Winter 2003

Active disturbance cancellation in nonlinear dynamical systems using neural networks

John C. Canfield
University of New Hampshire, Durham

ACTIVE DISTURBANCE CANCELLATION IN NONLINEAR DYNAMICAL SYSTEMS USING NEURAL NETWORKS

by

John C. Canfield

B.S. University of New Hampshire, 1995
M.S. University of New Hampshire, 1999

DISSERTATION

Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Doctor of Philosophy
in
Engineering: Electrical

December, 2003

This dissertation has been examined and approved.

L G Kraft

Dissertation Director, L. Gordon Kraft,
Professor of Electrical Engineering

Michael J Carter

Michael J. Carter,
Associate Professor of Electrical Engineering

[illegible]

Paul W. Latham, II,
Instructor of Electrical Engineering

[illegible]

W. Thomas Miller, III,
Professor of Electrical Engineering

[illegible]

Kondagunta U. Sivaprasad
Professor of Electrical Engineering

[illegible]

Kelly J. Black
Associate Professor of Mathematics

10/3/03

Date

DEDICATION

To My Parents

and

To the Children and Sisters

at the Saint Charles Children's Home

PREFACE

The objective of this dissertation is to provide the reader with sufficient background to elucidate the motivation behind the current research and to provide adequate technical detail so that the results of this research are accessible for use in practical applications and continuing research efforts. To that end, this dissertation contains a significant extent of background material intended to provide the reader with an overview of the more popular conventional methods of active disturbance cancellation. Additionally, the documented shortcomings of these methods provide motivation for the novel disturbance cancellation techniques introduced in the later chapters.

The field of active disturbance cancellation has a substantial historical background. In fact, the first patent for active control of sound was issued in 1936 although practical means for its implementation were not available at the time [61][68]. Since then, a tremendous variety of approaches have been developed. Chapter one is dedicated to providing a brief overview of the most common of these techniques. As an introduction to this subject, section 1.1 describes several classical approaches to disturbance rejection in linear time invariant control theory. This objective of this section is to provide an understanding of how the present research fits into the context of classical linear control theory. However, the described methods are strictly limited in applicability to time invariant systems. In contrast, most physical systems, particularly in acoustic and vibration control systems, experience significant temporal variation. Sections 1.2 through 1.4 review several active disturbance cancellation methodologies based on adaptive linear filter techniques which allow their use in time varying systems. Particular emphasis is placed on implementation of such approaches in systems containing significant transport delays, resonance, and non-minimum phase dynamics. These conditions represent some of the challenging aspects encountered in practical active vibration and noise control systems.

Special emphasis is placed on the acoustic cancellation problem although the presented techniques are applicable to a wide variety of systems.

Section 1.6 begins with a derivation of the LMS algorithm for adaptation of discrete-time linear filters. The LMS algorithm requires a measure of the filter error in order to provide convergent adaptation. However, in disturbance cancellation algorithms this signal is unavailable due to the presence of cancellation path dynamics and therefore, the LMS algorithm can not be directly applied to such systems. Instead, an extension of the algorithm known as the Filtered-X LMS method is commonly utilized. Section 1.6 provides a derivation of the Filtered-X LMS algorithm for the case where the secondary path is represented by an FIR filter. This algorithm serves as the basis for more analogous nonlinear techniques which are introduced in later chapters. Section 1.6 also presents a variety of refinements to the LMS update which are commonly utilized in active disturbance cancellation systems. These techniques serve as inspiration for the development of analogous enhancements of the nonlinear neural network methods presented in later chapters.

Section 1.6 and 1.7 provide background on the field of application for active disturbance cancellation systems. The two main categories of applications are active noise control and active vibration cancellation. However, such techniques are also useful for augmenting conventional control algorithms in order to enhance their robustness against external disturbances. Examples are provided of typical applications in each of these categories. Additionally, section 1.7 provides a tutorial review on the acoustics of active noise control. This section describes the fundamental approaches to active noise control and outlines the capabilities and limitations of each.

Chapter 2 is dedicated to providing a thorough background of the CMAC neural network and its use in modeling nonlinear dynamical systems. Section 2.1 provides a basic description of the CMAC algorithm and illustrates its relation to other feedforward neural network architectures. In particular, it is shown that the CMAC is particularly well suited to control applications due to its efficient implementation and robust convergence properties.

Additionally, in such applications, the limited generalization capabilities of the CMAC are readily surmounted by extra training given the large quantity of training data typically available in such applications. Section 2.2 provides a detailed description of the CMAC algorithm. The presented material is largely tutorial, but emphasis is placed on features and extensions of the CMAC network which are of particular significance in typical disturbance cancellation applications. Sections 2.3 and 2.4 describe the time delay neural network architecture and illustrate its ability to model nonlinear dynamical systems. In particular, these sections provide some fundamental guidelines as to the representational capabilities of the time delay CMAC network, focusing primarily on those requirements which are particular significance in active disturbance cancellation systems. Section 2.5 concludes the chapter with an analysis of the convergence properties of a single input dimension CMAC network. This section provides additional insight into the nature of the CMAC adaptation algorithm.

Chapter 3 is devoted to an analysis of the use of time delay CMAC neural networks for feedforward active disturbance cancellation in nonlinear systems. In this application, the conventional CMAC weight adaptation cannot be utilized directly since the output error is not known due to the presence of secondary path dynamics. Instead, suitable modifications to the standard CMAC weight update algorithm are derived which provide convergent adaptation for a variety of secondary paths. The simplest case is presented in section 3.2 which considers the case of a nonlinear forward signal path in combination with a purely linear secondary path. Convergence of the time delay CMAC in this case is proven analytically given a bound on the learning gain. This result is extended in section 3.3 to provide a similar conclusion for a class of static nonlinear secondary paths.

Sections 3.4 and 3.5 consider the impact of time delays in the secondary path. The presence of this delay adds a phase shift in the error signal path which can result in instability of the weight adaptation. An algorithm for the stable adaptation of the time delay CMAC in the case of secondary path delay is presented and an analytical bound on the maximum stable learning gain is derived. This result indicates that the maximum learning gain must be reduced in proportion to the length of the delay and the number of overlapping weights between nearby data points. The Band Limited CMAC introduced in this section allows for this technique to be applied for use in the case of narrowband cancellation with a general dynamical linear secondary path.

Section 3.6 describes the Filtered-X Backpropagation algorithm developed by Snyder and Tanaka [97]. This technique allows a multi-layer perceptron to be trained for active disturbance cancellation in nonlinear dynamical systems. A simplified derivation of the algorithm is provided in order to show its relation to the analogous CMAC algorithms introduced in later sections. The primary disadvantage of the Filtered-X Backpropagation algorithm is its significant computational overhead which limits the upper boundary of the effective cancellation frequency range. Additionally, the multi-layer perceptron often suffers from slow and unreliable convergence in many practical systems. These shortcomings provide motivation for introduction of the Filtered-X CMAC algorithm in section 3.7. The Filtered-X CMAC algorithm utilizes a time delay CMAC network to model the secondary path. Numerically determined gradients through this model are then used to provide a means for adaptation of a second time delay CMAC network which serves as the active disturbance cancellation compensator. Thus, the Filtered-X CMAC algorithm provides an alternative means for implementing active disturbance cancellation in systems characterized by a nonlinear secondary path which does not suffer from the limitations of the Filtered-X Backpropagation algorithm.

In many practical applications, despite the presence of a nonlinear forward path, the secondary path can be adequately modeled as a linear dynamical system. In such cases, the Reduced Filtered-X CMAC algorithm offers the benefits of lower computational overhead and more robust convergence properties. This algorithm is presented in section 3.9. Its use is demonstrated via simulation for a variety of nonlinear systems. In section 3.10, the convergence of the Reduced Filtered-X CMAC algorithm is considered analytically. It is shown that convergence of the algorithm is guaranteed given certain bounds on the learning gain.

Chapter 4 presents three alternative approaches for utilization of the time delay CMAC neural network in active disturbance cancellation systems. The regenerative CMAC approach described in section 4.1 is an extension of the analogous technique used in the case of adaptive linear compensators. This technique eliminates the need for an estimate of the original disturbance by functioning solely from the measured error signal. In section 4.2, a reinforcement learning algorithm is presented which allows the time delay CMAC to be trained without requiring any prior knowledge or model of the secondary path. However, this significant advantage comes at the expense of substantially reduced rates of convergence. Finally, section 4.3 describes the use of a recurrent time delay CMAC for disturbance cancellation. As with the regenerative approach, this technique eliminates the need for an estimate of the primary disturbance signal. The three techniques presented in this chapter are more experimental in comparison to the feedforward techniques of chapter 3. Each presentation is limited to a basic description of the algorithm and a representative simulation indicating basic feasibility of the concept.

Chapter 5 describes application of the Filtered-X CMAC algorithm to a laboratory acoustic duct system. Section 5.1 provides a brief analysis of the acoustic environment presented by such systems. In particular, the frequencies of higher order modes are determined in order to be able to restrict experiments to the fundamental mode of operation. Sections 5.2 and 5.3 provide two refinements to the algorithm designed to increase the efficiency of the implementation. Specifically, section 5.2 describes a means by which the computational overhead per iteration can be reduced at the expense of proportionally reduced rates of convergence. Section 5.3 shows how the dimension of the CMAC can be reduced in the case of narrowband disturbance cancellation. Finally, section 5.4 describes details of the laboratory implementation and documents the performance of the system. The primary goal of this laboratory platform is to show that the Filtered-X CMAC algorithm can be effectively applied to a practical physical system.

Chapter 6 provides a concluding commentary as well as a description of several practical concerns regarding implementation of the Filtered-X CMAC algorithm. Section 6.2 provides an estimate of the computational requirements of the algorithm which is a key factor in determining the upper frequency of cancellation given particular hardware capabilities. These results reveal that the algorithm is only marginally more computationally intensive than conventional linear adaptive techniques. Section 6.3 presents an overview of several means for continuous adaptation of the secondary path which have been developed. The applicability of each to the Filtered-X CMAC and Reduced Filtered-X CMAC algorithms is considered. Finally, section 6.4 reveals how the reduced Filtered-X CMAC algorithm can be extended for application in systems requiring multiple actuators and multiple error sensors.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

ACTIVE DISTURBANCE CANCELLATION IN NONLINEAR DYNAMICAL SYSTEMS USING NEURAL NETWORKS

by

John C. Canfield

University of New Hampshire, December, 2003

A proposal for the use of a time delay CMAC neural network for disturbance cancellation in nonlinear dynamical systems is presented. Appropriate modifications to the CMAC training algorithm are derived which allow convergent adaptation for a variety of secondary signal paths. Analytical bounds on the maximum learning gain are presented which guarantee convergence of the algorithm and provide insight into the necessary reduction in learning gain as a function of the system parameters. Effectiveness of the algorithm is evaluated through mathematical analysis, simulation studies, and experimental application of the technique on an acoustic duct laboratory model.

CHAPTER 1

ACTIVE DISTURBANCE CANCELLATION

1.1 Disturbance Cancellation in Linear Time Invariant Systems

In a general sense, the purpose of a control algorithm is to create activation signals for some plant or process in order to achieve a desired behavior in the outputs of that system. In open-loop control, the activation signals are created solely from an *a priori* model of the system, whereas in feedback control systems, a measure of the system output is continuously utilized to improve the performance of the controller. Generally, the use of feedback provides superior performance particularly in the presence of modeling uncertainties and external disturbances.

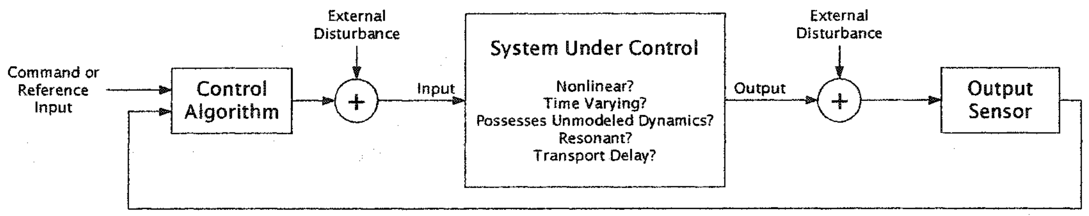


Figure 1.1 – A block diagram illustrating the general control system.

An illustration of the generic control system is shown in figure 1.1. In practical systems, there are often many factors which complicate the design of a suitable control algorithm. One commonly faced difficulty is the presence of nonlinear plant characteristics as might be caused by actuator saturation and sensor limitations. Additionally, many physical systems have time varying dynamics due to component aging and environmental factors. However, even if the system is linear and time variant, the presence of unmodeled dynamics, plant resonances, and transport delay commonly complicate practical control system design. In addition to non-ideal features associated with the plant, another common complication is the presence of external disturbances which can be introduced at different points in the system. Such disturbances often represent actual external stimulation sources, but can also be utilized to model quantization effects and measurement noise.

Under the assumption that the system is linear and time invariant, the compensator and plant can be modeled by transfer functions and the feedback control system is concisely represented by the block diagram of figure 1.2. In this case, $U(s)$ represents the command input, $C(s)$ represents the compensator, and $P(s)$ represents the plant transfer function. Control systems possessing a single input and a single output can be classified into two categories. Tracking controllers, or servosystems, are designed with the objective that the system output follow the command input with as little error as possible. In contrast, regulation controllers have a zero or constant reference input and the primary objective is to maintain the system output at some fixed operating point in the presence of internal and external disturbances.

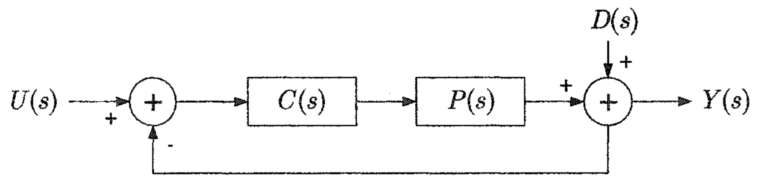


Figure 1.2 – Linear feedback control for disturbance rejection.

The most common approach to disturbance rejection in linear control system design is the use of a local feedback loop to minimize the impact of the disturbance on the closed-loop system output. As an example of this approach, consider the system of figure 1.2, where an external disturbance, $D(s)$, enters the system at the plant output. The compensator,

$C(s)$, can be designed so as to minimize the influence of the disturbance on the closed loop system output, $Y(s)$. For the sake of the present discussion, it is assumed that this system represents a regulation controller with an input $U(s) = 0$. In that case, the transfer function from the disturbance input to the closed loop system output is given by equation (1.1).

$$H(s) = \frac{Y(s)}{D(s)} = \frac{1}{1 + C(s)P(s)} \tag{1.1}$$

Using the final value theorem, the steady state output for this system of figure 1.2 can be expressed as shown in (1.2). To achieve zero steady state error, the limit in this expression must exist and be equal to zero.

$$\lim_{t\to\infty} y(t) = \lim_{s\to 0} sY(s) = \lim_{s\to 0} \frac{sD(s)}{1 + C(s)P(s)} \tag{1.2}$$

In most introductory texts on linear control theory, the topic of disturbance cancellation is primarily focused on the steady state error in response to signals which can be represented as integrals of the impulse function. For a disturbance of this form, $D(s) = s^{-A}$, and the steady state error will be zero as long as the open loop transfer function, $C(s)P(s)$, contains a factor s^{-A}. For example, in the case of a step disturbance, $D(s) = s^{-1}$, the steady state error will be zero if the compensator contains a single free integrator. In general, the steady state error can be reduced to zero for disturbances of the form $D(s) = s^{-A}$ simply by adding free integrators to the compensator. However, this modification must be performed while simultaneously ensuring that stability of the system is not adversely affected.

In many practical situations the disturbance signal can not be represented as an impulse function or its integrals. For example, a common disturbance signal in many practical applications is more accurately represented as a periodic noise source. In this case, the disturbance can be modeled by a Fourier series and the resultant response of the system in figure 1.2 is gauged by the frequency response given in equation (1.3).

$$H(jw) = \frac{Y(jw)}{D(jw)} = \frac{1}{1 + C(jw)P(jw)} \tag{1.3}$$

Complete rejection of a periodic disturbance signal would require an infinite loop gain at the frequency of each harmonic contained in the Fourier series representation of the disturbance. Consequently, complete disturbance cancellation in the feedback architecture is impossible for any periodic disturbance signal. However, it is possible to attain significant attenuation of a periodic disturbance utilizing the feedback approach. The extent of cancellation is determined by the magnitude of the loop gain evaluated at the frequencies contained in the disturbance. However, in practical applications, it is often difficult to increase the loop gain while simultaneously maintaining stability of the feedback loop, particularly in cases where the plant contains significant phase loss or transport delay.

In applications where an estimate of the disturbance can be obtained directly, an alternate disturbance rejection technique is possible. In this case, a measurement of the disturbance is utilized to create an additional plant input designed to reduce the effect of the disturbance signal on the system's output. This approach is depicted in figure 1.3 where $C(s)$ represents the compensator and $G(s)$ represents the possibility of a linear transformation between the actual disturbance and the estimated value available to the control algorithm. This approach is known as feedforward compensation since the compensator is not contained within a feedback loop. As a result, a significant advantage of this technique is that it eliminates the possibility for system level instabilities. That is, the only way in which the cancellation algorithm can become unstable is due to an instability in the compensator transfer function itself, since the output of the compensator has no influence on the signal at the input of the compensator.

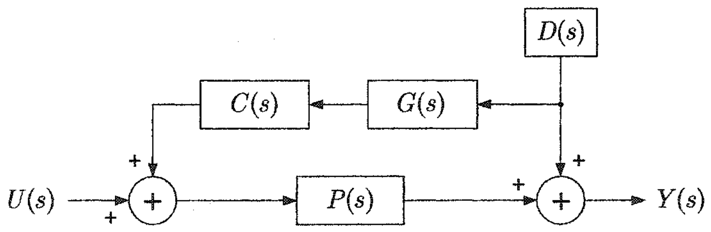


Figure 1.3 – Linear feedforward disturbance cancellation.

The transfer function from the disturbance input to the system output for the feedforward cancellation algorithm is given in equation (1.4). If the compensator is designed such that $C(s) = -G^{-1}(s)P^{-1}(s)$, then the overall resulting transfer function is identically zero. Thus, a fundamental advantage of this approach over the feedback disturbance cancellation technique is that, theoretically, the disturbance can be perfectly eliminated using a finite gain compensator. However, in practice, since the compensator must be stable and causal, complete cancellation is only realizable if the transfer function $G(s)P(s)$ is proper and minimum phase.

$$\frac{Y(s)}{D(s)} = 1 + G(s)C(s)P(s) \tag{1.4}$$

The fundamental distinction between the feedback and feedforward algorithms is that the feedforward approach requires the presence of a direct estimate of the external disturbance source while the feedback approach utilizes no knowledge as to the nature of the disturbance signal. A third approach which represents a compromise between these extremes is based on the development of a disturbance model which is used to create an estimate of the actual disturbance signal. In this methodology, the form of the disturbance signal must be known, but an actual estimate of the disturbance is not required. In practice, there are many techniques which can be utilized to model the disturbance and the most appropriate choice depends on the nature of the disturbance. In cases where the disturbance can be represented as a combination of a small number of sinusoids or step functions, one practical solution is to utilize an augmented state space observer as shown in figure 1.4. This approach is based on the assumption that an *a priori* linear model for the disturbance can be derived. The states of the disturbance model are appended to the plant state vector and a linear observer is created for the augmented state vector. The original disturbance signal can then be estimated directly from the augmented state vector. The primary disadvantage of this approach is that it requires an accurate model of both the plant and disturbance signal in order to provide effective cancellation.

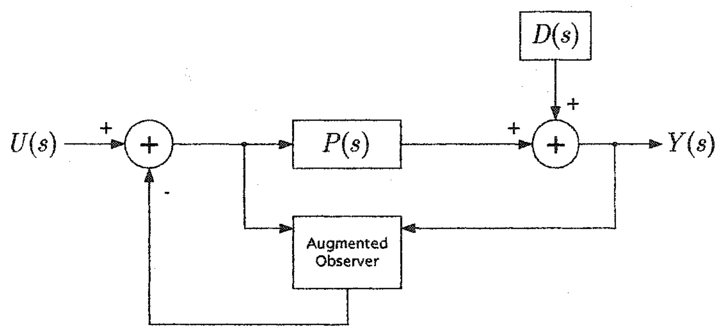


Figure 1.4 – Disturbance modeling via an augmented state space vector.

The disturbance rejection methods presented in this section are limited in applicability to cases where the system is linear and time invariant. In the remainder of this chapter it will be shown that these techniques can be extended for use in time varying linear systems through the use of an adaptive linear compensator.

1.2 Broadband Feedforward Cancellation

The block diagram of figure 1.5 depicts the standard framework utilized for feedforward disturbance cancellation. An acoustic cancellation system is illustrated to provide an intuitive basis for the discussion. In general, however, the same techniques are applicable to a broad range of applications beyond active noise control. A sampling of other common applications will be presented in section 1.6. The acoustic system is of special interest since it represents a particularly challenging application given the large transport delays and sharp resonances typical of such systems. Furthermore, this context is directly applicable to the experimental acoustic laboratory system introduced in chapter 5.

In the present section, it is assumed that all components of the system are linear and can therefore be represented by transfer functions in the Laplace domain as shown in figure 1.5. An acoustic disturbance propagates through the forward path represented by $P_1(s)$ to create a residual error signal which is monitored by the error sense microphone. A second microphone is utilized to obtain a direct estimate of the original disturbance which is used by the compensator, $C(s)$, to produce an appropriate cancellation signal. The compensator is designed to produce a cancellation signal which minimizes the power in the residual disturbance at the error sense microphone. Notice that the compensator can only influence the signal at the error sense microphone via the secondary path, $P_2(s)$. This transfer function is unavoidable in physical systems as it represents the combined dynamics of the cancellation actuator and the signal path from the cancellation source to the error sense microphone.

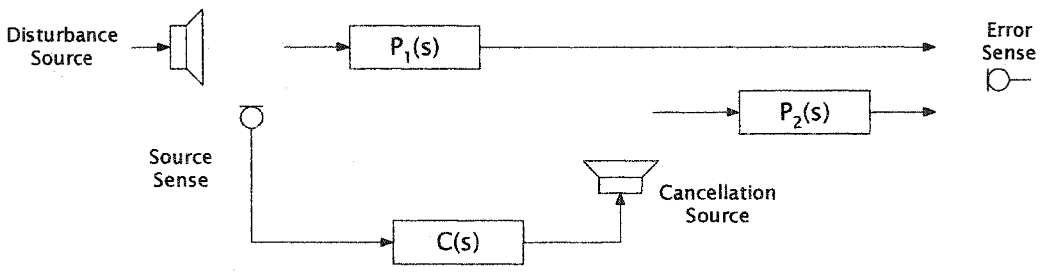


Figure 1.5 – Broadband feedforward disturbance cancellation.

In most active noise and vibration cancellation systems, the dynamical properties of the actuators and signal paths vary significantly over time. For example, in acoustic cancellation systems, the dynamics of the system are generally a function of temperature, humidity, and physical changes in the environment. Additionally, there are typically many longer term changes in the system, including the effects of aging on the signal paths and gradual changes in the bandwidth and sensitivity of speakers and microphone elements. Furthermore, relatively minor parametric changes can lead to quite significant changes in the signal propagation properties of many acoustic systems. This is particularly the case in systems with significant resonances, where minor parametric fluctuations can lead to a shift in the resonant frequencies, resulting in vast changes in the response at the disturbance frequencies. The presence of time varying factors renders static model-based compensation techniques inadequate for many disturbance cancellation problems.

A common means by which active disturbance cancellation can be effected in time varying systems is through the use of an adaptive linear compensator. A discussion of techniques which can be used for the adaptation of such a compensator is postponed until section 1.5. In the present section, the discussion will focus instead on the fundamental limitations of linear feedforward disturbance cancellation under the assumption that a means exists for providing convergent adaptation to the optimal compensator transfer function.

Given the feedforward disturbance rejection system of figure 1.5, the error signal, $E(s)$, is given by equation (1.5) where $D(s)$ represents the disturbance signal. It is evident from this expression that complete cancellation of the disturbance occurs if the compensator implements the relationship given in equation (1.6). If this is possible, the error signal will be identically zero independent of the nature of the disturbance signal.

$$E(s) = D(s)P_1(s) + D(s)C(s)P_2(s) \tag{1.5}$$

$$C(s) = -P_1(s)P_2(s)^{-1} \tag{1.6}$$

In acoustic cancellation systems, the relatively slow propagation speed of acoustic disturbances results in transport delays which are generally a dominant feature of the transfer characteristics of the signal paths. In such cases, both $P_1(s)$ and $P_2(s)$ will have a significant linear phase component. As a result, there is no guarantee that the ideal compensator expressed in (1.6) is causal. For example, consider the simplified situation where both the forward and cancellation paths are represented by pure time delays as given by equations (1.7) and (1.8), respectively.

$$P_1(s) = e^{-sT_1} \tag{1.7}$$

$$P_2(s) = e^{-sT_2} \tag{1.8}$$

For this system, the ideal compensator transfer function is given by equation (1.9). Notice that this expression is causal only under the condition $T_1 > T_2$. Therefore, in order that the compensator may be implemented in hardware, the system must be such that delay through the forward path is greater than the delay through the secondary path. In practical applications, the transport delay associated with the secondary path includes the computational overhead of the compensator implementation and any latency associated with the input and output data conversions.

$$C(s) = \frac{e^{-sT_1}}{e^{-sT_2}} = e^{-s(T_1 - T_2)} \tag{1.9}$$

A complication commonly arises in broadband feedforward acoustic cancellation systems due to the presence of a feedback path from the cancellation speaker to the disturbance source sensor. Figure 1.6 depicts a feedforward cancellation system which explicitly shows this feedback path as represented by the transfer function $F(s)$. The presence of this additive component in the source sense signal leads to two potential problems. First, it corrupts the estimate of the disturbance signal thereby potentially affecting the cancellation performance. Second, and potentially a more deleterious effect, is that this extraneous feedback path creates the potential for an unstable feedback loop consisting of the closed loop created by transfer functions $F(s)$ and $C(s)$. This problem can be alleviated in some systems via physical modifications designed to reduce the efficacy of the feedback path. For example, unidirectional microphones and directive speakers can reduce the degree to which the cancellation speaker impacts the source sense signal. Alternatively, an algorithmic solution to the problem is often utilized as illustrated in figure 1.7 [62]. In this case, a second adaptive linear filter, $F'(s)$, is trained off-line to model the actual feedback path, $F(s)$. This model of the feedback path is used to estimate the contribution of the feedback signal which is then subtracted from the compensator input.

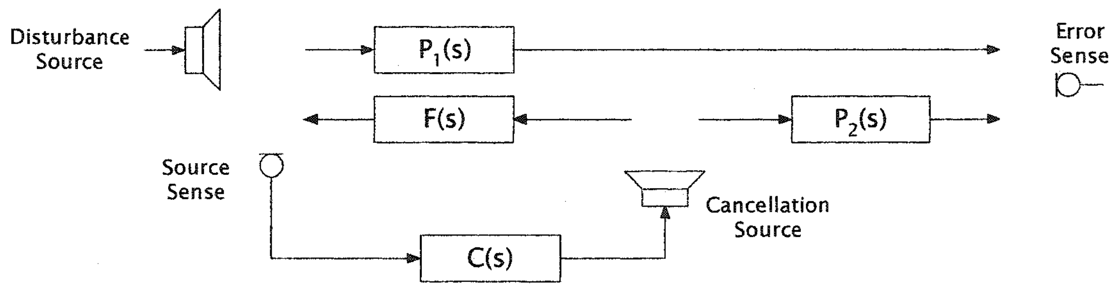


Figure 1.6 – The intrinsic feedback path in broadband feedforward cancellation.

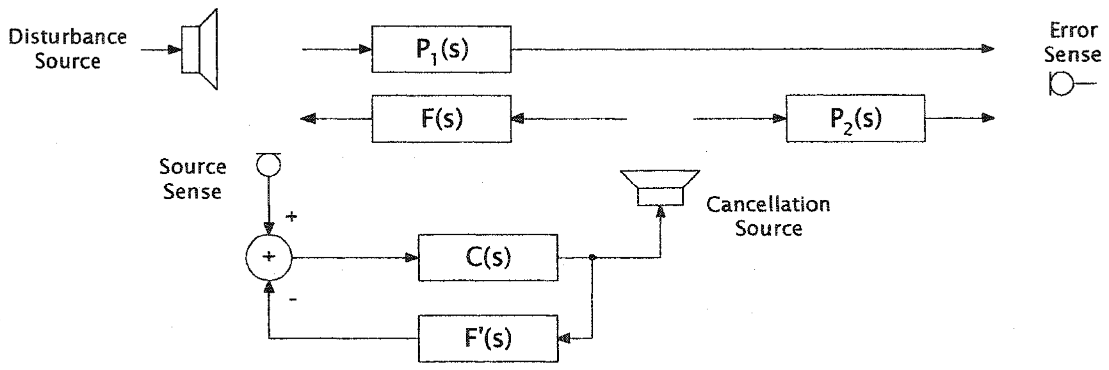


Figure 1.7 – Model based elimination of the acoustic feedback path.

The broadband feedforward cancellation approach described in this section has been successfully employed in a variety of acoustic cancellation problems [26][43][80][104]. In most applications the adaptive compensator is implemented as a Finite Impulse Response (FIR) filter due to its advantages of unconditional stability and global convergence of the adaptation algorithm. However, it is also possible to implement the compensator as an Infinite Impulse Response (IIR) filter. The IIR compensator is particularly well suited for use in resonant systems since the IIR filter provides a much more compact representation of such dynamics as compared to the FIR filter. That is, the dynamical system can be represented utilizing a smaller set of filter coefficients.

Utilization of a linear compensator results in the inability to provide effective cancellation in cases where significant nonlinearity is present in the actuators, sensors, or transmission paths of the system. For example, it is commonly the case that nonlinearities in the forward path result in the production of harmonics of the frequencies present in the disturbance signal. Since the linear compensator is able to produce a response only at the frequencies represented in the original disturbance, it is not possible to provide cancellation of such harmonics. In many acoustic environments, these harmonics can be even larger in amplitude than the fundamental due to the presence of high frequency resonances in the transmission path. To provide cancellation in such cases, a nonlinear dynamical compensator is required. This provides the motivation for the use of neural networks in this role as addressed in chapters 3 and 4.

1.3 Feedforward Cancellation of Periodic Disturbance Signals

In this section, the feedforward cancellation system is reconsidered under the assumption that the disturbance source is a periodic signal. This *a priori* knowledge of periodicity in the disturbance source leads to several potential advantages over the broadband feedforward cancellation system presented in the previous section. The resultant cancellation algorithm is commonly referred to as narrowband feedforward cancellation. However, the approach is applicable to any periodic disturbance source, even those possessing broad spectral content.

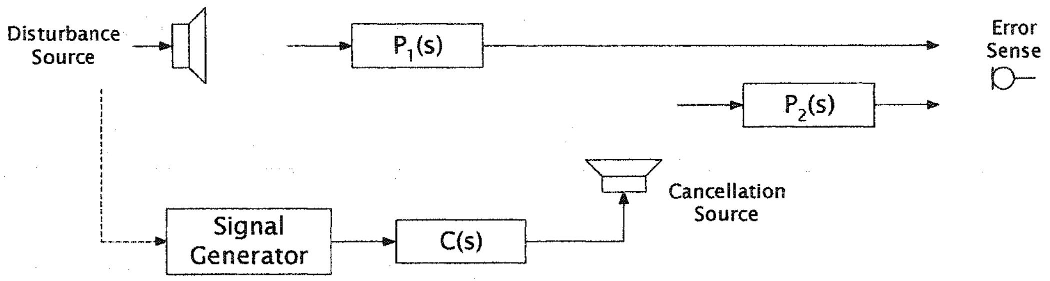


Figure 1.8 – Narrowband Feedforward Cancellation.

In many applications where the disturbance is known to be periodic, it is possible to utilize a reference signal that is not a direct measurement of the disturbance signal, as depicted in figure 1.8. For example, to cancel the fundamental tone produced by a piece of rotational machinery, it is often possible to utilize a sensor to directly generate a sinusoid at the frequency of rotation. The resultant signal can then be utilized as the input to the compensator. There are several advantages to the use of a non-acoustic reference signal sensor in active noise cancellation systems. First, this eliminates the need for use of a reference microphone in a caustic environment, for example, in the exhaust manifold of an automobile engine in the case of an active muffler system. Second, non-acoustic sensors are often more reliable and can eliminate the environmental variability associated with sense microphones. Finally, the use of a non-acoustic reference signal eliminates the possibility for the cancellation to reference signal feedback instability as described in the previous section.

Another general advantage of narrowband feedforward cancellation is that it allows for the selective cancellation of a periodic offending disturbance without interfering with other sounds in the environment. For example, engine noise can be reduced for a heavy equipment operator while other sounds, including warning alarms and even conversation, are unaffected. The narrowband feedforward approach also eliminates the causality constraint requiring that the delay through the forward path be longer than the delay through the secondary path. To illustrate this point, consider a forward path represented by a pure time delay, $P_1(s) = e^{-sT_1}$, and a secondary path represented by another pure time delay, $P_2(s) = e^{-sT_2}$. Assuming that the reference signal is an exact replica of the disturbance source, perfect cancellation will occur if the compensator implements the transfer function given in (1.10).

$$C(s) = -\frac{e^{-sT_1}}{e^{-sT_2}} = -e^{-s(T_1-T_2)} \tag{1.10}$$

In the case of broadband feedforward cancellation, this led to the restriction that $T_1 > T_2$. However, if the disturbance signal is periodic with period T then the compensator output can be delayed by any multiple of this period without affecting the output signal. Therefore, the ideal narrowband compensator transfer function is given by (1.11) where $k \in \{0,1,\cdots\}$. As a result, the parameter k can always be chosen to ensure that the compensator is causal given any combination of forward and cancellation path delays.

$$C(s) = -e^{-s(T_1-T_2)}e^{-skT} = -e^{-s(T_1+kT-T_2)} \tag{1.11}$$

An additional advantage in assuming periodicity of the disturbance signal results from the fact that the transfer function implemented by the compensator is only required to be valid over the range of frequencies present in the disturbance signal. Consequently, for narrowband disturbance signals, the order of the compensator can be substantially lower than the comparable compensator required in the case of broadband signals. This generally leads to an improved rate of convergence and reduces the computational overhead of the algorithm.

1.4 Feedback Disturbance Cancellation

The feedforward disturbance cancellation techniques described in the previous sections relied on the presence of an estimate of the disturbance signal. In contrast, the feedback approach presented in this section provides disturbance cancellation utilizing only the error signal measured at the location where the attenuation is desired. This is of fundamental importance in applications where an estimate of the disturbance is not available. For example, consider the case of an active noise cancellation system designed to reduce the noise created by turbulent air flow around an aircraft cabin. In this situation, the source of the acoustic disturbance is distributed over the entire surface of the aircraft and it is not possible to provide a suitable reference signal.

The general feedback disturbance cancellation algorithm is presented in the block diagram of figure 1.9. An unobservable disturbance source produces an undesired acoustic signal at the error sense after propagating through the forward path, $P_1(s)$. The active compensator, $C(s)$, is designed to produce an appropriate output signal such that after propagation through the secondary path, $P_2(s)$, the resultant cancellation signal destructively interferes with the original disturbance thereby reducing the magnitude of the error signal.

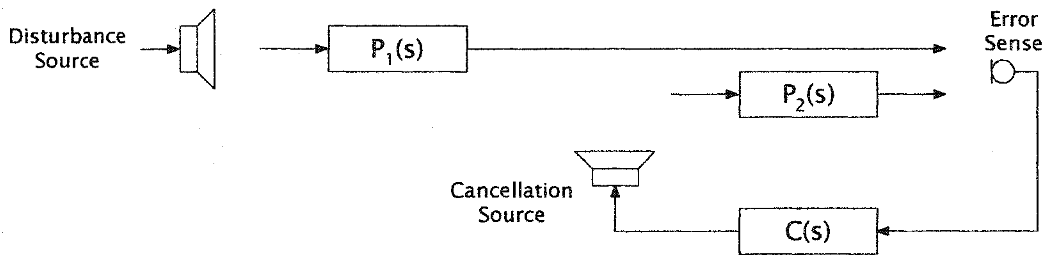


Figure 1.9 – Block diagram illustrating feedback disturbance cancellation.

In acoustic applications, a fundamental limitation of the feedback cancellation algorithm results from the transport delay which is present in the secondary path. This delay between the sensing of an error and the quickest possible response at the cancellation point can not be compensated by a causal filter. Thus, feedback cancellation is not well suited for broadband cancellation in any system where the secondary path delay is significant. However, if the disturbance is known to be periodic, this architecture can provide cancellation even in the case of substantial secondary path delay.

The feedback cancellation system can be represented by the block diagram of figure 1.10. Notice that this is simply a linear regulator control system with a zero reference input. As a result, linear control theory can be utilized to explore the limitations of this architecture.

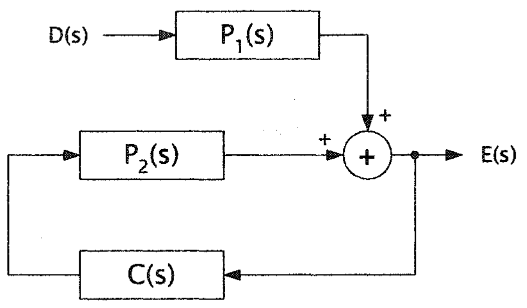


Figure 1.10 – Block diagram of the linear feedback disturbance cancellation algorithm.

From the block diagram of figure 1.10, the closed loop transfer function from the disturbance signal, $D(s)$, to the error signal, $E(s)$, can be written as shown in equation (1.12). The disturbance rejection provided by the feedback loop is maximized by designing $C(s)$ in such a way as to minimize the magnitude of the overall transfer function, $H(s)$. This goal is attained by maximizing the magnitude of the loop gain term $-C(s)P_2(s)$. However, it is not possible to achieve complete cancellation since this would require an infinite loop gain. In practical applications, the feedback approach is often severely limited in terms of the cancellation bandwidth and the degree of cancellation which can be attained. These limitations are a direct result of stability and causality constraints imposed by the system.

$$H(s) = \frac{E(s)}{D(s)} = \frac{P_1(s)}{1 - C(s)P_2(s)} \tag{1.12}$$

In order to ensure stability of the feedback loop, the open loop gain must fall below unity at the point where the open loop path reaches 360 degrees of phase shift. This condition is given in equation (1.13) where the open loop transfer function is expressed for a sinusoidal excitation of frequency ω.

$$|C(j\omega)P_2(j\omega)| < 1 \quad \forall\ \omega \ni \angle C(j\omega)P_2(j\omega) \geq 360 \text{ degrees} \tag{1.13}$$

To illustrate the effects of secondary path delay on the feedback loop stability, consider the case of an acoustic cancellation system where the sense microphone is mounted 20 cm from the cancellation speaker. For the sake of this example, the cancellation speaker resonances, secondary path dynamics, and microphone transfer function are ignored and it is assumed that the entire secondary path can be represented by a pure time delay as given in equation (1.14).

$$P_2(s) = e^{-s(571\times10^{-6})} \tag{1.14}$$

The time delay in (1.14) corresponds to the transport delay associated with propagation of the sound wave from the cancellation speaker to the error microphone. At 25°C and 50% relative humidity, this delay is approximately 571 µs as shown. The Bode plot for this secondary path is depicted in figure 1.11.

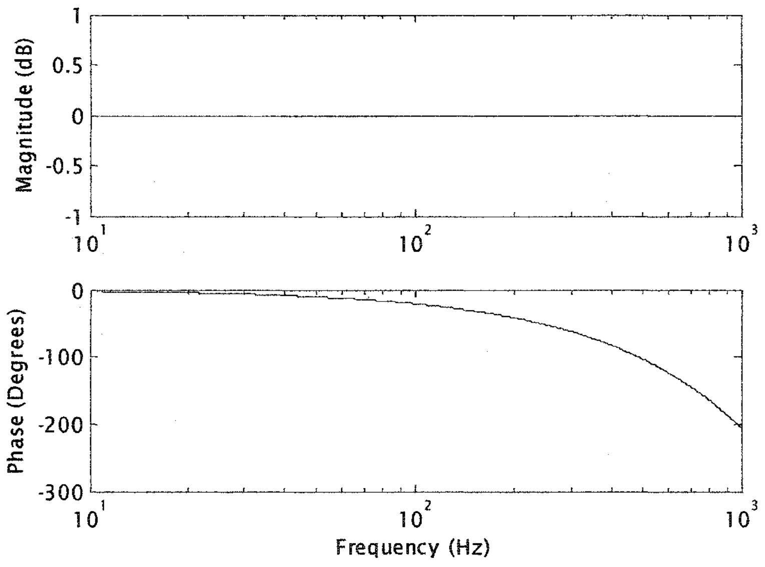


Figure 1.11 – Bode plot of pure time delay secondary path.

The magnitude response of the secondary path is simply equal to unity at all frequencies, while the phase response is linearly increasing with frequency. The compensator employs negative feedback and therefore will contribute a minimum of 180 degrees of phase shift. Therefore, to ensure stability, the frequency response of the compensator must be designed such that its gain falls below unity well before the point at which the secondary path approaches 180 degrees of phase shift. For the time delay associated with 20 cm spacing, this occurs at about 800 Hz. Since disturbance rejection is optimized by maximizing the compensator gain, this limitation puts a fundamental upper limit on the useful frequency range of the feedback approach.

A suitable feedback compensator can be designed using the conventional frequency domain approach as illustrated in figure 1.12. Figure 1.12a depicts the secondary path Bode plot for the case of a pure time delay. The compensator transfer function is designed with the objective of maximizing the loop gain $-C(s)P_2(s)$ at low frequencies but reducing the gain below unity before the open loop phase shift reaches 180 degrees. A typical compensator which achieves this goal is pictured in figure 1.12b, while the resultant aggregate open loop transfer function is shown in figure 1.12c. Notice that the additional phase introduced by the compensator must also be factored into the stability calculation. It is tempting to utilize an extremely sharp cutoff characteristic for the compensator in order to extend the bandwidth of the high gain region. However, the additional phase associated with the sharper roll-off generally more than offsets the benefit of the faster magnitude roll-off [62].

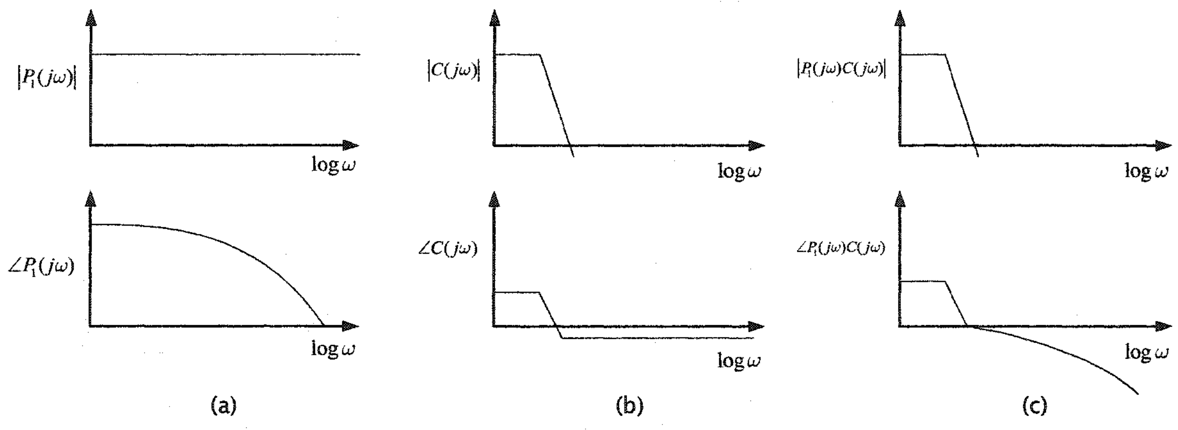


Figure 1.12 – Bode plots for secondary path (a), compensator (b), and open-loop transfer function (c).

