.1.2 Controlling a subset of principal components

An important result of implementing control in terms of the principal components is the ability to specify individual step size parameters for each PC. This feature can be used, for example, to prevent a given PC from being controlled, simply by setting the corresponding step size parameter, α_i, to zero. This raises the possibility of controlling a subset of the PCs, where convenient selection criteria are used to select members of the subset. Two candidate criteria that immediately come to mind are selecting PCs to minimize the norm of the control effort, and selecting PCs to maximize reduction of the error term. A more formal discussion of selection criteria based on statistical considerations will be given in the next chapter.

When the step size parameter for the ith PC is set to zero, there is no need to transform the PC control output, v_i, to the actuator inputs, and likewise, there is no reason to compute the corresponding error term, ζ_i. As a result, the corresponding columns of the transformation matrices **U** and **V** can be eliminated, which reduces the number of required computations for each iteration of the control algorithm. If c denotes the number of PCs with non-zero step size parameters, where $c \leq r$, **V** can be reduced to an $(r \times c)$ matrix, and **U** to an $(m \times c)$ matrix.

Selectively controlling a subset of the PCs can be used to limit the total control effort of the control system. Equation 4.11 shows that the effort for a PC is inversely proportional to the square of the singular value for the PC, so a simple effort penalty is to not control PCs with very small singular values. This approach is especially useful when the control system is ill-conditioned, and the last few singular values are very small. The benefit of this approach is that the control weights for the PCs that are controlled are allowed to converge to their optimum values. In contrast, a uniform effort penalty in the filtered-x LMS algorithm will affect all PCs of the control system (Rossetti et al. 1996; Elliott et al. 1992). This was demonstrated in Fig. 3.3, in which the uniform effort penalty prevented both controller coordinates from reaching their respective minimum values. An additional benefit of implementing an effort penalty by eliminating PCs from the controller is the reduced computational burden, once the PCs are removed. In contrast, implementing an effort with the filtered-x LMS algorithm will increase the number of computations at each iteration.

The more sophisticated threshold effort penalty (Rossetti et al. 1996), discussed in Chapter 3, can also be easily implemented in terms of the PC-LMS algorithm. The threshold effort penalty is a non-uniform effort penalty that affects only ill-conditioned principal components of the control system (see Eq. 3.47). The same effect can be achieved in PC coordinates, assuming the first $\rho \leq r$ PCs are well-conditioned, and the last $\rho + 1 - r$ PCs are ill-conditioned, as follows:

$$v_i(n+1) = \begin{cases} v_i(n) - \alpha_i \zeta_i(n) & \text{for } i = 1, \ldots, \rho \\ (1 - \alpha_i \beta) v_i(n) - \alpha_i \zeta_i(n) & \text{for } i = \rho + 1, \ldots, r \end{cases} \tag{4.33}$$

Thus the effort penalty only affects elements $(\rho + 1) - r$ of the control system.

The selection of which PCs to control could also be done on the basis of reduction in the error norm. Each PC reduces the error norm in direct proportion to $|p_i|^2$, as shown in Equation 4.11. A criteria based on error reduction would require keeping all PCs with sufficiently large values of p_i.

Regardless of the criteria used to select PCs, controlling a subset of the r PCs will introduce a bias into the solution relative to the case where all PCs are controlled. An example of this bias is shown in Fig. 4.3. This figure shows the convergence behavior of a two-element tap weight vector when only the first principal component is controlled. The error surface is identical to the surface shown in previous chapters (see Figs. 2.2, 3.3 and 3.4). The weight vector was initialized at the right side of the plot where $w_1 = w_2 = 0$, and only the first principal component was adapted (i.e. the step size parameter for the second PC, $\alpha_2 = 0$). The weight vector converged along a path parallel to the first principal axis, ξ_1, and minimized the cost function in this direction but did not move in the direction of the second principal axis, ξ_2. The final solution of the weight vector is clearly different from the weight vector that minimizes the error surface, at the center of the plot. In some applications it is preferable to sacrifice error reduction in return for reduced control effort or reduced controller complexity.

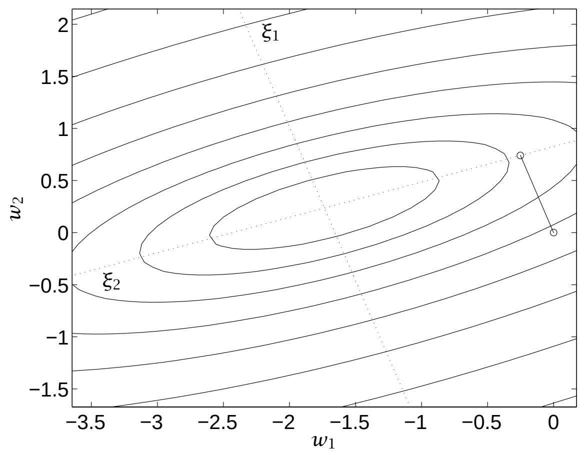

Figure 4.3: Convergence of first principal component only

4.2 Time domain weight update: PC-LMS

A sampled data version of the frequency domain principal component LMS algorithm is given here. The reference signal consists of a single frequency, so a sampled data version of the algorithm is obtained by expressing each complex frequency domain value as a two-coefficient FIR filter. A two-coefficient FIR filter is sufficient to produce any amplitude and phase shift indicated by a complex, frequency domain variable. While this is a simple approach for producing a time-domain version of the algorithm, the resulting formulation cannot be used to analyze the effect of delays in the control system (Nelson and Elliott 1992).

The assumptions used to derive the time domain equations here are identical to those for the frequency domain algorithm: the input time series are assumed to be stationary, there are no transients, and the reference signal is deterministic. As discussed in the previous section, the number of PCs included in the weight update equations can vary from 1 to r, so the variable c is used here to denote the number of PCs being controlled. For convenience, the symbols that have been used in previous sections to designate frequency domain variables are used here to denote their time domain counterparts. The context of the discussion should make it clear whether a particular variable refers to a frequency domain or time domain quantity.

The output of the ith principal component control filter is computed by filtering the reference input signal, x, with the ith PC weight value, v_i. The output of the ith PC at time n, denoted $y_i^{pc}(n)$, is given by

$$y_i^{pc}(n) = v_{i0}(n)\; x(n) + v_{i1}(n)\; x(n-1) \tag{4.34}$$

where (v_{i0}, v_{i1}) are the FIR filter coefficients that produce the amplitude and phase shift described by the frequency domain weight, v_i, at the frequency ω. The total number of multiplications required for this step of the algorithm is 2 times the number of PCs being controlled, or $(c * 2)$.

The outputs of the r control actuators are computed by multiplying the PC outputs by the transformation matrix $\mathbf{V}$. The time domain representation of the $(i,j)th$ element of the frequency domain transformation matrix $\mathbf{V}$ will be denoted here by the two FIR filter weights, (v_{ij0}, v_{ij1}), where the first subscript corresponds to the actuator output, the second subscript to the principal component, and the third to the time index. The output of the ith actuator is therefore

$$y_i(n) = \sum_{j=1}^{c} \left(v_{ij0}\, y_j^{pc}(n) + v_{ij1}\, y_j^{pc}(n-1) \right) \tag{4.35}$$

The transformation from PC coordinates to physical actuator outputs requires $(r * c * 2)$ multiplications.

The actuator outputs, y_i, are filtered by the error path transfer functions, which introduce a delay between the actuator outputs and the error sensor inputs. The outputs of the error sensors are then transformed to principal coordinates by the transformation matrix $\mathbf{U}^H$. The time domain representation of the (i, j)th element of the frequency domain matrix $\mathbf{U}$ will be denoted (u_{ij0}, u_{ij1}), where the first subscript is the error sensor, the second is the principal component, and the third is the time index. The mapping of the m error sensors onto the ith principal component is given by

$$\zeta_i(n) = \sum_{j=1}^{m} \left(u_{ji0} e_j(n) + u_{ji1} e_j(n-1) \right) \tag{4.36}$$

This coordinate transformation requires $(c * m * 2)$ multiplications.

The time domain update equation for the principal component filter weights, (v_{i0}, v_{i1}) is written

$$v_{i0}(n+1) = v_{i0}(n) - \alpha_i \zeta_i(n) x(n) \tag{4.37}$$

$$v_{i1}(n+1) = v_{i1}(n) - \alpha_i \zeta_i(n) x(n-1) \tag{4.38}$$

where α_i is the step size parameter for the ith PC. Updating the coefficients for the c PCs requires $(c * 2)$ multiplications.

A comparison of the multiplications required for each iteration of the PC-LMS and multichannel filtered-x LMS algorithms is shown in Table 4.1. Although the PC-LMS algorithm may appear to require more computations than the filtered-x LMS algorithm, due to the transformations to and from principal coordinates, in fact PC-LMS requires fewer computations per iteration. Assuming $c = r$, and subtracting the total computations for the two algorithms reveals that filtered-x LMS requires $2r(m - r + 1)$ more computations per iteration than PC-LMS. The difference in computational requirements is most pronounced when there are many more error sensors than actuators, i.e. $m >> r$. The difference between the computational requirements of the two algorithms also increases if an effort penalty is implemented. In PC-LMS, an effort penalty can be implemented by eliminating the last few PCs from the controller. This will reduce the number of PCs that must be transformed by the matrices **V** and **U**, and hence will reduce the number of computations required per iteration. In contrast, if a uniform effort penalty is implemented in the filtered-x LMS algorithm, an additional r computations will be required per iteration in order to multiply each weight by the factor $\mu\beta$.

The time domain equations given here provide some insight into the time delays that are introduced by the transformation matrices **V** and **U**. These delays are represented by Eqs. 4.35 and 4.36, which are, respectively, transformations from PCs to actuators, and from error sensors to PCs. In order for the weight update equations (Eqs. 4.37 and 4.38) to be valid, the PC weights must be stationary, or approximately so, from computation of the control inputs in Eq. 4.36 to computation of the error terms in Eq. 4.38. This includes physical delays due to the error path transfer functions. This requirement must be considered when the step size parameter α_i is computed for each PC. The experimental results discussed later confirm the necessity of reducing the step size parameter below what might be expected according to the frequency domain analysis, in order to maintain stable convergence.

Table 4.1: Comparison of computations for PC and filtered-x algorithms

Operation	computations
compute PC output	2c
compute actuator output	2cr
transform error inputs	2cm
update PC weights	2c
Total	2c(m+r+2)

(a) PC-LMS

Operation	computations
compute actuator output	2r
compute filtered-x	2rm
update weight	2rm
Total	2r(2m+1)

(b) Multichannel filtered-x

4.3 Robustness to errors in transfer function model

The robustness of the PC-LMS algorithm to errors in the estimate of **H** is considered here. The transfer function matrix **H** is usually not known in a practical implementation of the PC-LMS algorithm, so it must be estimated. The estimate will be denoted as $\hat{\mathbf{H}}$. The transformation matrices for principal component LMS are computed from the SVD of $\hat{\mathbf{H}}$, so it will be important to determine the sensitivity of the PC-LMS algorithm to modeling errors in $\hat{\mathbf{H}}$. A similar robustness analysis of the convergence behavior of the filtered-x LMS algorithm, in Chapter 3, established that the algorithm would converge if the real parts of the eigenvalues of the matrix $(\hat{\mathbf{H}}^H\mathbf{H})$ were positive. The analysis here will demonstrate that PC-LMS has precisely the same robustness to modeling errors as filtered-x LMS. The effect of modeling errors on the residual error of the controller will also be discussed.

From the analysis in Chapter 3 (see Eq. 3.21), a frequency domain expression for the convergence of **w**, when errors are present in the transfer function model is written

$$\mathbf{w}(n) - \mathbf{w}_\infty = \left(\mathbf{I} - \mu\hat{\mathbf{H}}^H\mathbf{H}\right)(\mathbf{w}(n-1) - \mathbf{w}_\infty) \tag{4.39}$$

where $\mathbf{w}_\infty$ is the weight vector after an infinite number of iterations, assuming stable convergence. The weight vector does not converge to the optimum, $\mathbf{w}_{opt}$, due to errors in the transfer function estimate $\hat{\mathbf{H}}$. This equation describes the convergence of the difference between the actual weight vector, $\mathbf{w}(n)$, and the final weight vector. The difference converges to zero (assuming $\mu > 0$) if the real part of every eigenvalue of $(\hat{\mathbf{H}}^H\mathbf{H})$ is greater than zero (Morgan 1980; Boucher et al. 1991).

The convergence of the principal component weight vector, $\boldsymbol{\nu}$, can be derived from Eq. 4.39. We will assume the SVD of $\hat{\mathbf{H}}$ is written as

$$\hat{\mathbf{H}} = \mathbf{U}\mathbf{S}\mathbf{V}^H \tag{4.40}$$

Because the matrices **U** and **V** are unitary, the product of either one with a third matrix $(\mathbf{U}^H\mathbf{A})$ will not change the eigenvalues of **A**. Another useful property of **U** and **V** is they span the column and row spaces, respectively, of any $(m \times r)$ matrix. Using this property, the true transfer function matrix can be written as a product of the singular vectors of the estimated transfer function matrix as

$$\mathbf{H} = \mathbf{U}\mathbf{K}\mathbf{V}^H \tag{4.41}$$

In general, the matrix **K** is fully populated, and is not diagonal like the singular value matrix, **S**.