The fundamental point illustrated by this example is that the maximum cancellation frequency is severely limited in cases where the secondary path has a substantial transport delay. In fact, the cancellation bandwidth and extent of cancellation are inversely proportional to the transport delay in the secondary path. This suggests that successful application of feedback active noise control requires that the error microphone is placed as close as possible to the cancellation speaker. However, in practice there is a limit on how close the microphone can be placed due to the effects of non-uniform acoustic fields at close proximity to the speaker. One application which is well suited to this approach is the case of headphone active noise cancellation for personal hearing protection. In this application, it is possible to locate the microphone in extremely close proximity to the cancellation speaker.

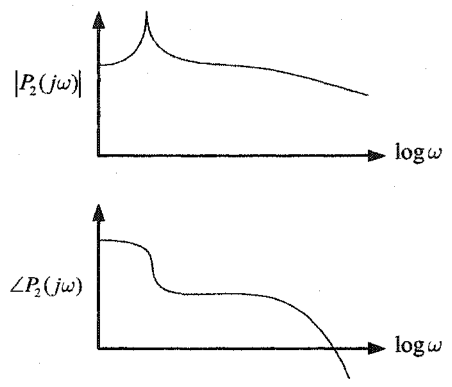


Figure 1.13 – Typical secondary path with resonance and transport delay.

The difficulties associated with acoustic feedback cancellation are exacerbated further in cases where resonances are present in the secondary path. Such resonant properties often result from the speaker dynamics as well as cavity resonances in the acoustic path. For example, consider the case of a secondary path characterized by a single low frequency resonant peak as illustrated in the Bode plot of figure 1.13. Most wide range speakers exhibit a significant resonant peak at a frequency between ten and several hundred Hertz. The presence of this resonant peak has two complicating influences on design of the feedback compensator. First, the resonant peak adds 180 degrees of phase shift at frequencies above resonance. Second, the amplitude of the resonant peak must be reduced below unity or must be located at a frequency at which there is less than 180 degrees of phase shift. One common alternative approach is to utilize an inverse notch filter which reduces the impact of the resonance. This is often difficult in practice since the resonance is likely to vary due to physical factors such as aging, temperature, and relative humidity. Additionally, the high-Q nature of typical acoustic resonances requires a high degree of accuracy in the inverse filter. Mismatch between the inverse filter and the resonance will result in degraded cancellation performance or even instability.

The regenerative feedback disturbance cancellation algorithm provides an alternative architecture which eliminates some of the limitations of the standard linear feedback approach [3][4]. This method utilizes a more sophisticated model of the secondary path enabling it to provide superior performance in cases where the secondary path possesses complicated dynamics. The regenerative cancellation architecture is shown in figure 1.14.

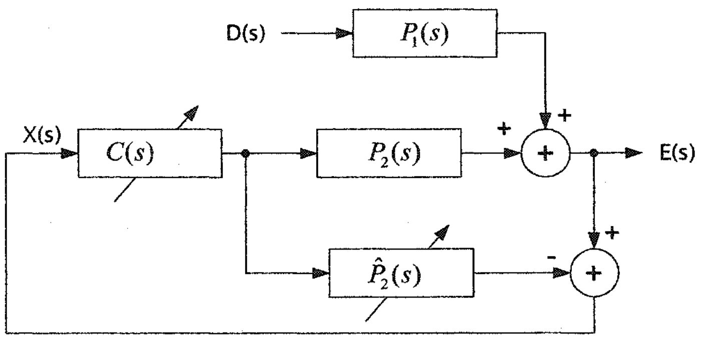


Figure 1.14 – Regenerative Feedback Disturbance Cancellation.

The regenerative cancellation algorithm utilizes a linear adaptive filter represented by $\hat{P}_2(s)$ to model the actual secondary path dynamics, $P_2(s)$. It is assumed that this model is created off-line prior to use of the cancellation system. Techniques for continuous on-line adaptation of the secondary path model will be presented in Chapter 6. The compensator input, $X(s)$, can be expressed as given by equation (1.15). Under the assumption that the secondary path model is a perfect representation of the actual dynamics, $\hat{P}_2(s) = P_2(s)$, this reduces to the result in equation (1.16).

$$X(s) = \frac{D(s)P_1(s)}{1 + C(s)P_2(s) - C(s)\hat{P}_2(s)} \tag{1.15}$$

$$P_2(s) = \hat{P}_2(s) \;\Rightarrow\; X(s) = D(s)P_1(s) \tag{1.16}$$

Thus, the regenerative architecture utilizes the secondary path model to eliminate the effects of the cancellation signal from the measured error signal. The resultant signal is used as input to the compensator. Consequently, $X(s)$ is equal to the original error signal independent of the output of the compensator and therefore, remains invariant during adaptation of the compensator. The error signal is given by equation (1.17). For complete disturbance cancellation $E(s) = 0$. This can be achieved if the compensator implements the transfer function given by (1.18).

$$E(s) = D(s)P_1(s) + D(s)P_1(s)C(s)P_2(s) \tag{1.17}$$

$$C(s) = -\frac{D(s)P_1(s)}{P_2(s)D(s)C(s)} = -\frac{1}{P_2(s)} \tag{1.18}$$

Therefore, under the assumption that the secondary path model is ideal, complete cancellation can be attained if the compensator implements an inverse of the secondary path. However, in two cases of practical significance, an inverse model of the secondary path can not be implemented. First, if the secondary path is non-minimum phase, then it will not possess a stable inverse. Second, if the secondary path contains a pure time delay, then its inverse will not be causal. However, when the inverse can be implemented, this approach provides the fundamental advantage that it alleviates the potential for instability in the feedback loop. This is due to the fact that the compensator output has no effect on the compensator input as long as the secondary path model is sufficiently accurate.

1.5 Compensator Adaptation with Secondary Path Dynamics

In most active disturbance cancellation problems, the dynamics of the system are significantly time varying as a result of changes in temperature, humidity, and the physical structure of the cancellation environment. Additionally, the signal paths encountered in typical applications are of high order and can not be conveniently derived via physical models. As a result, most active disturbance cancellation systems employ adaptive linear compensators which provide the capability to optimize the performance of the system without *a priori* knowledge of the system. The most widely used algorithm for this task is the Filtered-X Least Mean Square (LMS) algorithm which was developed independently for control applications and active noise control [62][107].

The Filtered-X LMS algorithm is an extension of the standard LMS algorithm which is widely used for the adaptation of linear adaptive filters. The standard LMS algorithm will be derived here for the insight it provides into the Filtered-X variant. Additionally, several refinements of the standard LMS algorithm are presented which will also prove useful in augmenting the performance of the Filtered-X Algorithm.

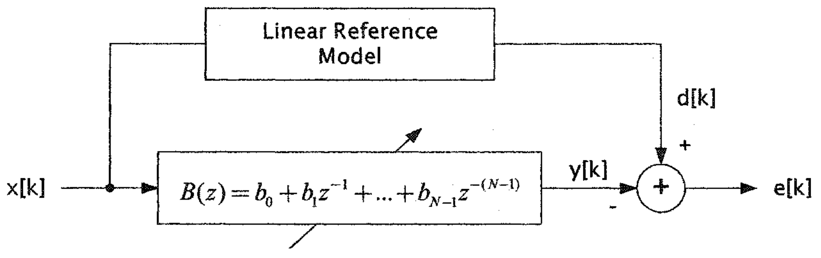


Figure 1.15 – The architecture of the conventional LMS adaptation algorithm.

Consider the adaptive linear filter shown in figure 1.15. The transfer function $B(z)$ represents an FIR filter whose coefficients, $b_0, b_1, \ldots, b_{N-1}$, are to be adapted in a manner which minimizes the difference between the filter output, $y[k]$, and the target signal $d[k]$. This filtering operation can be represented over the time interval $k \in [0, M]$ by the single matrix equation (1.19) using the definitions in (1.20) and (1.21).

$$\mathbf{y} = \mathbf{Xb} \tag{1.19}$$

$$\mathbf{X} = \begin{bmatrix} x[0] & x[-1] & \cdots & x[-N+1] \\ x[1] & x[0] & \cdots & x[-N+2] \\ \vdots & & & \vdots \\ x[M] & x[M-1] & \cdots & x[M-N+1] \end{bmatrix} \tag{1.20}$$

$$\mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ \vdots \\ y[M] \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} d[0] \\ d[1] \\ d[2] \\ \vdots \\ d[M] \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} e[0] \\ e[1] \\ e[2] \\ \vdots \\ e[M] \end{bmatrix} \tag{1.21}$$

The vector $\mathbf{e}$ as defined in equation (1.22) represents the error in the output of the filter over the time interval of interest. In addition, the sum squared error, S, as defined in (1.23) provides a single scalar error metric for gauging the performance of the adaptive filter over the entire time interval, $k \in [0, M]$.

$$\mathbf{e} = \mathbf{d} - \mathbf{X}\mathbf{b} \tag{1.22}$$

$$S = \sum_{k=0}^{M} e[k]^2 = \mathbf{e}^T\mathbf{e} = (\mathbf{d} - \mathbf{X}\mathbf{b})^T(\mathbf{d} - \mathbf{X}\mathbf{b}) \tag{1.23}$$

From equation (1.23), the aggregate error surface is a hyper-parabaloid in the filter parameter vector, $\mathbf{b}$. Therefore, there is a single value of the parameter vector which minimizes the sum squared error of the adaptive filter over the entire time interval. The solution, represented by the optimal parameter vector $\hat{\mathbf{b}}$, can be found via equation (1.24). The result, given in (1.25), represents the standard least squares solution.

$$\frac{\partial \mathbf{e}^T\mathbf{e}}{\partial \mathbf{b}} = -2\mathbf{X}^T\mathbf{d} + 2\hat{\mathbf{b}}\mathbf{X}^T\mathbf{X} = 0 \tag{1.24}$$

$$\hat{\mathbf{b}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{d} \tag{1.25}$$

This solution for the optimal filter parameter vector is known as the batch mode least squares method since all available data is utilized simultaneously to derive the result. The batch mode solution provides insight into the nature of the adaptive filter, but the storage requirements and computational delay associated with this technique make it ill-suited for real time implementations. Instead, an alternative approach known as the Least Mean Square (LMS) algorithm is typically utilized in such cases. The LMS algorithm solves this optimization problem iteratively on a sample by sample. Specifically, the LMS algorithm is a gradient descent approach whereby on each discrete time interval, all of the filter coefficients are updated with an increment that reduces the instantaneous squared error, $e[k]^2$. As a result, the update equation for arbitrary filter coefficient i takes the form shown in equation (1.26). The learning gain parameter η controls the rate of convergence and must be sufficiently small in order to ensure stability of the algorithm.

$$b_i[k+1] = b_i[k] - \eta \frac{\partial e^2[k]}{\partial b_i} \qquad \forall\ i \in \{0,1,\ldots,N-1\} \tag{1.26}$$

The partial derivative term in (1.26) is computed as shown in equation (1.27). The resulting LMS coefficient update is given in (1.28). This operation is performed on each filter coefficient at each discrete time interval.

$$\frac{\partial e[k]^2}{\partial b_i} = \frac{\partial}{\partial b_i}\{d[k]-y[k]\}^2 = -2\{d[k]-y[k]\}\frac{\partial y[k]}{\partial b_i} = -2e[k]x[k-i] \tag{1.27}$$

$$b_i[k+1] = b_i[k] + \eta e[k]x[k-i] \tag{1.28}$$

There are several refinements to the LMS algorithm which are of particular use in active disturbance cancellation applications. One category of enhancements is designed to reduce the computational overhead required by the algorithm. Although the LMS update in (1.28) is in itself quite simple, the presence of extremely long filters can still represent unmanageable computational burden at high sampling rates. For example, typical active noise control applications often require filters with hundreds of coefficients in order to model resonances in the signal paths. One such technique for reducing the computational demand of the algorithm is to eliminate the multiplication operations in (1.28). The resultant signed error LMS coefficient update is given in (1.29).

$$b_i[k+1] = b_i[k] + \eta\,\mathrm{sgn}\left(e[k]\right)x[k-i] \tag{1.29}$$

In fact, the update of (1.29) can be performed without any multiplication at all if a binary shift operation is utilized to implement the learning gain scaling. As a result, the entire weight update can be performed by a single addition or subtraction based on the sign of the error signal. In addition to reducing the computational overhead, this variant of LMS also allows the algorithm to be efficiently implemented in simple hardware which does not support intrinsic multiplication operations. The coefficient update can be further reduced by utilizing only the sign of both the error signal and the delayed input as shown in equation (1.30). This coefficient update can be implemented by simply utilizing an XOR logic gate to determine whether the fixed increment η is added or subtracted from the coefficient.

$$b_i[k+1] = b_i[k] + \eta\,\mathrm{sgn}\left(e[k]\right)\mathrm{sgn}(x[k-i]) \tag{1.30}$$

The computational reduction attained via the signed update LMS algorithms of (1.29) and (1.30) is largely dependent upon the nature of the hardware upon which the algorithm is implemented. For example, on a DSP with single cycle multiplications, these algorithms provide little benefit. However, in such cases, the computational overhead of the LMS algorithm can still be reduced through the use of partial update algorithms. Rather than updating all coefficients on each discrete time interval, the partial update algorithms select a subset of coefficients to be updated on each iteration. Therefore, in implementations on sequential processors, the computational burden per discrete time sample can be substantially reduced.

Two commonly utilized partial update algorithms are the periodic LMS algorithm and the sequential LMS algorithm [33]. The periodic LMS algorithm is described in equations (1.31) and (1.32).

$$l = N\left\lfloor \frac{k}{N} \right\rfloor \tag{1.31}$$

$$b_i[k+1] = \begin{cases} b_i[k] + \eta e[l]x[l-i] & \text{if } (k+i) \bmod N = 0 \\ b_i[k] & \text{otherwise} \end{cases} \tag{1.32}$$

On each discrete time step, k, a subset of $\lfloor L/N \rfloor$ coefficients is updated, where N is a fixed design parameter. For $N=1$ equation (1.31) is equivalent to $l=k$ and this algorithm reduces to the standard LMS update in which all of the weights are updated at each time step. At the other extreme, if $N=L$ then only a single weight is updated on any time step and the computational burden is reduced by a factor of $\frac{1}{N}$. Additionally, as N increases, the rate of convergence decreases proportionately. Thus, the parameter N controls the tradeoff between computational overhead and rate of convergence. Additionally, notice that this algorithm only utilizes every Nth value of the error signal. This is a result of the subsampling operation of (1.31). It has been shown that this algorithm is equivalent to performing a standard LMS update of all the weights at a sub-sampled rate [33].

The sequential LMS algorithm is given by equation (1.33). Just as in the previous algorithm, a subset of $\lfloor \frac{L}{N} \rfloor$ coefficients is updated on each discrete time step. However, unlike the periodic update algorithm, the sequential update algorithm always uses the current value of the error signal on each update. This reduces the computational overhead since it eliminates the need for subsampling the error signal.

$$b_i[k+1] = \begin{cases} b_i[k] + \eta e[k] x[k-i] & \text{if } (k-i) \bmod N = 0 \\ b_i[k] & \text{otherwise} \end{cases} \tag{1.33}$$

For a persistently exciting input signal, it has been shown that both of these algorithms are generally convergent with a rate of convergence proportional to the fraction of coefficients updated on each iteration. However, it has also been shown that the periodic LMS algorithm suffers from instability in many cases where the input has significant periodicity even in the case of signals for which the standard LMS algorithm is convergent [33]. This effect results from the presence of correlations between the cycling of the weight updates and the periodicity present in the input signal. Therefore, in cases where the input has limited spectral extent, the periodic LMS algorithm is a more appropriate choice.

An additional refinement of the LMS algorithm which is important in active disturbance cancellation is the leaky LMS variant of the algorithm as given in (1.34). The constant α represents a "forgetting" term and is usually picked to be slightly smaller than 1.0. Notice that in the absence of an error signal, the presence of α causes the coefficient to decay to zero. As a result, this modification stabilizes the coefficient updates in cases where the excitation signal is not persistently exciting. In contrast, the standard LMS update often leads to unbounded coefficient growth in the case of an input signal with insufficient spectral content.

$$b_i[k+1] = \alpha b_i[k] + \eta e[k]x[k-i] \tag{1.34}$$

With these refinements, the LMS algorithm provides a robust and computationally efficient means by which to adapt the coefficients of a linear filter. Unfortunately, the algorithm is not directly applicable to the compensators utilized in disturbance cancellation since the error in the compensator output is generally unknown. This dilemma is illustrated in the block diagram of figure 1.16 where the adaptive filter, $B(z)$, represents a linear compensator and $P_2(z)$ represents the secondary path. The goal in adaptation of the compensator is to minimize the observed error signal, $e[k]$. However, the LMS algorithm requires a measure of the error in the output of the compensator as represented by the conceptual signal, $e'[k]$. Notice that this situation arises in both the feedforward and the feedback cancellation architectures due to the unavoidable presence of the secondary path dynamics.

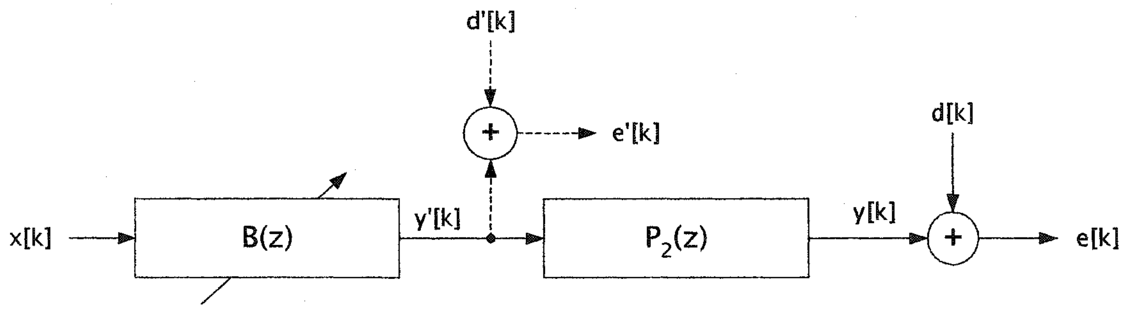


Figure 1.16 – Compensator adaptation in disturbance cancellation problems.

In order to modify the LMS algorithm for its use in the context of figure 1.16, the coefficient update must be derived in terms of the accessible error signal, $e[k]$. Thus, the form of the modified LMS algorithm is given in (1.35) where $b_0[k], b_1[k], \cdots, b_{M-1}[k]$ represent the coefficients of the adaptive filter $B(z)$ at time k.

$$b_i[k+1] = b_i[k] - \eta \frac{\partial e^2[k]}{\partial b_i[k]} \qquad \forall i \in \{0, 1, \cdots, M-1\} \tag{1.35}$$

In the present derivation, $P_2(z)$ is represented by an arbitrarily long FIR filter with L constant coefficients, $n_0, n_1, \cdots, n_{L-1}$. This choice simplifies the formulation, but is also a case of particular interest in disturbance cancellation problems since an FIR model of the secondary path is often used in practice. Additionally, given that the FIR filter can be arbitrarily long, this model can approximate any linear secondary path dynamics. However, more general derivations utilizing arbitrary linear secondary paths are available in the literature [39][62][107].

The partial derivative term in (1.35) can be expanded with the result given by equation (1.36).

$$\frac{\partial e^2[k]}{\partial b_i[k]} = 2e[k]\frac{\partial}{\partial b_i[k]}e[k] = 2e[k]\frac{\partial}{\partial b_i[k]}\left(d[k] - y[k]\right) \tag{1.36}$$

Equations (1.37) and (1.38) denote the filtering operations performed by the adaptive compensator and secondary path model, respectively. Substitution of these results into equation (1.36) yields the simplification expression given in (1.39).

$$y'[k] = \sum_{l=0}^{M-1} b_l[k]x[k-l] \tag{1.37}$$

$$y[k] = \sum_{j=0}^{L-1} n_j y'[k-j] \tag{1.38}$$

$$\begin{aligned}\frac{\partial e^2[k]}{\partial b_i[k]} &= 2e[k]\frac{\partial}{\partial b_i[k]}\left(d[k] - \sum_{j=0}^{L-1} n_j \sum_{l=0}^{M-1} b_l[k-j]x[k-j-l]\right) \\ &= -2e[k]\sum_{j=0}^{L-1} n_j \sum_{l=0}^{M-1} x[k-j-l]\frac{\partial}{\partial b_i[k]}b_l[k-j]\end{aligned} \tag{1.39}$$

If the filter coefficients vary slowly with respect to the length of the impulse response of $P_2(z)$ then the approximation in (1.40) is valid. Notice that this condition can be met by reducing the learning gain parameter.

$$\frac{\partial}{\partial b_i[k]} \cong \frac{\partial}{\partial b_i[k-j]} \quad \forall\ j \in \{0, 1, \cdots, L-1\} \tag{1.40}$$

With the approximation of (1.40), the expression for the partial derivative of the instantaneous squared error can be rewritten as shown in equation (1.41).

$$\frac{\partial e^2[k]}{\partial b_i[k]} = -2e[k]\sum_{j=0}^{L-1} n_j \sum_{l=0}^{M-1} x[k-j-i] \qquad (1.41)$$

Let $V(z)$ be defined as in (1.42). Notice that this equation represents the fact that $v[k]$ is a filtered version of $x[k]$ where the filtering is performed by a model of the cancellation path dynamics, $P_2(z)$.

$$V(z) \triangleq P_2(z)X(z) \qquad (1.42)$$

With the definition in (1.42) and the result in (1.41), the coefficient update in (1.35) can be written in its final form as shown in (1.43).

$$b_i[k+1] = b_i[k] + \eta e[k]v[k-i] \qquad \forall i \in \{0,1,\cdots,M-1\} \qquad (1.43)$$

Equation (1.43) reveals that the modified coefficient update is identical to the standard LMS algorithm except that the input terms, $x[k-i]$, are replaced with their filtered version, $v[k-i]$. This coefficient update is known as the Filtered-X LMS algorithm because it uses a filtered version of the input signal which was denoted as $x[k]$ in the original literature. The Filtered-X LMS algorithm is shown pictorially in figure 1.17b in comparison to the standard LMS algorithm in figure 1.17a. The LMS block in these diagrams represents the standard LMS update rule given in equation (1.28).

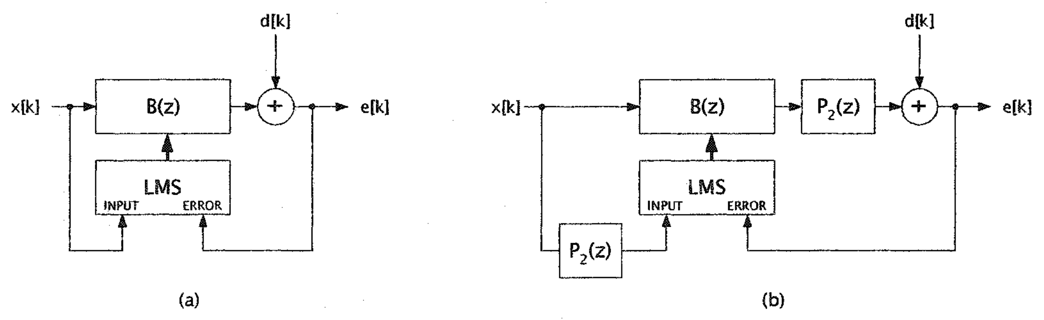


Figure 1.17 – Standard LMS (a) versus Filtered-X LMS (b).

A common intuitive explanation of the Filtered-X LMS algorithm is presented in figure 1.18. Consider the Filtered-X LMS algorithm of figure 1.17b. Under the assumption that the adaptive compensator $B(z)$ is slowly varying, it can be commuted with the secondary path dynamics, $P_2(z)$. Notice that in general this is not possible since the adaptive filter is nonlinear. However, by making the assumption that the adaptation is performed slowly, the adaptation inputs become decoupled in time from the input and output of the filter. Therefore, under this assumption, the order of the blocks are able to be swapped. The resulting block diagram is shown in figure 1.18a. If the secondary path model, $\hat{P}_2(z)$, is identical to the actual secondary path, then the block diagram can be further simplified to that shown in figure 1.18b. This final block diagram simply represents the standard LMS filter update algorithm.

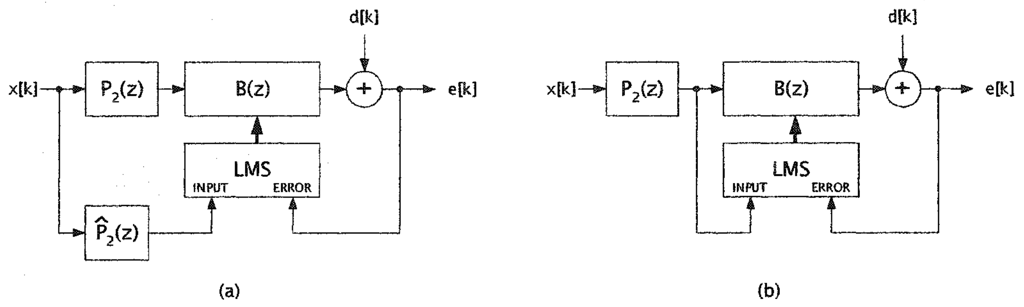


Figure 1.18 – Equivalent view of the Filtered-X LMS Algorithm under the assumption of slow coefficient adaptation.

1.6 Applications of Active Disturbance Cancellation

Active disturbance cancellation techniques are utilized in a wide range of practical applications. Additionally, the field of applications is likely to grow rapidly due to the availability of low cost, high speed digital signal processors. In fact, the fundamental concept of active sound cancellation was patented in 1936 [62][68], but practical applications of this technology have been enabled only more recently due to the availability of low cost hardware implementations for the required adaptive signal processing. This section provides a brief overview of some existing and potential future uses of this technology.

One the most publicized applications of active disturbance cancellation is acoustic noise control. In fact, many active noise control systems are presently available on the consumer market. One rapidly expanding application is in the automotive industry. For example, automobile manufacturer Honda provides an active noise cancellation system designed to reduce road noise using microphones and speakers mounted under the vehicle's seats. In addition, Siemens Automotive manufactures a system which utilizes a speaker mounted in the air intake manifold to suppress low frequency engine noise. This system provides significant attenuation at frequencies up to 600 Hz. Another commercial application of significant recent interest is the use of active noise control for the reduction of aircraft cabin noise. As an example, the Ultra Electronics corporation manufactures active noise control systems which have been implemented on a number of small propeller driven aircraft. A typical system on the Raytheon Beech 1900D utilizes 20 loudspeakers, 40 microphones, and a digital control system. Additionally, active noise control systems have been available commercially in a variety of personal hearing protection systems for many years.

The primary objective of most active noise control systems is the reduction of nuisance noise for personal comfort and minimization of operator distraction and fatigue. However, active noise cancellation can also lead to improved efficiency in many applications. For example, active noise control can reduce the need for heavy passive sound dampening materials allowing for lighter and more efficient vehicles and aircraft. Additionally, the use of active noise cancellation is being explored for replacement of the traditional baffled muffler found in automotive and small engines. The resultant reduction in engine back pressure is expected to result in a five to six percent increase in fuel efficiency for in-city driving [91].

It is likely that the demand for active noise control solutions will only expand in the future given increases in the density of population centers and the proliferation of noise generating devices characteristic of modern society. Additionally, there is a growing awareness of the negative health consequences associated with excessive noise exposure. It has long been known that prolonged exposure to sufficient intensity sound can result in permanent hearing loss. However, many additional health effects have been discovered only more recently. For example, recent studies have shown that exposure to loud noise increases the blood pressure of Rhesus monkeys with the effect lasting months after exposure to the noise has ended [89]. Other studies have shown links between noise exposure and gastrointestinal maladies [91] and anti-social behaviors including increased aggression [30].

Another significant area of application for active disturbance cancellation systems is in active vibration control. In general, active vibration control systems are designed to minimize the impact of mechanical disturbances in flexible structures. As an example, the Micromega Dynamics corporation provides systems for active vibration damping of suspension bridges using hydraulic actuators located within the cable terminations. Such systems are able to greatly reduce the effect of traffic and wind induced vibrations. Active vibration control is also becoming more commonly used in fabrication equipment for deep submicron integrated circuits [55][98]. As the resolution of the photolithography increases, minute vibrations represent a greater challenge in the manufacturing process. Active vibration control is often necessary at frequencies below which passive techniques are effective. In the future, active vibration control algorithms are also likely to play a key role in the design of smart structures. For example a identification algorithm can be utilized to identify structural damage points and active vibration control techniques can be utilized to reduce the stress within the damage zone. Such techniques provide a degree of fault tolerance in the design and consequently allow for a less conservative initial design. Active vibration has also been successfully applied to a variety of industrial processes. One interesting example is the use of active vibration control to reduce vibrations in circular saw blades during commercial sawing operations in order to reduce the saw kerf and thereby minimize the wood waste [109]. Another common industrial application is the use of active vibration control to reduce vibration caused by mass imbalance in rotational machinery. This can provide improved machine performance tolerances and increased bearing life. Active vibration control systems have also been proposed for reducing wind and even earthquake induced vibrations in buildings [47].

Active noise and vibration cancellation systems represent applications where the sole purpose of the control system is the reduction or elimination of the effects produced by an external disturbance. However, active disturbance cancellation techniques can also be utilized to augment the performance of many standard feedback controllers by adding immunity to external influences. As a simple illustration of this concept, consider the heating system controller illustrated in figure 1.19.

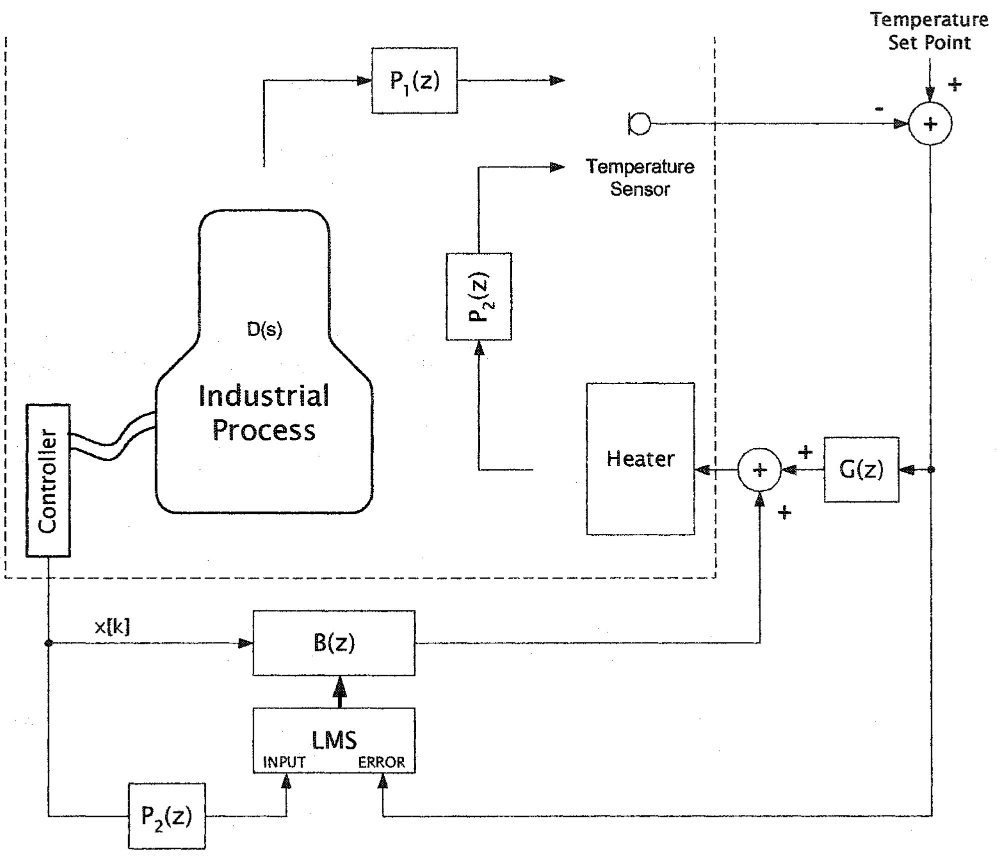


Figure 1.19 – Disturbance cancellation applied to a heating system.

A local feedback loop with compensator $G(z)$ is utilized to maintain a constant temperature at the sensor. An industrial process located within the room produces a significant amount of heat as a byproduct of its operation. The heat produced by the process varies over time and is represented by the function $D(s)$. This extra heat production represents an external disturbance to the primary heat source and its local feedback control loop. For example, consider the effects of a step change in $D(s)$. The resultant change in temperature will propagate through the room as represented by transfer function $P_1(z)$.

After a delay, this temperature disturbance will be recognized by the temperature sensor. The control loop will then adjust the heater to reduce this error. Given the delays caused by the dynamics of the heater and room, the disturbance can not be completely cancelled by the feedback controller and temperature fluctuations will be observed at the temperature sensor.

One means by which the temperature control can be improved is through the addition of an adaptive feedforward disturbance compensator as represented by the transfer function $B(z)$. The feedforward compensator utilizes information from the industrial process to adjust the heater output even before an error is sensed at the temperature sensor. The feedforward compensator must be adapted using the temperature error signal. This requires application of the Filtered-X LMS algorithm since the error in the compensator output is not available.

1.7 Principles of Acoustic Disturbance Cancellation

The research presented in this dissertation is primarily concerned with the control theory utilized for active disturbance cancellation. As a result, little emphasis is placed on the nature of the physical environment even though this factor is a crucial consideration for practical application of these techniques. For example, in a vibration control problem, the modal response of the structure generally must be understood in order to determine an appropriate placement of actuators and sensors. Given the particular significance of the active noise control application, this section provides a brief overview of the fundamental acoustical issues governing implementation of such systems. The primary objective is to provide a description of the basic categories of active noise control techniques and an understanding of their fundamental capabilities and limitations.

A sound wave is a periodic longitudinal disturbance which propagates through some compressible medium. An isolated acoustic disturbance in free space will create a spherical wave front which propagates radially outward in all directions at the speed of sound. The speed of sound is highly variable. In air, it depends on both the temperature and relative humidity. This is of particular concern in active noise cancellation systems as it represents a significant variability in the characteristics of signal propagation in the system.

In a confined space, reflections from the boundaries of the region create modal responses at sufficiently high frequencies. Each modal response represents a particular spatio-temporal distribution of pressure across the cavity. Modal responses are generally present whenever the wavelength of the acoustic wave approaches or decreases below the dimensions of the cavity. Consequently, the number of acoustic modes grows rapidly as the frequency of the sound wave increases. The acoustic duct is one example of a system which exhibits a modal response. The acoustic duct is often utilized for active noise control experiments, and additionally, there have been many practical active noise cancellation systems designed for quieting noise in heating and ventilation duct systems. At sufficiently low frequencies a duct is only able to propagate a plane wave in which the acoustic pressure is uniform across the duct cross section. However, at higher frequencies ducts allow propagation of higher order modes with much more complex wave fronts. For example, section 5.2 presents an analysis of the modal responses of a cylindrical duct system.

The variety of behaviors characteristic of sound waves in different physical settings allows categorization of active noise cancellation systems into three groups: global free space cancellation, cavity and duct cancellation, and zone-of-silence techniques [63].

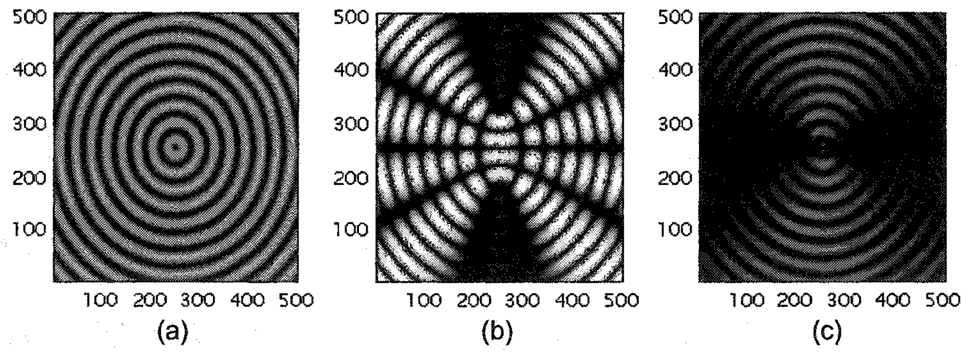


Figure 1.20 – Free Space Sound Fields.

Figure 1.20a depicts a spherical acoustic wave front emanating from a point source in free space. The gray and white circular bands represent alternating regions of compression and rarefaction. If a second acoustic source is introduced at some distance from the original, a complex sound field results as shown in figure 1.20b. The aggregate sound field has regions of intensity corresponding to the compression and rarefaction peaks of the original disturbance but also regions with twice the intensity due to the constructive interference between the two sources. As a result, the only case in which global reduction of the sound field is possible in free space (using a limited number of cancellation sources) is when the cancellation source is placed in close proximity to the source of acoustic disturbance. The resultant acoustic fields with a nearby cancellation source are depicted in figure 1.20c. In this case, waves from the two sources are largely in phase and cancellation is possible globally. As a general guideline, the cancellation source must be located within 0.1 wavelengths of the disturbance source in order to provide 20 dB global reduction in sound intensity at any given frequency [62].

In a confined space such as a room or equipment chassis, reflections from the boundaries produce modal responses at frequencies where the wavelength of the acoustic disturbance approaches the dimensions of the cavity. For example, figure 1.21 depicts two possible modal responses for a rectangular cavity of dimensions 2 X 4 X 4 meters, approximately the size of a typical household room.

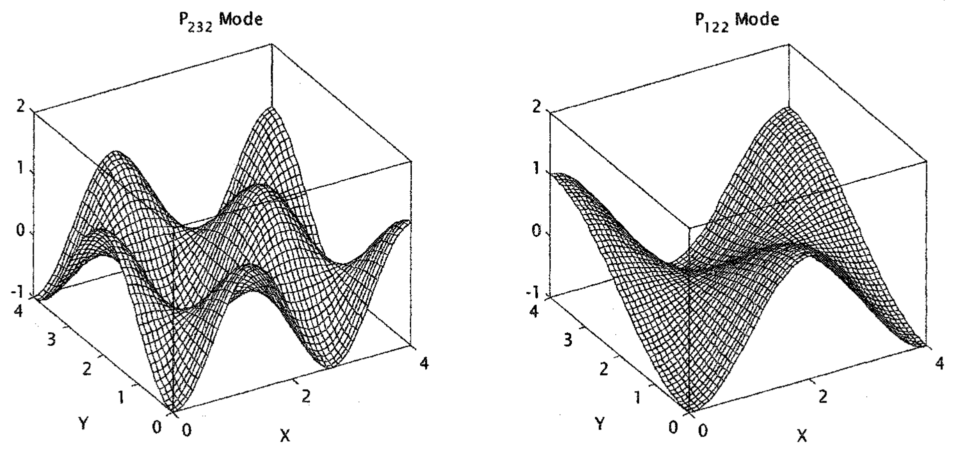


Figure 1.21 – Two Modal Sound Fields associated with a room of dimensions 2 X 4 X 4 meters.

The modal responses shown in figure 1.21 are for an acoustic disturbance with a frequency of 200 Hz and a corresponding wavelength of approximately 1.75 meters. The results shown in figure 1.21 were produced under the assumption that the walls are perfectly rigid. Notice that the room forms a cavity resonator with a complicated standing wave pattern. In any given mode, there are regions of oscillating pressure as well as points which experience no pressure fluctuations. It is possible for multiple modes to be excited simultaneously leading to a much more complicated overall pattern. The number of available modes increases rapidly with frequency. In fact, the number of modes below frequency f in a cavity of volume V is given by (1.44) where c represents the speed of sound [62]. Notice that the number of modes is proportional to the volume of the cavity and increases with the third power of the frequency.

$$N \cong \frac{4\pi V}{3c^3} f^3 \tag{1.44}$$

It is possible to design active noise cancellation systems to reduce the global intensity of the sound field associated with some set of modal responses. As a general guideline, an additional sensor and actuator is required for each additional mode [3]. Additionally, care must be taken in placement of the sensors and actuators such that the nulls of each mode are avoided. Additionally, in such cases, the cancellation system must be designed to accommodate multiple sensors as well as multiple cancellation actuators.

The zone-of-silence approach to active noise cancellation provides a localized cancellation of sound field intensity in the case of a complex wave pattern as produced by free space emanations or modal responses of a cavity. In this case, a single speaker is utilized to provide cancellation in a very small region of the overall sound field. Practical examples of this approach include personal hearing protection and head rest cancellation schemes which have been explored for use in automobiles and airlines. A typical cancellation zone will only be about a tenth of a wavelength in diameter [62]. Therefore, it is required that the cancellation speaker be located in close proximity to the desired point of silence in order to achieve any substantive cancellation.

Finally, it should be noted that active noise cancellation is most applicable at low frequencies (usually below 600Hz) for several reasons. First, at higher frequencies, the spacing requirements for free space and zone-of-silence techniques become prohibitive. Second, in acoustic cavity and duct based systems, the number of modes increases rapidly with increasing frequency which quickly makes active noise cancellation techniques unmanageable. Third, passive cancellation techniques become more effective at higher frequencies and often provide an adequate solution without the need for active control. Finally, most active noise cancellation systems are implemented digitally in hardware. The computational overhead results in an upper limit on the frequency at which cancellation can be performed.

CHAPTER 2

THE CMAC NEURAL NETWORK

2.1 Feedforward Neural Networks

Artificial neural networks can be classified into two groups based on the architecture of interconnections between the neurons. Recurrent networks are characterized by the use of local feedback connections. This imbues the network with an intrinsic history dependence. In contrast, feedforward neural networks produce a single output vector in response to any given input vector and therefore represent a memoryless mapping operation. The functionality of a feedforward neural network can be represented abstractly by the functional relationship shown in equation (2.1). Disregarding the slow time scale adaptation of the neural network relationship, this equation describes a static mapping between a between a set of input variables $x_1, x_2, \cdots, x_N$ and a scalar output y. A vector output can be accommodated by utilizing multiple expressions of this form operating in parallel on a common input vector.

$$y = f_{NN}(x_1, x_2, \cdots, x_N) \tag{2.1}$$

It is difficult to precisely define the factors which distinguish neural networks from amongst the larger body of general function approximation methods. However, there are several important factors which are typical of the majority of neural networks. One important distinction concerns the manner of adaptation. Most neural networks are able to adaptively improve their internal representations through training on individual input-output exemplar pairs. This capability allows neural networks to be utilized for continuous adaptation in an on-line fashion and is in contrast to batch mode learning which requires the entire training database to be present in advance. Additionally, a neural network is able to generalize beyond the data pairs used in its training in order to provide reasonable responses in areas of its input space which have not been explicitly trained. This feature distinguishes the neural network from a multivariate lookup table which simply stores exemplar pairs for later recall. Qualitatively, the neural network is generally comprised of a relatively simple algorithmic structure. This property facilitates use of neural networks in real-time applications as it represents a reduced computational complexity in comparison to other function approximation methods. Finally, the neural network is distinguished by the fact that it is a "black box" technique. That is, use of a neural network generally requires very little *a priori* knowledge of the relationship that is being learned. This capability is achieved by utilizing a function approximation with very flexible capabilities coupled with an adaptation algorithm which operates directly from the exemplar pairs representative of the relationship which is to be learned by the network.

These general properties of the neural network result from the use of processing units often referred to as neurons, in loose analogy to the biological nerve cells. Each neuron represents a single basis function defined on the input space and a combination of the active basis functions forms the output of the neural network. The fundamental differentiating factor between the different categories of feedforward neural networks is the nature of the basis function implemented in each neuron. The choice of basis function determines not only the representational capabilities of the network, but also the nature and extent of its ability to generalize. Additionally, the nature of the basis function has direct implications on the requirements of the training algorithm and therefore impacts the convergence properties of the neural network.

A wide variety of basis functions are appropriate for use in neural network algorithms. One of the most important considerations in this choice is whether the input domain of the basis function is local or infinite in extent. As an example of the latter case, the multilayer perceptron (MLP) utilizes a sigmoidal nonlinearity as its basis function. This results in neurons which are generally active over a large region of the input space. In contrast, the radial basis function (RBF) network generally utilizes basis functions which are active over only a very limited region of the input space. As a result, the impact of any given neuron is limited to a relatively small portion of the input domain. This fundamental difference in basis functions leads to several important tradeoffs in capabilities between the RBF and MLP networks.

In comparison with the RBF network, the MLP is generally capable of providing a more global approximation to the function being learned. As a result, the MLP can usually be trained with fewer exemplar pairs and often provides better generalization in untrained regions of the input space. However, the global nature of the basis functions can also lead to difficulties in convergence of the learning process. The most common algorithm for adaptation of the MLP is the backpropagation algorithm. This learning method is based on the use of differentiable basis functions in order to derive analytical error gradients for each weight as a function of the network output error. These error gradients are then utilized to incrementally adjust the interconnection weights in a manner which minimizes the error in the approximation. However, given the global extent and nonlinearity of the basis functions, the error surface generally contains local minima which can result in failed convergence of the training process. Many heuristic refinements have been proposed to speed convergence and elude local minima. Two popular techniques are the addition of a momentum term in the weight updates and the use of an adaptive learning gain. Despite these improvements, convergence remains a significant challenge in many practical applications.

Given the local nature of its basis functions, the RBF network generally requires the use of a greater number of neurons to cover a given region of the input space. Many RBF networks utilize Gaussian basis functions and there are several different algorithms which have been developed for the placement and shaping of the basis functions. In general, the optimal arrangement of basis functions will depend on the nature of the relationship being approximated. In practice, the basis functions are often placed in advance and the on-line adaptation is performed solely via adjustment of the weighting factors associated with each basis function. As a result, the on-line adaptation is a linear operation and therefore offers rapid and well understood convergence properties. However, if the initial basis function placement is inadequate for the relationship being learned, the adaptation will fail to produce zero error.

The Cerebellar Model Arithmetic Computer (CMAC) is a special case of a radial basis function neural network [2][16][75]. The CMAC network utilizes a fixed placement of basis functions distributed evenly across the entire input space. The CMAC basis function is a hypercube in the input domain and has a constant activation value across its entire extent. The fixed geometry and placement of the basis functions eliminates the computationally intensive basis function placement required for the Gaussian RBF network. Figure 2.1 depicts a representative view of basis functions covering a two dimensional input space. Figure 2.1a illustrates the irregular placement of Gaussian basis functions typical of the standard RBF network. In contrast, figure 2.1b shows the footprints of the basis functions defined for the corresponding CMAC neural network. In this two dimensional case, the CMAC basis functions are implemented as an overlapping pattern of squares. In this illustration, one set of basis functions is represented by the dashed lines and the other by solid lines. The exact arrangement of the CMAC basis functions is controlled by the network parameters and is fully described in section 2.2.

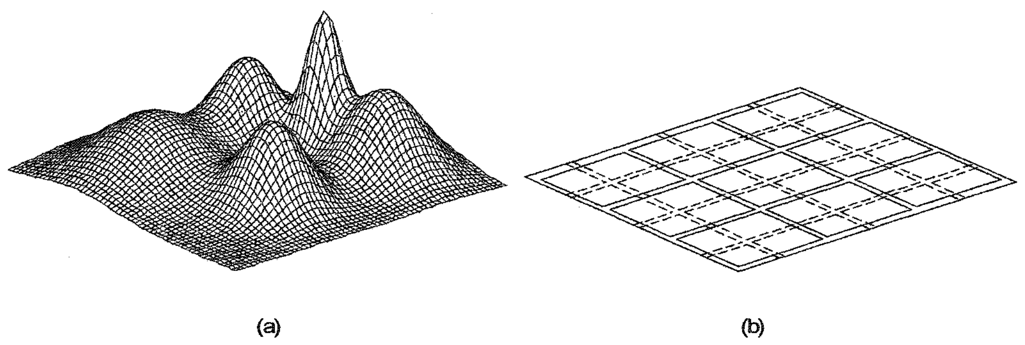


Figure 2.1 – Representative basis functions for Gaussian RBF network (a) and CMAC Network (b).

The fixed, regular structure of basis functions in the CMAC network results in extremely efficient algorithms for computation of the CMAC output and for adaptation of the mapping. In the RBF network, the most computational intensive aspect of these operations is the determination of the active set of basis functions as selected by the input vector. For example, in the Gaussian RBF network, this operation requires a search among all basis functions to find the activated set. In contrast, the active basis functions in the CMAC network can be computed directly from the input vector via an efficient calculation. Additionally, the CMAC network does not consume any resources for the placement or shaping of the basis functions, since these features are fully defined by parameters which are selected in advance.

The fixed, regular basis function placement of the CMAC network also leads to some potential disadvantages. One fundamental limitation is due to the fact that the degree of generalization in the CMAC network is uniform across the entire input space. Consequently, the CMAC basis function placement must be designed to provide sufficiently small generalization at the most irregular region of the mapping. As a result, the generalization will usually be suboptimal in more gradually varying regions of the mapping. Additionally, the CMAC mapping has many more free parameters than most other neural networks. As a result, the CMAC generally requires more training and offers more limited generalization capabilities. Additionally, for high dimensional input spaces, sparse memory compression techniques are often required to accommodate the large parameter space.

The advantages and disadvantages associated with the different feedforward neural network architectures are summarized in the spectrum presented in figure 2.2. The left end of the spectrum is characterized by the advantages of extensive generalization capabilities and minimal memory requirements. The associated disadvantages are high computational overhead and the potential for convergence difficulties. In contrast, the right end of the spectrum represents solutions which provide computational efficiency and superior convergence properties, but have severely limited generalization capabilities and substantial memory requirements. In addition, this spectrum represents the degree of parameterization associated with each function approximation technique. On the right is the most over-parameterized mapping while on the left is the least over-parameterized mapping.

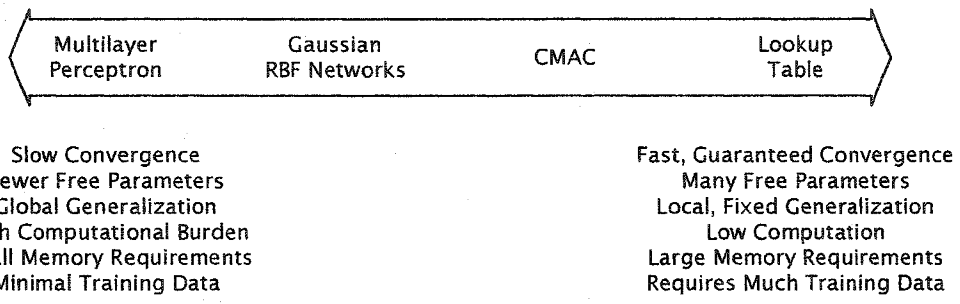


Figure 2.2 – Spectrum of tradeoffs between different artificial neural network architectures.

The optimal choice from this spectrum of neural network approximations is dependent upon the particular requirements of any given application. The most significant tradeoff in many applications is between the required computational speed and the extent of training data available. For example, consider the case of neural network handwriting recognition. In this application, recognition speed is often important but is usually not a fundamental limiting factor of the technology. Additionally, for a user customized system, the training database is relatively small and therefore, it is important to achieve the maximum extent of generalization possible. As a result, this application is best addressed by networks from the left side of the spectrum. In contrast, consider the typical active noise control application. In this case, computational speed is critically important in order to be able to operate in high bandwidth cancellation loops. Additionally, the amount of training data available is almost unlimited given the high sample rate of such systems. As a result, additional training can be utilized to compensate for reduced generalization capabilities. For these reasons, techniques from the right side of the spectrum are well suited to such applications.

2.2 The CMAC Algorithm

The RBF network is based on the fundamental principle that an arbitrary function defined on a multidimensional input space can be approximated by some combination of locally active basis functions defined on the same domain. In practice, the individual basis functions are often implemented as Gaussian functions and their combination is achieved through a weighted linear summation. The training or adaptation of the network involves the adjustment of the placement and geometry of each basis function and selection of the weighting received by each basis function in the output computation. This adaptation is performed in a manner designed to minimize the network's approximation error on a given set of training data. In many cases, the placement of the basis functions is performed off-line using the entire database of training data and the on-line adaptation is utilized solely for the adjustment of the linear weighting associated with each basis function. This considerably reduces the computational demand of on-line training at the expense of somewhat reduced flexibility in the function approximation capabilities of the network. However, even with this training procedure, the irregular placement of the basis functions results in a significant computational burden associated with determination of the active set of basis functions.

The CMAC algorithm represents a specialized form of a radial basis function network since it is based on the principle of function approximation through the weighted summation of locally active basis functions. However, unlike the Gaussian RBF network, the placement and geometry of the basis functions of the CMAC network are not dependent on the training data but are instead a parameterized feature of the network. The CMAC neural network operates under the assumption that the input space is bounded and the basis functions are distributed in a manner which uniformly covers the entire input space. Parameters of the CMAC network are used to control the degree of overlap and density of the basis functions on the input space. This fixed basis function placement greatly simplifies both the training and recall operations of the network.

The CMAC algorithm was first proposed as a computational model of adaptive control performed by the cerebellum [2]. Since then it has become popular in many engineering applications primarily due to the efficiency of its implementation and its robust convergence properties [23][60][70][73][76][77][83][92]. The remainder of this section will provide a detailed description of the CMAC algorithm. This presentation is intended to serve as a foundation for use of the CMAC for active disturbance cancellation in later chapters. In addition, many refinements to the basic CMAC network have been proposed, and those which are of particular significance to active disturbance cancellation will be addressed in this section.

Consider a CMAC network with N inputs and a single scalar output. The mapping performed by this network is described as $f_{cmac} : \mathbb{R}^N \to \mathbb{R}$. In order to enable the *a priori* placement of basis functions, the input space must be bounded in each dimension. In addition, the CMAC algorithm utilizes a quantized representation of the input vector. Since the CMAC inputs are bounded and quantized, there is only a finite set of possible inputs to the CMAC. The set of all such possible inputs is often referred to as the input lattice [3] and the actual map represented by the CMAC can be expressed as $f_{cmac} : \mathbb{N}^N \to \mathbb{R}$. As a result, the most general mapping that can be represented by the CMAC is the production of an arbitrary output for each cell in the input lattice. This is exactly the capability of a look-up table defined on the same lattice.

Consider the input vector $\mathbf{x}$ for an N input CMAC as given in (2.2). It is assumed that each component of the input vector is bounded such that $L < x_i < U$. In practical implementations, each input would have unique bounds but a common bound is assumed here to simplify the notation.

$$\mathbf{x} = \begin{bmatrix} x_0 & x_1 & \cdots & x_{N-1} \end{bmatrix}^T \in \mathbb{R}^N \tag{2.2}$$

The CMAC operates on a quantized version of the input space. In the present description, it is assumed that the quantization is uniform and is identical for all network inputs. As a result, the quantization can be completely described by a single quantization width parameter, Δ. In a practical applications, it is often advantageous to implement the coarse coding in a fashion which utilizes quantization levels which are placed optimally across the range of each input. For example, depending on the nature of the input signals, the quantization levels could be selected to provide logarithmic spacing or levels specifically placed to take advantage of *a priori* knowledge as to the nature of an input variable.

Under the present assumption of uniform input quantization, the coarse-coded input vector $\mathbf{q}$ is given by equation (2.3) where $\lfloor\cdot\rfloor$ represents the floor operation. Each element of the quantized input vector is bounded as shown by (2.4).

$$\mathbf{q} = \begin{bmatrix} q_0 & q_1 & \cdots & q_{N-1} \end{bmatrix} = \begin{bmatrix} \left\lfloor \frac{x_0 - L}{\Delta} \right\rfloor & \left\lfloor \frac{x_1 - L}{\Delta} \right\rfloor & \cdots & \left\lfloor \frac{x_{N-1} - L}{\Delta} \right\rfloor \end{bmatrix} \tag{2.3}$$

$$0 \le q_i \le \left\lfloor \frac{U - L}{\Delta} \right\rfloor \tag{2.4}$$

Computation of the CMAC output corresponding to a given input vector begins with a determination of the set of basis functions which cover the active lattice cell. Once this set of basis functions is determined, the associated weighting factors are summed to produce the network output. In the CMAC network, the placement of basis functions is controlled by the integer generalization parameter, $\beta \ge 1$. The generalization parameter directly determines the area covered by a single basis function which in turn affects the degree of extrapolation in the resultant function approximation. Generally, larger values of the generalization parameter are appropriate for functions which are more slowly varying across the input space. The generalization is a fixed feature and is uniform across the entire input space. As a result, it must be selected to be appropriate for the area requiring the finest generalization which can result in somewhat suboptimal performance elsewhere.

The arrangement of receptive fields in the CMAC network is conveniently viewed as the combination of β conceptual overlays of the input space. Each overlay consists of a subset of the entire set of basis functions. The collection of basis functions within each overlay provides complete and non-overlapping coverage of the input space. Therefore, any input vector will cause a single basis function from each overlay to be active and as a result, each input to the network will result in the activation of exactly β basis functions. Each basis function has a width of β lattice cells in each dimension and has a constant output of value 1 for any input within its boundaries. Thus, for a two input network the basis functions will have square regions of activation within the two dimensional input space. For a three input network each basis function will be active in a cube of the input space with dimension $\beta \times \beta \times \beta$. In higher dimensions each basis function will be a hypercube with dimension β along each axis. Within each overlay, the basis functions are stacked adjacent to one another across the entire input space. Additionally, the corners of the basis functions are offset along the hyperdiagonal in each consecutive overlay in order to produce overlap of basis functions between the different overlays.

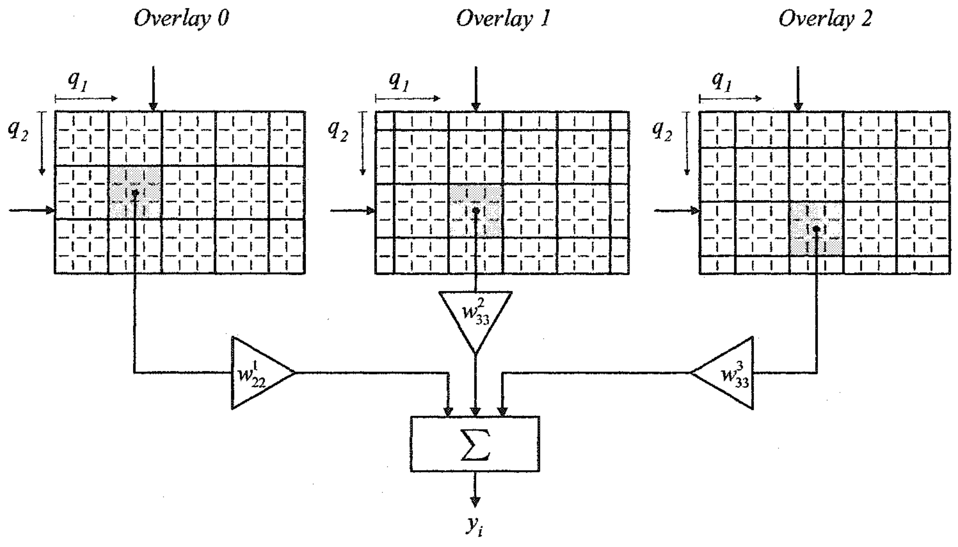


Figure 2.3 – The CMAC output calculation for a two input CMAC with $\beta = 3$.

In order to provide a qualitative understanding of the receptive field placement, consider the example of a two input CMAC with generalization width of $\beta = 3$ as shown in figure 2.3. In this case, there are three overlays and each basis function covers a square region of the input space with dimensions $\beta \times \beta = 3 \times 3$ lattice cells. The boundaries of the basis functions are shown by the solid lines while the dashed lines represent the boundaries of the lattice cells. Any given input vector for this network, $\mathbf{x} = \begin{bmatrix} x_2 & x_1 \end{bmatrix}^T$, is quantized to produce the vector $\mathbf{q} = \begin{bmatrix} q_2 & q_1 \end{bmatrix}^T$. The quantized input vector uniquely references a single cell on the input lattice as indicated by the arrows in figure 2.3. The active lattice cell is covered by a single basis function on each overlay and therefore a total of three basis functions are selected. These active basis functions are shaded gray in figure 2.3. Each basis function has an associated weighting factor and the CMAC output is calculated by summing the weighting factors associated with the active basis functions. The weight w_{ij}^p is associated with the basis function on overlay p at indices i, j.

For the CMAC of figure 2.3, each overlay can be represented by a two dimensional matrix of weights. In general, the weight matrix associated with overlay p can be represented by the function $W^p(\mathbf{d}_p)$ where $\mathbf{d}_p$ is a vector containing the indices of the selected basis function for overlay p. That is, the function $W^p(\mathbf{d}_p)$ references the weights associated with the vector of indices, $\mathbf{d}_p$. In the CMAC algorithm, these indices are computed from the quantized input vector by equation (2.5). Notice that there will be as many indices as there are input dimensions, N, since each overlay is N dimensional. Computation of the indices can be efficiently performed using fixed point hardware implementations since an integer division intrinsically provides the floor functionality as well. Additionally, if the quantization is restricted to a power of two, then the indices can be computed using only a binary right shift operation.

$$\mathbf{d}_p = \begin{bmatrix} \left\lfloor \frac{q_0 - p}{\Delta} \right\rfloor & \left\lfloor \frac{q_1 - p}{\Delta} \right\rfloor & \cdots & \left\lfloor \frac{q_{N-1} - p}{\Delta} \right\rfloor \end{bmatrix} \quad \forall\ p \in \{0, 1, \cdots, \beta - 1\} \tag{2.5}$$

The described algorithm is known as the binary CMAC since the output of each basis function is binary valued. The recall algorithm is summarized in figure 2.4 for the general case of an N dimensional input and a single real valued output. A vector output CMAC can be realized by utilizing multiple single output CMAC networks in parallel on the same input vector. However, the overall computational burden can be reduced through realization that the selection of the active basis functions is common to all of the parallel networks and therefore needs to be performed only once.

Recall Operation in the Binary CMAC: Consider a CMAC network with a real input vector $\mathbf{x} = \begin{bmatrix} x_0 & x_1 & \cdots & x_{N-1} \end{bmatrix}$ and a single real valued output, $y = f_{cmac}(\mathbf{x})$. Let the generalization parameter be $\beta \in \mathbb{N}$ and $\beta \geq 1$. Let input x_i be bounded such that $L_i \leq x_i \leq U_i$. Additionally let the quantization interval of input i be Δ_i. Let $W^p(\cdot)$ represent an N dimensional array of weights for each overlay $p \in \{0, 1, \cdots, \beta - 1\}$. With these definitions, the following equations describe the CMAC recall operation.

$$\mathbf{q} = \begin{bmatrix} q_0 & q_1 & \cdots & q_{N-1} \end{bmatrix} = \begin{bmatrix} \left\lfloor \dfrac{x_0 - L_o}{\Delta_0} \right\rfloor & \left\lfloor \dfrac{x_1 - L_1}{\Delta_1} \right\rfloor & \cdots & \left\lfloor \dfrac{x_{N-1} - L_{N-1}}{\Delta_{N-1}} \right\rfloor \end{bmatrix}$$

$$\mathbf{d}^p = \begin{bmatrix} \left\lfloor \dfrac{q_0 - p}{\beta} \right\rfloor & \left\lfloor \dfrac{q_1 - p}{\beta} \right\rfloor & \cdots & \left\lfloor \dfrac{q_{N-1} - p}{\beta} \right\rfloor \end{bmatrix} \quad \forall\, p \in \{0, 1, \cdots, \beta - 1\}$$

$$y = \sum_{p=0}^{B-1} W^p(\mathbf{d}^p)$$

Figure 2.4 – The binary CMAC recall operation.

The training or weight adaptation in the CMAC network provides a means of adjusting the values contained in the p weight matrices $W^p(\cdot)$ in a manner which minimizes the approximation error of the CMAC across some training data set. Usually, the weight adaptation is performed concurrently with recall operations on a sample by sample basis. For each output, the active weights are adjusted in a manner which reduces the squared output error associated with the present CMAC input. For example, suppose the current input vector is $\mathbf{x}_0$ and the desired output of the CMAC for input $\mathbf{x}_0$ is d. The resultant error in the CMAC approximation for this input is defined as $e = d - y$, where y is the CMAC output corresponding to input x_0. There are β weights, $w^0, w^1, \cdots, w^{\beta-1}$ which were used in computing the CMAC output. Therefore, to reduce the error associated with this input vector, each of these weights is adjusted in the direction of the negative gradient of the squared error. The resulting increment to be added to weight w^i is given by (2.6) where η represents a learning gain parameter.

$$\Delta w^i = -\eta \frac{\partial e^2}{\partial w^i} = 2\eta e \frac{\partial y}{\partial w^i} = 2\eta e \frac{\partial}{\partial w^i} \sum_{p=0}^{\beta-1} w^p = 2\eta e \quad i \in \{0,1,\cdots,\beta-1\} \tag{2.6}$$

Thus, each of the selected weights is simply updated with an increment proportional to the error in the output. Notice however that the contribution of each weight to the output is dependent upon the generalization parameter. That is, with a larger generalization, a fixed increment across all selected weights will have a greater influence on the output than the same increment with a smaller generalization. As a result, the effective learning gain in (2.6) varies with the generalization parameter. In order to make the learning gain parameter independent of the generalization width, the weight update is usually normalized by β.

Weight Update in the Binary CMAC: Consider a CMAC network with output $y = f_{cmac}(\mathrm{x})$ and generalization parameter $\beta \in \mathbb{N}$ and $\beta \geq 1$. Let $W^p(\cdot)$ represent the N dimensional array of weights for each overlay $p \in \{0,1,\cdots,\beta-1\}$ and let d be the desired or target output for the current input vector. Let $\mathrm{d_p}$ represent the pointers associated with the presented input vector. Then the following equation describes the CMAC weight update operation.

$$\text{for } p = 0,1,\cdots,\beta-1$$

$$W^p(\mathrm{d_p})\big|_{post-update} = W^p(\mathrm{d_p})\big|_{pre-update} + \frac{\eta}{\beta}(d-y) = W^p(\mathrm{d_p})\big|_{pre-update} + \frac{\eta}{\beta}e$$

Figure 2.5 – The binary CMAC weight update.

The CMAC weight update algorithm is given in figure 2.5. If the weight updates are performed concurrently with the CMAC recall then the only additional computation associated with the adaptation is the scaling of the output error by the learning gain and a single addition per weight. Notice that the adaptation algorithm is intrinsically local in nature since only weights within a β lattice cell radius are impacted by any weight update.

The selection of the active set of weights is a highly nonlinear operation. However, once the active weights are defined, the recall and training operations are entirely linear. This fact leads to analytical expressions for the CMAC recall and weight update operations which are often useful in the derivation of theoretical results. In this representation, all of the weights associated with the CMAC are represented by a single weight vector $\mathbf{w}$. A weight selection vector, $\mathbf{a}(\mathbf{x})$, is defined as a function of the current input vector, $\mathbf{x}$. The weight selection vector is binary valued and its length is equal to the length of the weight vector, $\mathbf{w}$. All elements of $\mathbf{a}(\mathbf{x})$ are zero valued except for β entries of value one which are in locations corresponding to the active weights. As a result, the CMAC recall is simply represented as the inner product of the weight selection vector and the weight vector as given in (2.7). In this formulation, the entire nonlinear mapping operation performed by the CMAC is contained within the weight selection vector.

$$y = \mathbf{a}(\mathbf{x})^T \mathbf{w} \tag{2.7}$$

With this convention, the weight update equation can be expressed as given in (2.8) where e represents the output error. It is also possible to derive batch mode recall and weight update expressions in the case where training and recall are performed simultaneously across an entire training data set. The implications of this formulation are explored further in section 2.5.

$$\mathbf{w}\Big|_{post-update} = \mathbf{w}\Big|_{pre-update} + \frac{\eta}{\beta}\mathbf{a}(\mathbf{x})e \tag{2.8}$$

The computational efficiency of the CMAC algorithm results directly from the uniform placement of the basis functions across the entire input space. However, this architecture also results in the presence of basis functions and their associated weights in regions of the input space which are never utilized in the mapping. Additionally, this feature of the CMAC leads to an enormous quantity of weighting factors especially in the case of large input dimensions. The dimensions of the weight matrices depend on the number of inputs, the range of the inputs, the quantization levels for the inputs, and the generalization parameter. The total number of required weights can be expressed as given in equation (2.9). Assuming the same number of quantized levels per input, this can be simplified to the expression given in (2.10).

$$N_w = \beta \prod_{i=0}^{N-1} \left\{ \left\lfloor \frac{1}{\beta} \left\lfloor \frac{U_i - L_i}{\Delta_i} \right\rfloor \right\rfloor + 1 \right\} \tag{2.9}$$

$$N_w \cong \frac{1}{\beta^{N-1}} (\text{\# of quantization levels per input})^{N-1} \tag{2.10}$$

From the result in (2.10) it is evident that the number of required weights decreases as the generalization width increases. Additionally, the number of weights required grows drastically with increasing input dimensions and increased quantization levels per input. To illustrate the rapid growth in the number of required weights with increasing input dimension, consider the results presented in figure 2.6. This illustration depicts the number of weights required for the case of 8 bit (256 level) inputs and various input dimensions, N. For a single input CMAC, the generalization width has no impact on the number of required weights. However, at larger input dimensions, increases in the generalization parameter leads to a significant reduction in the number of required weights. However, regardless of the value of the generalization parameter, the memory space required in high dimensions can quickly become unmanageable. For inputs with finer quantization, the problem is exacerbated further. In addition to the large memory space required for storage of the weights, this also has implications for most real-time hardware in that larger memories must often be implemented in slower off-chip memories.

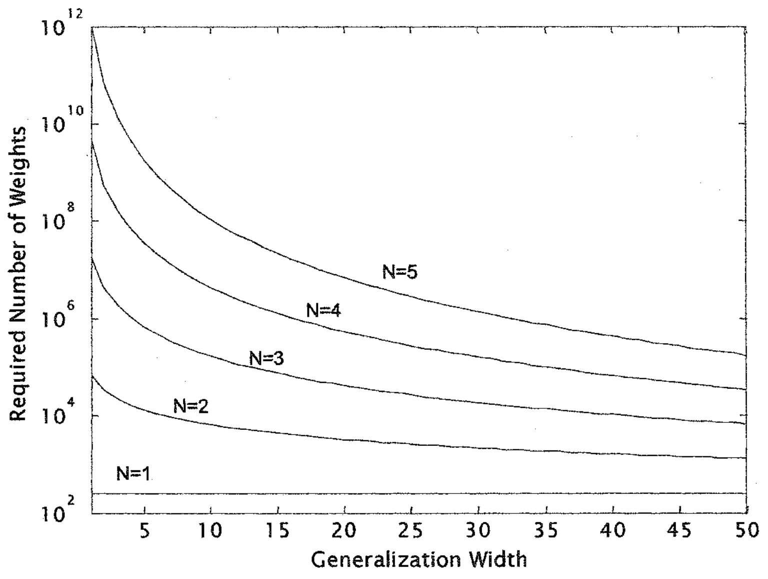


Figure 2.6 – Required CMAC weights versus input dimension and generalization.

In many applications, the volume of the input space over which the function approximation must be valid is a small fraction of the total input space. As a result, many weights in the CMAC representation of such functions are never referenced. In such cases, it is possible to reduce the memory requirements of the CMAC algorithm through the use of sparse memory techniques. The basic premise behind such methods is that a large, sparsely utilized memory can be adequately contained within a much smaller, more densely populated memory. One manner in which this can be accomplished is by mapping the memory space addressed by the CMAC into a much smaller physical memory space via a hashing function. The hashing function is time invariant function which provides a mapping such that consecutive addresses addressed in the virtual memory space are distributed in a random, uniform manner across the smaller physical memory space. It is assumed that the physical memory is chosen to be much larger than that required by the exercised portion of the input space and therefore most addressed memories will be stored and recalled without error. The occasional incidents in which two different virtual addresses map to the same physical address are referred to as a hash collisions and will result in a corruption of the CMAC output. There are several means by which the random hashing function can be implemented [79][95]. The CMAC implementations presented in later sections utilize either a fully implemented memory space with no hashing or hashing via a pseudorandom linear feedback shift register. There are also many lossless hashing algorithms which have been developed to eliminate or reduce the possibility of hash collisions. However, such methods also impose greater computational overhead on the algorithm.

There are many refinements which have been introduced to improve on different shortcomings of the standard binary CMAC algorithm. One important enhancement is an improved receptive field placement strategy [3][16][79]. In the original CMAC algorithm, the receptive fields of the β overlays are offset along the hyper-diagonal of the input space. This receptive field placement has the important property that for a single lattice cell movement in any direction, exactly one weight will be dropped and one weight will be added to the active weight set. This property is known as the uniform projection principle [16]. This condition also ensures that there are no common edges of receptive fields between overlays. For an input dimension of two or greater and generalization of three or greater there are other possible offsets between the basis function layouts on the different overlays which still obey the uniform projection principle. Given this extra degree of flexibility, it is necessary to choose some criteria for deciding upon the optimal overlay placement from all those which meet the uniform projection principle. To obtain the most spatially uniform generalization, the optimal placement strategy is that which produces the most even distribution of basis functions across the input space. The original diagonal displacement strategy is clearly a suboptimal choice in this regard. For example, figure 2.7a depicts the total area of the input space affected by a single training point denoted by the black square. This illustration is for the case of two input dimensions and a generalization width of ten. As a result of the standard receptive field placement, the selected basis functions are located along the diagonal of the input space thereby creating a spatially non-uniform update in the input space. Different methods for determining the optimal overlay placement have been presented [3][16] and are based on selecting the receptive field overlay offsets which maximize the average distance between basis function centers. In the case of the previous example, the optimal overlay placement found through these methods results in the activation pattern of figure 2.7b. Notice that the total impacted area is much more uniform than for the case of diagonal receptive field placement. This refinement in the CMAC algorithm comes at almost zero additional computation. The only modification is in the computation of the weight selection vectors as given in (2.11).

$$\mathbf{d}^p = \left[\left\lfloor \frac{q_0 - o_0^{N\beta}}{\beta} \right\rfloor \; \left\lfloor \frac{q_1 - o_1^{N\beta}}{\beta} \right\rfloor \; \cdots \; \left\lfloor \frac{q_{N-1} - o_{N-1}^{N\beta}}{\beta} \right\rfloor \right] \tag{2.11}$$

In this equation $o_i^{N\beta}$ represents the ith element of the optimal displacement vector. The superscript denotes the dependence of the optimal displacement vector on the input dimension and the generalization parameter. Tables of the optimal displacement vector for various values of the parameters exist in the literature [16].

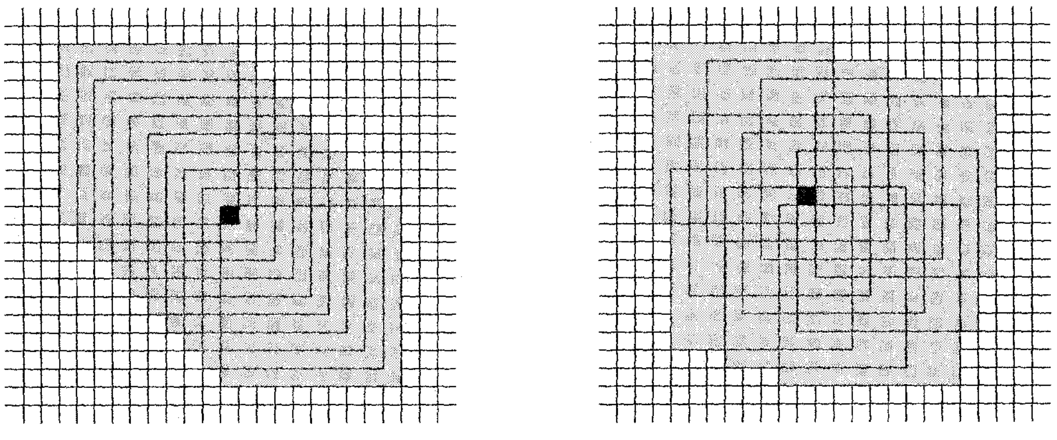

(a) Albus Receptive Field Placement
(b) Optimal Receptive Field Placement

Figure 2.7 – Influence of a single training point for Albus (a) and optimal (b) receptive field placements.

In the standard CMAC weight adaptation, the same weight increment is uniformly applied to the entire set of active weights. Although this is a computationally efficient solution, this weight update often leads to a wide variation in nearby weight values even though the output error is reduced to zero. This large variability in nearby weight values tends to reduce the generalization performance of the CMAC. Additionally, such weight variations reduce the reliability of partial derivative information from a trained CMAC model. In almost all applications, the standard CMAC algorithm is an over-parameterized function approximation. That is, there are more degrees of freedom than required to implement the desired mapping. As a result, the optimal weight vector solution is non-unique and there are many different sets of weights which will be able to reproduce the given set of training data with zero error. It is possible to utilize the extra degrees of freedom afforded by this flexibility in the weight values to impose constraints which minimize the differences between nearby weights. Such methods are known as weight smoothing algorithms. For the case of batch mode CMAC training a weight smoothing algorithm has been presented which provides a solution for the optimally smooth weight vector via a linear optimization over the entire training data set [21]. This approach has also been extended for use in on-line training [87]. An independent heuristic approach to weight smoothing has also been presented [95]. In this method, all of the addressed weights are adjusted by a small increment toward the mean value of the selected weights as given by (2.12). This additional term in the weight update equation tends to move all of the weights toward the average weight value thereby reducing weight differences. The parameter α controls the extent of the weight smoothing. This approach has the advantage that it is a very modest extension of the existing CMAC update rule.

$$W^p\big|_{smoothed} = (1-\alpha)W^p + \frac{\alpha}{\beta}\sum_{i=0}^{\beta-1} W^i \tag{2.12}$$

There are numerous additional refinements to the CMAC algorithm which have been developed. One of particular significance in many applications is the replacement of the binary receptive field with a continuous valued basis function in order to produce a continuous valued output [35][64]. Adaptive coding algorithms have been developed which enable CMAC input quantization to be adjusted to be finer in regions of the input space where greater resolution is required [34]. In addition, there have been many papers reporting on implementation aspects of the CMAC network [51][53][74]. Finally, there are many theoretical results regarding CMAC representational capabilities and convergence of the adaptation process [14][15][17][23][25][42][49][67][86][95][108].

In concluding this introduction to the CMAC algorithm, it is important to consider briefly the representational capabilities of the CMAC function approximation. Fundamentally, the quantized input map of the CMAC results in a different class of function approximation capabilities in comparison to those networks which utilize a continuous input space. Specifically, the class of mappings which can be represented by a CMAC network is necessarily a subset of a multivariate lookup table defined on the same input lattice. For a single input dimension, it has been shown that the CMAC is a universal approximator in this context [16]. That is, the CMAC can be trained to provide an arbitrary response at each input lattice cell. This capability is enabled by the fact that there are as many free parameters as there are input cells in the case of a single input dimension. However, for a multiple input CMAC, there are fewer free parameters then there are input lattice cells if the generalization parameter is greater than one. As a result, the CMAC is not able to represent an arbitrary multivariate lookup table with zero error. In fact, it has been shown that the only class of functions which can be represented with zero error are those which can be expressed as the sum of univariate lookup tables [16]. However, this result is somewhat pessimistic due to the fact that the complexity of the functional mapping increases with the finer generalization CMAC parameters. This does not allow for scaling of the network resources independent of the training data set. For example, in the MLP network, a well known result is that the two layer MLP is a universal function approximator given a sufficient number of hidden layer neurons. An somewhat analogous statement for the CMAC network is that given any discrete mapping $\{q_0, q_1, \cdots\} \rightarrow \{y_0, y_1, \cdots\}$ this relationship can be perfectly represented by a CMAC network given sufficiently small generalization and quantization width.

2.4 Time Delay and Recurrent CMAC Architectures

The feedforward neural network implements a static mapping from a vector $\mathbf{x} \in \mathbb{R}^N$ in the input space to a vector $\mathbf{y} \in \mathbb{R}^M$ in the output space. Assuming the network parameters are fixed, a certain input vector will always produce the same output response independent of the past history of inputs. Such a mapping is termed memoryless or static and is well suited for pattern classification and other time independent tasks. However, most control and identification tasks involve dynamical systems in which the response of the system is based not only on the current input to the system but also on the past history of inputs to the system. Neural network control and identification of such dynamical systems requires an architecture which has the capability to model such history dependent dynamics. In the case of linear discrete time systems, the finite impulse response (FIR) filter achieves this capability by transforming the time domain signal into a vector space comprised of delayed versions of the input signal as shown in figure 2.8b. This transformation to the delayed coordinate space allows the original linear dynamical system to be represented as a static linear mapping from a vector in the delayed coordinate space to the output space. The required dimension of the delayed coordinate vector depends on the nature of the system being represented, and it is possible for the FIR filter to model any linear dynamical system if the input dimension is allowed to be arbitrarily large.

Given the ability of a feedforward neural network to implement a multidimensional static mapping, it is possible to use it to replace the linear mapping of the FIR filter as shown in figure 2.8a. The resultant structure is known as the time delay neural network or tapped delay line neural network [90]. Given that the time delay neural network is a direct extension of the FIR filter, it is expected that its minimal capability is the emulation of a linear dynamical system. This will be the case as long as the neural network is capable of representing a linear mapping and the number of delays is sufficient for the system being modeled.

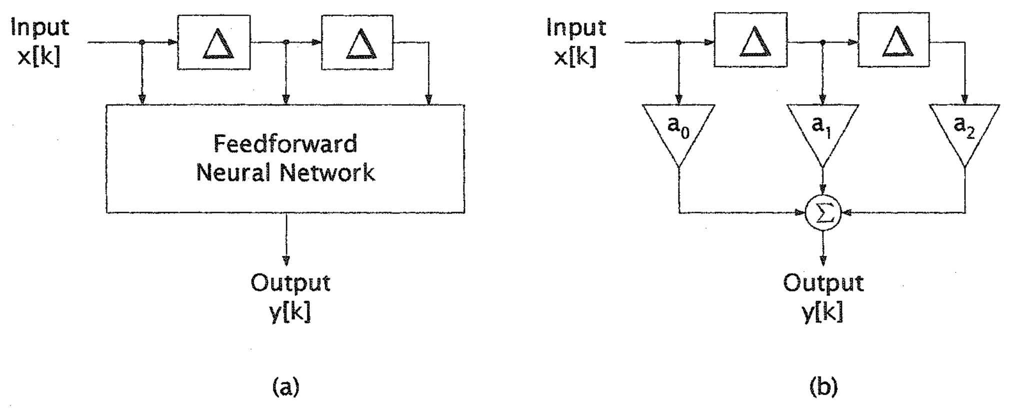


Figure 2.8 – Time Delay network in comparison to an FIR filter.

The primary advantage of the time delay neural network is that in addition to representing linear dynamical systems, this architecture also has the capability to model nonlinear, history dependent dynamics. Theoretical support for this capability is provided by nonlinear systems theory. In the study of nonlinear dynamical systems, it is commonly the case that the only observable information from some nonlinear process is a single time varying scalar function that is related in an unknown fashion to the state of the process. In such cases, a state space representation of the system can be obtained via a state space reconstruction which preserves the dynamics of the original system. Such a reconstruction is not unique. One possibility is to utilize a reconstruction space which is comprised of the signal and its derivatives [82]. Another choice which is generally more appropriate for physical implementations is to utilize the signal and time delayed versions of the signal as created by the tapped delay line in the time delay neural network. It has been shown that if the embedding dimension is chosen sufficiently large then the dynamics of the original system will be preserved in this reconstruction space [82]. The embedding delay is arbitrary in the case of unlimited, noiseless data, but an optimal delay value can be found given an actual data set. This result provides a secure theoretical basis for use of the time delay neural network as a model of nonlinear dynamical systems.

The lack of *a priori* assumptions in the time delay neural network model provides the capacity for modeling a diverse array of nonlinear dynamics. However, this flexibility also reduces the generalization capabilities of the model which will be an important consideration in many disturbance cancellation applications. For example, the FIR filter provides a global interpolation capability due to its assumption of linearity. This allows an FIR model to provide the correct response across the entire input space even if the training is constrained to a small region of the input domain. In contrast, the time delay neural network generally must be trained with data that is quite representative of the relationship which is to be implemented. Additionally, this implies that the time delay neural network generally must be trained over the entire region of the input space where it is expected to later function. To illustrate this point, consider the dynamical system described by (2.13) where $x[k]$ is the discrete time input signal and $y[k]$ is the corresponding output.

$$y[k] = \tfrac{1}{2}x[k] + \tfrac{1}{2}x[k-1] \tag{2.13}$$

The FIR model of this system has exactly two free parameters and the static mapping is a plane defined in the delayed coordinates space as represented by the coarse mesh in figure 2.9. Once these two parameters are chosen, the response is fixed across the entire input space. In contrast, consider the use of a two input Time Delay CMAC network to model the dynamics of (2.13). In this example, the time delay CMAC is trained to accomplish this task using the input signal given in (2.14).

$$x[k] = \tfrac{1}{8}\sin(0.083\pi k) \tag{2.14}$$

This input signal creates a circular closed orbit in the delayed coordinate space. The CMAC weights are adjusted in simulation to minimize the output error until reaching a total error of less than 0.1% throughout an entire period. The CMAC utilized in this comparison has an input quantization of 0.01 and a generalization width of 16. Upon completion of the training, the resultant mapping implemented by the CMAC is shown by the fine mesh surface in figure 2.9. Notice that the time delay CMAC mapping approximates the actual linear mapping only along the trajectory created by the input signal, $x[k]$. Outside of this trajectory, the network output is zero. This illustrates the fact that, unlike linear models, the time delay CMAC must be trained in all regions of the input space where it is expected to operate. For example, the time delay CMAC will produce zero output for an input equal to the training signal scaled by a factor of two. This fundamental attribute of the time delay neural network will have many implications on its use in active disturbance cancellation applications.

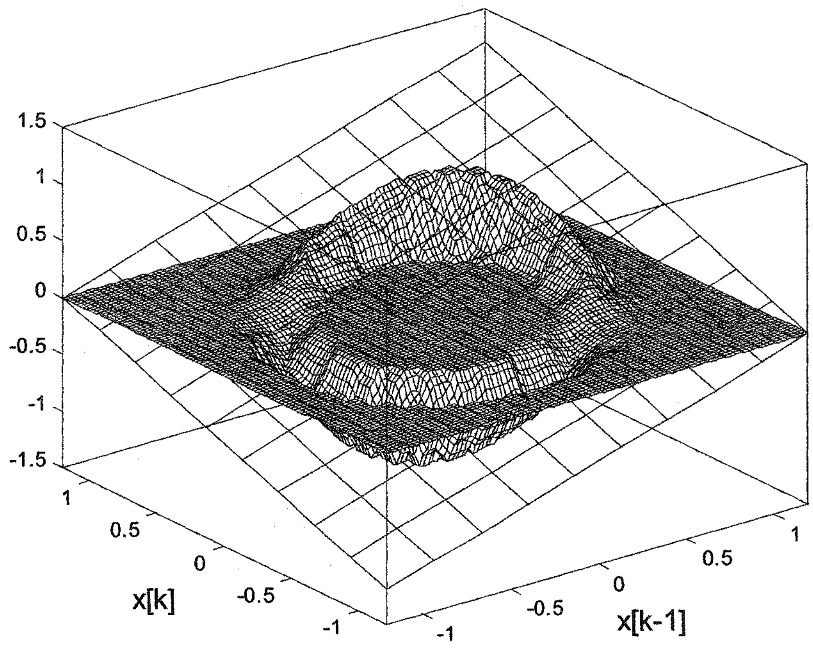


Figure 2.9 – Time delay CMAC map approximation to an FIR filter.

Many active disturbance cancellation systems require the modeling of nonlinear dynamical signal paths. The time delay CMAC can be utilized for this purpose given the conventional system identification arrangement of figure 2.10. In this application, the CMAC weights are continuously adjusted to minimize the difference between the CMAC output and the target signal produced by the reference model. The time delay CMAC is shown with a two dimensional input, but the same technique applies given any length tapped delay line. In practice, the required length of the tapped delay line will be dictated by the characteristics of the system being modeled. In general, systems containing transport delays and resonances will require a longer tapped delay line for accurate representation. As a reasonable guideline, in the case of linear systems, the delay through the tapped delay line must be at least as long as the impulse response of the system. Since the memory storage and computation of the CMAC are directly related to the dimensionality of the CMAC, the implementation of such a model can become unmanageable for the identification of complex signal paths. However, in many active disturbance cancellation applications, there are often situations in which the frequency content of the disturbance is restricted to a narrow band of frequencies. In such cases it is possible to develop a greatly simplified narrowband model that is only designed to model the signal path over the frequency band of interest. In the extreme case where the input signal is known to be a single fixed frequency sinusoid, the signal path model needs to be accurate at only a single frequency. It turns out that, this condition is directly applicable to many narrowband feedforward cancellation systems. Additionally, if the signal path model is continuously adapted, this simplified model will be able to track slow changes in the fundamental frequency. In this manner, the adaptive nature of the reduced order model compensates for its inability to serve as a broadband model.