Substituting Eqs. 4.40 and 4.41 into Eq. 4.39 yields

$$\mathbf{w}(n) - \mathbf{w}_\infty = \left(\mathbf{I} - \mu\mathbf{V}\mathbf{S}'\mathbf{U}^H\mathbf{U}\mathbf{K}\mathbf{V}^H\right)(\mathbf{w}(n-1) - \mathbf{w}_\infty) \tag{4.42}$$

$$\mathbf{w}(n) - \mathbf{w}_\infty = \mathbf{V}\left(\mathbf{I} - \mu\mathbf{S}'\mathbf{K}\right)\mathbf{V}^H(\mathbf{w}(n-1) - \mathbf{w}_\infty) \tag{4.43}$$

Using the definition $\boldsymbol{\nu} = \mathbf{V}^H\mathbf{w}$, this equation is rewritten as

$$\boldsymbol{\nu}(n) - \boldsymbol{\nu}_\infty = (\mathbf{I} - \mu\mathbf{S}'\mathbf{K})(\boldsymbol{\nu}(n-1) - \boldsymbol{\nu}_\infty) \tag{4.44}$$

which describes the convergence behavior of the principal component weight error vector, $(\boldsymbol{\nu}(n) - \boldsymbol{\nu}_\infty)$. Using the same convergence criteria as for Eq. 4.39, the weight error vector will converge to zero if the real parts of the eigenvalues of $(\mathbf{S}'\mathbf{K})$ are positive. However, Eq. 4.44 was obtained by factoring the unitary matrices $\mathbf{U}$ and $\mathbf{V}$ out of Eq. 4.39, and since the product of a unitary matrix with another matrix does not affect the eigenvalues of the matrix, the eigenvalues of $(\mathbf{S}'\mathbf{K})$ and $(\hat{\mathbf{H}}^H\mathbf{H})$ are equal. Therefore the PC-LMS algorithm has precisely the same robustness to modeling errors as the multichannel filtered-x LMS algorithm.

The preceding robustness analysis assumed the full set of PCs were included in the controller, so here we consider the robustness when only a subset of the PCs are controlled. For this analysis, it is helpful to express $\mathbf{K}$, from Eq. 4.41, as a perturbation of the singular value matrix of $\hat{\mathbf{H}}$,

$$\mathbf{K} = \mathbf{S} + \Delta\mathbf{S} \tag{4.45}$$

where $\Delta\mathbf{S}$ is an arbitrary matrix, in general. Substituting this expression for $\mathbf{K}$ into Eq. 4.44 produces

$$\boldsymbol{\nu}(n) - \boldsymbol{\nu}_\infty = (\mathbf{I} - \mu\mathbf{S}'(\mathbf{S} + \Delta\mathbf{S}))(\boldsymbol{\nu}(n-1) - \boldsymbol{\nu}_\infty) \tag{4.46}$$

$$= (\mathbf{I} - \mu(\mathbf{S}'\mathbf{S} + \mathbf{S}'\Delta\mathbf{S}))(\boldsymbol{\nu}(n-1) - \boldsymbol{\nu}_\infty) \tag{4.47}$$

A practical approach to this problem is to assume only the diagonal terms of $(\mathbf{S}'\mathbf{S} + \mathbf{S}'\Delta\mathbf{S})$ affect the stability of convergence, and not the off-diagonal elements. This assumption was shown by Omoto and Elliott (1997) to provide a reasonable guide for the stability of convergence. Assuming s_i and Δs_{ii} are the diagonal elements of $\mathbf{S}$ and $\Delta\mathbf{S}$, respectively, the criteria for stable convergence is (Omoto and Elliott 1997)

$$\mathrm{Re}(s_i^2 + s_i\Delta s_{ii}) > 0 \tag{4.48}$$

$$s_i > -\mathrm{Re}(\Delta s_{ii}) \tag{4.49}$$

The stability of convergence of the ith PC therefore depends only on the values in the ith diagonal of $\mathbf{S}$ and $\Delta\mathbf{S}$. This means the stability of the PC-LMS algorithm depends only on those PCs that are included in the weight update computations. Therefore, assuming convergence stability is determined only by the diagonal elements of $(\mathbf{S}'\mathbf{S} + \mathbf{S}'\Delta\mathbf{S})$, PCs not included in the weight updates will not cause convergence to go unstable. If the constraint in Eq. 4.49 is satisfied by every singular value except the jth singular value, the convergence of $\boldsymbol{\nu}$ will be stable as long as the step size parameter, α_j, is zero.

This discussion illustrates how the PC-LMS algorithm provides much greater flexibility for stabilizing the convergence of a multichannel controller than the filtered-x LMS algorithm. In filtered-x LMS, the effort penalty can be used to stabilize convergence by setting the penalty parameter β larger than the negative real parts of the eigenvalues of $(\hat{\mathbf{H}}^H\mathbf{H})$ (Boucher et al. 1991; Omoto and Elliott 1997). However, if the real part of one of the first few principal components were negative, the effort penalty would have to be set to a relatively large number to offset the negative eigenvalue. The large effort penalty would severely limit the control effort of all other principal components, and the reduction in the error norm would likely be very small. In contrast, if control were implemented using PC-LMS, the step size parameter for the unstable principal component could be set to zero, and the remaining principal components allowed to converge to their respective optimum values.

Errors in the estimated transfer function matrix, $\hat{\mathbf{H}}$, also affect the amount by which the cost function can be reduced. The error response after the controller has converged, written in terms of the untransformed controller coordinates is

$$\mathbf{e}_\infty = \left[\mathbf{I} - \mathbf{H}\left[\hat{\mathbf{H}}^H\mathbf{H}\right]^{-1}\hat{\mathbf{H}}^H\right]\mathbf{d} \tag{4.50}$$

Substituting the SVD of $\hat{\mathbf{H}}$, and Eq. 4.41 for $\mathbf{H}$, yields

$$\mathbf{e}_\infty = \left[\mathbf{I} - \mathbf{UKV}^H \left[\mathbf{VS'U}^H\mathbf{UKV}^H\right]^{-1} \mathbf{VS'U}^H\right] \mathbf{d} \tag{4.51}$$

$$\mathbf{e}_\infty = \mathbf{U}\left[\mathbf{I} - \mathbf{K}\,[\mathbf{S'K}]^{-1}\,\mathbf{S'}\right]\mathbf{U}^H\mathbf{d} \tag{4.52}$$

$$\boldsymbol{\zeta}_\infty = \left[\mathbf{I} - \mathbf{K}\,[\mathbf{S'K}]^{-1}\,\mathbf{S'}\right]\mathbf{p} \tag{4.53}$$

Rewriting $\mathbf{K}$ as a perturbation of $\mathbf{S}$, as in Eq. 4.45, produces

$$\boldsymbol{\zeta}_\infty = \left[\mathbf{I} - (\mathbf{S} + \Delta\mathbf{S})\,[\mathbf{S'}(\mathbf{S} + \Delta\mathbf{S})]^{-1}\,\mathbf{S'}\right]\mathbf{p} \tag{4.54}$$

$$= [\mathbf{I} - \mathbf{SS}^+]\,[\mathbf{I} + \Delta\mathbf{SS}^+]^{-1}\,\mathbf{p} \tag{4.55}$$

where $\mathbf{S}^+$ is the pseudo-inverse of $\mathbf{S}$. For small perturbations $\Delta\mathbf{S}$, this can be further simplified as (Omoto and Elliott 1997)

$$\boldsymbol{\zeta}_\infty = [\mathbf{I} - \mathbf{SS}^+]\,[\mathbf{I} - \Delta\mathbf{SS}^+]\,\mathbf{p} \tag{4.56}$$