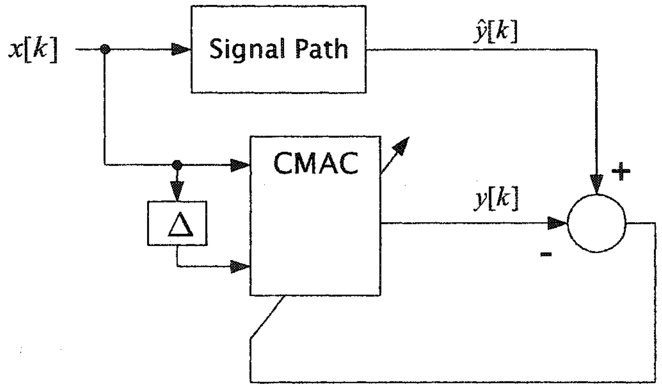


Figure 2.10 – Use of a TDCMAC as a dynamic signal path model.

To demonstrate the capabilities of a reduced order model, consider the corresponding linear case shown in figure 2.11. The reference signal path is defined by a twentieth order FIR filter with coefficients $\hat{b}_0, \hat{b}_1, \cdots, \hat{b}_{19}$. In general, an accurate broadband model of this path would require a model of equal length as shown in figure 2.11a.

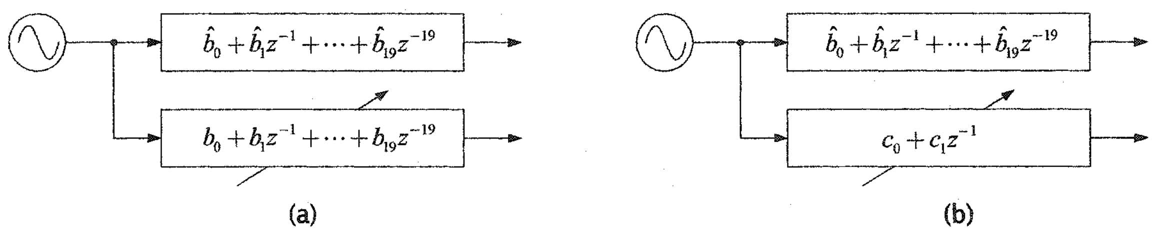

$\hat{b}_0+\hat{b}_1z^{-1}+\cdots+\hat{b}_{19}z^{-19}$
$b_0+b_1z^{-1}+\cdots+b_{19}z^{-19}$
(a)
$\hat{b}_0+\hat{b}_1z^{-1}+\cdots+\hat{b}_{19}z^{-19}$
$c_0+c_1z^{-1}$
(b)

Figure 2.11 – Broadband (a) and Narrowband (b) linear models.

However, suppose the model is only required to emulate the reference signal path for a single sinusoidal input of constant frequency. Then, since the signal path is linear, the output will be a single sinusoid of the same frequency with altered phase and magnitude. This transformation can be accomplished using a two input FIR filter as shown in figure 2.11b with coefficients c_0, c_1. To prove this result, consider the input signal represented as the sinusoid defined by (2.15). The corresponding output signal of any linear system will have the same frequency but generally a unique phase and amplitude as given by (2.16).

$$x[k] = A_1 \sin(wk + \phi_1) \tag{2.15}$$

$$z[k] = A_2 \sin(wk + \phi_2) \tag{2.16}$$

Given a two input FIR model with transfer function $H(z) = c_o + c_1 z^{-1}$, the filtering operation performed on the input signal, $x[k]$, results in the expression in (2.17). If the FIR model is to accurately represent the linear mapping $x[k] \rightarrow z[k]$, then the result of this filtering operation must meet the condition given in (2.18).

$$y[k] = c_0 A_1 \sin(wk + \phi_1) + c_1 A_1 \sin(w(k-1) + \phi_1) \tag{2.17}$$

$$A_2 \sin(wk + \phi_2) = c_0 A_1 \sin(wk + \phi_1) + c_1 A_1 \sin(w(k-1) + \phi_1) \tag{2.18}$$

Thus, if it is possible to solve (2.18) for constant filter coefficients c_0 and c_1 then the resultant transfer function $H(z)$ will perfectly model the higher order system for a sinusoidal input. To that end, the sinusoidal terms in (2.18) can be reduced using the trigonometric angle difference relation which yields the simplified expression of (2.19). Equating coefficients of the time varying $\sin(wk)$ and $\cos(wk)$ terms yields the linear system of equations in (2.20)-(2.21). The solution to this set of equations is given by (2.22) and (2.23).

$$\begin{aligned} A_2\cos\phi_2\sin(wk)+A_2\sin\phi_2\cos(wk) = {} & [c_0A_1\cos\phi_1+c_1A_1\cos(\phi_1-w)]\sin(wk) \\ & +[c_0A_1\sin\phi_1+c_1A_1\sin(\phi_1-w)]\cos(wk) \end{aligned} \tag{2.19}$$

$$c_0A_1\cos\phi_1+c_1A_1\cos(\phi_1-w)=A_2\cos\phi_2 \tag{2.20}$$

$$c_0A_1\sin\phi_1+c_1A_1\sin(\phi_1-w)=A_2\sin\phi_2 \tag{2.21}$$

$$c_0=\frac{A_2[\cos\phi_2-\sin\phi_2\cot(\phi_1-w)]}{A_1[\cos\phi_1-\sin\phi_1\cot(\phi_1-w)]} \tag{2.22}$$

$$c_1=\frac{A_2\sin\phi_2}{A_1\sin(\phi_1-w)}-c_0\frac{\sin\phi_1}{\sin(\phi_1-w)} \tag{2.23}$$

This result indicates that a two input FIR filter is able to model any linear transfer function in the case where the input is comprised of a single sinusoid. Additionally, if the FIR filter is continuously adaptive, this filter can be used as a model of the full linear system for any slowly varying sinusoidal disturbance. A similar, though somewhat more general, result holds for the time delay neural network. In a qualitative sense, the two input time delay neural network has the capability to represent a transformation from a periodic input signal to a periodic output signal of the same period. This includes the important ability to model the relationship between a fundamental disturbance as an input signal and an output signal comprised of the fundamental in combination with its harmonics. There are, however, certain constraints placed on the input and output signals which will be presented in more detail in the following section.

If the transfer function being modeled by a time delay neural network or FIR filter has a long duration impulse response, then the number of taps required in the model can become quite large. In the case of linear systems, a more compact model is possible using an infinite impulse response (IIR) filter as shown in figure 2.12b. The IIR filter utilizes internal feedback to provide memory capabilities which exceed the length of the tapped delay line input. It is possible to extend the IIR filter by replacing the linear mappings with feedforward neural networks as shown in figure 2.12a. The resulting recurrent neural network extends the capabilities of the IIR filter and is therefore well suited to modeling nonlinear systems with significant history dependence. However, online training of these architectures is generally more involved then in the FIR counterparts. Additionally, these architectures present the possibility of internal instability, which is not the case for the FIR architecture.

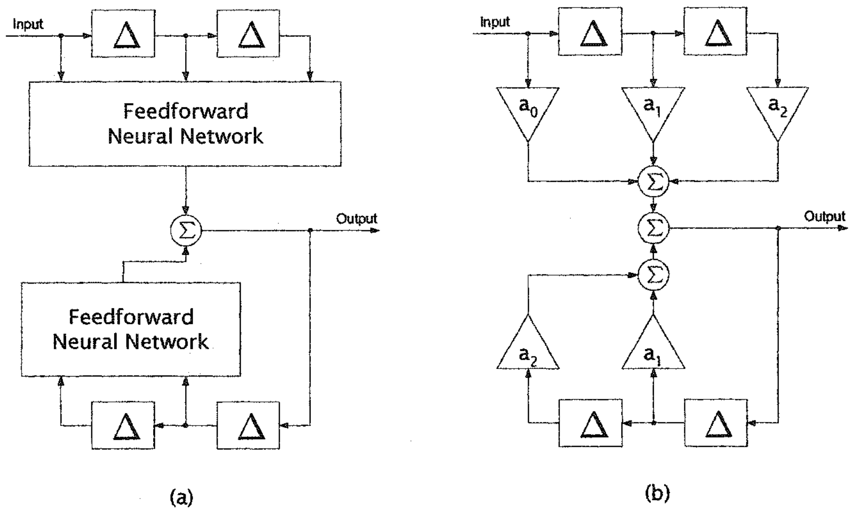


Figure 2.12 – Recurrent neural network and linear IIR models.

In addition to its capability of modeling a dynamical signal path, the IIR filter is able to function as a sinusoidal oscillator in the case where its poles are designed to be on the imaginary axis. Similarly, the recurrent time delay neural network can function as a nonlinear oscillator via the architecture shown in figure 2.13a.

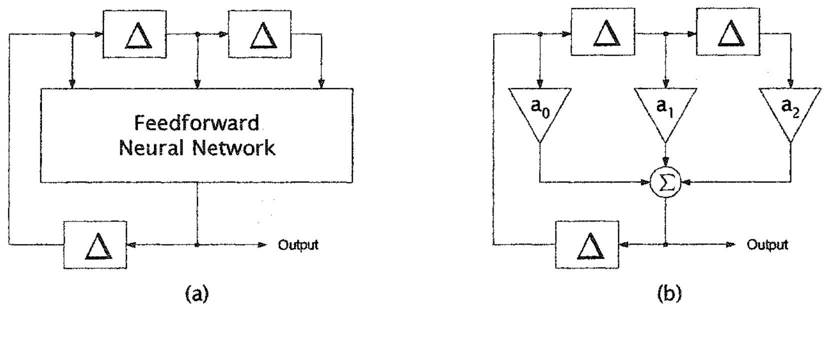


Figure 2.13 – Recurrent neural network and linear IIR oscillators.

2.4 Representational Capabilities of Time Delay Neural Networks

In the previous section, it was shown that the time delay neural network can be viewed as a generalization of a linear FIR filter. As a result, the time delay neural network has the ability to approximate any linear transfer function, and also possesses the ability to represent certain nonlinear dynamical systems. In this section, some basic representational capabilities of the time delay neural network are considered. These results are intended to provide some fundamental guidelines for application of the time delay neural network to disturbance cancellation systems.

The first result given in theorem 2.1 simply formalizes the fact that the time delay CMAC can represent any quantized FIR mapping with zero error. Since the CMAC operates on a quantized input lattice, it cannot implement a real valued FIR filter perfectly. However, the time delay CMAC does possess the capability to represent the FIR mapping with zero error at every lattice point. Since the input quantization levels can be reduced to any required level, this model can be made as accurate as required by the application. Additionally, the effect of quantization on linear filters is well understood as this issue arises in the digital hardware implementation of such filters. This result pertains to an ideal CMAC without any hashing considerations.

Theorem 2.1 – Time Delay CMAC Representation of an FIR Relationship: Let $Q\{\cdot\}$ represent the quantization of a real valued vector into the respective point on the input lattice of an ideal CMAC. Let coefficients $\mathbf{b} = [b_0, b_1, \cdots, b_{M-1}]^T$ define an arbitrary linear FIR filter. Then, an ideal time delay CMAC with M input taps can represent the quantized mapping $Q\{\mathbf{x}\} \to y = \mathbf{b}^T Q\{\mathbf{x}\}$ with zero error.

Proof:

The FIR mapping to be represented by the time delay CMAC can be expressed as in equation (2.24) where $q_i\{\cdot\}$ represents the CMAC quantization of the ith input.

$$y = \sum_{i=0}^{M-1} b_i q_i\{x_i\} = b_0 q_0\{x_0\} + b_1 q_1\{x_1\} + \cdots + b_{M-1} q_{M-1}\{x_{M-1}\} \qquad (2.24)$$

Equation (2.24) shows that the desired mapping is the sum of univariate lookup tables defined on the input lattice. It has been shown that an ideal multivariate CMAC can represent any function which is the sum of univariate lookup tables with zero error [16]. This shows that the mapping $Q\{\mathbf{x}\} \to y = \mathbf{b}^T Q\{\mathbf{x}\}$ can be represented with zero error. Therefore, the time delay CMAC can model the quantized version of any FIR relationship with zero error. $\square$

A two-input time delay neural network has the ability to represent a relationship between a periodic input and a periodic output of the same period, given certain conditions on the nature of the input signal. This capability is of particular interest in the case of narrowband disturbance cancellation since the reference signal in such systems is often in the form of a non-sinusoidal periodic signal. The proof of this result, presented in theorem 2.2, is based only on the assumption that the neural network is able to represent an arbitrary functional mapping, and does not directly address the capabilities of any specific neural network architecture. As a result, the derived results are necessary yet not sufficient conditions in the specific case of a time delay CMAC network. However, in practice this result provides a minimal set of guidelines for assistance in choosing appropriate parameters based on the nature of the input signal.

Theorem 2.2 – Representation of Periodic Signals with Time Delay Neural Networks: Let $f(\cdot)$ be a universal function approximator used to create a two-input time delay network with time delay T_S. Let $s(t)$ be a continuous periodic signal with period $T > 0$ and with a time derivative $\dot{s}(t)$ that is continuous. Let $v(t)$ be a continuous and periodic signal of the same period T. If the parametrically described contour $\{x_1 = s(t), x_2 = s(t + T_S)\}$ is simple (i.e. not self-intersecting) and the time derivatives $\dot{x}_1$ and $\dot{x}_2$ are not simultaneously zero for any $t \in [0,T]$, then the mapping $\left[s(t) \quad s(t-T_S) \right] \rightarrow v(t)$ can be represented by the time delay neural network.

Proof:

Consider the illustrative trajectory $\{x_1 = s(t), x_2 = s(t+T_S)\}$ shown in figure 2.14 and notice that the trajectory does not intersect. Additionally, the trajectory must be a closed orbit given that $\{x_1(t+T), x_2(t+T)\} = \{x_1(t), x_2(t)\}$.

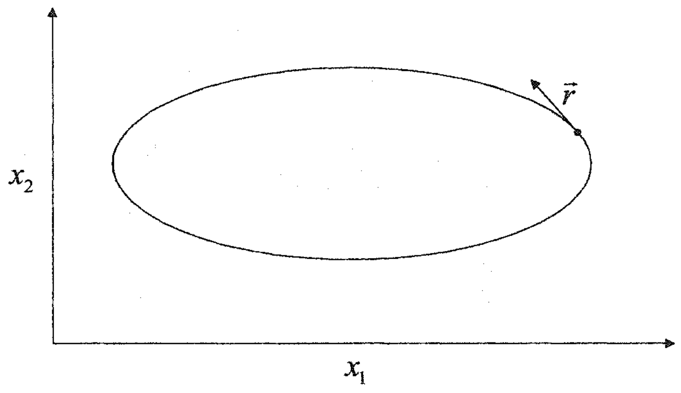


Figure 2.14 – Trajectory in the neural network input space.

The magnitude of the tangential velocity along this trajectory is expressed in equation (2.25). Given that the $\dot{x}_1$ and $\dot{x}_2$ exist and are never zero simultaneously, the magnitude of the tangential velocity must always be greater than zero as given in (2.26).

$$\left| \frac{dr}{dt} \right| = \sqrt{\left(\frac{dx_1}{dt} \right)^2 + \left(\frac{dx_2}{dt} \right)^2} \tag{2.25}$$

$$\left| \frac{dr}{dt} \right| > 0 \tag{2.26}$$

Also, given that the time derivatives $\dot{x}_1$ and $\dot{x}_2$ are continuous, the tangential velocity is a continuous function as well. Since the tangential velocity is continuous and can never equal zero by (2.26), it is therefore impossible for the trajectory to reverse direction. Additionally, the trajectory can not remain at any point for any finite amount of time given

(2.26). Therefore, for any given time t_0, the point $\{x_1(t_0) = s(t_0), x_2(t_0) = s(t_0 - T_S)\}$ is unique on the x_1, x_2 plane within any period. In conclusion, equation (2.27) holds.

$$\{x_1(t+\Delta), x_2(t+\Delta)\} \neq \{x_1(t), x_2(t)\} \quad \forall t \in [0,T) \tag{2.27}$$

Given (2.27), there are no two identical points in the domain of $f(\cdot)$. As a result, since a universal approximator can represent any proper function, $f(\cdot)$ can map to an arbitrary value for any point in the domain. As a result, the two input time delay neural network will be able to represent the mapping $\begin{bmatrix} s(t) & s(t-T_S) \end{bmatrix} \to v(t)$ with zero error. □

The result given in theorem 2.2 is a direct result of the fact that the neural network must implement a single-valued functional relationship. To illustrate the implications of theorem 2.2, two individual cases will be considered. The first case fails to meet the conditions of theorem 2.2 due to the presence of a self-intersecting orbit. The second case fails due to the presence of zero time derivatives in the input and delayed input signals.

Consider the signal $s(t)$ represented in (2.28) which consists of two harmonically related sinusoids. Notice that the signal $s(t)$ is periodic with a period of one. Additionally, $s(t)$ is continuous and has continuous time derivatives. For this example, the delay utilized in the tapped delay line is $T_S = 1.4$.

$$s(t) = \frac{1}{2}\sin(2\pi t) + \frac{1}{5}\sin(4\pi t) \tag{2.28}$$

Figure 2.15 depicts the delay space trajectory $\{x_1 = s(t), x_2 = s(t - T_S)\}$. Notice that the orbit is not simple and therefore the conditions of theorem 2.2 are not met. As a result, theorem 2.2 does not guarantee that an arbitrary periodic function mapping can be performed for the input signal $s(t)$. Qualitatively, this is due to the fact that representation of an arbitrary mapping $\begin{bmatrix} s(t) & s(t-T_S) \end{bmatrix} \to v(t)$ for an arbitrary periodic signal $v(t)$, would require the neural network to produce two different outputs for the same input vector at the point of overlap. Since this can not be accomplished by a single-valued functional mapping, it can not be implemented using a feedforward neural network.

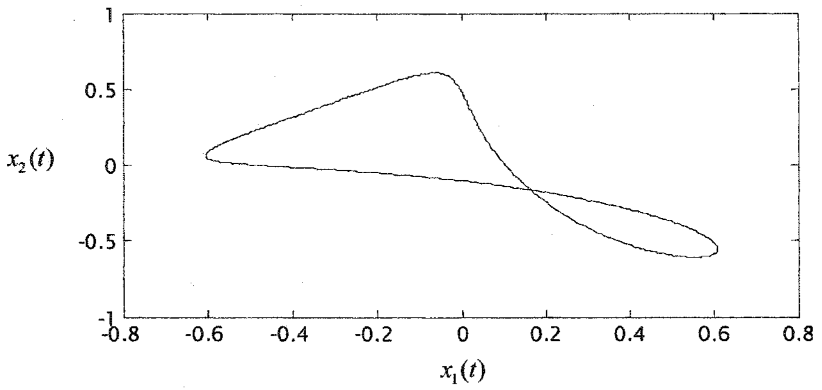


Figure 2.15 – Delay space trajectory showing an intersecting orbit.

The problem in this case can also be observed in the time domain as shown in figure 2.16. In this figure, the tapped delay line outputs are shown as a function of time over a single period of the input signal. At the two times marked by the solid vertical lines, the input vector $\{x_1 = s(t), x_2 = s(t - T_S)\}$ is identical. This corresponds to the overlap point in the delayed coordinate trajectory of figure 2.15. In general, the desired target signal $v(t)$ could be such that the neural network would be required to produce different values at each of these times. For the special case that $v(t)$ has the same value at both of these points, it would be possible to represent the mapping $\begin{bmatrix} s(t) & s(t - T_S) \end{bmatrix} \to v(t)$, however, this is not true in the general case.

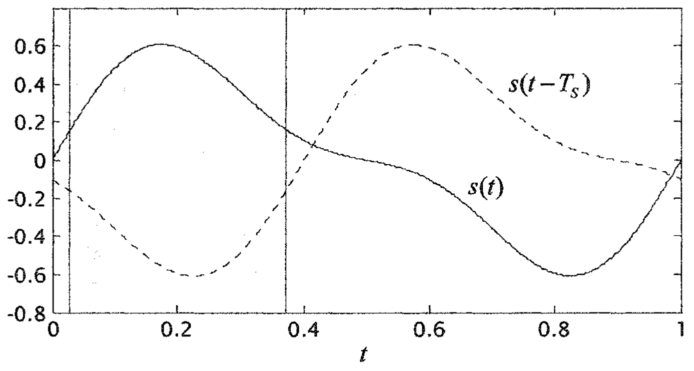


Figure 2.16 – Time domain input signals.

The geometry of the orbit in the embedding space is directly dependent upon the delay utilized in the tapped delay line. Therefore, the conditions of theorem 2.2 can be utilized to select an appropriate choice for the tapped delay line delay. For example, consider the input signal $s(t)$ from the previous example as defined in (2.28) but choosing $T_S = 1.0$. With this delay, the embedding space trajectory is given in figure 2.17 and the orbit is no longer self-intersecting. Therefore, in this case, the time delay neural network can implement an arbitrary periodic function mapping from this input signal to another periodic signal of the same period.

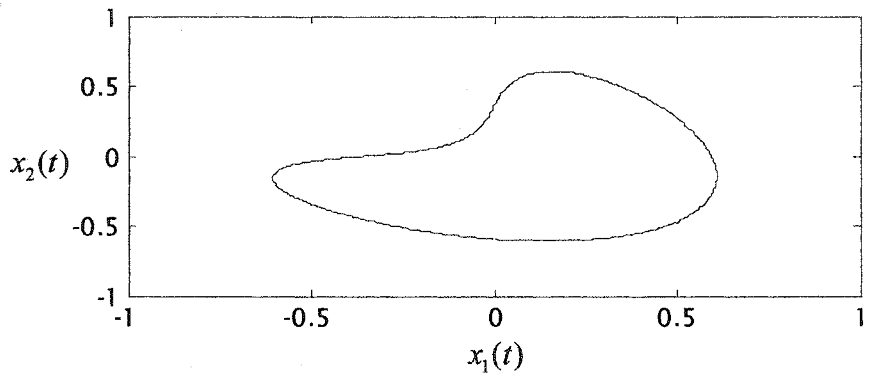


Figure 2.17 – Delay space trajectory for $T_S = 1.0$.

The previous example presented the case of a periodic signal which failed to meet the non-self-interesting orbit property specified by theorem 2.2. The next example presents a case where the failure is due to the presence of parametric time derivatives which are simultaneously zero. The input signal $s(t)$ is defined by the saturated sinusoid given in (2.29). The tapped delay line delay is chosen as $T_S = 0.1$.

$$s(t) = \begin{cases} \frac{1}{2} & \text{if } \sin(2\pi t) > \frac{1}{2} \\ -\frac{1}{2} & \text{if } \sin(2\pi t) < -\frac{1}{2} \\ \sin(2\pi t) & \text{otherwise} \end{cases} \tag{2.29}$$

Notice that the signal $s(t)$ is continuous and periodic with a period of one. Additionally, its projection in the delay space forms a simple orbit as shown in figure 2.18.

However, notice that the time derivative of $s(t)$ is not continuous and additionally, there are periods of time for which $\dot{s}(t) = 0$ and $\dot{s}(t - T_S) = 0$ as is evident from the time domain plots of the signals in figure 2.19. Therefore, this input signal does not meet the requirements of theorem 2.2 and in general, a periodic function mapping for this input function cannot be represented by a two input time delay neural network.

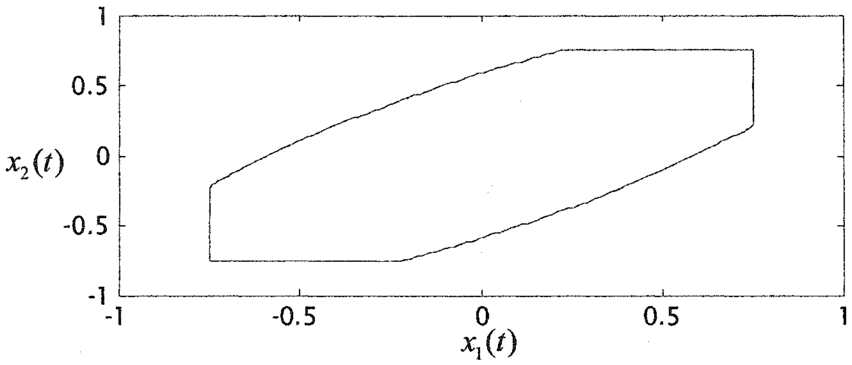


Figure 2.18 – Delay space trajectory for the system of equation (2.29).

The failure in this example occurs because the input space trajectory pauses at certain locations in the limit cycle of figure 2.18. Specifically, over the time intervals when both $\dot{s}(t) = 0$ and $\dot{s}(t - T_S) = 0$, there is no movement along the orbit. Consequently, there are periods of time when the orbit remains at points $(x_2, x_1) = (1,1)$ and $(x_2, x_1) = (-1,-1)$. Thus production of a different value of $v(t)$ within these intervals would violate the requirement that the neural network represents a proper function relationship.

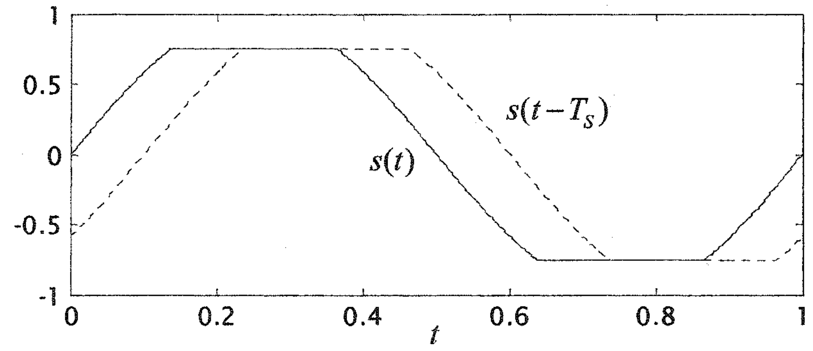


Figure 2.19 – Time domain trajectory of the system given in (2.29).

Another commonly utilized periodic signal which fails for the same reason is the square wave. Notice, however, that it is often possible to utilize a simple transformation to convert the input signal into one which meets the conditions of theorem 2.2. For example, the square wave can be converted into a signal which meets these conditions through the use of an appropriately chosen low pass filter.

In narrowband feedforward disturbance cancellation, the fundamental disturbance source is often represented by a single sinusoid. Due to the presence of nonlinearities in the system, the measured disturbance is often comprised by a linearly transformed version of the original tone along with components related to its harmonics. As a result, it is commonly necessary to represent such a relationship in the compensator of such a system. In such cases, the input signal can be represented by (2.30).

$$s(t) = A\sin(wt) \tag{2.30}$$

The resultant output signal is comprised of Q sinusoidal components each possessing an independent amplitude A_i and phase ϕ_i, as given by (2.31)

$$v(t) = \sum_{l=1}^{Q} A_i \sin(lwt + \phi_l) \tag{2.31}$$

With appropriate choice of the tapped delay line delay, the signal $s(t)$ meets the requirements of theorem 2.2 since the orbit is a closed ellipse. Additionally, $v(t)$ is periodic and has the same period as $s(t)$. Therefore, by theorem 2.2, this mapping can be represented by an ideal two input time delay neural network. This capability is of significant importance in the case of narrowband disturbance cancellation using a time delay CMAC compensator.

2.5 A Linear System Analysis of the Single Dimension CMAC

Most applications of the CMAC neural network in disturbance cancellation utilize a sequential training technique whereby the weights are updated upon presentation of each individual exemplar pair. An alternative training method is batch mode adaptation in which the weight changes associated with the entire training data set are made simultaneously. In this section, the convergence of the batch mode update is considered for a single dimension CMAC and a bound on the maximum stable learning gain is established. This approach provides additional insight into the nature of the CMAC representation. Furthermore, it is shown that the sequential update converges to the same result as the batch mode update given sufficient small learning gain.

For any given input vector, the CMAC output is a linear function of the active set of weights. Thus, the linear portion of the output computation and weight adaptation can be analyzed separately from the nonlinear mapping associated with the weight selection. Consider the simplified CMAC representation of figure 2.20 which depicts the simultaneous computation of all possible network outputs for a single dimension CMAC with seven weights, five outputs, and a generalization width of three. For the results presented in this section, it is assumed that sufficient memory resources are available so that hash coding is not required.

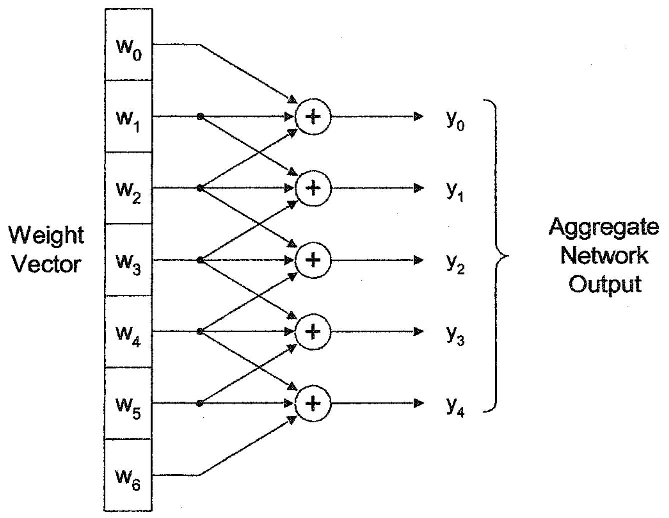


Figure 2.20 – A simplified representation of batch mode CMAC recall.

The simultaneous output calculation of figure 2.22 is a linear operation and can be represented concisely by the matrix equation (2.32) where $y[k]$ represents a column vector of outputs at time k and $w[k]$ represents the weight vector at time k. The matrix A represents the aggregate weight selection matrix. Each row of A represents the weight selection vector associated with a single input. For the CMAC of figure 2.20, the $\mathbf{A}$ matrix is given by equation (2.33).

$$\mathbf{y}[k] = \mathbf{A}\mathbf{w}[k] \tag{2.32}$$

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{2.33}$$

The batch mode weight update can also be written as a matrix equation as given in equation (2.34) where α represents the learning gain, and $\hat{\mathbf{y}}$ represents the target output vector.

$$\mathbf{w}[k+1] = \mathbf{w}[k] + \alpha\mathbf{A}^T\left(\hat{\mathbf{y}} - \mathbf{y}[k]\right) \tag{2.34}$$

Combination of equations (2.32) and (2.34) allows for the output vector to be expressed as a linear difference equation as given in equation (2.35).

$$\mathbf{y}[k+1] = \left[\mathbf{I} - \alpha\mathbf{A}\mathbf{A}^T\right]\mathbf{y}[k] + \alpha\mathbf{A}\mathbf{A}^T\hat{\mathbf{y}} \tag{2.35}$$

From (2.35), the stability of batch mode learning is dictated solely by the eigenvalues of the system matrix $\left[\mathbf{I} - \alpha\mathbf{A}\mathbf{A}^T\right]$. If the eigenvalues of the system matrix are all stable then the steady state solution is found by setting $\mathbf{y}_{ss} = \mathbf{y}[k+1] = \mathbf{y}[k]$ in (2.34). Assuming that the inverse $\left[\mathbf{A}\mathbf{A}^T\right]^{-1}$ exists, this results in equation (2.36) which shows that the steady state output vector is equal to the target vector. Therefore, the batch mode learning will converge to zero error as long as all eigenvalues of the system matrix are stable.

$$\begin{aligned} \mathbf{y}_{ss} &= \left[\mathbf{I} - \alpha\mathbf{A}\mathbf{A}^T\right]\mathbf{y}_{ss} + \alpha\mathbf{A}\mathbf{A}^T\hat{\mathbf{y}} \\ \Rightarrow \mathbf{y}_{ss} &= \left[\mathbf{A}\mathbf{A}^T\right]^{-1}\mathbf{A}\mathbf{A}^T\hat{\mathbf{y}} = \hat{\mathbf{y}} \end{aligned} \tag{2.36}$$

From (2.35), it is evident that the stability of the learning algorithm depends only on the learning gain and the nature of the weight selection matrix. To illustrate the structure of this matrix, consider its value for the case of figure 2.20 as given in equation (2.37).

$$\left[\mathbf{I}-\alpha\mathbf{A}\mathbf{A}^T\right]=\begin{bmatrix}(1-3\alpha) & -2\alpha & -\alpha & 0 & 0\\ -2\alpha & (1-3\alpha) & -2\alpha & -\alpha & 0\\ -\alpha & -2\alpha & (1-3\alpha) & -2\alpha & -\alpha\\ 0 & -\alpha & -2\alpha & (1-3\alpha) & -2\alpha\\ 0 & 0 & -\alpha & -2\alpha & (1-3\alpha)\end{bmatrix} \tag{2.37}$$

Notice that the system matrix of (2.37) is in Toeplitz form. This will be the case for any single dimensional CMAC independent of the network size, generalization width, and learning gain. From the fact that the matrix is symmetric, its eigenvalues are unconditionally real-valued. Before analyzing the stability of the weight update equation in general, consider the special case of a generalization width of one. In this case, the system matrix reduces to that given by equation (2.38). The eigenvalues in this case are simply equal to the diagonal entries of the matrix.

$$\left[\mathbf{I}-\alpha\mathbf{A}\mathbf{A}^T\right]=\begin{bmatrix}(1-\alpha) & 0 & 0 & 0 & 0\\ 0 & (1-\alpha) & 0 & 0 & 0\\ 0 & 0 & (1-\alpha) & 0 & 0\\ 0 & 0 & 0 & (1-\alpha) & 0\\ 0 & 0 & 0 & 0 & (1-\alpha)\end{bmatrix} \tag{2.38}$$

In this case, convergence of the weight update equation is guaranteed as long as the learning gain is chosen according to the familiar bound given in (2.39).

$$0<\alpha<2 \tag{2.39}$$

Derivation of a learning gain bound in the general case of $\beta>1$ is more complicated since the eigenvalues can not be readily calculated directly from the system matrix in an analytical fashion. However, given that the eigenvalues are all real valued, it is sufficient to establish eigenvalue bounds on the real axis alone. To provide insight into the dependence of the eigenvalue locations on the network parameters, the eigenvalues are calculated numerically for several example cases. Figure 2.21 depicts the location of the system matrix eigenvalues as a function of generalization width for a single dimensional CMAC network with 50 weights and a learning gain, α, of 0.005. At a generalization width of one, the system matrix is diagonal and all of the eigenvalues are located at $(1 - \eta) = 0.995$. As the generalization width is increased many of the eigenvalues become more negative and eventually some number become unstable. It is important to note that none of the eigenvalues become unstable by becoming greater than 1.0, but only by becoming more negative than -1.0.

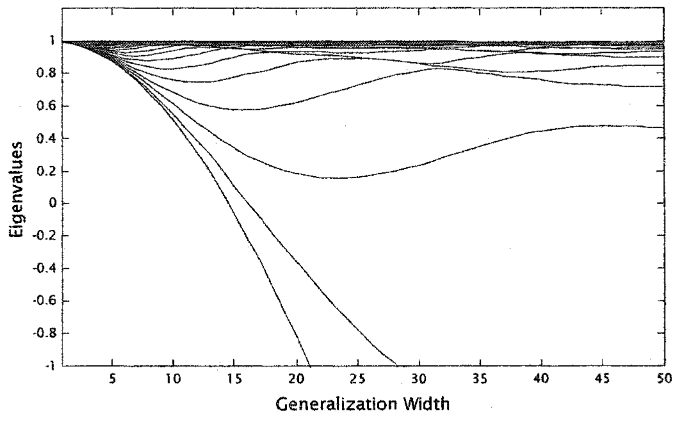


Figure 2.21 – Eigenvalue locations as a function of generalization for a width 50 and a learning gain of 0.005.

Figure 2.22 depicts the eigenvalue locations as a function of learning gain, α, for a single dimensional CMAC with 50 weights and a generalization width of 5. At a learning gain of zero, the system matrix is diagonal and all eigenvalues are identically equal to one. As the learning gain increases, the eigenvalues generally become more negative and eventually some become unstable by becoming more negative than -1.0. Once again, no eigenvalues become unstable in the positive direction.

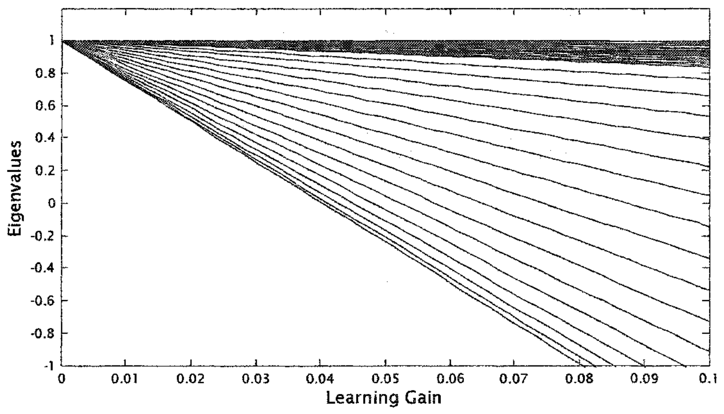


Figure 2.22 - Eigenvalue locations as a function of learning gain.

Figure 2.23 depicts a numerically calculated plot of the maximum stable learning gain as a function of both the generalization width and network size. This chart was created using a binary search algorithm for the learning gain within the range [0,2] to determine the learning gain that yielded a marginally stable update (i.e. an eigenvalue at -1.0). This binary search was repeated at every combination of generalization width and network size to generate the following graph. Notice that the maximum stable learning gain depends strongly on the generalization width, but only weakly on the network size.

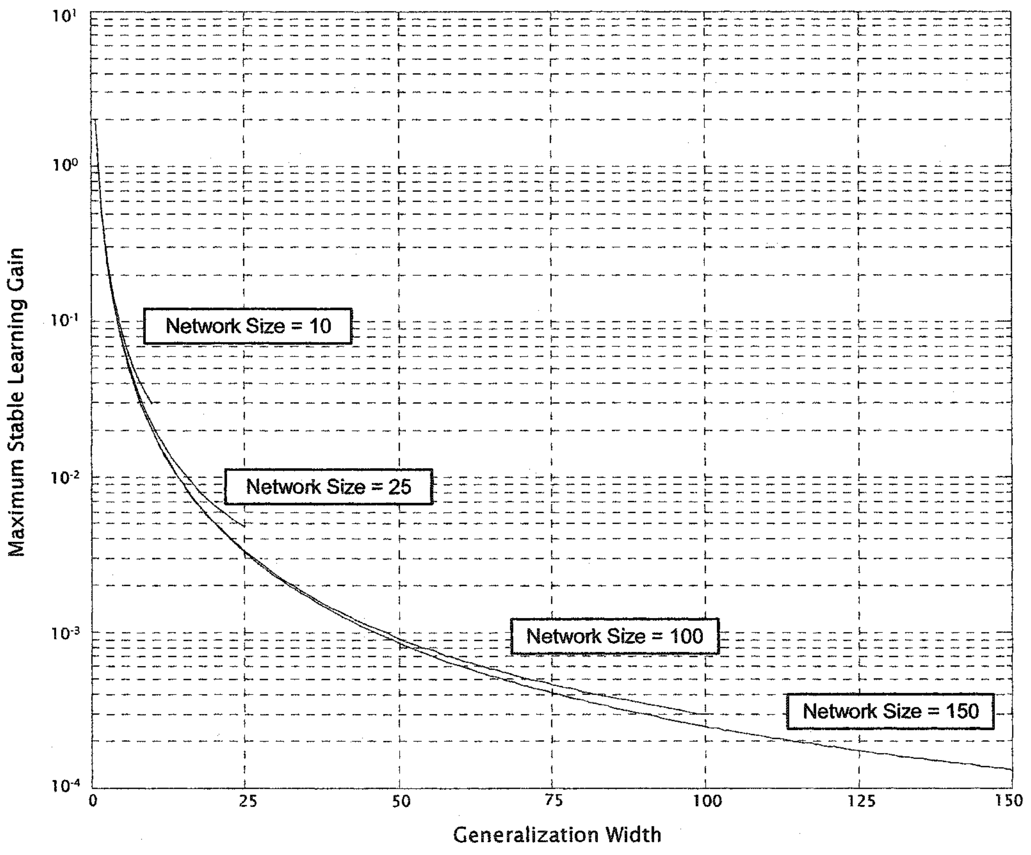


Figure 2.23 – Numerically determined maximum stable learning gain.

From the simulation results of figure 2.21 and 2.22, it appears that instability is always caused by eigenvalues which become unstable in the negative direction. Under this assumption, the stability of the weight update can be guaranteed by determining a lower bound for the eigenvalues of the system matrix on the real axis. Such a lower bound can be established using the Gershgorin theorem presented in figure 2.24 [99][102][103].

The Gershgorin Theorem: Given the real valued matrix $\mathrm{A} \in \mathbb{R}^{n \times n}$, its eigenvalues are all contained within the union $\bigcup_{i=1}^{n} \Gamma_i$ where Γ_i are circles in the complex plane as defined by the following equation.

$$\Gamma_i = \{z \in \mathbb{C} : |z - a_{ii}| \leq r_i\}, \quad r_i = \sum_{j=1, j \neq i}^{n} |a_{ij}|, \quad i = 1, \ldots, n$$

Figure 2.24 – The Gershgorin Theorem.

For the case of the general weight update system matrix, all of the Gershgorin circles are centered at $(1 - \beta\alpha)$ on the real axis where β represents the generalization width of the network and α represents the learning gain. Additionally, all eigenvalues are constrained to the real axis since the update matrix is symmetric. It is assumed that instability is caused solely by negatively decreasing eigenvalues as confirmed by simulation studies. Therefore, the most negative eigenvalue is found by determining the intersection of the largest radius Gershgorin circle with the real axis as shown in figure 2.25.

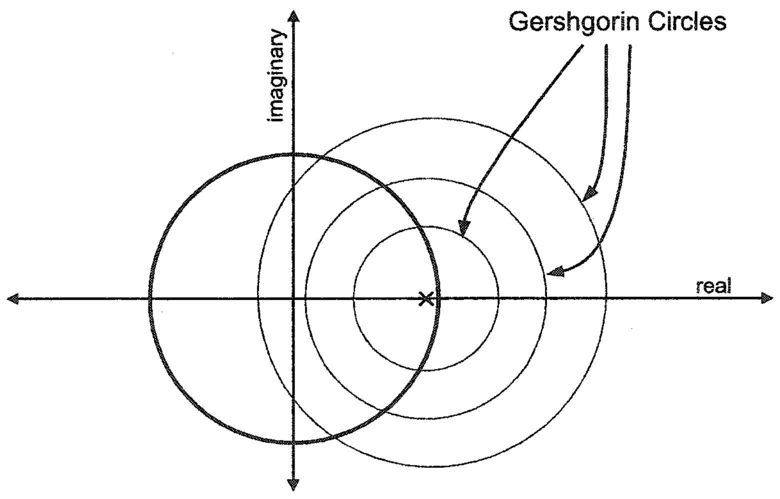


Figure 2.25 – Gershgorin circles for the general CMAC update matrix.