Expanding the individual terms of this matrix product, the elements of $\boldsymbol{\zeta}$ are given by (Omoto and Elliott 1997)

$$\zeta_{i,\infty} = \begin{cases} 0 & i = 1, \ldots, r \\ p_i + \sum\limits_{j=1}^{r} \dfrac{-\Delta s_{ij}}{s_j} p_j & i = r+1, \ldots, m \end{cases} \tag{4.57}$$

Assuming all r PCs are included in the weight update equations, the first r elements of $\boldsymbol{\zeta}$ will be cancelled by the control system. Each remaining $(r+1)$ through m element of $\boldsymbol{\zeta}$ will contain residual error in addition to p_i, as indicated by the summation term in Eq. 4.57. The additional residual error is due to errors in the plant model $\hat{\mathbf{H}}$.

If the ith PC is not included in the weight updates and is therefore left uncontrolled, the corresponding error element ζ_i will not be zero as in the top half of Eq. 4.57. Instead, the ith error term will be

$$\zeta_i = p_i + \sum_{j=1}^{r} \frac{-\Delta s_{ij}}{s_j} p_j \tag{4.58}$$

as in the bottom half of Eq. 4.57. Therefore, if a PC is eliminated from the controller when the transfer function model contains errors, the residual error will consist not only of the primary response p_i corresponding to the uncontrolled PC, but will also contain contributions from all of the first r PCs in the proportions indicated by Eq. 4.58. When the errors in the transfer function matrix are small such that Δs_{ij} is small, the additional residual errors in the summation term of Eq. 4.58 will also be small. In this case the benefit of eliminating a PC, such as limiting control effort or stabilizing convergence, may be worth a small increase in residual error.

4.4 Principal Component Analysis for Controller Design

This thesis emphasizes how the principal components of a control system can be used to implement feedforward control, but the principal components can also be used in the control design process to provide insight into efficient ways to drive the control actuators. Previous work has looked into how actuators can be grouped to reduce the controller degrees of freedom and reduce control spillover caused by individual actuators (Cabell et al. 1993; Cabell et al. 1995; Fuller and Carneal 1993). In this context, the right singular vectors in the matrix **V** can be interpreted as specifying actuator groupings. In certain applications, the groupings specified in **V** and the corresponding sensor responses in **U** can provide considerable insight into the mechanics of the control problem.

The application of principal component analysis to control design is described briefly in a numerical study in Appendix C. A multichannel control system was used to actively control sound radiated from the inlet of a turbofan engine. The control sources were acoustic sources mounted on the inside circumference of the inlet to the turbofan engine, and error microphones were located on a hemispherical surface outside the inlet. The symmetry of the problem, and the number of control sources needed to control the very high level of noise radiated from the inlet combine to cause control system conditioning problems.

The results of the analytical study in Appendix C illustrate how the principal components could be easily related to propagating modes in the duct. The first few principal components described actuator groupings that suggested how to wire the actuators together to maximize noise reduction while minimizing conditioning and complexity problems of the control system.

Chapter 5

Statistical Properties of Principal Components

The purpose of this chapter is to restate the multichannel feedforward control problem in terms of a multiple linear regression problem (Ruckman and Fuller 1995; Snyder and Hansen 1991) in order to apply formal variable selection techniques from linear regression to decide which principal components to control. The previous chapter illustrated how the PC-LMS algorithm could be used to control subsets of the principal components of the controller. This is useful for reducing the computational burden of the algorithm, so control can be applied with a very large control system containing many sensors and actuators. In addition, controlling a subset of the PCs is useful for limiting control effort or maximizing reduction of the error term in the cost function. The variable selection techniques described here have been used before to select a subset of actuator locations from a larger candidate set of actuator locations (Ruckman and Fuller 1993; Snyder and Hansen 1991). However, here we are concerned with selecting a subset of principal components to control from all of the principal components. In contrast to the actuator selection problem, the end result of selecting a subset of PCs will not be fewer actuators in the control system, but instead will be fewer degrees of freedom in the controller. The selection techniques are thus useful for maximizing the performance of the control system while limiting the complexity of the controller.

The chapter begins by describing feedforward control as a linear regression problem. The correspondence between the random error term in the standard linear regression model and random noise in the sensor measurements in active control is noted. This random noise term means the variance of the control filter weights can be computed. Accurate computation of control weights with high variances can be problematic when random noise is present in the sensors. This observation leads to one PC selection metric: choose principal components whose regression coefficients have small variances. Selection based on variance minimization will be shown to be equivalent to selection based on minimizing control effort. A second selection metric is based on the statistical confidence interval about a computed regression coefficient. The resulting selection metric is based on the correlation between each PC and the primary disturbance; if the correlation is less than a given multiple of the signal to noise ratio in the control system, the PC is eliminated from the controller.

It is usually impossible to find a single set of PCs that simultaneously minimizes variance and maximizes correlation (Jolliffe 1986), hence a simple procedure for combining the two criteria is discussed at the end of the chapter.

5.1 Linear Regression Problem

There are many valuable tools from regression analysis that can be applied to the feedforward active control, once the control problem is restated as a linear regression problem. The similarity between feedforward control and linear regression has been noted elsewhere (Snyder and Hansen 1991; Ruckman and Fuller 1995). The terminology and assumptions of regression analysis are discussed briefly here, but the subject of regression deserves a more thorough treatment. The reader is encouraged to see the references (e.g., see Draper and Smith (1981), Walpole and Myers (1985)) for more information on the capabilities and subtleties of linear regression.

In its simplest form, linear regression is the determination of parameters of a linear model that relate input variables to an output variable. The input variables are called predictors, and the output variable is called a dependent variable, since it depends on the predictors. A regression model includes a residual error term to denote aspects of the relationship between the predictors and dependent variable that is not linear, such as random or non-linear variation. As described in Ruckman and Fuller (1995), in the context of feedforward control the dependent variables are the negative of the sensor responses to the primary disturbance. This will be denoted as $(-\mathbf{d})$, using the same notation as in previous chapters. The feedforward control problem, written as a linear regression problem, is

$$-\mathbf{d} = \mathbf{H}\mathbf{w}_{opt} + \boldsymbol{\epsilon} \tag{5.1}$$

where $\mathbf{H}$ is the transfer function matrix between actuators and sensors at a single frequency, $\mathbf{w}_{opt}$ is a vector of optimal control inputs, and $\boldsymbol{\epsilon}$ is a vector of random error terms. For the control problem, the random error term represents random noise present in the sensor measurements, and thus provides a means to evaluate the sensitivity of the control solution to random sensor noise. This equation will be discussed here in the context of both the linear regression problem and the feedforward active control problem, hence the terms "regression coefficients" and "control filter weights" will be used interchangeably to refer to $\mathbf{w}$, and "primary response" and "dependent variable" will both refer to $\mathbf{d}$.

Some important properties of this regression model for feedforward control are noted below:

1. the model represents complex linear regression, which requires a simple extension of concepts from real-valued regression (Ruckman and Fuller 1995; Miller 1973).

2. the elements of $\mathbf{H}$ are assumed to be measured without error, which implies the transfer function matrix is known exactly. Standard linear regression does not consider the effect of errors in the matrix $\mathbf{H}$, although more sophisticated regression treatments do have this capability (Draper and Smith 1981). This assumption presents a dilemma for the active control problem, because the same sensors used to measure $\mathbf{d}$ and which are assumed to contain random noise, are also used to measure the transfer function matrix. The impact of this contradiction can be minimized if the variance of the elements in $\mathbf{H}$ is much greater than the variance of the random noise in the sensors (Draper and Smith 1981). In practical terms, the elements of $\mathbf{H}$ should cover as wide a dynamic range as possible in order to minimize the effect of random measurement errors.

3. the regression coefficient vector $\mathbf{w}_{opt}$ is unknown and can only be estimated from sampled data. One purpose of the model in Eq. 5.1 is to understand how random noise in the dependent variable $\mathbf{d}$ affects the accuracy of these estimates.

4. the residual errors, $\boldsymbol{\epsilon}$, are assumed to be randomly distributed with a mean of $E[\boldsymbol{\epsilon}] = \mathbf{0}$ and a covariance matrix of $V(|\boldsymbol{\epsilon}|) = \sigma^2\mathbf{I}$, where σ^2 is a real scalar. The error terms are therefore independent from sensor to sensor, and their magnitudes have equal variance, σ^2. As in Ruckman and Fuller (1995), only the variance of the magnitudes of the $\boldsymbol{\epsilon}$ terms is characterized, not

the variance of the phase. This does not imply phase invariance, but for computing confidence intervals on the regression coefficients, characterizing the magnitude is sufficient.

A least squares estimate for the optimal coefficient vector $\mathbf{w}_{opt}$ that minimizes $(\boldsymbol{\epsilon}^H\boldsymbol{\epsilon})$ is

$$\left(\mathbf{H}^H\mathbf{H}\right)\mathbf{w} = -\mathbf{H}^H\mathbf{d} \tag{5.2}$$

$$\mathbf{w} = -\left(\mathbf{H}^H\mathbf{H}\right)^{-1}\mathbf{H}^H\mathbf{d} \tag{5.3}$$

This result is identical to the solution for the optimal weight vector for the multichannel controller given in Eq. 3.5. The dependent variable, **d**, is assumed to contain a random component that changes each time **d** is measured. Because **d** changes, the weight vector **w** computed from Eq. 5.3 will also change, and the amount of change in **w** for a given change in **d** is the sensitivity of **w** to random noise in **d**. The next section discusses the value of choosing predictor variables that minimize the sensitivity of the coefficients to random noise in the dependent variable.

Just as the feedforward control problem was reformulated in terms of principal components, the regression problem can also be expressed in terms of the principal components (Jolliffe 1986). Substituting the singular value decomposition of **H** (i.e., $\mathbf{H} = \mathbf{U}\mathbf{S}\mathbf{V}^H$) into the normal equation in Eq. 5.3 yields

$$\mathbf{w} = -\left(\mathbf{V}\mathbf{S}'\mathbf{U}^H\mathbf{U}\mathbf{S}\mathbf{V}^H\right)^{-1}\mathbf{V}\mathbf{S}'\mathbf{U}^H\mathbf{d} \tag{5.4}$$

$$\mathbf{w} = -\mathbf{V}\left(\mathbf{S}'\mathbf{S}\right)^{-1}\mathbf{S}'\mathbf{U}^H\mathbf{d} \tag{5.5}$$

$$\boldsymbol{\nu} = -\left(\mathbf{S}'\mathbf{S}\right)^{-1}\mathbf{S}'\mathbf{p} \tag{5.6}$$

The principal component weight vector is defined as $\boldsymbol{\nu} = \mathbf{V}^H\mathbf{w}$, and the mapping of the desired response onto the principal components is defined as $\mathbf{p} = \mathbf{U}^H\mathbf{d}$. The regression problem has thus been restated in terms of the principal components of **H** such that the predictor variables correspond to the principal components, and the desired response corresponds to **p**. The vector $\boldsymbol{\nu}$ denotes the regression coefficients.

5.2 Variance of regression coefficients

The variances of the regression coefficients can be used to decide which PCs should be included in the controller and which PCs should not be controlled. The variance and control effort of a principal component are closely related, hence selecting PCs on the basis of variance minimization is nearly equivalent to control effort minimization. This discussion provides additional motivation for removing certain principal components from the control system.

The variance of a regression coefficient quantifies how much the coefficient can be expected to vary due to random noise in the dependent variable **d**. The variance of a control filter weight does not necessarily make sense in the context of the usual multichannel control problem because the formulation of the problem does not account for random noise in the error sensors. The variance of the coefficient vector could actually be quantified if the dependent variable were measured many times, and for each measurement a corresponding **w** (or $\boldsymbol{\nu}$) was computed. The collection of computed **w**'s would then have a mean and a variance. In the ideal case, the coefficient variances would be low, signifying an insensitivity to noise in the dependent variable. If the variances are high, however, this indicates the coefficients are sensitive to random noise in **d**, and could fluctuate wildly with very slight changes in the dependent variable. In statistical terms, there would be low confidence in a computed value of the regression coefficients when they have high variances.

The variance is a mathematical property of the regression coefficients, and its computation is quite simple. We assume the noise in each sensor has a mean value of zero, and the covariance matrix of the noise terms is given by $V(|\boldsymbol{\epsilon}|) = \sigma^2\mathbf{I}$. With these assumptions, a covariance matrix of the regression coefficients is defined as (Draper and Smith 1981)

$$V(|\mathbf{w}|) = \left|\left(\mathbf{H}^H\mathbf{H}\right)^{-1}\right|\sigma^2 \tag{5.7}$$

The variance of an individual coefficient is given by the corresponding diagonal element of the matrix in Eq. 5.7. A similar expression gives the covariance matrix for the coefficients, $\boldsymbol{\nu}$, in principal component regression, which is written

$$V(|\boldsymbol{\nu}|) = \left(\mathbf{S}'\mathbf{S}\right)^{-1}\sigma^2 \tag{5.8}$$

The matrix $(\mathbf{S}'\mathbf{S})^{-1}$ is diagonal with dimensions $(r \times r)$ (assuming the number of control actuators is given by r), hence the variance of the ith PC regression coefficient is simply

$$V(|\nu_i|) = \frac{\sigma^2}{s_i^2} \tag{5.9}$$

The variance is therefore inversely proportional to the square of the singular value. The singular values decrease from the first to the last PC, therefore the coefficient for the first PC has the lowest variance, and the coefficient for the last PC has the highest variance. If $(\mathbf{H}^H\mathbf{H})$ is ill-conditioned, the last few singular values could be close to zero and the variance of the corresponding regression coefficients will be very high. Ill-conditioning therefore inflates the variance of the regression coefficients of certain PCs of the control system. This is undesirable because it means the coefficients will be very sensitive to random noise in the sensor measurements.