To determine the nature of the Gershgorin circles in this problem, the general single dimension CMAC update equation matrix is given in equation (2.40). Notice that the width of the band depends directly on the generalization width parameter.

$$\begin{bmatrix} (1-\alpha\beta) & -(\beta-1)\alpha & -(\beta-2)\alpha & \vdots & -\alpha & 0 & \cdots \\ -(\beta-1)\alpha & (1-\alpha\beta) & -(\beta-1)\alpha & -(\beta-2)\alpha & \vdots & -\alpha & \cdots \\ -(\beta-2)\alpha & -(\beta-1)\alpha & (1-\alpha\beta) & -(\beta-1)\alpha & -(\beta-2)\alpha & \vdots & \cdots \\ \vdots & -(\beta-2)\alpha & -(\beta-1)\alpha & (1-\alpha\beta) & -(\beta-1)\alpha & -(\beta-2)\alpha & \cdots \\ -\alpha & \vdots & -(\beta-2)\alpha & -(\beta-1)\alpha & (1-\alpha\beta) & -(\beta-1)\alpha & \cdots \\ 0 & -\alpha & \vdots & -(\beta-2)\alpha & -(\beta-1)\alpha & (1-\alpha\beta) & \cdots \\ 0 & 0 & -\alpha & \vdots & -(\beta-2)\alpha & -(\beta-1)\alpha & \cdots \\ 0 & 0 & 0 & -\alpha & \vdots & -(\beta-2)\alpha & \cdots \\ 0 & 0 & 0 & 0 & -\alpha & \vdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & -\alpha & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{2.40}$$

As shown in figure 2.25, all of the Gershgorin circles are centered at the point $(1-\alpha\beta)$ since all of the diagonal entries are equal. As a result, the eigenvalue bound will be solely determined by the Gershgorin circle with the largest radius. From matrix (2.40), the largest Gershgorin radius can be written as given in (2.41).

$$r_{\max} = 2\alpha(\beta-1) + 2\alpha(\beta-2) + 2\alpha(\beta-3) + \cdots + 2\alpha = \alpha(\beta-1)\beta \tag{2.41}$$

Using the result in (2.41), the lower bound on the eigvenvalues along the real axis is given by equation (2.42).

$$\lambda_{\min} \geq (1-\alpha\beta) - \alpha(\beta-1)\beta = 1-\alpha\beta^2 \tag{2.42}$$

Finally, to ensure stability it is required that $\lambda_{\min} > -1$. This condition results in the learning gain bound given in equation (2.43). Notice that for the case of $\beta = 1$ this result agrees with the earlier result of $0 < \alpha < 2$.

$$0 < \alpha < \frac{2}{\beta^2} \tag{2.43}$$

This result indicates that to maintain the stability of the batch mode update, the learning gain must be normalized by the square of the generalization. Based on this result, the weight update equation can be modified to eliminate the dependence of the generalization on the learning gain parameter as shown in equation (2.44). In this case, the weight update will result in stable adaptation as long as the modified learning gain α' is chosen according to the bound $0 < \alpha' < 2$. It is interesting to note that this is smaller than the maximum stable learning gain in the case of sequential update.

$$\mathbf{w}[k+1] = \mathbf{w}[k] + \frac{\alpha'}{\beta^2}\mathbf{A}^T(\hat{\mathbf{y}} - \mathbf{y}[k]) \tag{2.44}$$

Figure 2.26 provides a comparison of the derived analytical learning gain bound superimposed on the previous numerically derived bounds. Notice that the Gershgorin bound is consistent with the numerical estimates. Additionally, the Gershgorin bound appears to be equal to the infinite network size generalization of the numerical bounds. This is a reasonable result given that the Gershgorin bound is independent of network size.

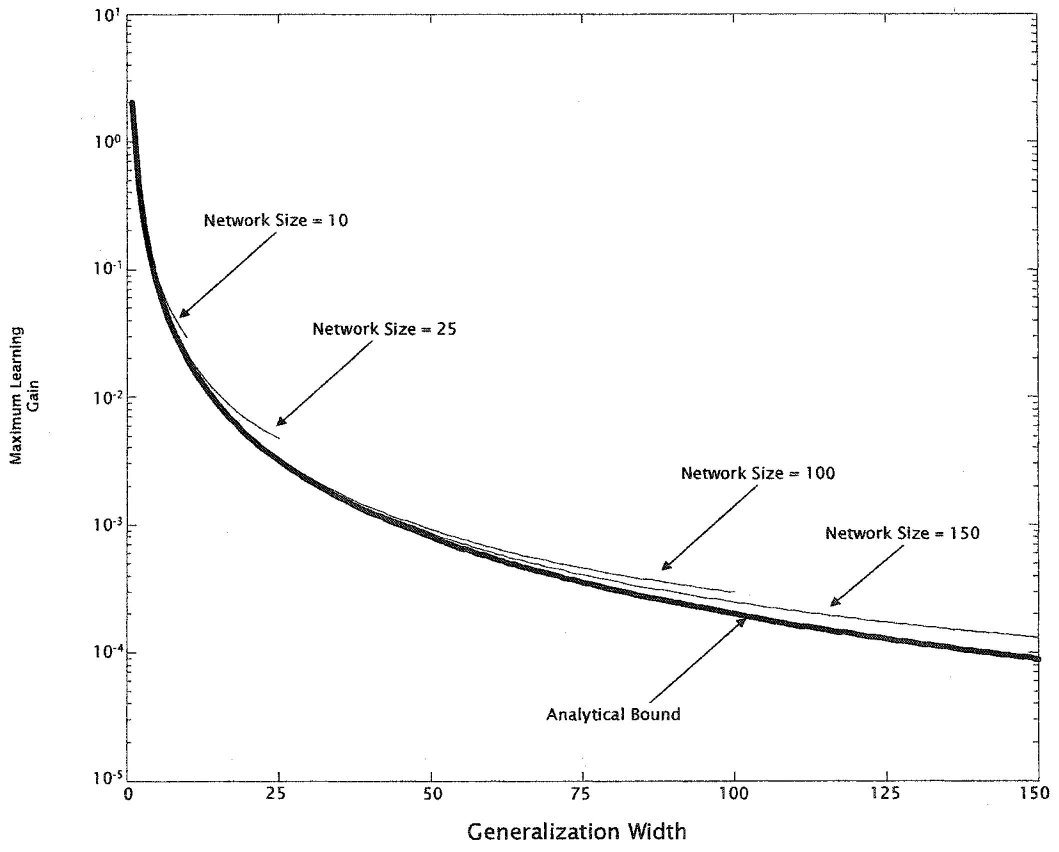


Figure 2.26 – Analytical bound plotted with previous numerically determined bounds.

The plot in figure 2.27 depicts the eigenvalues of a CMAC network of width 25 and learning gain 0.005 for different values of the generalization parameter. Figure 2.28 depicts the same results except for a network of width 100. The vertical line in each plot depicts that maximum stable generalization width as calculated via the analytical bound. Notice that the analytical bound provides an accurate bound in each case and the bound is closer to the numerical result in the case of the larger network.

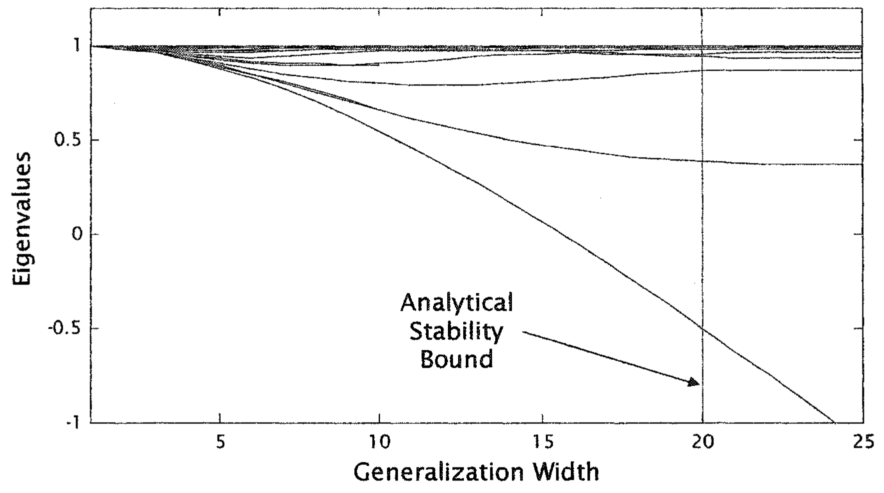


Figure 2.27 – Eigenvalues versus generalization width for a network with 25 weights and learning gain 0.005.

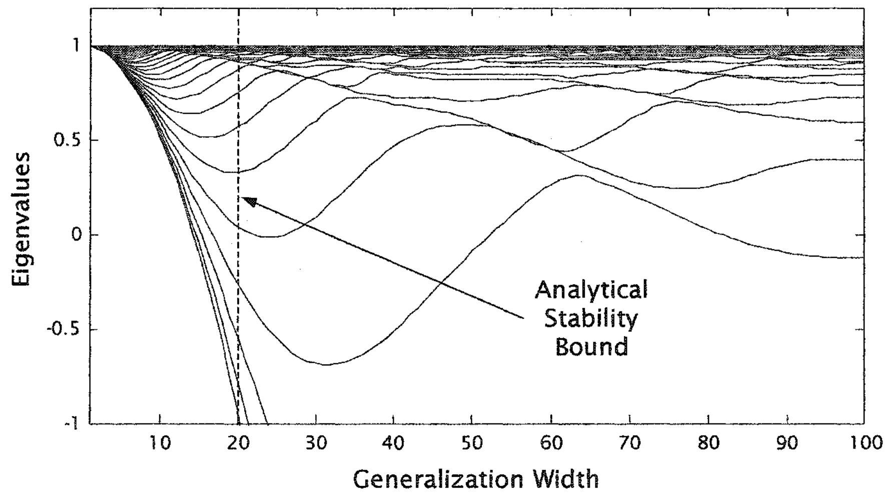


Figure 2.28 - Eigenvalues versus generalization width for a network with 100 weights and learning gain 0.005.

Another important aspect of the weight adaptation is the rate of convergence. In general, the convergence is more rapid as the learning gain is increased. This can be explained in terms of the eigenvalue plots by the fact that the eigenvalues move toward the origin as the learning gain is increased. However, if the learning gain is increased sufficiently, the most negative eigenvalues decrease to the neighborhood of -1.0 where they represent very slowly decreasing underdamped eigenvalues. Consequently, this can lead to an overall reduced rate of convergence. As a result, it is reasonable to expect that the optimal convergence rate is obtained for some intermediate learning gain, rather than exactly at the maximum stable learning gain.

The plot of figure 2.29 depicts the convergence rate versus learning gain for three different CMAC networks with 50 weights and generalization width of three. In each case, the initial weight vector is all zeros and the target vector is comprised of random numbers uniformly selected from the interval [0,1]. Convergence is defined here as achieving a total sum squared error over the entire vector of outputs of 0.01. The graph of figure 2.29 depicts the number of iterations required to reach convergence as a function of learning gain. The vertical line represents the analytical stability bound, above which the network is unstable. The maximum number of cycles tested was 100,000 and this results in the flat curve seen in the unstable region.

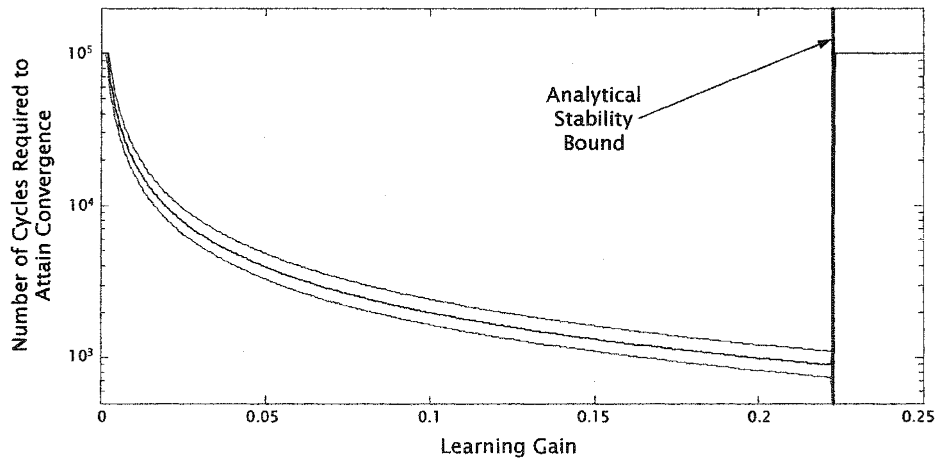


Figure 2.29 – Rate of Convergence versus learning gain.

This result suggests that the optimal convergence rate is attained using a learning gain that is the largest gain which maintains stability. However, figure 2.30 presents rate of convergence results for the same network except instead of having randomly generated target vectors, the target vector is comprised of all ones. Notice that the convergence is much faster in this case and the nature of the convergence curve is fundamentally different in that the fastest convergence occurs at some point below the largest stable gain.

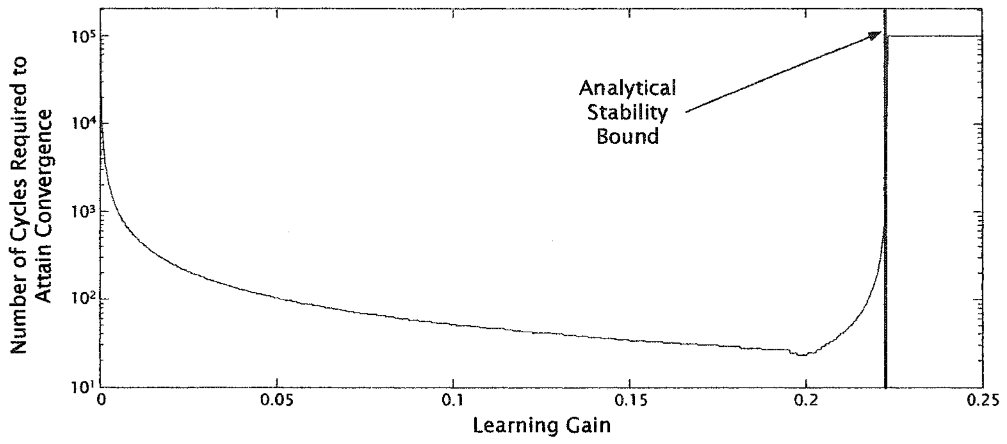


Figure 2.30 – Convergence rate versus learning gain for uniform target vector.

In conclusion, despite the fact that stability of the CMAC update is independent of the data being learned, the convergence rate is highly dependent upon the target data. At first consideration this seems implausible since the nature of the transient response of a linear system is determined by its eigenvalues, independent of the initial conditions and external inputs. However, the explanation is that not all of the eigenvalues necessarily contribute to the rate of convergence. Furthermore, the set of eigenvalues which are expressed in the output is dependent upon the target vector. The reason for this is that each eivenvalue governs the nature of the response in the direction associated with its eigenvector. If the initial conditions are chosen to lie on a line defined by a single eigenvector, then the dynamics will be governed solely by the associated eigenvalue.

As an example of this phenomenon in a physical linear system, consider the simple circuit of figure 2.31. Solving the circuit equations for the two node voltages indicates that the system has two eigenvalues: one slow eigenvalue at $\frac{-1}{RC}$ and one fast eigenvalue at $\frac{-2}{R_2C} - \frac{1}{RC}$. If the system begins with initial conditions $V_1 = 1$ and $V_2 = 1$ then only the dynamics associated with the slow eigenvalue will be expressed. However, if the initial conditions are chosen as $V_1 = 1$ and $V_2 = -1$ then the dynamics are totally governed by the fast eigenvalue and evolve along its associated eigenvector.

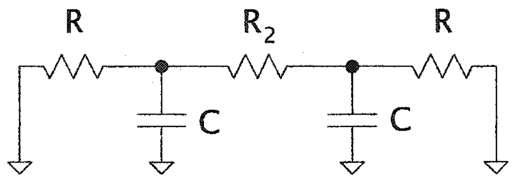


Figure 2.31 – Simple circuit to demonstrate the dependence of initial conditions on the transient response.

The coupling in the circuit in figure 2.31 is qualitatively similar to the effect of overlapping receptive fields in the CMAC weight update. Specifically, if training points with overlapping receptive fields are similar in value, then the convergence will be more rapid. If nearby training points are dissimilar, the convergence rate will decrease. As a result, smoother functional relationships will generally experience more rapid convergence rates than rapidly varying maps. This agrees with the previous simulation results for the cases of a constant valued target versus the randomly generated target vector.

The batch mode update analyzed to this point is not well suited for use in real-time active disturbance cancellation applications. Instead, in such cases it is advantageous to utilize a sequential update algorithm whereby the error at each time interval is used to update only the selected weights at that instant. This reduces the storage requirements and distributes the training computation evenly over time. The weight dynamics associated with sequential update are different than for the batch mode updates even if the same training data are utilized. This is due to the fact that in the sequential update there is an opportunity for interference between the weight updates produced by different training pairs within a single training epoch. However, it will be shown that for sufficiently small learning gain, the sequential weight update algorithm converges to the batch mode weight update result.

The sequential weight update will be considered for the case where the entire set of M training pairs is presented to the network in a fixed sequence. The sequential weight update is given in equation (2.45) where $\mathbf{a}[k]$ represents the weight selection vector at discrete time interval k. The weight selection vector is binary valued and contains exactly β non-zero elements. Additionally, $y[k]$ and $\hat{y}[k]$ represent the scalar CMAC output and scalar target value at time k. The CMAC output is calculated via the inner product of the weight selection vector and the weight vector as given in (2.46).

$$\mathbf{w}[k+1] = \mathbf{w}[k] + \alpha \mathbf{a}[k](\hat{y}[k] - y[k]) \tag{2.45}$$

$$y[k] = \mathbf{a}[k]^T \mathbf{w}[k] \tag{2.46}$$

Given that there are only M training pairs, there are also only M distinct values for the weight selection vector and target output. These will be denoted as $\mathbf{a}_0, \mathbf{a}_1, \cdots, \mathbf{a}_{M-1}$ and $\hat{y}_0, \hat{y}_1, \cdots, \hat{y}_{M-1}$, respectively. As a result, an entire epoch of training can be represented by the set of M weight update equations given in (2.47) – (2.50).

$$\mathbf{w}[k+1] = \mathbf{w}[k] + \alpha \mathbf{a}_0 (\hat{y}_0 - y[k]) \tag{2.47}$$

$$\mathbf{w}[k+2] = \mathbf{w}[k+1] + \alpha \mathbf{a}_1 (\hat{y}_1 - y[k+1]) \tag{2.48}$$

$$\mathbf{w}[k+3] = \mathbf{w}[k+2] + \alpha \mathbf{a}_2 (\hat{y}_2 - y[k+2]) \tag{2.49}$$

$$\vdots$$

$$\mathbf{w}[k+M] = \mathbf{w}[k+M-1] + \alpha \mathbf{a}_{M-1} (\hat{y}_{M-1} - y[k+M-1]) \tag{2.50}$$

Given that $y[k+i] = \mathbf{a}_i^T \mathbf{w}[k+i]$ this set of weight updates can be rewritten as given by (2.51) – (2.54).

$$\mathbf{w}[k+1] = \alpha \mathbf{a}_0 \hat{y}_0 + \left(I - \alpha \mathbf{a}_0 \mathbf{a}_0^T\right) \mathbf{w}[k] \tag{2.51}$$

$$\mathbf{w}[k+2] = \alpha \mathbf{a}_1 \hat{y}_1 + \left(I - \alpha \mathbf{a}_1 \mathbf{a}_1^T\right) \mathbf{w}[k+1] \tag{2.52}$$

$$\mathbf{w}[k+3] = \alpha \mathbf{a}_2 \hat{y}_2 + \left(I - \alpha \mathbf{a}_2 \mathbf{a}_2^T\right) \mathbf{w}[k+2] \tag{2.53}$$

$$\vdots$$

$$\mathbf{w}[k+M] = \alpha \mathbf{a}_{M-1} \hat{y}_{M-1} + \left(I - \alpha \mathbf{a}_{M-1} \mathbf{a}_{M-1}^T\right) \mathbf{w}[k+M-1] \tag{2.54}$$

This set of equations can be combined into the single result given in (2.55) which expresses the aggregate change in the weight vector over an entire epoch of sequential training updates.

$$\mathbf{w}[k+M] = \alpha \sum_{i=0}^{M-1} \prod_{j=M-1-i}^{j=1} \left(I - \alpha \mathbf{a}_j \mathbf{a}_j^T\right) \mathbf{a}_i \hat{y}_i + \prod_{j=M-1}^{j=0} \left(I - \alpha \mathbf{a}_j \mathbf{a}_j^T\right) \mathbf{w}[k] \tag{2.55}$$

Under the assumption that the learning gain is small, higher order terms in α can be neglected. After elimination of terms of α^2 and greater, this expression reduces to that given in equation (2.56).

$$\mathbf{w}[k+M] = \alpha \sum_{i=0}^{M-1} \mathbf{a}_i \hat{y}_i + \left[\mathbf{I} - \alpha \sum_{i=0}^{M-1} \mathbf{a}_i \mathbf{a}_i^T\right] \mathbf{w}[k] \tag{2.56}$$

The cumulative weight update equation can be further reduced via algebraic manipulation to yield the expression in (2.57).

$$\mathbf{w}[k+M] = \mathbf{w}[k] + \alpha \sum_{i=0}^{M-1} \mathbf{a}_i \left(\hat{y}_i - \mathbf{a}_i^T \mathbf{w}[k]\right) \tag{2.57}$$

The batch mode weight selection matrix, $\mathbf{A}$, can be expressed as a composition of the individual weight selection vectors as given in (2.58). Additionally, the batch mode output vector, $\mathbf{y}$, is given by (2.59).

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_0^T \\ \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_{M-1}^T \end{bmatrix} \tag{2.58}$$

$$\mathbf{y} = \begin{bmatrix} y_0 & y_1 & \cdots & y_{M-1} \end{bmatrix}^T = \begin{bmatrix} \mathbf{a}_0^T \mathbf{w}[k] & \mathbf{a}_1^T \mathbf{w}[k] & \cdots & \mathbf{a}_{M-1}^T \mathbf{w}[k] \end{bmatrix}^T \tag{2.59}$$

Let k' denote a new discrete time index which is updated once per training epoch. Then, using (2.58) and (2.59), the approximate sequential weight update equation is given by (2.60) under the assumption that $\alpha \ll 1$.

$$\mathbf{w}[k'+1] = \mathbf{w}[k'] + \alpha \mathbf{A}\left(\hat{\mathbf{y}} - \mathbf{y}[k]\right) \tag{2.60}$$

Notice that equation (2.60) is exactly equal to the batch mode weight update equation given previously in (2.34). As a result, this shows that for sufficiently small learning gain, the sequential weight update algorithm will have properties which are well approximated by those derived for the batch mode update.

CHAPTER 3

FEEDFORWARD DISTURBANCE CANCELLATION USING NEURAL NETWORKS

3.1 Introduction

The concept of feedforward active disturbance cancellation was introduced in chapter two. In particular, it was shown that this approach is capable of effective cancellation in many linear time varying systems through use of an adaptive linear compensator. However, in many practical systems, the effectiveness of this approach is limited due to the presence of nonlinearities in the actuators and transmission paths of the system. For example, one common manifestation of nonlinearities in vibration and acoustic cancellation systems is the presence of energy at harmonics of the fundamental disturbance source. Effective cancellation in such environments is only possible by utilizing a compensator which is capable of representing the nonlinear dynamics of the system. This provides the motivation in the present chapter for considering the use of a time delay neural network as an adaptive compensator in feedforward disturbance cancellation.

The feedforward time delay CMAC disturbance cancellation architecture is shown in figure 3.1. The signal $u[k]$ represents a disturbance which propagates through the generally nonlinear history-dependent dynamics of $g(\cdot)$ and $f_2(\cdot)$ to produce the observed error signal, $e[k]$. Usually, an exact measurement of the original disturbance is not available and instead some related estimate of that signal is used. This possibility is represented in the block diagram by the function $h(\cdot)$ which represents a nonlinear distortion of the actual disturbance signal. The time delay CMAC utilizes the estimate of the original disturbance to generate a cancellation signal which influences the original disturbance at the summing junction. The functions $f_1(\cdot)$ and $f_2(\cdot)$ represent the path of the cancellation signal which is commonly referred to as the secondary path. The first component of this path, $f_1(\cdot)$, represents the dynamics associated with the cancellation actuator and the signal transmission path which is unique to the cancellation signal. The second component, represented by $f_2(\cdot)$, represents the portion of the secondary path which is common to both the original disturbance signal and the secondary cancellation signal. Finally, $f_3(\cdot)$ represents the possibility for dynamics introduced in the sensing of the error signal. This includes any distortion produced by the error sensor as well as the delay and quantization associated with conversion of the error signal into the digital domain.

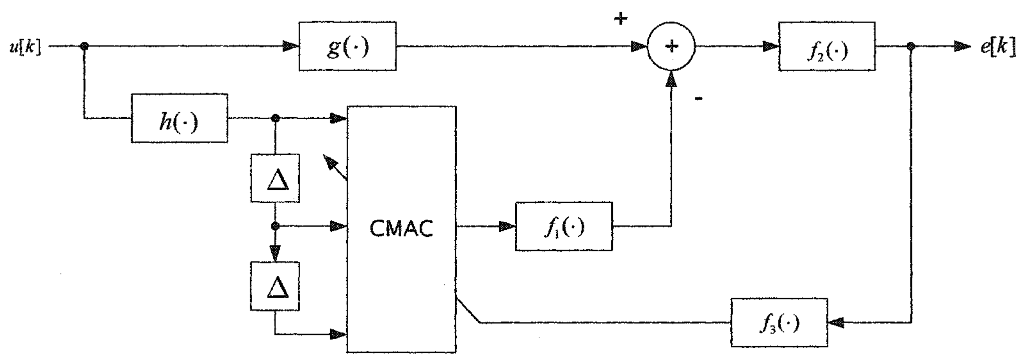


Figure 3.1 – The Time Delay CMAC feedforward disturbance cancellation algorithm.

In order to provide attenuation of the error signal, the CMAC must be trained such that the combined cancellation path comprised of $h(\cdot)$, the time delay CMAC, and $f_1(\cdot)$, approximates the forward path, $g(\cdot)$. Given the nonlinear mapping capabilities of the CMAC, this architecture can be used to accommodate a wide range of nonlinearities in both the forward path as well as the reference signal path. This architecture provides a particular advantage in the case of narrowband feedforward cancellation since the time delay CMAC allows for the use of a reference signal which is not linearly related to the actual disturbance.

For example, a suitable reference signal can be provided by a sensor generated sawtooth waveform at the fundamental frequency of the disturbance.

The usual method of training the CMAC neural network is via an incremental weight adjustment designed to implement a stochastic gradient descent of the instantaneous output error. However, in the present case, the output error is unknown due to the presence of the secondary path dynamics represented by $f_1(\cdot)$ and $f_2(\cdot)$. If an inverse model of the secondary path were available, it could be utilized to transform the observed error signal, $e[k]$, into the corresponding error at the output of the compensator. However, the secondary path often contains time delays and non-minimum phase dynamics which render the inverse noncausal and unstable, respectively. As a result, in order to provide effective adaptation of the CMAC compensator, the weight update algorithm must be revised to directly utilize the observed error signal, $e[k]$. In general, the modified weight update algorithm will be dependent upon the nature of the secondary path. In this chapter, suitable modifications to the training algorithm will be presented which ensure stability of the adaptation for a variety of secondary paths including those characterized by: linear gains, static nonlinearities, time delay, linear dynamical systems, and full nonlinear dynamics.

3.2 Static Linear Secondary Path

In this section, performance of the time delay CMAC compensator will be considered in the case of a secondary path which is comprised solely of static linear gain elements. Stability of the weight update algorithm will be considered in the context of figure 3.2 where the nonlinear portion of the forward path is itself represented by a time delay CMAC. This technique has been utilized previously to attain convergence results for open and closed loop CMAC learning [49]. In essence, this approach circumvents the issue of whether a particular relationship can be represented by the time delay CMAC by focusing instead on the convergence of the algorithm under the assumption that the desired relationship can be modeled by a CMAC with appropriately chosen parameters. The parameters of the forward path model (CMAC A) are unconstrained and therefore, with appropriate choices, this model can represent a wide variety of nonlinear dynamical signal paths. These parameters include the CMAC receptive field placement, the generalization width, the input coarse-coding structure, the tap delay, and the number of taps in the tapped delay line.

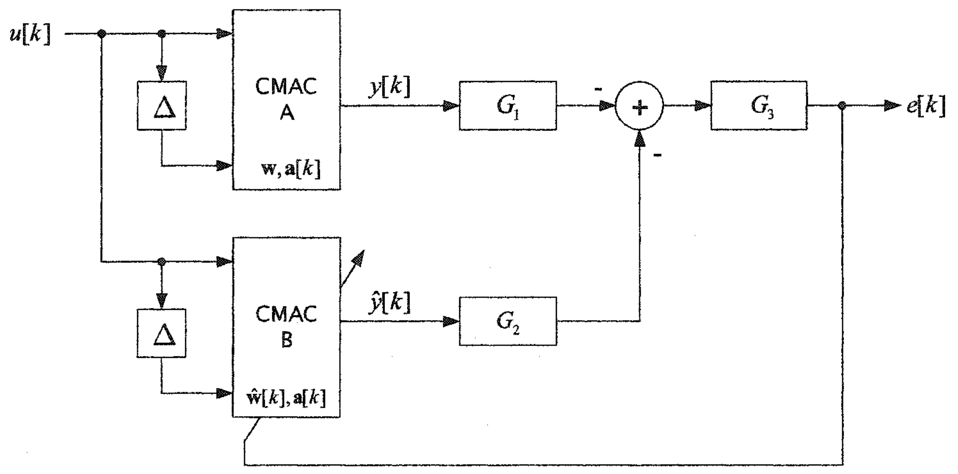


Figure 3.2 – Time Delay CMAC compensator with linear secondary path.

At each discrete time instant, the CMAC outputs are computed as a linear combination of the β active weights, where β represents the CMAC generalization parameter. This is true independent of the input dimension of the CMAC. A convenient representation of the CMAC is afforded by organizing the complete set of CMAC weights into a single vector. Using this convention, the selection of the active weight set can be performed by a time varying binary weight selection vector, $\mathrm{a}[k]$, which contains exactly β elements of value 1 at the locations of the active weights. As a result, the CMAC output at each instant of time is concisely represented by the inner product of the weight selection vector and the weight vector. Thus, the nonlinearity of the CMAC mapping is contained entirely in the formulation of the vector $\mathrm{a}[k]$. Utilizing this technique, the output of CMAC A is given by equation (3.1) where w represents the weight vector for the forward path CMAC. It is assumed that the forward path model is not time varying and therefore, the weight vector is constant valued.

$$y[k] = \mathbf{a}[k]^T \mathbf{w} \tag{3.1}$$

The fundamental assumption to be utilized in this approach is that the architecture of the compensator is a direct match to the forward path model. Specifically, it is assumed that all parameters of time delay CMAC A are identical to the corresponding parameters of CMAC B. With this assumption, and given that both networks have the same input vector, the weight selection vector is also identical in both cases. As a result, the output of CMAC B is given by (3.2) where $\hat{\mathbf{w}}[k]$ represents the time varying compensator weight vector.

$$\hat{y}[k] = \mathbf{a}[k]^T \hat{\mathbf{w}}[k] \tag{3.2}$$

The error signal to be minimized by the compensator is given by equation (3.3) and the compensator weight update is performed using the standard CMAC algorithm given in (3.4) except that the observed error signal, $e[k]$, is used to replace the CMAC output error. The parameter α specifies the learning gain, while the parameter β represents the generalization width.

$$e[k] = G_3\left(G_1 y[k] - G_2 \hat{y}[k]\right) = G_1 G_3 \mathbf{a}[k]^T \mathbf{w} - G_2 G_3 \mathbf{a}[k]^T \hat{\mathbf{w}}[k] \tag{3.3}$$

$$\hat{\mathbf{w}}[k+1] = \hat{\mathbf{w}}[k] + \frac{\alpha}{\beta}\mathbf{a}[k] e[k] \tag{3.4}$$

Given this description of the operation of the disturbance cancellation algorithm of figure 3.2, it will be shown that the error signal $e[k]$ is convergent to zero under the condition that the learning gain parameter satisfies the bound given in (3.5).

$$0 < \alpha < \frac{2}{G_2 G_3} \tag{3.5}$$

Notice that the learning gain bound depends only on the gain within the path of the cancellation signal. Additionally, for $G_2 = 1$ and $G_3 = 1$ this result reduces to the standard CMAC output error learning gain bound of $0 < \alpha < 2$. Before presenting the proof of this learning gain bound, Lemma 3.1 is derived as an intermediate result for use in this and later theorems.

Lemma 3.1 – Convergence of a Discrete Increment: Given a discrete time signal $V[k]$ such that $V[k]$ is a decreasing sequence (i.e. $V[k+1] \leq V[k]$) and $V[k]$ is bounded below (i.e. $V[k] \geq K \in \mathbb{R}\ \forall k$), then $\lim_{k\to\infty}(V[k+1]-V[k]) = 0$.

Proof:

Since $V[k]$ is a decreasing sequence which is bounded below, its limit exists and is finite as given by equation (3.6) [65].

$$\lim_{k\to\infty} V[k] = L \geq K \tag{3.6}$$

Let an arbitrary $\varepsilon > 0$ be given. Let $\varepsilon' = \frac{1}{2}\varepsilon > 0$. Then statement (3.7) follows from the definition of the limit in (3.6).

$$\exists\, K' \ni |V[k]-L| < \varepsilon' \quad \forall k > K' \tag{3.7}$$

Use of the triangle inequality and the definition of ε' yields the inequality in (3.8).

$$\begin{aligned}&|V[k+1]-V[k]| = |(V[k+1]-L)+(-V[k]+L)| \\ &\leq |V[k+1]-L| + |-V[k]+L| \\ &= |V[k+1]-L| + |V[k]-L| < 2\varepsilon' = \varepsilon\end{aligned} \tag{3.8}$$

It has been shown that given any arbitrary $\varepsilon > 0$, there exists K' such that $|V[k+1]-V[k]| < \varepsilon$ for all $k > K'$. Therefore, $\lim_{k\to\infty}(V[k+1]-V[k]) = 0$. □

Theorem 3.1 – Convergence of CMAC with a Linear Gain Secondary Path: Consider the block diagram of figure 3.2 with dynamics specified in equations (3.1) through (3.4). Implicit in these equations is the fact that CMAC A and CMAC B have identical architectures. Let $\beta > 0$ denote the common CMAC generalization parameter. It is assumed that the path gains are strictly positive. That is, $G_1 > 0$, $G_2 > 0$, and $G_3 > 0$. Under these assumptions, if the learning gain is chosen such that $0 < \alpha < \frac{2}{G_2 G_3}$ then $\lim_{k\to\infty} e[k] = 0$, thereby proving that the system is convergent.

Proof:

Let $\mathbf{D}[k]$ be the weight difference vector scaled by the respective path gain to the observed error signal as defined in equation (3.9). Additionally, let $V[k]$ be as defined by (3.10) and notice that it is non-negative for any k.

$$\mathbf{D}[k] \triangleq G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k] \tag{3.9}$$

$$V[k] \triangleq \mathbf{D}[k]^T \mathbf{D}[k] \geq 0 \tag{3.10}$$

Notice that $V[k]$ is a measure of the aggregate difference in the CMAC outputs over all possible inputs. To determine the variation in $V[k]$ per discrete time interval, its value at time $k+1$ can be derived as shown in equations (3.11)-(3.15).

$$V[k+1] = \mathbf{D}[k+1]^T \mathbf{D}[k+1] \tag{3.11}$$

$$= \{G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k+1]\}^T \{G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k+1]\} \tag{3.12}$$

$$= \left\{G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k] - G_2 G_3 \frac{\alpha}{\beta} e[k]\mathbf{a}[k]\right\}^T \left\{G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k] - G_2 G_3 \frac{\alpha}{\beta} e[k]\mathbf{a}[k]\right\} \tag{3.13}$$

$$= V[k] + G_2^2 G_3^2 \frac{\alpha^2}{\beta^2} e^2[k]\mathbf{a}[k]^T \mathbf{a}[k] - 2G_2 G_3 \frac{\alpha}{\beta} e[k]\mathbf{a}[k]^T \{G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k]\} \tag{3.14}$$

$$= V[k] + G_2^2 G_3^2 \frac{\alpha^2}{\beta^2} e^2[k]\mathbf{a}[k]^T \mathbf{a}[k] - 2G_2 G_3 \frac{\alpha}{\beta} e^2[k] \tag{3.15}$$

It follows directly from the CMAC algorithm that $\mathbf{a}[k]^T \mathbf{a}[k] = \beta$. With this result, the expression in (3.15) can be further reduced as shown in (3.16). Additionally, with the given bound on the learning gain, the increment $V[k+1] - V[k]$ is strictly non-positive although it may be zero, specifically in the case where $e[k] = 0$.

$$V[k+1] - V[k] = \frac{\alpha}{\beta} G_2^2 G_3^2 e^2[k] \left(\alpha - \frac{2}{G_2 G_3}\right) \leq 0 \tag{3.16}$$

From (3.16), $V[k]$ is a decreasing sequence and $V[k]$ is bounded below by zero. Therefore, the conditions of Lemma 3.1 hold and the limit in (3.17) exists.

$$\lim_{k\to\infty} V[k+1]-V[k] = \lim_{k\to\infty} \frac{\alpha}{\beta} G_2^2 G_3^2 \left(\alpha - \frac{2}{G_2 G_3} \right) e^2[k] = 0 \qquad (3.17)$$

Since the multiplicative constant of $e^2[k]$ is strictly non-zero, this result implies that $\lim_{k\to\infty} e^2[k] = 0$ and therefore, $\lim_{k\to\infty} e[k] = 0$. □

Notice that the result of theorem 3.1 indicates that the stability of the compensator weight update depends only on the loop gain of the secondary path, G_2G_3, and not on the forward path gain G_1. Additionally, for any finite value of gain in the secondary path, the learning gain can be set so as to guarantee convergence of the algorithm. This result is independent of the architecture of the tapped delay line and CMAC parameters.

For a qualitative explanation of this proof, consider the illustration of figure 3.3 which depicts $V[k]$ as a function of time. Result (3.16) follows directly from the operational equations of the CMAC and shows that $V[k]$ is a decreasing function. That is, on each discrete time interval, $V[k]$ either decreases or remains the same value. Since $V[k]$ is also strictly non-negative by its definition, it must converge to some (unknown) limiting value, $L \geq 0$.

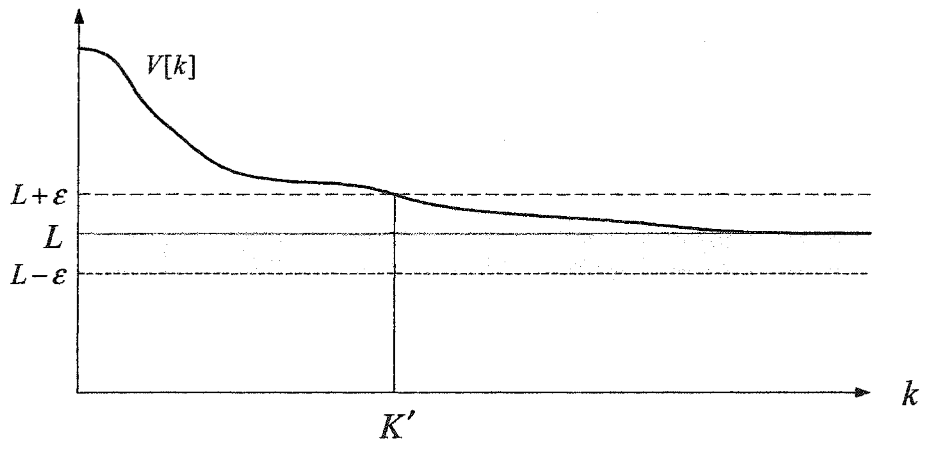


Figure 3.3 – Convergence of $V[k]$.

Given the existence of a limit for $V[k]$, there exists some time K' such that for all $k > K'$ the sequence $V[k]$ is constrained to remain within the shaded region of figure 3.3. As a result, any increment in $V[k]$ must be bounded by 2ε. However, $e^2[k]$ is linearly related to the increment in $V[k]$ as shown by (3.16). As a result, $e^2[k]$ is also bounded. Since this bound can be made arbitrarily tight by going out further in time, this proves that $e^2[k] \to 0$. Notice that, despite the fact that $V[k]$ is monotonically decreasing, the error signal $e^2[k]$ is generally not monotonically decreasing.

As an alternative to the presented proof of Theorem 3.1, it is reasonable to consider $V[k]$ as an energy function and directly apply the second method of Lyapunov. However, there are several difficulties associated with this approach. First, the energy function increment, $V[k+1] - V[k]$, can equal zero even if $V[k]$ does not equal zero. Therefore, the LaSalle Invariance principle must be employed to show that the increment could not remain zero for all time [3]. Additionally, the discrete time system for $\mathbf{D}[k]$ is not an autonomous or time independent system since the weight selection vector varies independent of the states $\mathbf{D}[k]$ and therefore represents an external forcing term. That is, the next state vector $\mathbf{D}[k+1]$ is not solely a function of $\mathbf{D}[k]$, but also depends on $\mathbf{a}[k]$ which varies independent of $\mathbf{D}[k]$. This must be taken into consideration by utilizing the Lyapunov method for time variant systems. Finally, a fundamental dilemma in this approach is that the successful application of the Lyapunov method would show that $\mathbf{D}[k] \to \mathbf{0}$ as $k \to \infty$. This is a sufficient but not necessary condition to show that $e[k] \to 0$ as $k \to \infty$. Specifically, it is not necessary for all weight differences to converge to zero in order for the observed error to converge to zero. There are two reasons for this. First, there can be dormant weights which are not accessed and therefore do not contribute to the output in the long term. Second, there are different linear combinations of weights that produce the same CMAC output. These two cases are demonstrated using a simplified three weight CMAC network.

Consider the three weight reference and target CMAC weight vectors of figure 3.4 where the weights happen to have the values shown at some discrete time instant. It is assumed that the CMAC input signal is a constant so that the same three weights are selected at every time interval and let $G_1 = G_2 = G_3 = 1$. Since the reference and compensator CMAC have the same output, the error signal $e[k]$ will be equal to zero for all time. However, $\mathbf{D}[k] = [1 \quad 0 \quad 1]^T \neq \mathbf{0}$. Thus, different linear combinations of weights can yield zero error without requiring the weight vectors to match. Since the CMAC generally represents a significantly under-specified function approximation, it will usually be the case that there will be many weight vectors which produce the same outputs over the training data set. Therefore, the error can be reduced to zero without requiring that the reference and compensator weight vectors match identically.

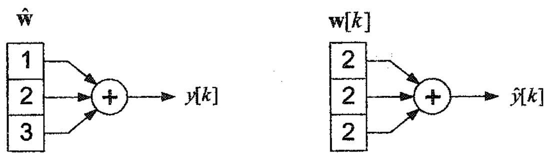


Figure 3.4 – Different weight vectors producing identical outputs.

A second example of this phenomenon occurs when certain dormant weights are not accessed in the long term and therefore due not contribute to the long term error. Consider the situation illustrated in figure 3.5 where it is assumed that the CMAC input is a constant which addresses the first three weights as shown. With the weight values shown, the outputs match and as a result, the error will be zero. However, the weight vectors do not match and therefore $\mathbf{D}[k] = [0 \quad 0 \quad 0 \quad 9]^T \neq \mathbf{0}$. Such dormant weights can be produced in two ways. First, they can be created as a result of differing initial conditions on weights which are never visited. Second, they can result from weights which are visited a finite number of times, but then never visited again (as perhaps on an initial transient response). As a result, the weight values will never match, but since they are not accessed as time goes to infinity, they have no contribution to the observed error in the long term.

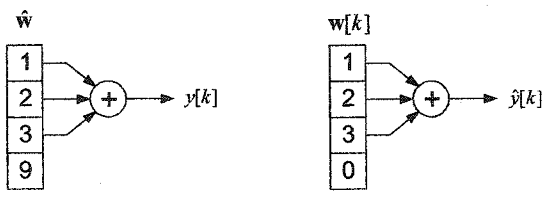


Figure 3.5 – Dormant weights with identical network outputs.

These examples indicate that there are many cases where the weight vector of the compensator will not converge to the weight vector of the reference CMAC and yet the observed error is still convergent to zero. To accommodate such situations, the proof of Theorem 3.1 is based directly on the definition of the limit of the error signal, $e[k]$.