The variance of the regression coefficient of a PC is closely related to the magnitude of the control effort for the PC. The norm of the control effort of the ith PC was discussed in Chapter 3 and was shown to be

$$|\nu_i|^2 = \frac{|p_i|^2}{s_i^2} \tag{5.10}$$

where p_i is the correlation between the ith principal component and the desired response. The control effort and variance are therefore both inversely proportional to the square of the singular values, and ill-conditioning in $(\mathbf{H}^H\mathbf{H})$ is equally detrimental in terms of control effort and coefficient variance. One difference between the effort and the variance is that even if s_i is small, the effort could also be small if p_i were very small. In contrast, the variance would still be large if s_i were small, because the variance is a property of the control system and is independent of the primary response.

Ill-conditioning in regression is called multicollinearity, since it is a result of two or more predictor variables being related by a near-constant linear function (Jolliffe 1986; Ruckman and Fuller 1993). Equations 5.9 and 5.10 illustrate how multicollinearity results in large control efforts and large coefficient variances. A simple procedure to reduce the effects of multicollinearity is to eliminate principal components with “small” singular values from the regression. To decide when a singular value is “small,” it is helpful to first normalize all singular values by the maximum singular value. The ith normalized singular value is defined to be

$$\eta_i = \frac{s_{max}}{s_i} \quad \text{for } i = 1, \ldots, r \tag{5.11}$$

A threshold value, denoted $\eta^\star$, is then specified and any normalized singular value larger than the threshold is eliminated from the regression problem. The value of $\eta^\star$ is problem dependent, but some guidance in specifying its value is given in the references, such as Ruckman (1994), who suggests a value of 30, or Rossetti et al. (1996), who suggest a value of 100. Examination of the control effort relation in Eq. 5.10 and the variance relation in Eq. 5.9 should be helpful in selecting an appropriate threshold.

This selection metric is simple to apply when it is used to decide which PCs should be included in the regression, but is more complicated when it is used to decide which actuators should be included in the regression. This is because the variance of each PC regression coefficient is independent of every other PC regression coefficient. In contrast, the variance of a regression coefficient for an actuator is dependent on the other actuators in the regression, hence eliminating one actuator will affect the variances of all other coefficients. Various methods can be used to decide which actuators should be eliminated from the regression based on variance considerations, and Ruckman (1994) has applied variance inflation factors to the actuator selection problem. These methods are generally based on assigning a portion of the variance of each PC to the actuators, and the actuator with the highest total apportioned variance is then eliminated from the regression model (Ruckman 1994).

The drawback of variable selection based on variance minimization alone is that it ignores the predictive capability of each PC. This means a PC that is very useful for predicting the desired response might have a high variance and would therefore be eliminated from consideration if selection were based solely on variance reduction (Jolliffe 1986). A more thorough selection procedure would also consider the predictive ability of each PC, and this criterion is discussed in the next section.

5.3 Significance of regression variables

The correlation between a principal component and the dependent variable, or primary response, is a useful selection criterion. The amount of correlation can be quantified as significant or insignificant using a statistical test known as the t-test. A principal component is judged to be insignificant if a confidence interval about its regression coefficient includes zero. In the context of multichannel feedforward control, this test is equivalent to applying a signal-to-noise ratio threshold to the correlation between each PC and the primary disturbance, and eliminating PCs whose correlation falls below the threshold. A statistical test for the importance of a predictor variable in a regression model is closely related to the variance of the corresponding regression coefficient. To determine the importance of a predictor variable, we go one step beyond calculating the variance and assume the probability distribution of the random noise terms in the dependent variable $\mathbf{d}$ is known. The regression coefficients are linear functions of the dependent variable (see Eqs. 5.3 and 5.6), hence knowing the distribution of the random terms in $\mathbf{d}$ is sufficient to characterize the distribution of the regression coefficients.

An example of the probability distribution of a complex regression coefficient is shown in Fig. 5.1. A three-dimensional normal distribution of a complex variable is plotted, where the height above the complex plane corresponds to the probability of the underlying complex value. The mean value of the variable occurs at the maximum value of the probability distribution, which in this example occurs where the real and imaginary parts of the variable are zero. The surface is symmetric about its mean value because only the magnitudes of the $\boldsymbol{\epsilon}$ terms have been characterized. The black line on the surface denotes a radius two standard deviations from the mean value. From the theory of the normal distribution (e.g., see Walpole and Myers (1985)), approximately 95% of the measured values of this variable will fall inside the black circle.

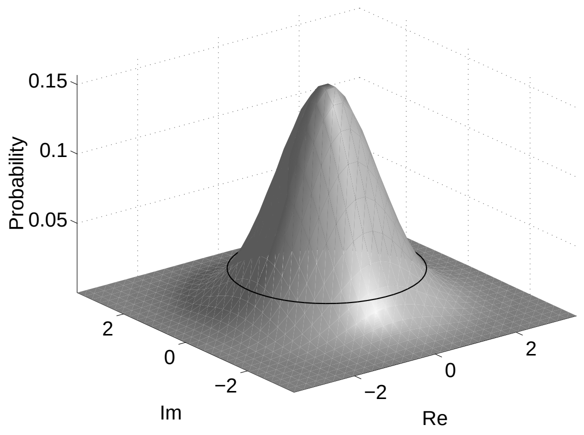

Figure 5.1: Probability distribution for magnitude of complex variable

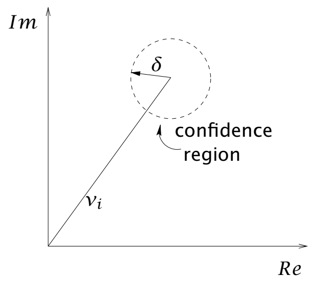

Figure 5.2: Confidence region about complex variable

In the context of the active control problem, the surface in Fig. 5.1 describes the probability distribution of a single regression coefficient, w_i or ν_i. The mean value at the center of the plot corresponds to the true value of the coefficient, while the probability distribution describes the likelihood of computing any other coefficient value. When a confidence interval is computed about a regression coefficient, the variance of that coefficient and the shape of its probability distribution are assumed to be known, but the true value of the coefficient is not known. The confidence interval describes a region about the computed coefficient that is likely to contain the true value of the coefficient. For example, 95% of the values of a normally distributed random variable fall within two standard deviations of the mean value of the variable. Therefore, given an arbitrary value drawn from a normal distribution, an interval with a radius of 2σ about the sampled value will contain the mean value of the variable 95% of the time. In this example, 95% corresponds to the significance level of the confidence interval. As the significance level gets smaller, the size of the interval also gets smaller. For complex regression coefficients, the confidence interval is a circular region in the complex plane. An example of a circular confidence interval about a complex value is shown in Fig. 5.2 as the dotted circle with a radius of δ about the complex value, ν_i (Ruckman and Fuller 1995). The following discussion will illustrate how the radius of the confidence region depends on the variance of $|\nu_i|$ and the significance level of the confidence region.

The random terms in $\boldsymbol{\epsilon}$ are assumed to be normally distributed, in addition to the previous assumptions about their mean and variance. As discussed in Ruckman (1994), this assumption may not be correct for pressure measurements expressed in linear units. However, the errors in making this assumption are small, and the benefits of using a more accurate distribution may not justify the added complexity (Ruckman 1994).

In order to apply the t-test, we must decide on a significance level for the test. The usual notation in statistical texts for a significance level is to refer to a $100(1-\alpha)\%$ level of confidence, so a given confidence level is specified by the value α (not to be confused with the adaptation rate in the PC-LMS weight update equation in Chapter 4). In addition to the significance level, the number of degrees of freedom of the regression is important, and a controller with m sensors and r actuators has $(m-r-1)$ degrees of freedom (Ruckman 1994; Draper and Smith 1981). The significance and degrees of freedom are used to compute the value of the t-statistic, or Student's probability distribution, which will be denoted here by $t_{(\alpha,m-r-1)}$. Values of the Student distribution are usually tabulated in the appendices of a statistical textbook (e.g., see Draper and Smith (1981), Walpole and Myers (1985)).

Assuming the random terms are normally distributed, a $100(1-\alpha)\%$ confidence interval for the regression coefficient w_i is written (Draper and Smith 1981; Ruckman 1994)

$$|w_i| \ \pm\ t_{(\alpha/2,m-r-1)} \sqrt{\mathrm{Var}(|w_i|)} \tag{5.12}$$

Note that the t-statistic is actually computed at a significance level of $\alpha/2$ because the confidence interval describes both positive and negative variation about $|w_i|$. As a result the overall significance level is divided in half so the total confidence interval has a significance of α. The confidence interval for a PC regression coefficient is of greater interest since it will be used as a selection metric. Rewriting Eq. 5.12 in terms of the ith PC regression coefficient v_i, and substituting the variance of that coefficient from Eq. 5.9, the confidence interval for the ith PC regression coefficient is

$$|v_i| \ \pm\ t_{(\alpha/2,m-r-1)} \sqrt{\sigma^2/s_i^2} \tag{5.13}$$

$$\pm\ t_{(\alpha/2,m-r-1)}\, \sigma/s_i \tag{5.14}$$

At a fixed significance level $\alpha/2$, the size of the confidence interval increases as the singular value decreases, therefore the size of the confidence interval is largest for the last few principal components.

The t-test for selecting predictor variables is based on removing a variable from the regression if the confidence interval for the corresponding regression coefficient contains zero (Walpole and Myers 1985; Draper and Smith 1981). If the interval contains zero, there is a chance the true value of the coefficient is zero and hence the predictor variable is not important in the regression. Because the magnitude of the regression coefficient, $|v_i|$, is always positive, only the negative side of the confidence interval can contain zero. A PC should therefore be eliminated from the regression equation if

$$|v_i| - t_{(\alpha/2,m-r-1)}\, \sigma/s_i \le 0 \tag{5.15}$$

$$|v_i| \le t_{(\alpha/2,m-r-1)}\, \sigma/s_i \tag{5.16}$$

The solution for the ith regression coefficient can be substituted from Eq. 5.6 to produce

$$\left| \frac{-p_i}{s_i} \right| \le t_{(\alpha/2,m-r-1)}\, \sigma/s_i \tag{5.17}$$

$$|p_i| \le t_{(\alpha/2,m-r-1)}\, \sigma \tag{5.18}$$

where p_i denotes the mapping of the primary response **d** onto the ith principal component. The ith PC should therefore be eliminated from the regression equation if the mapping of the desired response, **d**, onto the ith PC is less than a multiple of the standard deviation of the random noise in the sensors, σ. Note that this expression was obtained from the confidence interval, which is closely related to the variance of a regression coefficient, but the final selection metric depends only on the correlation between a PC and the primary response. If this correlation is not significantly greater than correlation that could be produced by random effects, the PC should be eliminated from the regression.

As the variance σ^2 of the random term gets larger, the selection criteria in Eq. 5.18 gets more stringent, meaning the correlation between a PC and the primary must be even larger. A larger noise variance means the possibility of a high random correlation between a PC and the primary disturbance increases as well, and hence the selection criteria must increase as well. In addition, as the significance level of the test increases, the value of the t statistic increases and therefore the selection criteria also gets more stringent. As the significance of the test increases, the experimenter is asking for greater confidence that a PC that passes the selection criteria in Eq. 5.18 is in fact important in the regression solution. Taking this to an extreme, at a 100% level of confidence, every PC would have to be eliminated because there is a small chance, perhaps vanishingly small, that any amount of correlation between a PC and the primary could be due to entirely random effects.

5.4 Procedure for selecting PCs

Each of the selection criterion discussed in this chapter has its benefits, but finding a set of PCs that satisfies both criteria simultaneously may not be possible (Jolliffe 1986). Variance minimization is useful for reducing the effects of multicollinearity in a control system, and the t-test is valuable for selecting which PCs will be useful for cancelling the primary response at the error sensors. A simple procedure to use both criteria to select PCs is described here.

One possibility for combining the selection criteria is to eliminate PCs one by one starting with the smallest singular value and proceeding to the largest. When a PC is reached that satisfies the significance criteria in Eq. 5.18, selection stops and all remaining PCs are kept (Jolliffe 1986). This method would probably retain too many PCs, and the final subset could still contain high variance PCs. Another approach that may work best for feedforward control problems, where limiting the control effort is sometimes critical, is to apply the two selection criteria sequentially. This requires knowledge of the primary disturbance, the SVD of the transfer function matrix $\mathbf{H}$, and the variance of the random noise at the sensors. The first step is to eliminate PCs that require excessive control effort, in order to satisfy any practical limits on control effort. This can be done using the normalized singular values defined in Eq. 5.11. A good starting value for the selection threshold, $\eta^\star$, is 50. At this stage of the selection process, it is preferable to include too many PCs, rather than eliminate potentially useful PCs.

After the control effort metric is applied, additional PCs can be eliminated from the subset by applying the significance criteria in Eq. 5.18. A good starting value for the significance level for the test is 95%, which corresponds to $\alpha = 0.05$. The significance level can be increased or decreased depending on whether or not the final subset of PCs should be more or less significant in the presence of random noise.