In summary, it has been shown that the nonlinear feedforward CMAC cancellation algorithm shown in figure 3.2 is asymptotically stable for appropriate choice of the learning gain. The proof is based on the fact that the nonlinear signal path can be modeled as a time delay CMAC. The Time Delay CMAC has the ability to model a wide variety of nonlinear behaviors, and therefore, this is not an unreasonable assumption. Additionally, it was required that the compensator CMAC have the same parameters as the reference CMAC. In practical applications, this means that there may be some experimentation required to set the compensator CMAC parameters to adequate values. Notice that the result of theorem 3.1 holds regardless of the length of the tapped delay line input to the reference CMAC.

3.3 Static Nonlinear Secondary Path

In this section, the approach used in the case of a linear secondary path will be employed to determine the stability of the algorithm for a secondary path comprised of a static nonlinearity. This situation is depicted in figure 3.6 where $f(\cdot)$ represents a memoryless nonlinearity while linear gains G_1 and G_2 allow for a difference in gain between the forward and cancellation paths. As in previous section, it is assumed that the parameters of CMAC A are appropriately chosen to model the actual forward path and that the parameters of CMAC B are chosen to match identically.

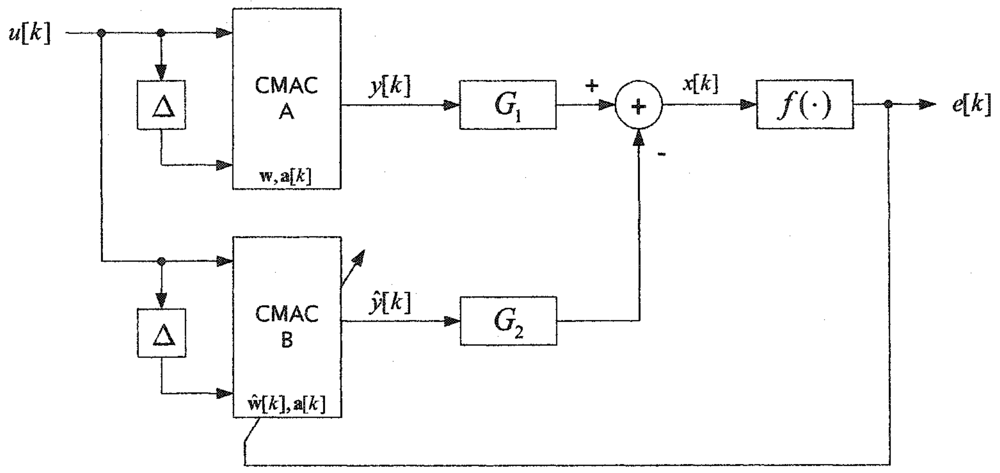


Figure 3.6 – Feedforward CMAC disturbance cancellation with a static nonlinear secondary path.

The operational equations for this system are given by equations (3.18)-(3.21). The CMAC output equations are given in (3.18) and (3.19). The error signal is computed via equation (3.20). The weight update in (3.21) is based on the standard CMAC algorithm except that the observed error signal is used in place of the CMAC output error.

$$y[k] = \mathbf{a}[k]^T \mathbf{w} \tag{3.18}$$

$$\hat{y}[k] = \mathbf{a}[k]^T \hat{\mathbf{w}}[k] \tag{3.19}$$

$$e[k] = f\left(x[k]\right) = f\left(G_1 y[k] - G_2 \hat{y}[k]\right) \tag{3.20}$$

$$\hat{\mathbf{w}}[k+1] = \hat{\mathbf{w}}[k] + \frac{\alpha}{\beta}\mathbf{a}[k]e[k] \tag{3.21}$$

It is necessary to impose certain constraints on $f(\cdot)$ in order to establish the convergence of this algorithm. Specifically, it is assumed that there exists some parameter $m > 0$ such that the nonlinearity is bounded according to (3.22)-(3.24).

$$mx > f(x) > 0 \ \forall x > 0 \tag{3.22}$$

$$mx < f(x) < 0 \ \forall x < 0 \tag{3.23}$$

$$f(0) = 0 \tag{3.24}$$

If these conditions are met, then the error signal will be convergent to zero as long as the learning gain is chosen according to the bound in (3.25).

$$\alpha < \frac{2}{mG_2} \tag{3.25}$$

Depicted graphically, the nonlinearity must be contained within the shaded region of figure 3.7. Since the constant m can be chosen arbitrarily large, the region of allowable nonlinear functions can be readily extended. However, an increase in the parameter m requires a proportional reduction in the learning gain. Notice that there are no additional constraints on $f(\cdot)$ aside from these bounds. In particular, it is not required that $f(\cdot)$ be differentiable or even continuous. The proof of this result is given in Theorem 3.2.

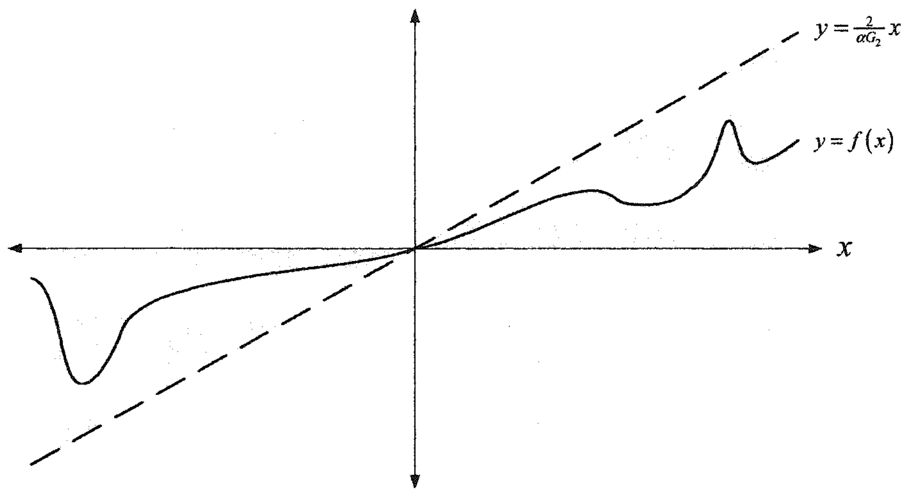


Figure 3.7 – Boundary of permissible nonlinear plant functions.

Theorem 3.2 – CMAC Convergence with a Static Nonlinearity in the Secondary Path: Consider the feedforward disturbance cancellation system shown in figure 3.6 with dynamics specified by equations (3.18)-(3.21). Implicit in these equations is the fact that CMAC A and CMAC B have identical architectures. Let $\beta > 0$ denote the common generalization width parameter. It is assumed that there exists some $m > 0$ such that $mx > f(x) > 0 \ \forall x > 0$, and $-mx < f(x) < 0 \ \ \forall x < 0$. Additionally, $f(0) = 0$. It is also assumed that $G_1 > 0$ and $G_2 > 0$.

Under these conditions, if the learning gain parameter is chosen such that $0 < \alpha < \frac{2}{mG_2}$, then $\lim_{k\to\infty} e[k] = 0$.

Proof:

Let $\mathbf{D}[k]$ be the weight difference vector defined by (3.26). Additionally, let $V[k]$ be defined by (3.27) and notice that $V[k]$ is non-negative for all k.

$$\mathbf{D}[k] \triangleq G_1\mathbf{w} - G_2\hat{\mathbf{w}}[k] \tag{3.26}$$

$$V[k] \triangleq \mathbf{D}[k]^T\mathbf{D}[k] \geq 0 \tag{3.27}$$

Using these definitions and the operational equations (3.18)-(3.21), it is possible to derive an expression for the change in $V[k]$ over any discrete time interval as follows.

$$V[k+1] = \mathbf{D}[k+1]^T\mathbf{D}[k+1] \tag{3.28}$$

$$= \{G_1\mathbf{w} - G_2\hat{\mathbf{w}}[k+1]\}^T\{G_1\mathbf{w} - G_2\hat{\mathbf{w}}[k+1]\} \tag{3.29}$$

$$= \left\{G_1\mathbf{w} - G_2\hat{\mathbf{w}}[k] - G_2\frac{\alpha}{\beta}e[k]\mathbf{a}[k]\right\}^T\left\{G_1\mathbf{w} - G_2\hat{\mathbf{w}}[k] - G_2\frac{\alpha}{\beta}e[k]\mathbf{a}[k]\right\}^T \tag{3.30}$$

$$= V[k] + G_2^2\frac{\alpha^2}{\beta^2}e^2[k]\mathbf{a}[k]^T\mathbf{a}[k] - 2G_2\frac{\alpha}{\beta}e[k]\mathbf{a}[k]^T\{G_1\mathbf{w} - G_2\hat{\mathbf{w}}[k]\} \tag{3.31}$$

Given that $\mathbf{a}[k]^T\mathbf{a}[k] = \beta$ and $e[k] = f(x[k])$ this expression can be utilized to derive the following result for the incremental change in $V[k]$ as given by equation (3.32).

$$V[k+1] - V[k] = \frac{G_2^2\alpha^2}{\beta}f(x[k])^2 - \frac{2G_2\alpha}{\beta}x[k]f(x[k]) \tag{3.32}$$

Inequalities (3.33) and (3.34) follow directly from the bounds on $f(\cdot)$. Additionally, a combination of these inequalities provides the result in (3.35).

$$|f(x[k])| < |mx[k]| \tag{3.33}$$

$$x[k]f(x[k]) \geq 0 \tag{3.34}$$

$$x[k]f(x[k]) > \frac{1}{m}f(x[k])^2 \tag{3.35}$$

The inequality in (3.35) can be utilized to place a bound on the expression in (3.32) as the following result shows. The final inequality in this expression follows directly from the learning gain bound.

$$\begin{aligned} V[k+1]-V[k] &= \tfrac{G_2^2\alpha^2}{\beta} f(x[k])^2 - \tfrac{2G_2\alpha}{\beta} x[k] f(x[k]) \\ &\le \frac{G_2^2\alpha^2}{\beta} f(x[k])^2 - \frac{2G_2\alpha}{\beta m} f(x[k])^2 \\ &= \frac{G_2^2\alpha}{\beta} f(x[k])^2 \left(\alpha - \frac{2}{G_2 m}\right) \le 0 \end{aligned} \tag{3.36}$$

Since $V[k]$ is a decreasing sequence by (3.36) and since $V[k]$ is bounded below by zero, the conditions of Lemma 3.1 are met and the limit (3.37) exists.

$$\lim_{k\to\infty} \left(V[k+1]-V[k]\right) = 0 \tag{3.37}$$

By the sequence squeeze theorem [65] and the inequality shown in (3.36), the limit in (3.38) exists.

$$\lim_{k\to\infty} \frac{G_2^2\alpha}{\beta}\left(\alpha - \frac{2}{G_2 m}\right) e^2[k] = 0 \tag{3.38}$$

The multiplicative constant of $e^2[k]$ in (3.38) is strictly non-zero given the learning gain bound, and therefore $\lim_{k\to\infty} e^2[k] = 0$ and $\lim_{k\to\infty} e[k] = 0$. □

3.4 Band-Limited CMAC Compensators for Dynamical Systems

The systems considered to this point in the chapter have been characterized by secondary paths which contained only static or memoryless transfer properties. In most practical applications, this is far from the case. In fact, most physical secondary paths contain significant and unavoidable transport delay as well as phase delay which varies with frequency. For example, consider the case where the signal paths are represented by linear dynamical systems as shown by figure 3.8. In this illustration, the secondary path is modeled by the linear transfer function $P_2(s)$ while $P_1(s)$ represents the forward disturbance signal path. The transfer function $P_3(s)$ represents a linear distortion of the disturbance source which accounts for the fact that the disturbance signal may not be directly accessible to the compensator.

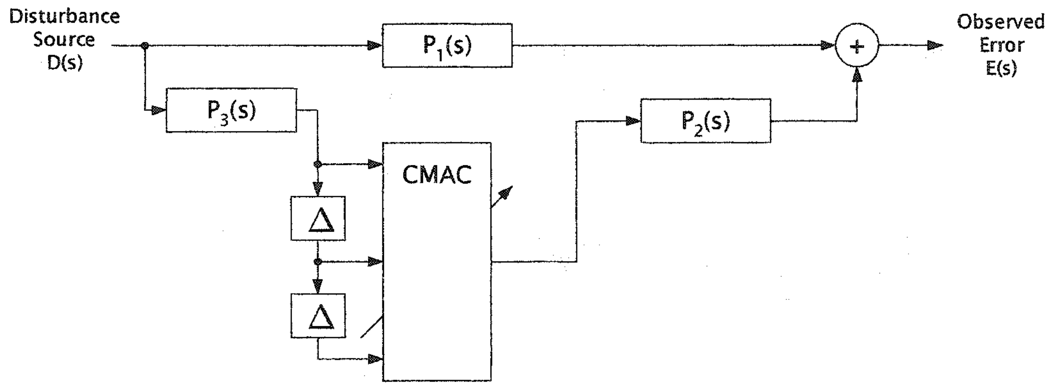


Figure 3.8 – CMAC Compensation with Dynamical Secondary Path.

Implementation of the pictured disturbance cancellation algorithm requires a means to update the CMAC network in a manner which reduces the observed error signal. In the previous two sections, the standard CMAC weight update was used directly with the observed error signal. However, this was only possible because there was no phase delay present in the secondary path in either of those cases. In the case of figure 3.8, direct use of the CMAC will often lead to instability and lack of convergence. As an example, consider the extreme situation where the secondary path, $P_2(s)$, is comprised of a 180 degree phase shift. In that case, the error signal will be inverted, leading directly to instability of the weight update.

The standard CMAC weight update requires a measure in the error of the CMAC output. In the system of figure 3.8, this signal is not observable and the only available error signal is that measured at the other end of the secondary path. One possible solution to provide convergent adaptation is to utilize an inverse model of the secondary path to transform the observed error signal into an error signal at the CMAC output by essentially inverting the dynamics of the secondary path. This approach is shown in figure 3.9. Unfortunately, this approach is not feasible in most active disturbance cancellation problems since a stable, causal inverse model of the secondary path does not exist. A causal inverse will not exist if the secondary path contains a pure time delay as is commonly the case in acoustic cancellation systems. A stable inverse will not exist if the secondary path is non-minimum phase as is also common in active noise and vibration control applications.

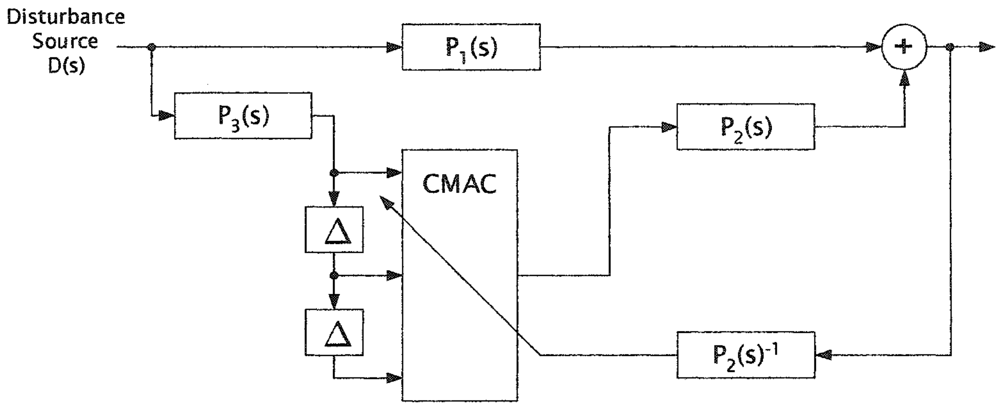


Figure 3.9 – CMAC adaptation with inverse secondary path model.

As described in Chapter 2, this same obstacle is faced in the case of a linear adaptive compensator. In that case, a common and effective solution is to utilize the Filtered-X LMS algorithm. However, this technique is not applicable to the case of a nonlinear adaptive element. To demonstrate its failure in this case, consider use of the Filtered-X LMS algorithm with the adaptive filter replaced by a time delay CMAC network as shown in figure 3.10.

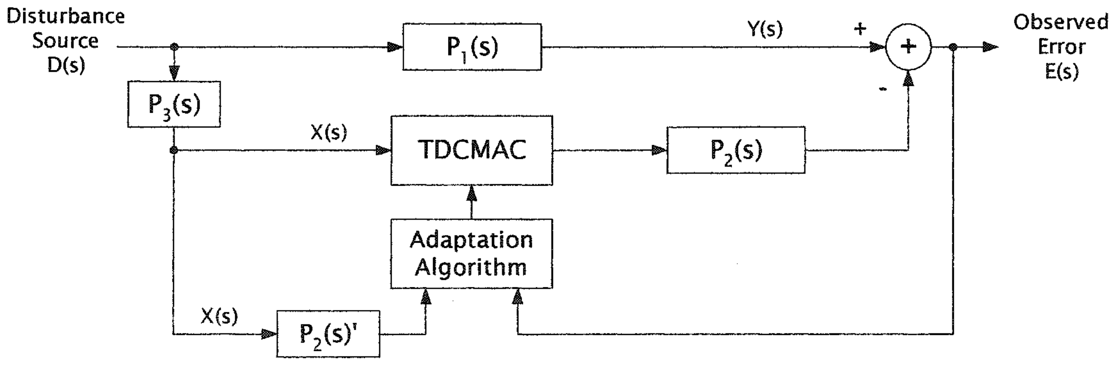


Figure 3.10- Attempt at using Filtered-X LMS Technique.

The transfer function $P_2(s)'$ is a model of the true secondary path $P_2(s)$. The CMAC adaptation is performed using pointers formed by a filtered version of the input. To achieve perfect cancellation, the time delay CMAC must transform its input $X(s)$ into the signal $Y(s)P_2^{-1}(s)$ as represented by the mapping given in (3.39).

$$X(s) \rightarrow Y(s)P_2^{-1}(s) \tag{3.39}$$

However, the actual input to output relationship being used to train the CMAC network is given by (3.40), assuming $P_2(s)' = P_2(s)$.

$$X(s)P_2(s) \rightarrow Y(s) \tag{3.40}$$

Notice that these two mappings are linearly related since multiplying each side of (3.39) by $P_2(s)$ results in (3.40). Therefore, if the compensator is constrained to represent only a linear relationship, the implementation of the mapping (3.40) implies that the relationship given in (3.39) is realized as well. Thus, in the case of an adaptive linear compensator, this algorithm is appropriate. However, a nonlinear compensator such as the time delay CMAC does not have the constraint of linearity and therefore, training on the relationship in (3.40) does not generally result in the implementation of the desired mapping of (3.39).

However, it is possible to utilize this technique directly in the special case where the secondary path is comprised of a pure time delay as shown in figure 3.11. In this case, the desired mapping to be learned is given by (3.41) and the relationship used for the adaptation is given by (3.42).

$$X(s) \rightarrow Y(s)e^{sT} \tag{3.41}$$

$$X(s)e^{-sT} \rightarrow Y(s) \tag{3.42}$$

Notice that these two mappings represent the same functional relationship at two different instants of time. Given that the CMAC mapping is time invariant (neglecting the slow time scale adaptation), adaptation using a delayed version of the input and output signals will produce the same final result as if the original signals were used. This technique can be directly extended to the case where the secondary path is represented by a pure time delay in combination with a static mapping. As long as the mapping does not reverse sign then there will generally be a learning gain which allows for stable adaptation in this case.

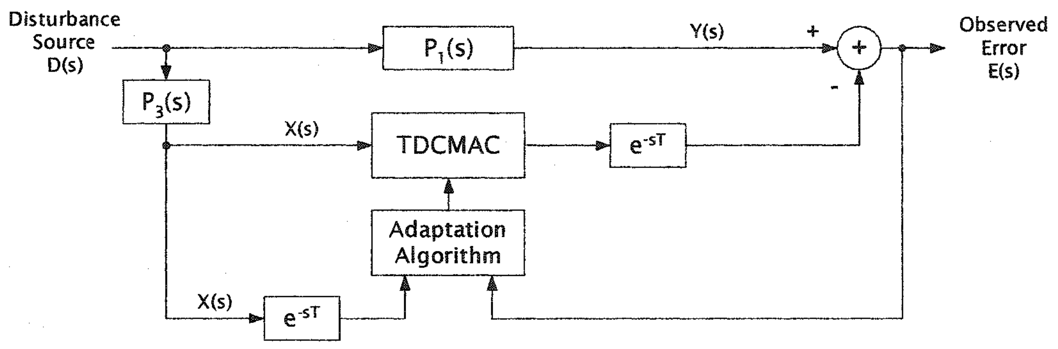


Figure 3.11 – CMAC compensation of a plant with transport delay.

Another potential application of this approach is the case of a narrowband disturbance signal and a secondary path which has a relatively linear phase response across the frequency band of interest. In this case, the secondary path can be adequately approximated by a static gain in combination with a time delay. As a result, the time delay method of figure 3.11 can be utilized if the training pointer delay is chosen to estimate the average phase delay through the secondary path at the frequency of interest. However, it is important to note that a stability problem can still arise in this case. This can occur if the secondary path phase shift is sufficient to induce instability at any frequency, even outside the frequency band of interest. Even though the fundamental disturbance may be well removed from such unstable frequencies, it is possible that energy at the unstable frequencies can be introduced by noise or as a result of the adaptation process itself. If sufficient energy is introduced at the unstable frequencies, the architecture can become unstable.

As an example of this phenomenon, consider the simulation result presented in figure 3.12. In this case, the disturbance source is a single sinusoid and the secondary path is represented by a second order linear transfer function. The phase delay of the disturbance through the secondary path was used as a time delay for the training pointers. Notice that the error signal converges to near zero error initially. However, application of an external noise source during the time interval t=190 to t=200 results in the introduction of noise at frequencies which make the adaptation unstable. As a result, the weight update process becomes unstable and remains unstable even after the disturbance is removed. In other cases, the effects of this phenomenon are not so dramatic. Often it can result in a higher level of residual noise rather than full instability. Additionally, the effects can often be seen without the introduction of an external disturbance due simply to noise created by the training process.

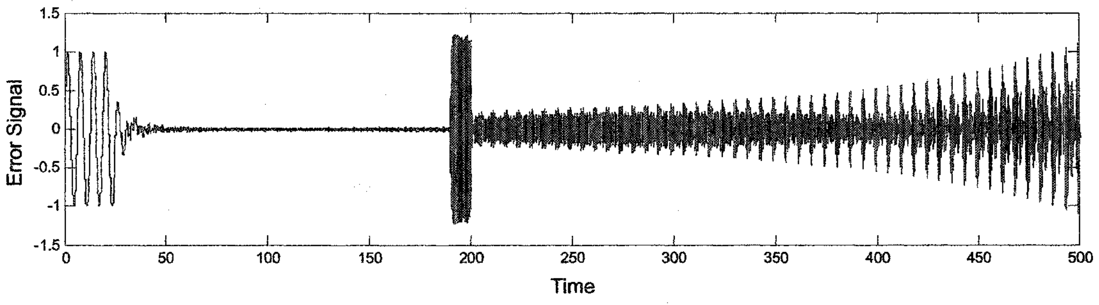


Figure 3.12 – Error signal for the case of a second order linear secondary path.

A means to eliminate this potential instability is provided by the band limited CMAC presented in figure 3.13. In this algorithm, the input to the CMAC is filtered by a linear bandpass filter designed to pass only the narrow frequency band of interest. Additionally, the output of the CMAC is filtered so that it is unable to produce frequencies outside of the range of interest. By using linear phase digital FIR filter implementations, the total phase response of the filters is simply a time delay. This time delay can then be added to the plant phase estimate time delay. As a result of the band pass filters, the CMAC input and output are confined to the frequency band of interest. This reduces the possibility that the CMAC training process will introduce noise which will result in unstable learning. Additionally, as a result of the input and output filtering, noise from external sources will be eliminated from the feedback path as well.

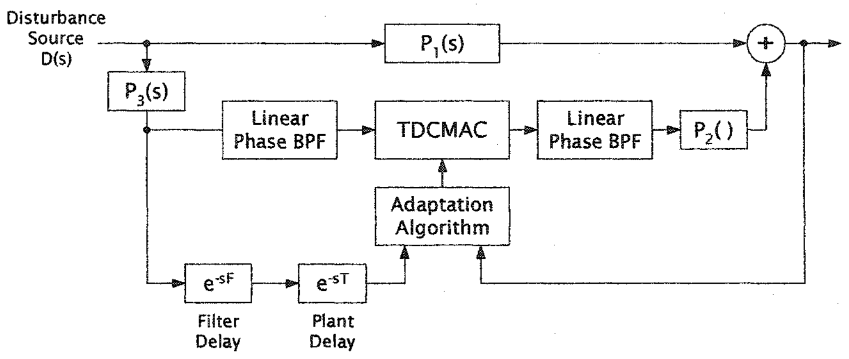


Figure 3.13 – Band-limited CMAC Compensator.

Figure 3.14 depicts simulation results for the band limited CMAC implemented on the same secondary path as used in the simulation of figure 3.12. Notice that in this case the introduction of external noise at the CMAC output over the time interval of t=350 to t=375 no longer results in instability of the learning process.

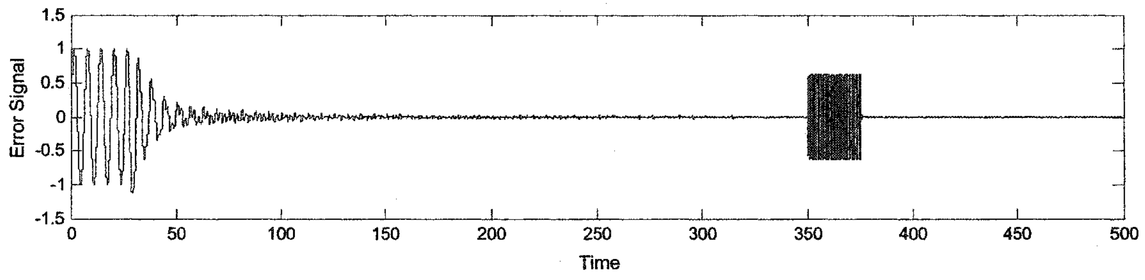


Figure 3.14 – Simulation of the band limited CMAC.

3.5 Time Delay Secondary Path

In the previous section, it was shown that the presence of a time delay in the secondary path could be accommodated by the learning algorithm through the addition of a model of that delay in the training input. In this section the convergence of this algorithm is considered in more detail. In general the presence of the delay complicates the convergence proof considerably. In fact, it will be shown the convergence no longer depends solely on the learning gain but also on the degree of overlap in neighboring inputs, the length of the time delay, and the rate of variation in the error signal.

Convergence of the algorithm will be considered in the context of figure 3.15. The forward path is represented by the nonlinear dynamics of time delay CMAC A and linear gain G_1. The secondary path is modeled by the linear gain G_3 and a pure transport delay of M discrete time intervals. The compensator, realized by CMAC B, is represented by two conceptual networks to indicate that the weight adaptation is performed using a different input than is used for generating the compensator output. Specifically, the input used for the weight adaptation is a delayed version of the compensator input where the delay is chosen to match the secondary path delay. It is assumed that all parameters of the compensator CMAC are identical to the corresponding parameters of the forward model CMAC.

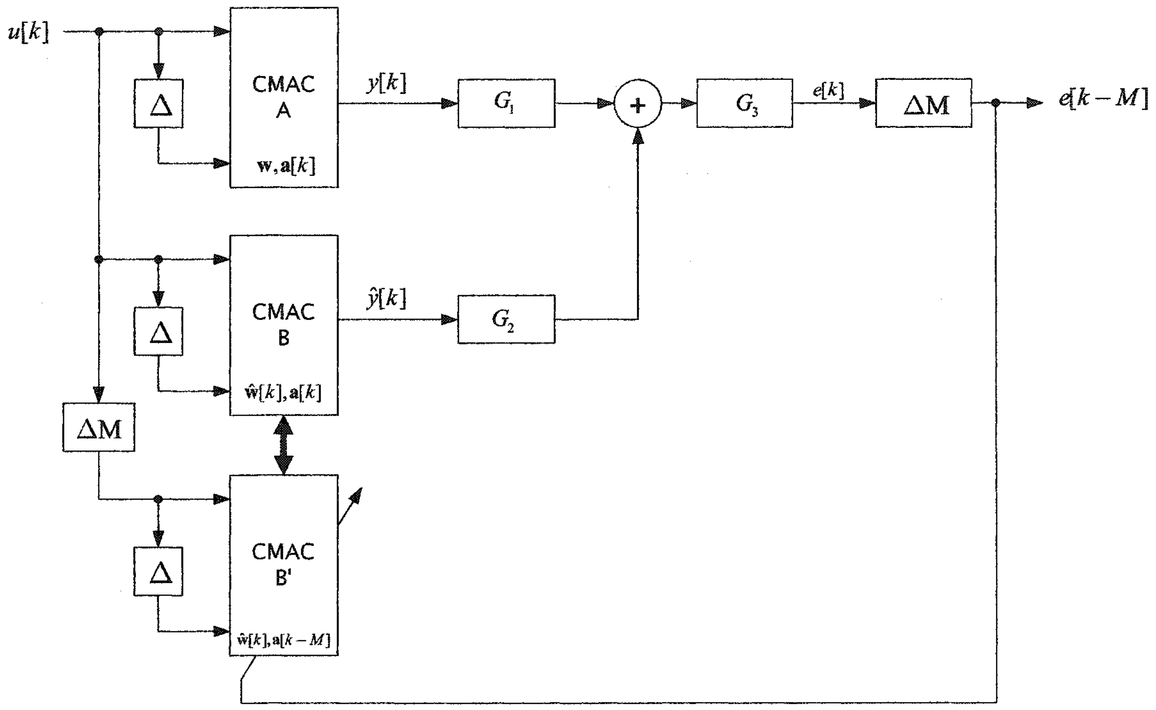


Figure 3.15 – CMAC Disturbance Cancellation with Secondary Path Time Delay.

The reference and compensator CMAC outputs are given by equations (3.43) and (3.44), respectively. The error signal is calculated via equation (3.45) and the compensator adaptation is performed using the weight update given in (3.46). Notice that the weight selection vector used for the weight update, $\mathbf{a}[k-M]$, is simply a delayed version of the selection vector used for computing the compensator output.

$$y[k] = \mathbf{a}[k]^T \mathbf{w} \tag{3.43}$$

$$\hat{y}[k] = \mathbf{a}[k]^T \hat{\mathbf{w}}[k] \tag{3.44}$$

$$e[k] = G_3\left(G_1 y[k] - G_2 \hat{y}[k]\right) \tag{3.45}$$

$$\hat{\mathbf{w}}[k+1] = \hat{\mathbf{w}}[k] + \frac{\alpha}{\beta}\mathbf{a}[k-M]e[k-M] \tag{3.46}$$

For any given input vector, the CMAC network references a set of β weights which lie within the selected receptive field. In the case where the same weights are referenced by two different inputs, there is said to be a receptive field overlap between those two training inputs. In the present architecture, the CMAC parameters are generally picked such that there is significant overlap between inputs which occur nearby in time. This provides generalization between adjacent time instants which usually results in smoother and more rapid convergence. However, in the case of the time delay update algorithm, overlapping receptive fields can impair the stability of the algorithm. The reason for this is that any given weight update is not reflected in the error signal until M samples after the adjustment is made. Therefore, adjustments made on the weights referenced by multiple nearby inputs will be performed without knowledge of previous updates already implemented on those weights. As a result, weights which are referenced by multiple nearby points can become over-compensated leading to instability in the adaptation process.

It will be shown that conditions for guaranteeing the convergence of this algorithm can be established. In general the learning gain must be reduced based on the degree of receptive field overlap, the amount of secondary path time delay, and the rate of variation in the error signal. However, before considering the general convergence result, a simplified case is considered where it is assumed that there is no receptive field overlap between any two CMAC inputs within any M time step interval. In this case, it will be shown that the algorithm is convergent as long as the learning gain is chosen such that $0 < \alpha < \frac{2}{G_2 G_3}$. Therefore, if there is no receptive field overlap, the learning gain bound is the same as in the case where there is no time delay present. This indicates that the potential deleterious effects to the convergence are directly a result of overlapping weights. In practice, overlapping receptive fields could be eliminated in most cases by choosing sufficiently fine input quantization and sufficiently small generalization with respect to the spacing of points in the sampled disturbance signal. However, such parameter choices would limit the CMAC generalization to the point where the convergence would be adversely affected.

Theorem 3.3 – CMAC Convergence with a Time Delay Secondary Path for the case of Non-Overlapping Receptive Fields: Consider the feedforward disturbance cancellation system of figure 3.15 with dynamics described by equations (3.43)-(3.46). Implicit in these equations is the fact that CMAC A and CMAC B have identical architectures. Let $\beta > 0$ denote the common CMAC generalization parameter. It is assumed that there is no receptive field overlap in any M time step window. More precisely, it is required that $\mathbf{a}[k]^T \mathbf{a}[k-i] = 0 \quad \forall i \in \{1, 2, \cdots, M\}$. It is assumed that $G_1 > 0$, $G_2 > 0$, and $G_3 > 0$. Under these conditions, if the learning gain parameter is chosen such that $0 < \alpha < \frac{2}{G_2 G_3}$ then $\lim_{k \to \infty} e[k] = 0$ showing that the algorithm is convergent.

Proof:

Let $\mathbf{D}[k]$ and $V[k]$ be defined by (3.47) and (3.48) and notice that $V[k]$ is non-negative by its definition.

$$\mathbf{D}[k] \triangleq G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k] \tag{3.47}$$

$$V[k] \triangleq \mathbf{D}[k]^T \mathbf{D}[k] \geq 0 \tag{3.48}$$

An expression for the increment in $V[k]$ can be derived as shown by the following expressions, utilizing the fact that $\mathbf{a}[k-M]^T \mathbf{a}[k-M] = \beta$.

$$V[k+1] = \mathbf{D}[k+1]^T \mathbf{D}[k+1] \tag{3.49}$$

$$= \{G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k+1]\}^T \{G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k+1]\} \tag{3.50}$$

$$= \left\{ G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k] - G_2 G_3 \frac{\alpha}{\beta} e[k-M] \mathbf{a}[k-M] \right\}^T \cdot \left\{ G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k] - G_2 G_3 \frac{\alpha}{\beta} e[k-M] \mathbf{a}[k-M] \right\} \tag{3.51}$$

$$V[k+1] - V[k] = \frac{-2G_2 G_3 \alpha}{\beta} e[k-M] \mathbf{a}[k-M]^T (G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k]) + \frac{\alpha^2 G_2^2 G_3^2}{\beta} e^2[k-M] \tag{3.52}$$

Notice that the last term in (3.52) contains the product of the weight selection vector at time $k-M$ and the weight vectors at time k. To simplify this expression, it is necessary to derive an expression for the compensator weight vector at time k as a function of the weight vector at time $k-M$. To that end, consider equations (3.53) – (3.55) which depict the weight updates performed at consecutive time instants.

$$\hat{\mathbf{w}}[k] = \hat{\mathbf{w}}[k-1] + \frac{\alpha}{\beta} \mathbf{a}[k-M-1] e[k-M-1] \tag{3.53}$$

$$\hat{\mathbf{w}}[k-1] = \hat{\mathbf{w}}[k-2] + \frac{\alpha}{\beta} \mathbf{a}[k-M-2] e[k-M-2] \tag{3.54}$$

$$\vdots$$

$$\hat{\mathbf{w}}[k-M+1] = \hat{\mathbf{w}}[k-M] + \frac{\alpha}{\beta} \mathbf{a}[k-2M] e[k-2M] \tag{3.55}$$

These equations can be combined into a single expression by starting with (3.55) and then back-substituting into each of the previous equations. The result of this operation given in (3.56) represents the total change in the compensator weight vector from time $k-M$ to time k.

$$\hat{\mathbf{w}}[k] = \hat{\mathbf{w}}[k-M] + \frac{\alpha}{\beta} \sum_{i=1}^{M} \mathbf{a}[k-M-i] e[k-M-i] \tag{3.56}$$

Substituting this result into equation (3.51) yields the modified $V[k]$ update equations given in (3.57) and (3.58).

$$\begin{aligned} V[k+1] - V[k] &= \frac{-2G_2 G_3 \alpha}{\beta} e[k-M] \mathbf{a}[k-M]^T (G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k-M]) + \\ &\frac{2\alpha^2 G_2^2 G_3^2}{\beta^2} e[k-M] \mathbf{a}[k-M]^T \sum_{i=1}^{M} \mathbf{a}[k-M-i] e[k-M-i] + \frac{\alpha^2 G_2^2 G_3^2}{\beta} e^2[k-M] \end{aligned} \tag{3.57}$$

$$V[k+1]-V[k]=\frac{-2G_2G_3\alpha}{\beta}e^2[k-M]+\frac{\alpha^2G_2^2G_3^2}{\beta}e^2[k-M] \\ +\frac{2\alpha^2G_2^2G_3^2}{\beta^2}\sum_{i=1}^{M}\mathbf{a}[k-M]^T\mathbf{a}[k-M-i]e[k-M]e[k-M-i] \tag{3.58}$$

However, given the assumption of non-overlapping receptive fields, $\mathbf{a}[k]^T\mathbf{a}[k-i]=0\ \forall i\in\{1,2,\cdots,M\}$, the expression in (3.58) reduces to (3.59).

$$V[k+1]-V[k]=\frac{G_2^2G_3^2\alpha^2}{\beta}e^2[k-M]-\frac{2G_2G_3\alpha}{\beta}e^2[k-M] \\ =\frac{G_2^2G_3^2\alpha}{\beta}\left(\alpha-\frac{2}{G_2G_3}\right)e^2[k-M]\le 0 \tag{3.59}$$

Since $V[k]$ is a decreasing sequence by (3.59) and since $V[k]$ is bounded below by zero, the conditions of Lemma 3.1 are met and statement (3.60) follows.

$$\lim_{k\to\infty}\frac{G_2^2G_3^2\alpha}{\beta}\left[\alpha-\frac{2}{G_2G_3}\right]e^2[k-M]=0 \tag{3.60}$$

Since the multiplicative constant of $e^2[k-M]$ in (3.60) is strictly non-zero, this implies $\lim_{k\to\infty}e^2[k-M]=0$ and therefore, $\lim_{k\to\infty}e[k]=0$. □

In order to obtain a result a more general result, it is necessary to impose a bound on the rate of change of the error signal. Specifically, it is assumed that there exists some value of R which serves as a bound on the rate of change of the error signal over any M time step interval as given in (3.61).

$$|e[k-i]|\le R|e[k]| \quad \begin{array}{l}\forall i\in\{1,2,\cdots,M\}\\ \forall k\ni e[k]\ne 0\end{array} \tag{3.61}$$

The value of R can be made arbitrarily large to accommodate rapidly varying error signals. Initially, with no contribution from the compensator, the error signal is completely specified by the influence of the disturbance signal at the error sensor. Therefore, this signal can be utilized to estimate the value of R. Additionally, if the convergence is reasonably uniform, this value will be a reasonable estimate of the parameter through adaptation of the compensator.

It is assumed that the receptive field overlap in any M time step interval is bounded by some constant ϕ. Notice that this is not a restriction since it is possible to choose $\phi = \beta$ to allow for the maximum possible overlap. Under these conditions, it will be shown that the algorithm is convergent if the learning gain is chosen subject to the bound given in (3.62). The proof of this result is presented in Theorem 3.4.

$$0 < \alpha < \frac{2}{G_2 G_3 \left[1 + 2MR\left(\frac{\phi}{\beta}\right)\right]} \tag{3.62}$$

This bound indicates that to ensure convergence, the learning gain must be reduced given increased length of the delay M, the rate of change of the error signal R, and the degree of overlap between receptive fields ϕ. In the case of zero time delay this result reduces to the bound presented in section two for the case of a linear gain secondary path. Additionally, for zero overlap ($\phi = 0$) this result reduces to the bound presented for the case of non-overlapping receptive fields given in Theorem 3.3.

Theorem 3.4 – CMAC Convergence with a Time Delay Secondary Path: Consider the feedforward disturbance cancellation system of figure 3.15 with dynamics described by equations (3.43)-(3.46). Implicit in these equations is the fact that CMAC A and CMAC B have identical architectures. Let $\beta > 0$ denote the common CMAC generalization parameter. It is assumed that there exists some constant R such that $|e[k-i]| \leq R|e[k]|$ for all $i \in \{1, 2, \cdots, M\}$ and for all k such that $e[k] \neq 0$. Additionally, it is assumed that the maximum receptive field overlap in any M time step window is limited to ϕ. More precisely, it is required that $\mathrm{a}[k]^T \mathrm{a}[k-i] \leq \phi \ \ \forall i \in \{1, 2, \cdots, M\}$. Additionally, let $G_2 > 0$ and

$G_3 > 0$. Under these conditions, and with the learning gain selected according to the bound (3.62), then $\lim_{k\to\infty} e[k] = 0$ showing that the system is convergent.

$$0 < \alpha < \frac{2}{G_2 G_3 \left[1 + 2MR\left(\frac{\phi}{\beta}\right)\right]} \tag{3.63}$$

Proof:

Let $\mathbf{D}[k]$ and $V[k]$ be defined by (3.64) and (3.65) and notice that $V[k]$ is non-negative by its definition.

$$\mathbf{D}[k] \triangleq G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k] \tag{3.64}$$

$$V[k] \triangleq \mathbf{D}[k]^T \mathbf{D}[k] \geq 0 \tag{3.65}$$

The value of $V[k]$ at time $k+1$ can be expressed using the weight update equation of (3.46) to yield the following expressions in (3.66)-(3.68).

$$V[k+1] = \{G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k+1]\}^T \{G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k+1]\} \tag{3.66}$$

$$\begin{aligned} &= \left\{G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k] - G_2 G_3 \frac{\alpha}{\beta} e[k-M]\mathbf{a}[k-M]\right\}^T \\ &\quad \left\{G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k] - G_2 G_3 \frac{\alpha}{\beta} e[k-M]\mathbf{a}[k-M]\right\} \end{aligned} \tag{3.67}$$

$$\begin{aligned} &= V[k] + G_2^2 G_3^2 \frac{\alpha^2}{\beta^2} e^2[k-M]\mathbf{a}[k-M]^T \mathbf{a}[k-M] - \\ &\quad 2G_2 G_3 \frac{\alpha}{\beta} e[k-M]\mathbf{a}[k-M]^T \{G_1 G_3 \mathbf{w} - G_2 G_3 \hat{\mathbf{w}}[k]\} \end{aligned} \tag{3.68}$$

It follows directly from the CMAC algorithm that $\mathbf{a}[k-M]^T \mathbf{a}[k-M] = \beta$. Using this fact and the previous result of (3.56), the expression in (3.68) can be rewritten to yield (3.69).

$$\begin{aligned} V[k+1] - V[k] &= \frac{G_2^2 G_3^2 \alpha^2}{\beta} e^2[k-M] - \frac{2G_2 G_3 \alpha}{\beta} e^2[k-M] \\ &+ \frac{2G_2^2 G_3^2 \alpha^2}{\beta^2} \sum_{i=1}^{M} \mathbf{a}[k-M]^T \mathbf{a}[k-M-i] e[k-M] e[k-M-i] \end{aligned} \tag{3.69}$$

From the assumptions on the maximum overlap and the maximum rate of change of the error signal, inequalities (3.70) and (3.71) can be written.

$$\mathrm{a}[k-M]^T\mathrm{a}[k-M-i] \le \phi \quad \forall\, i \in \{1,2,\cdots,M\} \tag{3.70}$$

$$e[k-M-i]e[k-M] \le |e[k-M-i]e[k-M]| \le Re^2[k-M] \tag{3.71}$$

These results can then be utilized to obtain a bound on the expression in (3.69) given by expression (3.72). The second inequality is due to the bound on the learning gain.

$$\begin{aligned} V[k+1]-V[k] &\le \frac{G_2^2G_3^2\alpha^2}{\beta}e^2[k-M] - \frac{2G_2G_3\alpha}{\beta}e^2[k-M] + \frac{2G_2^2G_3^2\alpha^2}{\beta^2}M\phi Re^2[k-M] \\ &= \frac{G_2^2G_3^2\alpha}{\beta}\left(1+\frac{2MR\phi}{\beta}\right)\left(\alpha - \frac{2}{G_2G_3\left[1+\frac{2MR\phi}{\beta}\right]}\right)e^2[k-M] \le 0 \end{aligned} \tag{3.72}$$

Since $V[k]$ is a decreasing sequence by (3.72) and since $V[k]$ is bounded below by zero, the conditions of Lemma 3.1 are met and statement (3.73) follows.

$$\lim_{k\to\infty}\left(V[k+1]-V[k]\right) = 0 \tag{3.73}$$

By the sequence squeeze theorem and the inequality in (3.72) the limit given in (3.74) exists.

$$\lim_{k\to\infty}\frac{G_2^2G_3^2\alpha}{\beta}\left(1+2MR\frac{\phi}{\beta}\right)\left[\alpha - \frac{2}{G_2G_3\left(1+\frac{2MR\phi}{\beta}\right)}\right]e^2[k-M] = 0 \tag{3.74}$$

Since the multiplicative constant of $e^2[k-M]$ in (3.74) is strictly non-zero, this implies $\lim_{k\to\infty} e^2[k-M] = 0$ and it follows directly that $\lim_{k\to\infty} e[k] = 0$. □

It is expected that the learning gain bound derived in Theorem 3.4 is conservative since the average overlap will generally be below the maximum overlap specified by the parameter ϕ. Additionally, the parameter predicting the maximum rate of change of the error signal, R, must be chosen for the worst case variation in the error signal and most often, the actual variation will be less. These derived bounds are sufficient to guarantee the stability of the algorithm in each case. However, they do not represent necessary conditions.