If the selection criteria are applied in this manner, they should produce a small set of PCs selected out of a much larger candidate set that will be useful for reducing the primary disturbance at the error sensors within acceptable control effort bounds.

Chapter 6

Frequency Dependence of Principal Components

This chapter examines the frequency variation of the principal components of a feedforward control system. The principal components correspond to the left and right singular vectors from a singular value decomposition (SVD) of the transfer function matrix between control actuators and error sensors at a single frequency. Because the dynamics of most systems change with frequency, the transfer function matrix will also change with frequency. The effect of this frequency variation on the left and right singular vectors and the singular values of the transfer function matrix are discussed. This information will provide insight into what types of changes in the singular vectors produced by frequency variation will cause the PC-LMS controller to go unstable. This analysis is based on the derivatives of the eigenvectors of a matrix dependent on a parameter (Lancaster 1964; Nelson 1976). The singular vectors are actually the eigenvectors of $(\mathbf{H}^H\mathbf{H})$, and $(\mathbf{H}\mathbf{H}^H)$, so expressions for the eigenvector derivatives can be used to study the variation of the singular vectors with frequency.

In order to relate the analytical derivative expressions to the properties of a control system, the derivatives are evaluated for transfer function matrices generated for three control systems: a simply supported vibrating plate; a simply supported plate radiating sound into a half-space; and a cylindrical, enclosed shell that radiates sound into the enclosed cavity. In each case, transfer function matrices between control actuators and error sensors are computed across a wide range of frequencies, and the variation of the singular values and singular vectors with frequency are then related to the dynamics of the relevant system. The simply supported vibrating plate is examined in the greatest detail, because it is the simplest of the three models and the eigenvector derivatives are easily related to the dynamics of the plate.

The results of this analysis can be used to infer the robustness of the principal component LMS (PC-LMS) algorithm to errors in the transfer function matrix that are due to frequency variation. These errors will occur if the transfer function matrix is measured at one frequency, but is then used to control a disturbance at a different frequency. Other sources of error in the transfer function matrix, due to temperature variation or pressurization of an aircraft fuselage, for example, are not considered, although these factors could also produce significant changes in the singular vectors.

6.1 Eigenvalue and eigenvector derivatives

An analytical expression for the variation of the principal components of a transfer function matrix is developed by considering the derivatives of the eigenvalues and eigenvectors of a matrix dependent on a parameter. The matrix can be either of the products, $(\mathbf{H}^H\mathbf{H})$ or $(\mathbf{H}\mathbf{H}^H)$, where by definition, $\mathbf{V}$ contains the eigenvectors of the former and $\mathbf{U}$ contains the eigenvectors of the latter. Either matrix product will be denoted here by $\mathbf{A}$, since the particular one is irrelevant for these derivations. The eigenvalue problem for $\mathbf{A}$ is written

$$[\mathbf{A}(\omega) - \lambda(\omega)\mathbf{I}]\mathbf{x}(\omega) = \mathbf{0} \tag{6.1}$$

where λ is an eigenvalue, $\mathbf{I}$ is the identity matrix, and $\mathbf{x}$ is an eigenvector. The terms in the equation are assumed to be functions of frequency, ω.

If the individual elements of $\mathbf{A}$ are analytic functions of frequency, then the eigenvalues and eigenvectors of $\mathbf{A}$ vary continuously with frequency (Lancaster 1964; Watkins 1991). For the current application, $\mathbf{A}$ denotes either $(\mathbf{H}\mathbf{H}^H)$ or $(\mathbf{H}^H\mathbf{H})$, so the analyticity requirement should be satisfied as long as there is some damping in the system being studied. Since the eigenvalues and eigenvectors of $\mathbf{A}$ are continuous, we can compute their derivatives. Expressions for the derivatives can be derived using either a perturbation method (Meirovitch 1980) or by direct differentiation (Lancaster 1964; Nelson 1976), which will be used here.

Differentiating Eq. 6.1, for the ith eigenvalue-eigenvector pair, with respect to ω yields (Nelson 1976)

$$\frac{\partial}{\partial\omega}\left([\mathbf{A} - \lambda_i\mathbf{I}]\mathbf{x}_i\right) = \mathbf{0} \tag{6.2}$$

$$\left[\frac{\partial\mathbf{A}}{\partial\omega} - \frac{\partial\lambda_i}{\partial\omega}\mathbf{I}\right]\mathbf{x}_i + [\mathbf{A} - \lambda_i\mathbf{I}]\frac{\partial\mathbf{x}_i}{\partial\omega} = \mathbf{0} \tag{6.3}$$

Pre-multiplying by the eigenvector $\mathbf{x}_i^H$ produces

$$\mathbf{x}_i^H\left[\frac{\partial\mathbf{A}}{\partial\omega} - \frac{\partial\lambda_i}{\partial\omega}\mathbf{I}\right]\mathbf{x}_i + \mathbf{x}_i^H[\mathbf{A} - \lambda_i\mathbf{I}]\frac{\partial\mathbf{x}_i}{\partial\omega} = \mathbf{0} \tag{6.4}$$

Assuming $\mathbf{A}$ is a Hermitian matrix, $\mathbf{A} = \mathbf{A}^H$. With this assumption, the second term in Eq. 6.4 is zero by definition of the eigenvalue problem. Simplifying yields

$$\frac{\partial\lambda_i}{\partial\omega} = \mathbf{x}_i^H\frac{\partial\mathbf{A}}{\partial\omega}\mathbf{x}_i \tag{6.5}$$

which is the derivative of the ith eigenvalue with respect to ω. The term $\partial\mathbf{A}/\partial\omega$ is computed by differentiating the individual elements of the matrix $\mathbf{A}$ with respect to ω.

The derivative of the ith eigenvector is obtained from Eq. 6.3. Rewriting and substituting for $\partial\lambda_i/\partial\omega$ from Eq. 6.5,

$$\begin{aligned}
[\mathbf{A} - \lambda_i\mathbf{I}]\frac{\partial\mathbf{x}_i}{\partial\omega} &= \left[\frac{\partial\lambda_i}{\partial\omega}\mathbf{I} - \frac{\partial\mathbf{A}}{\partial\omega}\right]\mathbf{x}_i & (6.6)\\
&= \left[\mathbf{x}_i^H\frac{\partial\mathbf{A}}{\partial\omega}\mathbf{x}_i - \frac{\partial\mathbf{A}}{\partial\omega}\right]\mathbf{x}_i & (6.7)\\
&= \mathbf{x}_i\left[\mathbf{x}_i^H\frac{\partial\mathbf{A}}{\partial\omega}\mathbf{x}_i\right] - \frac{\partial\mathbf{A}}{\partial\omega}\mathbf{x}_i & (6.8)\\
&\equiv \boldsymbol{\phi}_i & (6.9)
\end{aligned}$$