As a result, the derived bounds could be quite conservative in nature. A brief consideration of this subject is provided via comparison of the analytical bounds with simulation results for the case of a time delay secondary path.

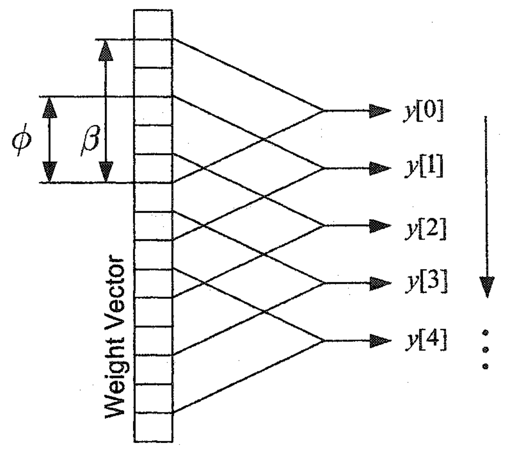


Figure 3.16 – Single Dimension CMAC used in simulations.

The presented simulations are for the single dimension CMAC as depicted conceptually in Figure 3.16. The generalization width is $\beta = 30$ and the input signal is designed to sequentially activate receptive fields with an overlap of exactly ϕ weights. The secondary path is comprised of a pure time delay of M samples. Notice that this simulation corresponds to the block diagram of figure 3.15 with the gains chosen as $G_2 = 1$ and $G_3 = 1$. The maximum stable learning gain for a given set of parameter values ϕ and M was determined using a binary search algorithm over the range $\alpha \in [0,2]$.

Figure 3.17 depicts the maximum stable learning gain as a function of the degree of overlap for two different values of the secondary path delay, M. In each case, the solid line represents the analytical result calculated via equation (3.63) while the dotted line is the numerically computed bound from simulation. From these results, it is evident that in cases where the overlap is comparable to the generalization width, the analytical and simulation bounds are in close agreement. However, in the case of long secondary path delays and small values of overlap, the analytical bound underestimates the corresponding simulation result.

The explanation for this discrepancy is in the worst case assumption made by the analytical bound that all receptive fields within any M time step interval necessarily overlap all other fields within that window by an amount ϕ. In practice, it is unlikely that such complete overlap will actually occur. In the present case of a single dimension CMAC, it is not even possible to achieve such an extent of overlapping receptive fields. In the case of a multi-dimensional CMAC it is possible to have a greater extent of overlap. However, it still becomes less likely to have complete receptive field overlap as the secondary delay increases or extent of overlap decreases.

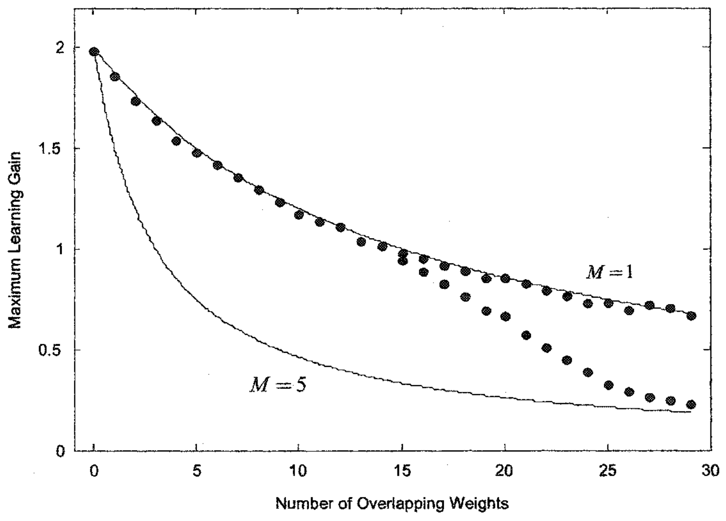


Figure 3.17 – Numerical results for time delay secondary path.

Figure 3.18 depicts the maximum stable learning gain as a function of the secondary path delay for two different values of the overlap parameter, ϕ. For the reasons previously described, the analytical bound tends to be more conservative as the overlap decreases or the secondary path delay increases. For example, in the case of $\phi = 5$, there is no overlap between samples for a time delay of more than one. As a result, the maximum stable learning gain does not decrease for delays of longer duration. In contrast, the analytical result makes the assumption that even in the case of longer delays, all receptive fields within any M sample window will have $\phi = 5$ overlapping weights. As a result, the analytical bound is more conservative. Tighter analytical bounds could be attained by utilizing more realistic models of the overlap distribution throughout the M step time window. However, such results would require more assumptions to be made regarding the nature of the input signal.

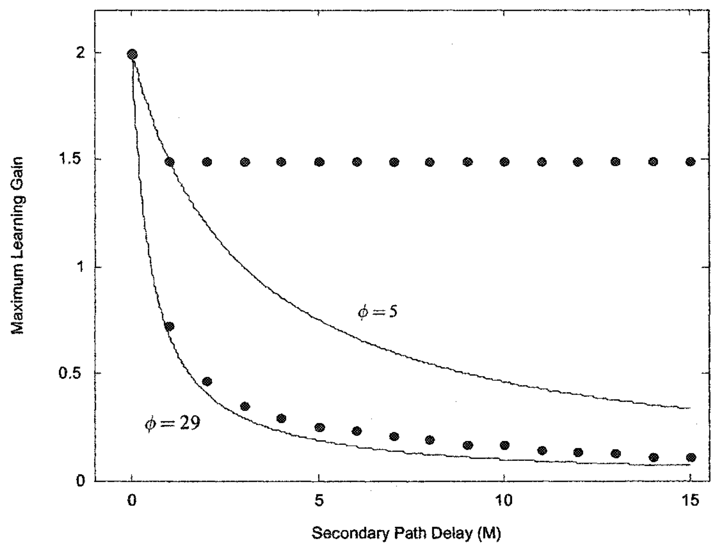


Figure 3.18 – Numerical and analytical stability bounds.

In conclusion, the presented bounds provide a guarantee of stability in the case of sufficiently small learning gain. Additionally, the derived results provide an indication of the variation of the maximum learning gain with the degree of overlap, generalization width, rate of change of the error signal, and secondary path delay. These parameters can be estimated based on the characteristics of the system and used to appropriately de-rate the learning gain to ensure convergence.

3.6 The Filtered-X Backpropagation Algorithm

This section presents a method for utilizing the multi-layer perceptron (MLP) in feedforward disturbance cancellation for cases where the secondary path is nonlinear and history dependent. This approach was developed by Snyder and Tanaka [97] and more recent refinements have been proposed to improve its convergence [11]. It is included here as background material, but also to serve as inspiration for the Filtered-X CMAC algorithms to be introduced later in this chapter. The basic architecture used by the Filtered-X Backpropagation algorithm is shown in figure 3.19.

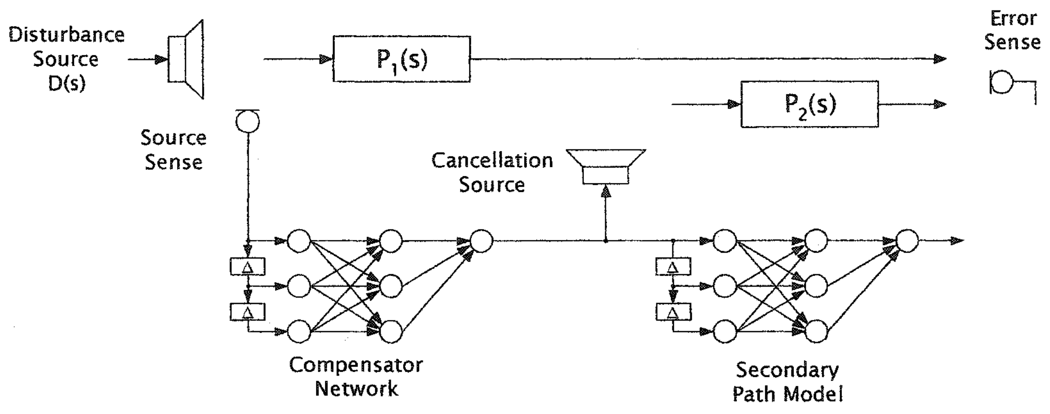


Figure 3.19 – The Filtered-X Backpropagation Algorithm.

The Filtered-X Backpropagation algorithm utilizes a time delay MLP network to serve as a general nonlinear dynamical compensator. The most common technique for training a MLP network is the backpropagation algorithm which is based on a gradient descent of the error surface. That is, each weight is adjusted in a manner which reduces the instantaneous error in the network output. However, in the system of figure 3.19, the error in the compensator output is not directly known since the only available error signal is located at the output of the secondary path. The solution proposed in the Filtered-X Backpropagation algorithm utilizes a second time delay MLP network to model the secondary path. This secondary path model is adapted off-line and its weights are assumed to be constant during adaptation of the compensator. Assuming that this neural network is an accurate model of the actual secondary path, it can then be utilized to backpropagate the observed error signal to the compensator. The resultant weight update rule for the compensator will be derived in this section. However, the approach utilized here differs from that used in the original derivation. This alternate approach, though only valid for a fixed architecture, yields qualitative insight into the algorithm and provides a closer analogy to the Filtered-X CMAC algorithm which will be introduced in the following section.

The weight update utilized in the Filtered-X Backpropagation algorithm is an extension of the standard backpropagation algorithm utilized for training a standard feedforward MLP [44][90]. Consider the single output, two layer MLP network as shown in figure 3.20. The neuron activation function, $f(\cdot)$, is assumed to be a differentiable sigmoidal function. There are two layers of network weights. In layer one, weight $w_{ij}[k]$ connects input i to hidden layer neuron j. In the second weight layer, weight $v_l[k]$ connects the output of hidden layer neuron l to the output neuron. It is assumed that there are M inputs to the network and N hidden layer neurons.

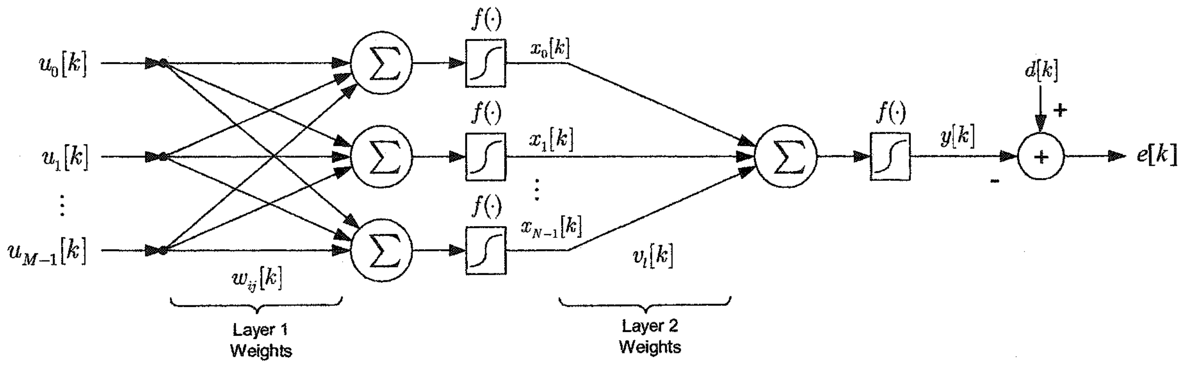


Figure 3.20 – A two layer MLP network with a single output.

With appropriate network weight values, the MLP network can represent a desired static mapping between an input vector $\begin{bmatrix} u_0[k] & u_1[k] & \cdots & u_{M-1}[k] \end{bmatrix}$ and an output value $y[k]$. The operational equations for this network are given in (3.75) and (3.76).

$$y[k] = f\left(\sum_{l=0}^{N-1} v_l[k] x_l[k]\right) \tag{3.75}$$

$$x_l[k] = f\left(\sum_{i=0}^{M-1} w_{il}[k] u_i[k]\right) \quad \forall\, l \in \{0,1,\cdots,N-1\} \tag{3.76}$$

The weight update algorithm must be derived separately for each layer. For a weight $v_l[k]$ in layer two, the weight update will take the form of (3.77) where η represents a learning gain parameter.

$$v_l[k+1] = v_l[k] - \eta \frac{\partial e^2[k]}{\partial v_l[k]} \quad \forall\, l \in \{0,1,\cdots,N-1\} \tag{3.77}$$

Let $f'(\cdot)$ denote the derivative of the activation function. Then, the partial derivative in the weight update equation can be reduced via the chain rule and equation (3.75) to produce the result given in (3.78) and (3.79).

$$\frac{\partial e^2[k]}{\partial v_l[k]} = 2e[k]\frac{\partial}{\partial v_l[k]}(d[k]-y[k]) = -2e[k]\frac{\partial}{\partial v_l[k]} f\left(\sum_{i=0}^{N-1} v_i[k]x_i[k]\right) \tag{3.78}$$

$$= -2e[k]f'\left(\sum_{i=0}^{N-1} v_i[k]x_i[k]\right)x_l[k] \tag{3.79}$$

The weight update algorithm for each weight contained in the first layer is of the form (3.80) where η again represents a learning gain parameter. The partial derivative can be represented as given by (3.81) and reduced to the expression (3.85).

$$w_{hj}[k+1] = w_{hj}[k] - \eta \frac{\partial e^2[k]}{\partial w_{hj}[k]} \quad \begin{array}{l} \forall\, h \in \{0,1,\cdots,M-1\} \\ \forall\, j \in \{0,1,\cdots,N-1\} \end{array} \tag{3.80}$$

$$\frac{\partial e^2[k]}{\partial w_{hj}[k]} = 2e[k]\frac{\partial}{\partial w_{hj}[k]}(d[k]-y[k]) = 2e[k]\frac{\partial}{\partial w_{hj}[k]}\left\{-f\left(\sum_{i=0}^{N-1} v_i[k]x_i[k]\right)\right\} \tag{3.81}$$

$$= -2e[k]f'\left(\sum_{i=0}^{N-1} v_i[k]x_i[k]\right)\frac{\partial}{\partial w_{hj}[k]}\sum_{i=0}^{N-1} v_i[k]x_i[k] \tag{3.82}$$

$$= -2e[k]f'\left(\sum_{i=0}^{N-1} v_i[k]x_i[k]\right)\frac{\partial}{\partial w_{hj}[k]}\sum_{i=0}^{N-1} v_i[k]f\left(\sum_{l=0}^{M-1} w_{li}[k]u_l[k]\right) \tag{3.83}$$

$$= -2e[k]f'\left(\sum_{i=0}^{N-1} v_i[k]x_i[k]\right)\sum_{i=0}^{N-1} v_i[k]f'\left(\sum_{l=0}^{M-1} w_{li}[k]u_l[k]\right)\frac{\partial}{\partial w_{hj}[k]}\sum_{l=0}^{M-1} w_{li}[k]u_l[k] \tag{3.84}$$

$$= -2e[k]f'\left(\sum_{i=0}^{N-1} v_i[k]x_i[k]\right)v_j[k]f'\left(\sum_{l=0}^{M-1} w_{lj}[k]u_l[k]\right)u_h[k] \tag{3.85}$$

Given these results, the weight updates can be written as (3.86) and (3.87) for first and second layer weights respectively. In each case, the factor of two has been absorbed into the learning gain coefficient.

$$w_{hj}[k+1] = w_{hj}[k] + \eta e[k] f'\left(\sum_{i=0}^{N-1} v_i[k] x_i[k]\right) v_j[k] f'\left(\sum_{l=0}^{M-1} w_{lj}[k] u_l[k]\right) u_h[k] \tag{3.86}$$

$$v_j[k+1] = v_j[k] + \eta e[k] f'\left(\sum_{i=0}^{N-1} v_i[k] x_i[k]\right) x_j[k] \tag{3.87}$$

These coefficient update equations can then be utilized to adapt the coefficients of the network shown in figure 3.20 to minimize the mapping error across a training data set. Notice that there are terms which are common to both weight updates. Standard implementations of the gradient descent learning usually utilize a backpropagation of these common terms to reduce the computational overhead of the algorithm. The general backpropagation algorithm for any number of hidden layers can be found in numerous sources [44][90].

The off-line adaptation of the secondary path model can be performed by direct implementation of the standard backpropagation algorithm. However, the backpropagation algorithm cannot be directly utilized to adapt the weights of the compensator network since the error associated with the compensator output is unobservable. Instead, a modified weight update must be derived based on a gradient descent of the observable error at the output of the secondary path. Figure 3.21 presents a alternative block diagram to the system of figure 3.19. In this figure, multiple conceptual compensator networks have been utilized to eliminate the tapped delay line associated with the secondary path model. Each conceptual network represents the operation of the actual compensator at a delayed instant of time. Since the weights are updated on every discrete time interval, each conceptual network has a slightly different set of weights. For example, the weight $b_j[k]$ in network I becomes weight $b_j[k-1]$ in network II, the value of the same weight at the last discrete time step. Notice however, that this block diagram is mathematically equivalent to the actual block diagram of figure 3.19. The variable M represents the number of taps in the tapped delay line of the secondary path model. Since each input to the secondary path model is fed by a separate instantiation of the compensator network, there are exactly M conceptual compensator networks. The tapped delay line of the compensator has P taps and the compensator has Q hidden layer neurons.

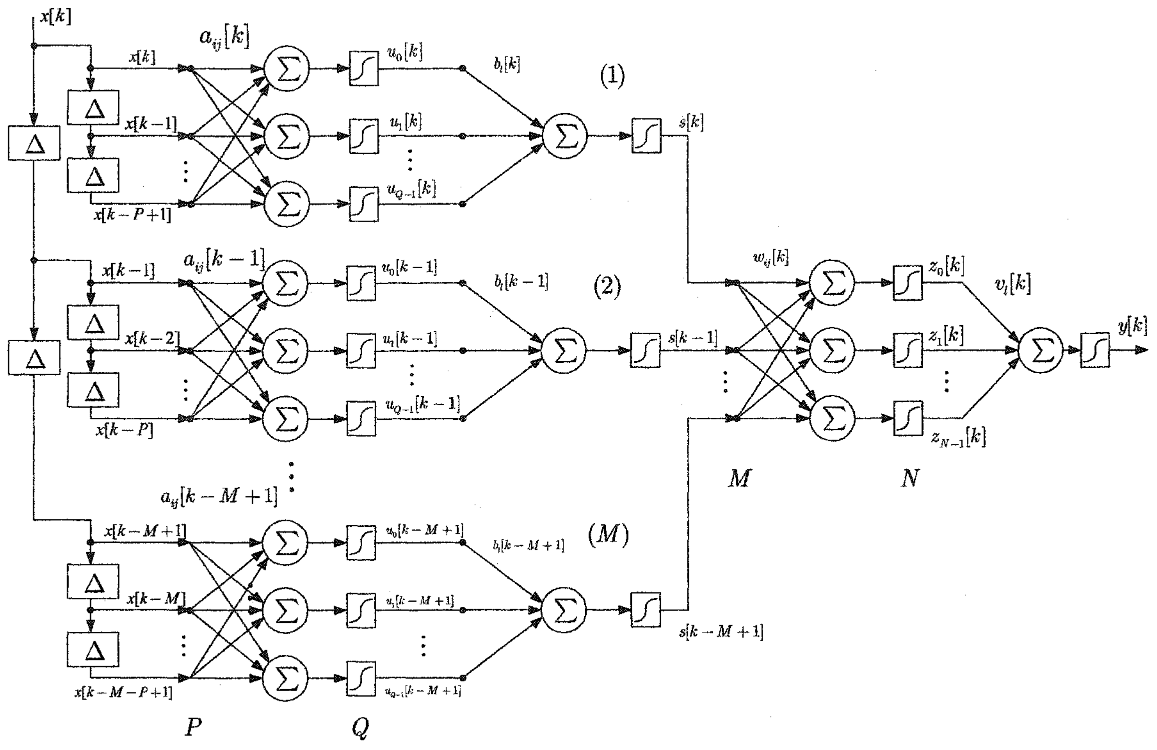


Figure 3.21 – Filtered-X BP Compensator Unwrapped in Time.

In figure 3.21, the weight $a_{ij}[k]$ represents the connection between compensator input i and hidden neuron j while weight $b_j[k]$ represents the connection between neuron j and the output neuron of the compensator. Additionally, let $\hat{r}$ denote the value of the respective signal r before the sigmoidal nonlinearity. That is, $\hat{u}_0[k]$ is defined by $\hat{u}_0[k] = f(\hat{u}_0[k])$. With these definitions, the output of the network is given by (3.88). Notice that since the weights of the secondary path model are assumed to be fixed, they are represented without a time index.

$$y[k] = f\left(\sum_{e=0}^{N-1} v_e z_e[k]\right) \tag{3.88}$$

The output, $z_e[k]$, of each hidden layer neuron in the secondary path model is given by (3.89). The output of each instantiation of the compensator network, $s[k-g]$, is expressed in (3.90). Finally, equation (3.91) describes the output of each hidden layer neuron in each instantiation of the compensator. Thus, equations (3.89)-(3.91) fully describe the mapping through the aggregate signal path comprised of the compensator and secondary path model.

$$z_e[k] = f\left(\sum_{g=0}^{M-1} w_{ge} s[k-g]\right) \tag{3.89}$$

$$s[k-g] = f\left(\sum_{h=0}^{Q-1} b_h[k-g] u_h[k-g]\right) \tag{3.90}$$

$$u_h[k-g] = f\left(\sum_{m=0}^{P-1} a_{mh}[k-g] x[k-g-m]\right) \tag{3.91}$$

Given the analytical description of this mapping in (3.89)-(3.91), it is possible to derive a gradient descent weight update rule for each layer of the compensator network. This will be performed independently for each layer of weights in the compensator network. For any given weight in the second layer of the compensator, the weight update will take the form (3.92) where $e[k]$ is not the error in the compensator output but rather the error as measured at the output of the secondary path. In active disturbance cancellation systems, the desired output is zero and therefore, $e[k] = -y[k]$.

$$b_j[k+1] = b_j[k] - \eta \frac{\partial e^2[k]}{\partial b_j[k]} \tag{3.92}$$

The partial derivative term of (3.92) can be rewritten using the chain rule to generate equation (3.93). Notice that $\hat{y}[k]$ represents the signal in the output neuron immediately after the summing junction.

$$\frac{\partial e^2[k]}{\partial b_j[k]} = -2e[k]\frac{\partial}{\partial b_j[k]} f\left(\sum_{e=0}^{N-1} v_e z_e[k]\right) = -2e[k] f'(\hat{y}[k]) \frac{\partial}{\partial b_j[k]} \sum_{e=0}^{N-1} v_e z_e[k] \tag{3.93}$$

Using (3.89) to replace $z_e[k]$ yields expressions (3.94)-(3.96).

$$= -2e[k] f'(\hat{y}[k]) \frac{\partial}{\partial b_j[k]} \sum_{e=0}^{N-1} v_e f(\hat{z}_e[k]) = -2e[k] f'(\hat{y}[k]) \sum_{e=0}^{N-1} v_e f'(\hat{z}_e[k]) \frac{\partial}{\partial b_j[k]} \sum_{g=0}^{M-1} w_{ge} s[k-g] \tag{3.94}$$

$$= -2e[k] f'(\hat{y}[k]) \sum_{e=0}^{N-1} v_e f'(\hat{z}_e[k]) \sum_{g=0}^{M-1} w_{ge} \frac{\partial}{\partial b_j[k]} f(\hat{s}[k-g]) \tag{3.95}$$

$$= -2e[k]f'(\hat{y}[k])\sum_{e=0}^{N-1} v_e f'(\hat{z}_e[k])\sum_{g=0}^{M-1} w_{ge} f'(\hat{s}[k-g])\frac{\partial}{\partial b_j[k]}\sum_{h=0}^{Q-1} b_h[k-g]u_h[k-g] \tag{3.96}$$

Assuming that the weights change slowly with respect to the length of the secondary path tapped delay line, the approximation in (3.97) is valid.

$$\frac{\partial b_j[k-g]}{\partial b_j[k]} \cong \frac{\partial b_j[k]}{\partial b_j[k]} = 1 \quad \forall g \in \{0,1,\cdots,M-1\} \tag{3.97}$$

With this assumption, equation (3.96) can be rewritten as (3.98) and the weight update rule for the output layer weights of the compensator is given by (3.99).

$$= -2e[k]f'(\hat{y}[k])\sum_{e=0}^{N-1} v_e f'(\hat{z}_e[k])\sum_{g=0}^{M-1} w_{ge} f'(\hat{s}[k-g])\, u_j[k-g] \tag{3.98}$$

$$b_j[k+1] = b_j[k] + \eta e[k]f'(\hat{y}[k])\sum_{e=0}^{N-1} v_e f'(\hat{z}_e[k])\sum_{g=0}^{M-1} w_{ge} f'(\hat{s}[k-g])\, u_j[k-g] \tag{3.99}$$

In a similar fashion, given assumption (3.100), the weight update rule can be expressed as given in (3.101).

$$\frac{\partial a_{tn}[k-g]}{\partial a_{tn}[k]} \cong \frac{\partial a_{tn}[k]}{\partial a_{tn}[k]} = 1 \quad \forall g \in \{0,1,\cdots,M-1\} \tag{3.100}$$

$$\begin{aligned} &a_{tn}[k+1] = a_{tn}[k] + \\ &\eta e[k]f'(\hat{y}[k])\sum_{e=0}^{N-1} v_e f'(\hat{z}_e[k])\sum_{g=0}^{M-1} w_{ge} f'(\hat{s}[k-g])\sum_{h=0}^{Q-1} b_h f'(\hat{u}_j[k-g])\, x[k-g-t] \end{aligned} \tag{3.101}$$

Equations (3.99) and (3.101) represent the Filtered-X Backpropagation weight updates. These equations show that it is possible to perform a gradient descent update on the compensator using only the observed error signal. In a practical implementation, the redundant computations in these weight updates can be eliminated by backpropagating the common terms for use by both updates. In comparison to the standard backpropagation weight updates of (3.86) and (3.87), the Filtered-X update equations have an additional summation with M terms. In effect, this provides a parallel update of the M conceptual compensator networks.

The primary disadvantages of this approach are its large computational overhead and the possibility of convergence problems. The likelihood of convergence problems can be reduced by utilizing many refinements which have been made to the backpropagation algorithm including the addition of a momentum term in the learning gain and implementation of an adaptive learning rate. However, the computational overhead of the algorithm remains a significant concern in real time applications. This is particularly true given the significant order of typical secondary path models required for modeling time delays and resonant dynamics.

3.7 The Filtered-X CMAC Algorithm

The Filtered-X Backpropagation algorithm introduced in the previous section extends the Filtered-X LMS technique for use with a nonlinear compensator, in that case, the multi-layer perceptron. This provides the capability for effective disturbance cancellation with nonlinearities in the forward path as well as in the cancellation path. However, the Filtered-X Backpropagation algorithm suffers from several deficiencies intrinsic to the multi-layer perceptron. These include a slow rate of convergence, the potential lack of convergence due to the presence of local minima in the error surface, and the significant computational overhead of the algorithm which limits the maximum cancellation frequency in real-time implementations.

This section presents a method by which nonlinear feedforward disturbance cancellation can be implemented using a Time Delay CMAC neural network. The CMAC, described previously in section 2.3, is a type of radial basis function neural network which utilizes a uniform receptive field placement with static localized generalization. Through this architecture the CMAC attains fast and robust convergence at the expense of reduced generalization capabilities. This tradeoff is ideally suited to the case of active noise and vibration control where there is typically an abundance of training data available and additional training can be utilized to compensate for the limited generalization capabilities of the CMAC.

The Filtered-X CMAC algorithm utilizes two CMAC neural networks with tapped delay line inputs as shown in figure 3.22. CMAC A is used as a nonlinear compensator while CMAC B is trained off-line to represent the secondary path transfer function. The number of taps in each delay line must be chosen appropriately based on the nature of the system and the signal content of the disturbance source. This issue was previously discussed in section 2.4. In this section, the general case is considered using a compensator with N taps and a secondary plant model with M taps. For the system shown in figure 3.22, M=3 and N=2.

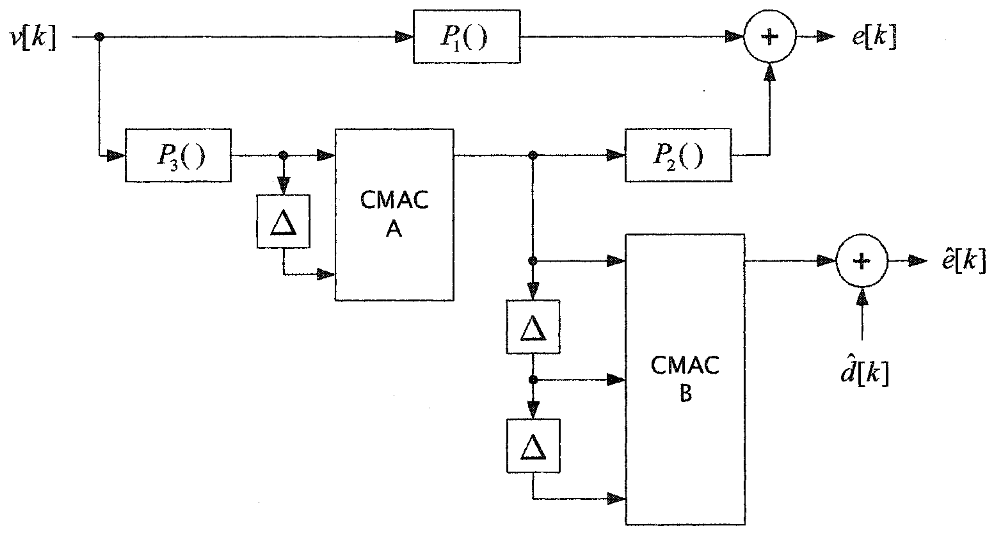


Figure 3.22 – The Filtered-X CMAC Algorithm.

The standard CMAC weight update requires a measure of the error in the network output. Since this signal is unavailable here, a means of computing the necessary compensator weight updates for the minimization of the observed error must be derived. If CMAC B is a perfect model of the secondary path then the observable error, $e[k]$, can be viewed as having been measured at the output of CMAC B as represented by signal $\hat{e}[k]$. In this case, $\hat{d}[k]$ represents the contribution of the primary disturbance as measured at the error sense. Thus, minimizing the instantaneous error signal $\hat{e}[k]^2$ will be the same as minimizing the error signal $e[k]^2$. The tapped delay line associated with the secondary path model can be conceptually eliminated by unwrapping the compensator network in time as shown in figure 3.23. The three conceptual networks represent the operation of the single actual compensator at the present time and all of the delayed time instants as required for input to the secondary path model. It is clear from this view that the current output error is due not only to the currently addressed compensator weights but also to the M sets of weights corresponding to previous compensator inputs.

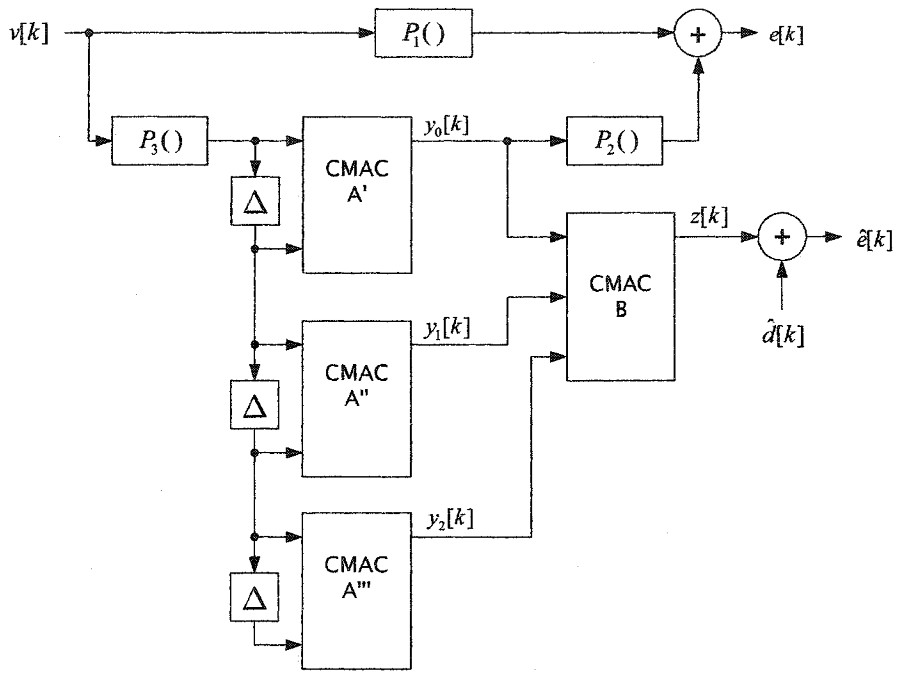


Figure 3.23 – The Filtered-X CMAC Algorithm unwrapped in time.

Operation of the CMAC can be represented by the matrix equation (3.102) where $w[k]$ represents a single column vector of all of the network weights and $a[k]$ represents the binary weight selection vector computed via the standard CMAC algorithm as a function of the current inputs. The weight selection vector contains exactly β values equal to 1.0 and all other elements are zero, where β represents the generalization parameter. Both $w[k]$ and $a[k]$ are functions of the discrete time index k.

$$y[k] = \mathbf{a}[k]^T \mathbf{w}[k] \tag{3.102}$$

The standard CMAC weight update is based on a stochastic gradient descent which incrementally reduces the instantaneous squared error in the output of the CMAC. This is given in (3.103) where β represents the learning gain parameter and e[k] is a measure of the error in the CMAC output.

$$\mathbf{w}[k+1] = \mathbf{w}[k] - \alpha \frac{\partial e^2[k]}{\partial \mathbf{w}[k]} \tag{3.103}$$

If $r[k]$ represents the target output value then the CMAC output equation in (3.103) can be used to express the error as given by (3.104).

$$e[k] = r[k] - y[k] = r[k] - \mathbf{a}[k]^T \mathbf{w}[k] \tag{3.104}$$

Substituting (3.104) into (3.103) and combining scalar gain terms into the learning gain results in the standard CMAC weight update given by (3.105).

$$\mathbf{w}[k+1] = \mathbf{w}[k] - \alpha e[k]\mathbf{a}[k] \tag{3.105}$$

Thus, on each iteration, the standard CMAC weight update increments each of the selected weights with a term proportional to the present output error. However, in the disturbance cancellation architecture of figure 3.23, the output error is not observable. Instead, it is necessary to derive a weight update which incrementally reduces the instantaneous squared error at the error sense, $\hat{e}^2[k]$. This is expressed in (3.106).

$$\mathbf{w}[k+1] = \mathbf{w}[k] - \alpha \frac{\partial\left(\hat{e}^2[k]\right)}{\partial \mathbf{w}[k]} \tag{3.106}$$

The partial derivative in (3.106) can be expanded via the chain rule with the result shown in (3.107). Notice that the time index is fixed with respect to the gradient descent of the error surface.

$$\frac{\partial\left(\hat{e}^2[k]\right)}{\partial \mathbf{w}[k]} = -2\hat{e}[k]\frac{\partial z[k]}{\partial \mathbf{w}[k]} = 2\hat{e}[k]\sum_{i=0}^{M-1} \frac{\partial z[k]}{\partial y_i[k]} \frac{\partial y_i[k]}{\partial \mathbf{w}[k]} \tag{3.107}$$

The partial derivative terms related to the relationship between the inputs and output of the secondary path model can be approximated numerically as given in (3.108). These can then be computed via an additional perturbation and evaluation of CMAC B given that the secondary path model is trained prior to adapting the compensator.

$$\frac{\partial z[k]}{\partial y_i[k]} \cong \frac{\Delta z[k]}{\Delta y_i[k]} = \nabla_i[k] \tag{3.108}$$

Assuming that changes in the weight vector occur at a rate which is slow compared to the length of the secondary path delay line M, then the approximation (3.109) is justified. Notice that an analogous condition applies to the Filtered-X LMS algorithm in that the filter coefficients are assumed to vary slowly with respect to the signal dynamics.

$$\frac{\partial}{\partial \mathbf{w}[k]} \cong \frac{\partial}{\partial \mathbf{w}[k-i]} \qquad \forall\, i \in \{0,1,\cdots,M-1\} \tag{3.109}$$

With this assumption and the CMAC output equation (3.102), it is possible to write expression (3.110).

$$\frac{\partial y_i[k]}{\partial \mathbf{w}[k]} = \frac{\partial}{\partial \mathbf{w}[k]}\left(\mathbf{a}[k-i]^T \mathbf{w}[k-i]\right) \cong \mathbf{a}[k-i]^T \tag{3.110}$$

Finally, substituting (3.110) and (3.108) into (3.106) and absorbing the factor of 2 into the learning gain constant allows the Filtered-X CMAC weight update to be written as in equation (3.111).

$$\mathbf{w}[k+1] = \mathbf{w}[k] + \alpha \hat{e}[k] \sum_{i=0}^{M-1} \nabla_i \mathbf{a}[k-i] \tag{3.111}$$

Thus, the Filtered-X CMAC weight update is performed on M pairs of delayed pointers with each update weighted by the respective numerically estimated gradient ∇_i through the secondary path model. This is in contrast to the standard CMAC weight update in (3.105) which modifies only the weights associated with the current input. From a computational point of view, there are M times as many weights to be updated on each cycle. However, the weight selection vectors need only be calculated once and then shifted for use on the lagged updates. The quality of the CMAC partial derivative computation can be significantly improved by CMAC weight smoothing methods which have been recently introduced [21][87][95].

3.8 An Alternative Derivation of Filtered-X LMS Algorithm

In this section, the approach developed for derivation of the Filtered-X CMAC algorithm is applied to a linear FIR compensator. The result is an alternative derivation of the familiar Filtered-X LMS algorithm. This reveals that the Filtered-X CMAC algorithm represents a generalization of the Filtered-X LMS algorithm. Additionally, this derivation provides additional insight into the function of the Filtered-X LMS technique.

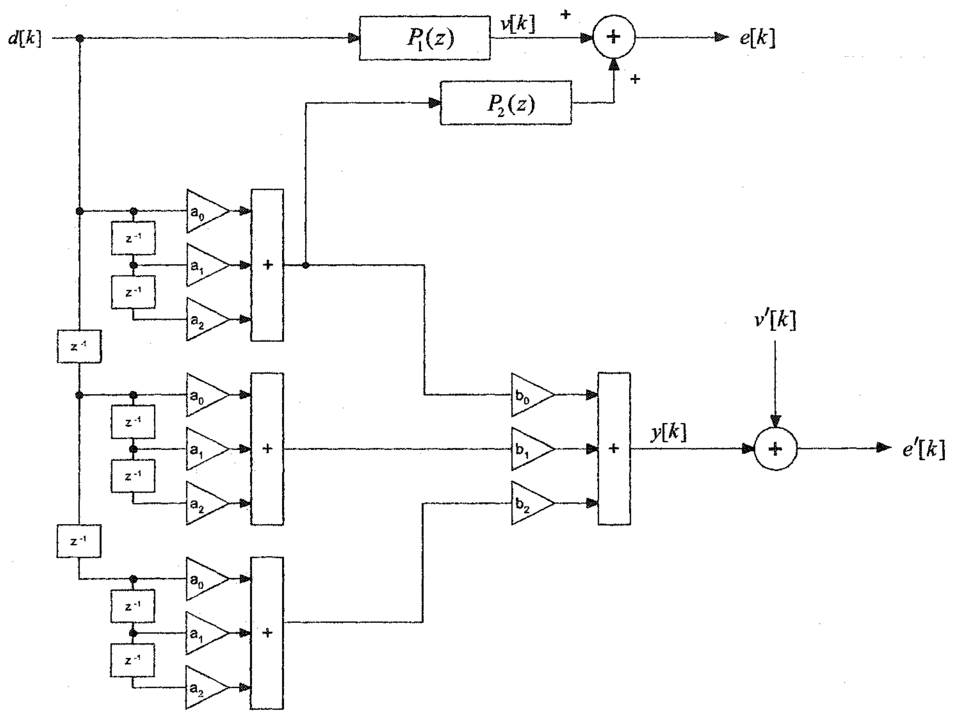


Figure 3.24 – A FIR cancellation system unwrapped in time.

Consider the linear cancellation architecture shown in figure 3.24. This corresponds to the Filtered-X CMAC algorithm with the compensator and secondary path model replaced by linear FIR filters. The compensator is described by the coefficients $a_0, a_1, \cdots, a_{N-1}$ while the secondary path model is described by filter coefficients $b_0, b_1, \cdots, b_{M-1}$. It is assumed that the secondary path model is adapted off-line prior to cancellation so that the coefficients are considered to be constant during cancellation. The compensator filter is shown unwrapped in time in order to depict the dependence of the present error on past compensator outputs. The compensator coefficients can then be adjusted via a gradient descent of the instantaneous squared output error. Assuming that the secondary path model is ideal, minimization of $\hat{e}^2[k]$ is identical to minimization of the observed error, $e^2[k]$. In the following derivation the general case is considered where the compensator has N coefficients and the secondary path model has M coefficients. The form of the compensator coefficient update is given in (3.112), where α represents the learning gain parameter.

$$a_i[k+1] = a_i[k] - \alpha \frac{\partial\left(\hat{e}^2[k]\right)}{\partial a_i[k]} \quad \forall\, i \in \{0,1,\cdots,N-1\} \tag{3.112}$$

The secondary path model output, $z[k]$, can be represented by the double linear filtering operation shown in (3.113).

$$z[k] = \sum_{i=0}^{M-1} b_i y_i[k] = \sum_{i=0}^{M-1} b_i \sum_{j=0}^{N-1} a_j[k-l]x[k-j-l] \tag{3.113}$$

Using this expression for the secondary path model output, it is possible to simplify the partial derivative in the update equation (3.112) as shown by the result in (3.114). Notice that $\hat{d}[k]$ is independent of the compensator output and therefore its derivative with respect to any compensator coefficient is zero.

$$\begin{aligned}\frac{\partial \hat{e}^2[k]}{\partial a_i[k]} &= 2\hat{e}[k]\frac{\partial}{\partial a_i[k]}\left(\hat{d}[k] - z[k]\right) = -2\hat{e}[k]\frac{\partial z[k]}{\partial a_i[k]} \\ &= -2\hat{e}[k]\frac{\partial}{\partial a_i[k]}\sum_{i=0}^{M-1} b_i \sum_{j=0}^{N-1} a_j[k-l]x[k-j-l]\end{aligned} \tag{3.114}$$

Assuming that the rate of change of the coefficients is relatively slow then the approximation (3.115) can be made.

$$\frac{\partial a_i[k-l]}{\partial a_i[k]} \cong 1 \tag{3.115}$$

Application of this approximation to (3.114) yields the expression in (3.116).

$$\frac{\partial \hat{e}^2[k]}{\partial a_i[k]} = -2\hat{e}[k]\sum_{i=0}^{M-1} b_i x[k-i] \tag{3.116}$$

Substitution of (3.116) into the learning rule of (3.112) yields the compensator coefficient update given in equation (3.117).

$$a_i[k+1] = a_i[k] + \alpha\hat{e}[k]\sum_{i=0}^{M-1} b_i x[k-l] \quad \forall\, i \in \{0,1,\cdots,N-1\} \tag{3.117}$$

Let $x'[k]$ be equal the input signal, $x[k]$, filtered by a perfect FIR model of the secondary path as given by (3.118).

$$x'[k] = \sum_{i=0}^{M-1} b_i x[k-i] \tag{3.118}$$

This allows the coefficient update rule to be written in its final form as shown in equation (3.119). Notice that this is exactly the Filtered-X LMS algorithm. That is, this result is the standard LMS update performed using a filtered version of the input signal.

$$a_i[k+1] = a_i[k] + \alpha\hat{e}[k]x'[k-l] \quad \forall\, i \in \{0,1,\cdots,N-1\} \tag{3.119}$$

This result demonstrates that the Filtered-X CMAC algorithm can be viewed as an extension of the Filtered-X LMS algorithm. Additionally, the conceptual block diagram of figure 3.24 provides insight into operation of the standard Filtered-X LMS algorithm. It is evident that the observed error signal at any instant is influenced not only by the present contents of the tapped delay line of the compensator, but also delayed versions of that vector. As a result, the compensator coefficients must be updated according to the error associated with each set of delayed input vectors.

3.9 The Reduced Filtered-X CMAC Algorithm

In many practical applications, the secondary path can be modeled using a linear dynamical system. This simplification results in reduced computational overhead and better convergence properties in comparison to the use of time delay CMAC secondary path model as described in section 3.7. In sections 3.4 and 3.5, it was shown that the time delay secondary path could be used in the case of a linear secondary path and narrowband disturbance. However, in cases where the disturbance signal is broadband or the phase characteristic varies rapidly with frequency, as in a resonant system, then the secondary path cannot be approximated using a time delay model. The Reduced Filtered-X CMAC algorithm presented in this section can be utilized with any disturbance signal in combination with any linear dynamical secondary path, including those which contain resonance and time delay. Additionally, the accuracy requirements of the secondary path model are not stringent. As a result, in practice most mild nonlinearities do not have significant impact on the convergence of the algorithm even when the secondary path is approximated by a linear model.

The Reduced Filtered-X CMAC algorithm is depicted in the block diagram of figure 3.25. The forward path is represented by $P_1(\cdot)$ while the secondary path is represented by the linear transfer function, $P_2(s)$. It is assumed that FIR filter B is adapted off-line to serve as a model of the actual secondary path.

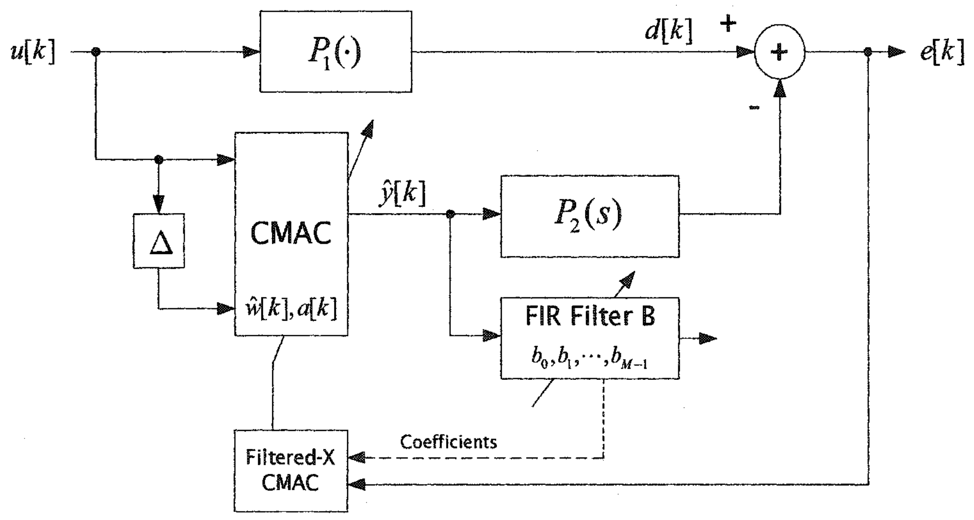


Figure 3.25 – The Reduced Filtered-X CMAC Algorithm.

The error signal is computed by the filtering operation given in (3.120) where b_i represent the secondary path filter coefficients for $i = 0, 1, \cdots, M-1$. The CMAC output is given in equation (3.121) using the standard notation for the weight vector and weight selection vector.

$$e[k] = \sum_{i=0}^{M-1} b_i \left(d[k-i] - y[k-i]\right) \tag{3.120}$$

$$y[k] = \mathbf{a}[k]^T \mathbf{w}[k] \tag{3.121}$$

The standard CMAC weight update algorithm incrementally reduces the error in the output of the CMAC. However, in the present case, the desired CMAC output is unknown and instead the objective is the minimization of the observable error, $e[k]$. Therefore, the CMAC weight update must be revised to directly implement a gradient descent of the squared error signal $e[k]^2$. The modified adaptation takes the form of (3.122).

$$\mathbf{w}[k+1] = \mathbf{w}[k] - \alpha \frac{\partial e^2[k]}{\partial \mathbf{w}[k]} \tag{3.122}$$

Using expressions (3.122) and (3.120), the partial derivative of the squared error can be derived as given in the following result.

$$\frac{\partial e^2[k]}{\partial \mathbf{w}[k]} = 2e[k]\frac{\partial e[k]}{\partial \mathbf{w}[k]} = -2e[k]\sum_{i=0}^{M-1} \frac{\partial}{\partial \mathbf{w}[k]} \mathbf{a}[k-i]^T \mathbf{w}[k-i] \tag{3.123}$$

Under the assumption that the weight vector varies slowly with respect to the filter impulse response, the approximation in (3.124) is justified.

$$\frac{\partial}{\partial \mathbf{w}[k]} \cong \frac{\partial}{\partial \mathbf{w}[k-i]} \qquad \forall i \in \{0,1,\cdots,M-1\} \tag{3.124}$$

Under this assumption the weight update of (3.123), can be expressed in its final form as given by (3.125). In this expression, the factor of two has been absorbed into the learning gain parameter. Additionally, the learning gain parameter has been normalized by the generalization parameter.

$$\mathbf{w}[k+1] = \mathbf{w}[k] + \frac{\alpha}{\beta}\sum_{i=0}^{M-1} b_i \mathbf{a}[k-i] e[k] \tag{3.125}$$

This weight update algorithm reflects the fact that the output error at any time is a function not only of the present CMAC output, but also the CMAC output at previous time instances. As a result, weight updates are performed for M sets of delayed pointers. This technique is referred to as the Reduced Filtered-X CMAC algorithm since it can be viewed as an extension of the Filtered-X LMS algorithm utilized in the case of a linear adaptive compensator. An implementation of the Reduced Filtered-X CMAC algorithm is shown in figure 3.26. A model of the secondary path, represented by FIR Filter B, is adapted offline prior to operation of the compensator. The coefficients of the secondary path model are utilized in conjunction with the error signal to update the compensator CMAC via the weight update algorithm of (3.125).

Despite the use of a linear secondary path model, the reduced algorithm is still applicable in many nonlinear systems. In particular, the impact of nonlinearities in the locations specified by dashed blocks in figure 3.125 will be considered here.

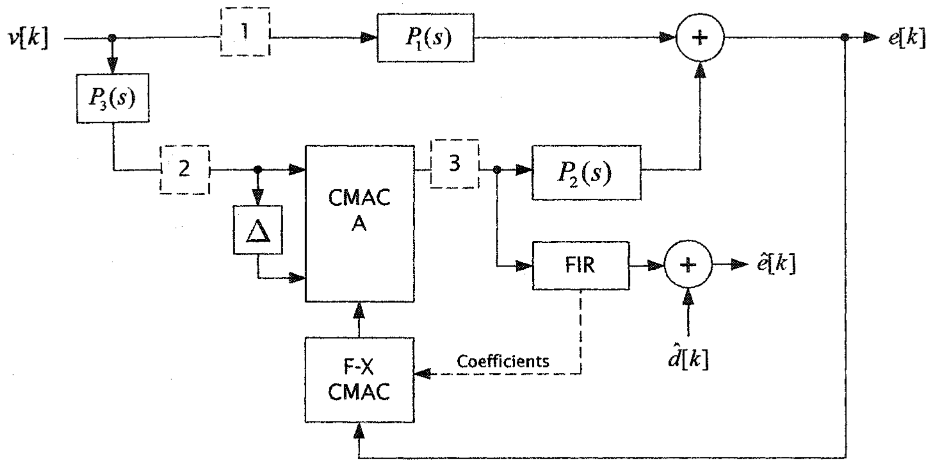


Figure 3.26 – Nonlinearities and the Filtered-X CMAC algorithm.

The first nonlinearity to be considered is located in the forward transmission path as indicated by block '1' in figure 3.26. Its presence results in disturbance signals at the error sense which are not linearly related to the reference signal and therefore require a nonlinear compensator for their removal. However, given that the cancellation path is linear in this case, there is no need for a nonlinear secondary path model and the reduced Filtered-X CMAC algorithm is adequate.

To illustrate a specific example of this case, we now consider a narrowband active noise cancellation application in simulation. The reference signal is a single tone sinusoid phase-locked to the primary 300 Hz disturbance. Often such a reference signal is generated via a non-acoustic sensor such as a tachometer on a rotational disturbance source. The forward path consists of a nonlinearity which passes the fundamental and generates the first three harmonics with amplitudes equal to half the primary disturbance. The fundamental and its harmonics then propagate through the remainder of the forward path which is modeled by a first order linear low pass filter. The cancellation path is a resonant linear fourth order system with a 2 millisecond transport delay which is approximated by a 300th order linear FIR model. Figure 3.27 depicts the reduction in the measured disturbance e[k] after the CMAC weight adaptation is enabled at time 50 milliseconds. Figure 3.28 shows three periods of the fundamental disturbance overlaid on the reference signal. Finally, figure 3.29 depicts the compensator output and secondary path output after convergence is attained.

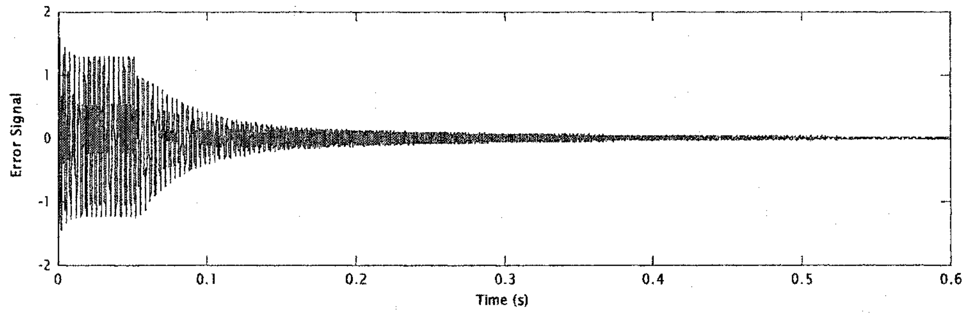


Figure 3.27 – Error signal convergence for a feedforward harmonic-generating nonlinearity. Parameters for this simulation are: N=2, ρ=20, quantization width = 0.03, $\eta = 0.055$, disturbance amplitude = 1.0, sample period = 50 μs.

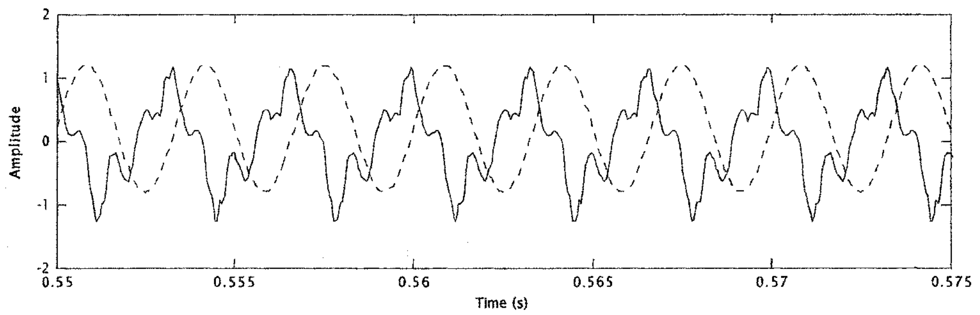


Figure 3.28 – Reference Signal (dashed line) and compensator output (solid line). Note the harmonic content of the disturbance which is not linearly related to the reference signal.

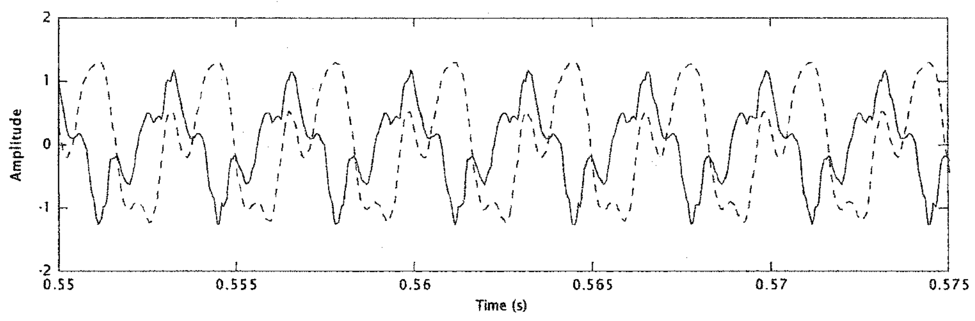


Figure 3.29 – Compensator output (solid line) and secondary path output (dashed line).

The second nonlinearity to be considered is located in the reference signal path as shown by block '2' in figure 3.26. This nonlinearity represents a transformation of the CMAC inputs. A thorough investigation of allowable nonlinearities in this location is beyond the scope of the present research. However, simulations indicate that most soft nonlinearities in this path can be accommodated. As a fundamental guideline, in order to ensure that the desired CMAC mapping is one-to-one, the CMAC input vector should represent a closed non-self-intersecting orbit in the embedding space with no regions of zero tangential velocity. However, a fundamental point is that those nonlinearities in location '2' which can be accommodated are unaffected by whether a linear or nonlinear secondary path model is utilized. Therefore, the Reduced Filtered-X CMAC algorithm is an appropriate choice in the case of such a nonlinearity.

Finally, a nonlinearity located in the cancellation path is considered. This is represented by block '3' in figure 3.26. At first consideration, it might seem that the reduced algorithm is not well suited to this case given its use of a linear secondary path model. However, it can in fact be successfully applied in cases where the nonlinear secondary path can be represented by a static nonlinearity followed by a linear dynamical system and an FIR model of the linear portion of the secondary path can be extracted. A common example of such a situation is the case of a static nonlinearity generated by the cancellation path actuator (e.g. dead zone, nonlinear gain) followed by a linear transmission path.

The process of modeling the linear portion of the cancellation path can be complicated in such cases, however, it is often possible to either derive the linear portion via a physical model or extract the model using a small excitation signal to reduce the impact of a large signal nonlinearity. Additionally, simulation studies suggest that the secondary path model does not require great accuracy to ensure convergence. As a guideline, in the case of the Filtered-X LMS algorithm, the secondary path model can have phase errors up to ±90 degrees and the system will still be convergent [62].

To illustrate a particular example, consider the case of a secondary path containing a dead-band nonlinearity followed by a fourth order resonant linear system with transport delay as used in the previous simulation. The dead-band nonlinearity is of width 1.0 and is centered at the origin. The disturbance and reference signals are the same as used in the previous simulation. Figure 3.30 provides an indication of the degree of distortion produced by the nonlinearity by showing its effect on the disturbance signal.

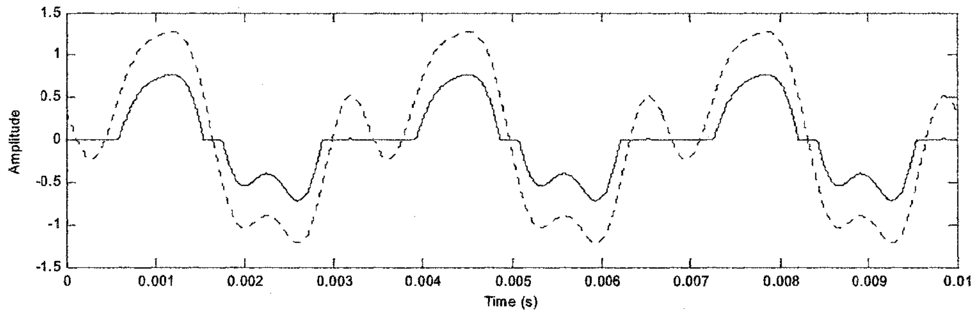


Figure 3.30 – This figure depicts the disturbance signal (dashed line) and the disturbance after passing through the cancellation path nonlinearity (solid line).

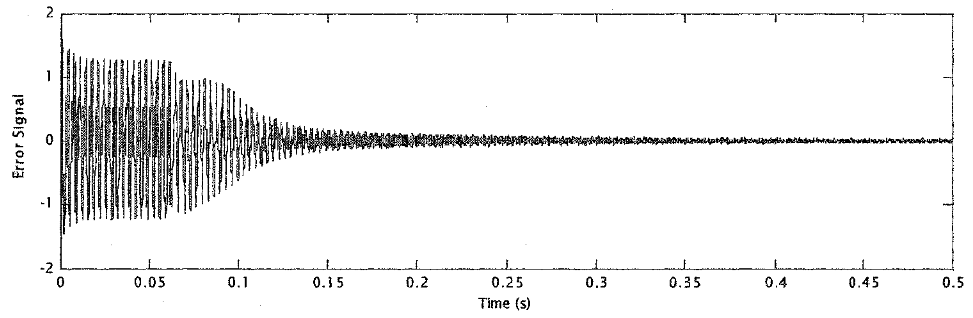
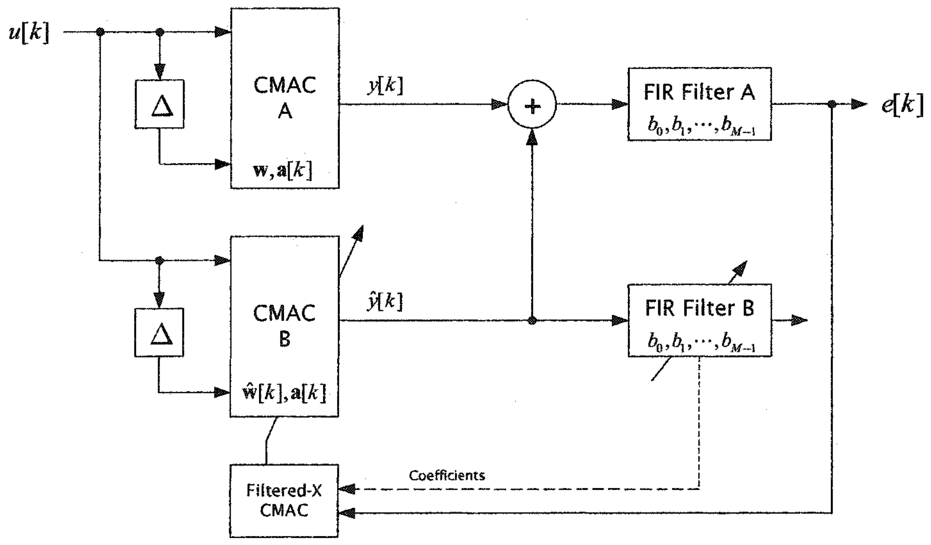


Figure 3.31 – Error signal versus time for the case of a deadband nonlinearity in the cancellation path. All parameters are the same as used in the previous simulation. Adaptation is enabled at time 50 ms.

3.10 Convergence of the Reduced Filtered-X CMAC Algorithm

In this section, the convergence of the reduced Filtered-X CMAC algorithm is considered analytically. Figure 3.32 represents a typical application of the algorithm where a disturbance source $u[k]$ propagates through a nonlinear dynamical plant modeled by CMAC A. The compensator is represented by CMAC B which monitors the original disturbance source and influences the disturbance signal at the summing junction. To facilitate the analysis, it is assumed that both the original disturbance and the cancellation signal pass collectively through a common secondary path represented here by M tap FIR filter A. Thus, the block diagram considered by this proof is slightly less general than the system used in derivation of the algorithm in section 3.9. The compensator is updated via the reduced Filtered-X CMAC algorithm. It is assumed that a secondary path model has been identified off-line as represented by FIR filter B. Additionally, it is assumed that the secondary path model is a perfect model of the actual secondary path.



$u[k]$
Δ
CMAC A
$\mathbf{w},\mathbf{a}[k]$
$y[k]$
+
FIR Filter A
$b_0, b_1, \cdots, b_{M-1}$
$e[k]$
Δ
CMAC B
$\hat{\mathbf{w}}[k],\mathbf{a}[k]$
$\hat{y}[k]$
FIR Filter B
$b_0, b_1, \cdots, b_{M-1}$
Coefficients
Filtered-X CMAC

Figure 3.32 – Block diagram utilized in the convergence proof.

Given that both CMAC networks share the same input vector, the weight selection vector for each network is identical. The output equation for CMAC A and CMAC B are given by (3.126) and (3.127) respectively.

$$y[k] = \mathbf{a}[k]^T \mathbf{w} \tag{3.126}$$

$$y[k] = \mathbf{a}[k]^T \hat{\mathbf{w}}[k] \tag{3.127}$$

The weight update equation for the compensator CMAC is performed via the reduced Filtered-X CMAC algorithm as given in (3.128). Operation of the secondary path filter is defined in (3.129).

$$\hat{\mathbf{w}}[k+1] = \hat{\mathbf{w}}[k] + \frac{\alpha}{\beta}\sum_{i=0}^{M-1} b_i \mathbf{a}[k-i] e[k] \tag{3.128}$$

$$e[k] = \sum_{i=0}^{M-1} b_i x[k-i] = \sum_{i=0}^{M-1} b_i \left(y[k-i] - \hat{y}[k-i] \right) \tag{3.129}$$

The present analysis requires an additional assumption on the nature of the error signal. Specifically, it is assumed that the rate of change of the error signal is bounded over any time interval of length M, as given in (3.130).

$$\begin{aligned} |e[k-i]| \le R|e[k]| \quad & \forall i \in \{1,2,\cdots,M\} \\ & \forall k \ni e[k] \ne 0 \end{aligned} \tag{3.130}$$

Additionally, it is assumed that the maximum overlap between any two inputs in an M time step interval is ϕ. For complete generality, it is possible to pick $\phi = \beta$ to allow any possible overlap. To further simplify the results, the parameter b represents a bound on the magnitude of the filter coefficients such that $|b_i| < b \quad \forall\, i \in \{0,1,\cdots,M-1\}$. With these definitions, it will be shown that the algorithm is convergent as long as the learning gain is chosen according to the bound given in (3.131). From this result it is evident that the learning gain must be reduced to accommodate increased secondary path length, larger degrees of overlap, increased filter coefficient magnitudes, and larger rates of change in the error signal. The proof of this result is provided in Theorem 3.5.

$$0 < \alpha < \frac{\left(\frac{2}{Mb^2}\right)}{\left[1 + \left(\frac{\phi}{\beta} R(M-1) + R + \frac{\phi}{\beta}\right)(M-1)\right]} \tag{3.131}$$

In the case of $M = 1$, this algorithm reduces to a simple linear gain secondary path with gain b_0. In this case, the learning gain bound reduces to that given in (3.132). This is the same result as derived in the case of a linear gain secondary path given that $G_1 = 0$ and $G_2 = 0$ and given the extra factor of b_0 inherent in the Filtered-X CMAC weight update.

$$0 < \alpha < \frac{2}{b_0^2} \tag{3.132}$$

In the case where there is no restriction on the extent of receptive field overlap, $\phi = \beta$, and the learning gain bound reduces to the expression given in (3.133).

$$0 < \alpha < \frac{2}{M^2 b^2 \left[RM - R + 1\right]} \tag{3.133}$$

Theorem 3.5 – Convergence of the Reduced Filtered-X CMAC Algorithm: Consider the feedforward disturbance cancellation system of figure 3.22 with dynamics described by equations (3.126)-(3.129). Implicit in these equations is the fact that CMAC A and CMAC B have identical architectures. Let $\beta > 0$ denote the common CMAC generalization parameter. It is assumed that there exists some constant R such that $\left|e[k-i]\right| \le R\left|e[k]\right|$ for all $i \in \{1, 2, \cdots, M\}$ and for all k such that $e[k] \ne 0$. Additionally, it is assumed that the maximum receptive field overlap in any M time step window is limited to ϕ. More precisely, it is required that $\mathrm{a}[k]^T \mathrm{a}[k-i] \le \phi \ \ \forall i \in \{1, 2, \cdots, M\}$. Let b_i represent the secondary path coefficients for $i \in \{0, 1, \cdots, M-1\}$. Additionally let these coefficients be bounded such that $\left|b_i\right| < b \ \ \forall\, i \in \{0, 1, \cdots, M-1\}$.

$$0 < \alpha < \frac{\left(\dfrac{2}{Mb^2}\right)}{\left[1 + \left(\frac{\phi}{\beta} R(M-1) + R + \frac{\phi}{\beta}\right)(M-1)\right]} \tag{3.134}$$

Under the stated assumptions, with the learning gain parameter chosen according to bound (3.134) then $\lim_{k \to \infty} e[k] = 0$ showing that the system is convergent.

Proof:

Let $V[k]$ be the inner product of the weight difference vector as given by (3.135) and notice that this quantity is non-negative.

$$V[k] \triangleq \left(\mathbf{w} - \hat{\mathbf{w}}[k]\right)^T \left(\mathbf{w} - \hat{\mathbf{w}}[k]\right) \geq 0 \tag{3.135}$$

Using the Filtered-X weight update of equation (3.128), an expression for $V[k+1]$ can be derived as shown in the following expressions.

$$V[k+1] = \left(\mathbf{w} - \hat{\mathbf{w}}[k+1]\right)^T \left(\mathbf{w} - \hat{\mathbf{w}}[k+1]\right) \tag{3.136}$$

$$= V[k] - 2e[k]\frac{\alpha}{\beta}\sum_{i=0}^{M-1} b_i \mathbf{a}[k-i]^T \left(\mathbf{w} - \hat{\mathbf{w}}[k]\right) + \frac{\alpha^2}{\beta^2} e^2[k] \sum_{i=0}^{M-1}\sum_{j=0}^{M-1} b_i b_j \mathbf{a}[k-i]^T \mathbf{a}[k-j] \tag{3.137}$$

To simplify this expression further it is necessary to derive an expression for $\mathbf{w}[k-i]$ as a function of $\mathbf{w}[k]$. To that end, consider equations (3.138)-(3.140) which represent the weight updates performed at consecutive intervals of time.

$$\hat{\mathbf{w}}[k] = \hat{\mathbf{w}}[k-1] + \frac{\alpha}{\beta}\sum_{j=0}^{M-1} b_j \mathbf{a}[k-1-j] e[k-1] \tag{3.138}$$

$$\hat{\mathbf{w}}[k-1] = \hat{\mathbf{w}}[k-2] + \frac{\alpha}{\beta}\sum_{j=0}^{M-1} b_j \mathbf{a}[k-2-j] e[k-2] \tag{3.139}$$

$$\vdots$$

$$\hat{\mathbf{w}}[k-i+1] = \hat{\mathbf{w}}[k-i] + \frac{\alpha}{\beta}\sum_{j=0}^{M-1} b_j \mathbf{a}[k-i-j] e[k-i] \tag{3.140}$$

Combining equations (3.138)-(3.140) into a single equation yields the result in (3.141). Notice that for the case $i = 0$ the last term in this expression is equal to zero.

$$\hat{\mathbf{w}}[k] = \hat{\mathbf{w}}[k-i] + \frac{\alpha}{\beta}\sum_{l=1}^{i}\sum_{j=0}^{M-1} b_j \mathbf{a}[k-l-j] e[k-l] \tag{3.141}$$

Substituting this expression into (3.137) yields the result given in (3.142)

$$\begin{aligned} V[k+1] - V[k] = {} & \frac{-2\alpha e[k]^2}{\beta} + \frac{2\alpha^2 e[k]}{\beta^2}\sum_{i=1}^{M-1}\sum_{l=1}^{i}\sum_{j=0}^{M-1} b_i b_j \mathbf{a}[k-i]^T \mathbf{a}[k-l-j] e[k-l] \\ & + \frac{\alpha^2 e^2[k]}{\beta^2}\sum_{i=0}^{M-1}\sum_{j=0}^{M-1} b_i b_j \mathbf{a}[k-i]^T \mathbf{a}[k-j] \end{aligned} \tag{3.142}$$

It is possible to split the summations in (3.142) according to whether the term represents complete or partial overlap receptive field overlap. That is all terms containing $\mathbf{a}[k-i]^T\mathbf{a}[k-j]$ for $i=j$ are isolated from those terms containing $\mathbf{a}[k-i]^T\mathbf{a}[k-j]$ where $i\neq j$. The result is given in (3.143).

$$\begin{gathered}V[k+1]-V[k]=\frac{-2\alpha e[k]^2}{\beta}+\frac{2\alpha^2}{\beta^2}\sum_{i=1}^{M-1}\sum_{l=1}^{i}\sum_{\substack{j=0\\ j\neq i-l}}^{M-1} b_i b_j \mathbf{a}[k-i]^T\mathbf{a}[k-l-j]e[k]e[k-l]+\\ \frac{2\alpha^2}{\beta}\sum_{i=1}^{M-1}\sum_{l=1}^{i} b_i b_{i-l} e[k]e[k-l]+\frac{\alpha^2 e^2[k]}{\beta^2}\sum_{i=0}^{M-1}\sum_{\substack{j=0\\ j\neq i}}^{M-1} b_i b_j \mathbf{a}[k-i]^T\mathbf{a}[k-j]+\frac{\alpha^2 e^2[k]}{\beta}\sum_{i=0}^{M-1} b_i^2\end{gathered} \tag{3.143}$$

Inequalities (3.144)-(3.146) follow directly from the conditions on the maximum coefficient magnitude, the maximum rate of change of the error signal, and the maximum overlap.

$$|b_i|<b\Rightarrow b_i b_j\leq b^2 \tag{3.144}$$

$$|e[k-l]|\leq R|\mathrm{e}[k]|\Rightarrow e[k]e[k-l]\leq R\,\mathrm{e}^2[k] \tag{3.145}$$

$$\mathbf{a}[k]^T\mathbf{a}[k-l]\leq\phi \tag{3.146}$$

With these inequalities, it is possible to establish a bound on the expression in (3.143) as shown in (3.147).

$$\begin{gathered}V[k+1]-V[k]\leq\frac{-2\alpha e^2[k]}{\beta}+\frac{2\alpha^2}{\beta^2}\sum_{i=1}^{M-1}\sum_{l=1}^{i}\sum_{\substack{j=0\\ j\neq i-l}}^{M-1} b^2\phi R e^2[k]+\\ \frac{2\alpha^2}{\beta}\sum_{i=1}^{M-1}\sum_{l=1}^{i} b^2 R e^2[k]+\frac{\alpha^2 e^2[k]}{\beta^2}\sum_{i=0}^{M-1}\sum_{\substack{j=0\\ j\neq i}}^{M-1} b^2\phi+\frac{\alpha^2 e^2[k]}{\beta}\sum_{i=0}^{M-1} b^2\end{gathered} \tag{3.147}$$

Notice that all of the terms in (3.147) are independent of the summation indices. Given this fact and the result in (3.148), this bound can be rewritten as shown in (3.149).

$$\sum_{i=1}^{M-1}\sum_{l=1}^{i}1=\tfrac{1}{2}\left(M^2-M\right) \tag{3.148}$$

$$\begin{gathered}V[k+1]-V[k]\leq\frac{-2\alpha e^2[k]}{\beta}+\frac{\alpha^2 b^2\phi R}{\beta^2}\left(M^2-M\right)(M-1)e^2[k]+\\ \frac{2\alpha^2 b^2 R}{\beta}\left(M^2-M\right)e^2[k]+\frac{\alpha^2 b^2\phi}{\beta^2}M(M-1)e^2[k]+\frac{\alpha^2 b^2}{\beta}Me^2[k]\end{gathered} \tag{3.149}$$

By factoring out common terms, the bound in (3.149) can be rewritten as given by (3.150).

$$\begin{aligned} V[k+1]-V[k] &\leq \frac{\alpha}{\beta}e^2[k]\Big[-2+\alpha\Big(Mb^2+Mb^2\frac{\phi}{\beta}R(M-1)^2 \\ &+Mb^2R(M-1)+Mb^2\frac{\phi}{\beta}(M-1)+\Big)\Big] \leq 0 \end{aligned} \tag{3.150}$$

The second inequality in (3.150) follows directly from the learning gain bound. From (3.150) and given that $V[k] \geq 0$, the conditions of Lemma 3.1 are met and therefore statement (3.151) is valid.

$$\lim_{k\to\infty}\left(V[k+1]-V[k]\right)=0 \tag{3.151}$$

By the sequence squeeze theorem and inequality (3.150), the limit given in (3.152) exists.

$$\begin{aligned} &\lim_{k\to\infty}\frac{\alpha}{\beta}e^2[k]\Big[-2+\alpha b^2R(M^2-M)+\frac{\alpha}{\beta}b^2\phi R(M^2-M)(M-1) \\ &+\alpha Mb^2+\frac{\alpha}{\beta}b^2\phi M(M-1)\Big]=0 \end{aligned} \tag{3.152}$$

Since the multiplicative constant of $e^2[k]$ is strictly non-zero due to the learning gain bound, this implies $\lim_{k\to\infty}e^2[k]=0$ and therefore, $\lim_{k\to\infty}e[k]=0$. □

CHAPTER 4

ALTERNATIVE NEURAL NETWORK DISTURBANCE CANCELLATION ARCHITECTURES

4.1 Introduction

This chapter presents three additional architectures for active disturbance cancellation utilizing the time delay CMAC network. Section 4.2 describes a regenerative CMAC algorithm which provides narrowband cancellation without requiring a reference signal estimate of the primary disturbance. This approach is an extension of the regenerative technique described in section 1.4 for the case of linear adaptive systems. Section 4.3 describes a disturbance cancellation algorithm in which the CMAC network is trained via reinforcement learning rather than the conventional adaptation based on a gradient descent of the error surface. This allows the compensator to be trained without knowledge of the phase characteristics of the secondary path. Finally, section 4.4 describes a means by which the time delay CMAC can be used as a recurrent nonlinear oscillator and adapted to provide an appropriate cancellation signal. As with regenerative cancellation, this technique does not require an estimate of the original disturbance. In addition, this approach does not require a model of the secondary path in cases where the secondary path does not introduce phase distortion in the error signal. The algorithms presented in this chapter are of a more experimental nature than the feedforward techniques described in the previous chapter. The description of these techniques is limited to a brief presentation of the algorithm and representative simulations indicating basic functionality of each technique. General conclusions regarding stability, applicability, and utility of the presented techniques have not been established.

4.2 Regenerative Feedback Neural Network Cancellation

In section 1.4, the regenerative approach to feedback active disturbance cancellation was introduced. In the present section, this technique is extended for use with the time delay CMAC compensator. The basic architecture is shown in figure 4.1. Notice that the only input signal to the cancellation algorithm is provided by the error signal. This represents a significant advantage over the feedforward approach since a reference signal is often difficult or even impossible to obtain in many practical systems.

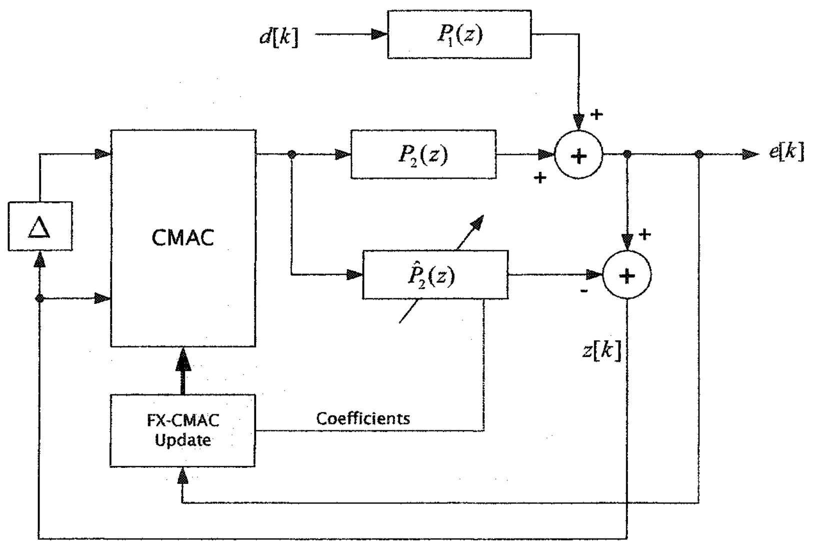


Figure 4.1 – Regenerative Feedback Neural Network Cancellation.

Initially, it is assumed that the secondary path can be represented as a linear dynamical system. An adaptive filter represented by $\hat{P}_2(z)$ is trained off-line to serve as a model of the cancellation path dynamics. During operation, the secondary path model serves two purposes. First, it is utilized to remove the compensator's contribution in the error signal. As a result, the signal $z[k]$ remains equal to the original error signal $e[k]$ independent of the CMAC output. Second, the coefficients of the secondary path model are utilized to provide convergent adaptation of the CMAC compensator through use of the Reduced Filtered-X CMAC algorithm which was presented in section 3.9. However, even when utilizing the linear secondary path model, this algorithm is capable of functioning in a variety of nonlinear systems. A preliminary survey of the nature of acceptable nonlinear systems is presented in this section.

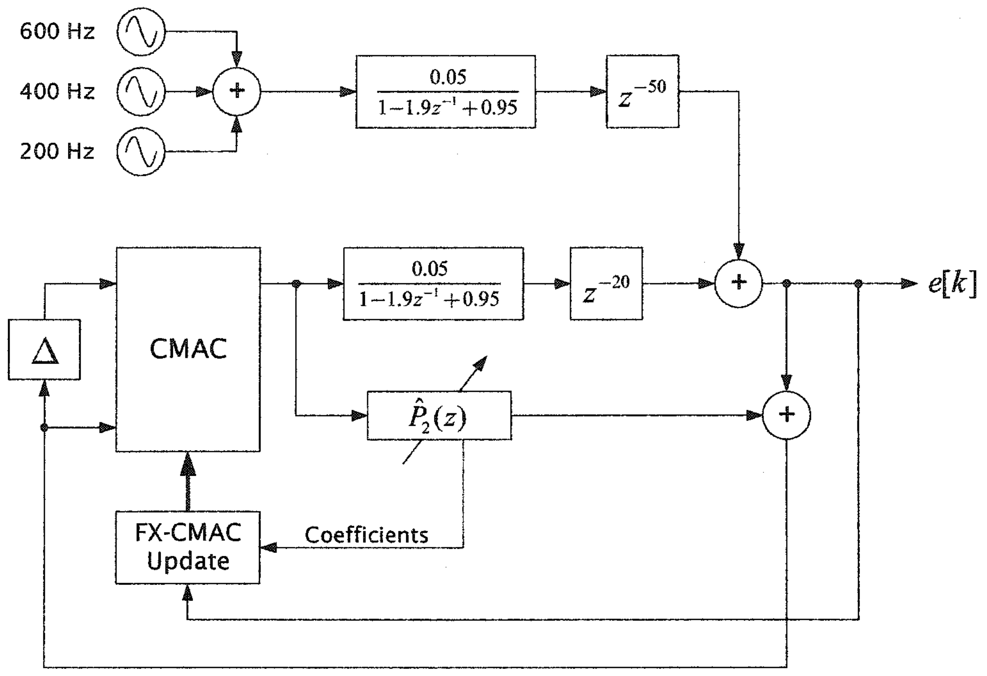


Figure 4.2 – Regenerative CMAC cancellation in a linear system.

Before considering the effects of system nonlinearities, the basic feasibility of this approach is demonstrated via simulations of the system shown in figure 4.2. The disturbance source is represented as the superposition of three harmonically related sinusoids. This composite disturbance propagates through a forward signal path represented by a second order resonant system in combination with a transport delay of 50 discrete time intervals. The secondary path is represented by a resonant second order system in followed by a pure time delay of 20 discrete time intervals. Given the significant resonance of the secondary path, the FIR model must contain a large number of coefficients. In these simulations, a 300th order FIR filter is utilized.

Figure 4.3 depicts the convergence of the error signal during adaptation of the compensator while figure 4.4 depicts the disturbance and cancellation signals after a significant level of convergence has been attained. The parameters used in these simulations were as follow: Sample Rate = 20 µs, Pointer Delay = 75 Samples, Learning Gain = 0.05, Generalization = 50, Quantization Width = 0.02, Disturbance Frequencies = 200Hz, 400Hz, 600Hz, and corresponding Disturbance Amplitudes = 1.0, 0.5, 0.5.

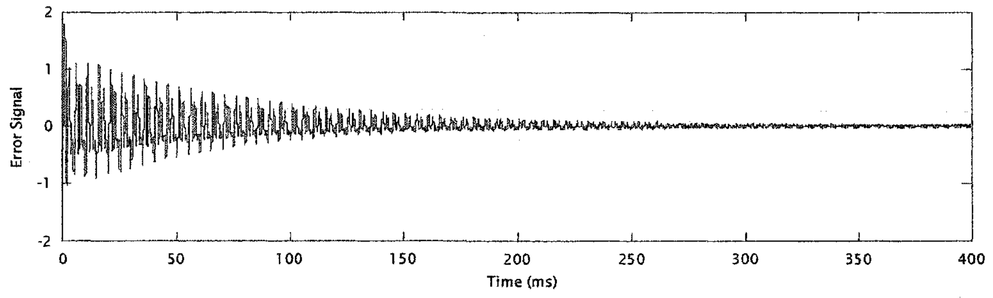


Figure 4.3 – Error Signal during adaptation for the system of figure 4.2.

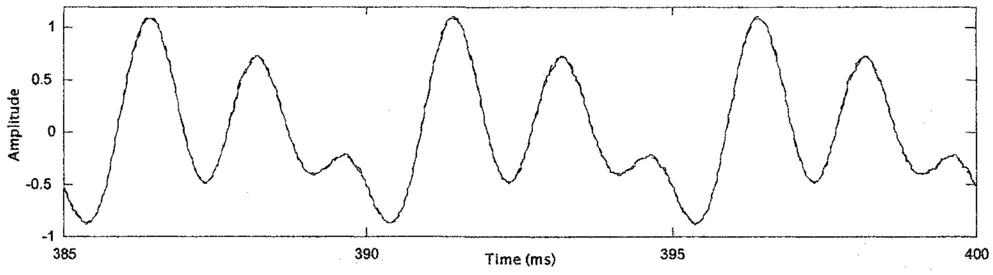


Figure 4.4 – Disturbance signal (solid) and Cancellation signal (dashed) at the error sense summing junction. On this scale, the two can barely be distinguished from each other.

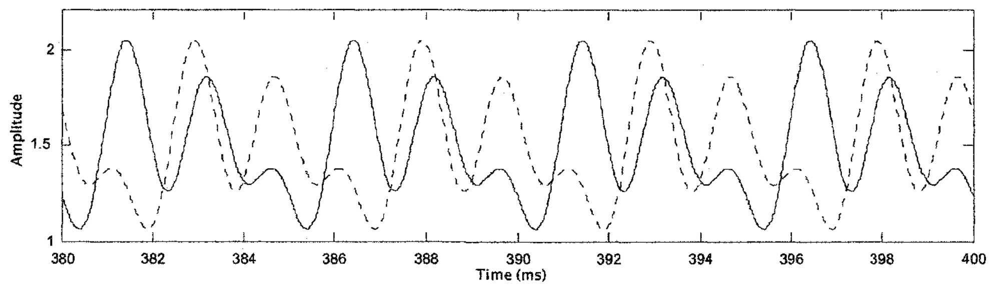


Figure 4.5 – CMAC inputs for the system of figure 4.2.

One important aspect of the regenerative technique is that a nonlinear compensator is not required in order to provide effective cancellation in the case of nonlinearities in the forward path. This is due to the fact that the compensator must only represent the transfer function relating the disturbance measured at the error sense to the required cancellation signal. This transfer function depends only on elements in the secondary path. However, nonlinearities in the cancellation loop generally do require the use of a nonlinear compensator. In the remainder of this section the presence of nonlinearities in the locations marked by dashed boxes in figure 4.6 will be considered.

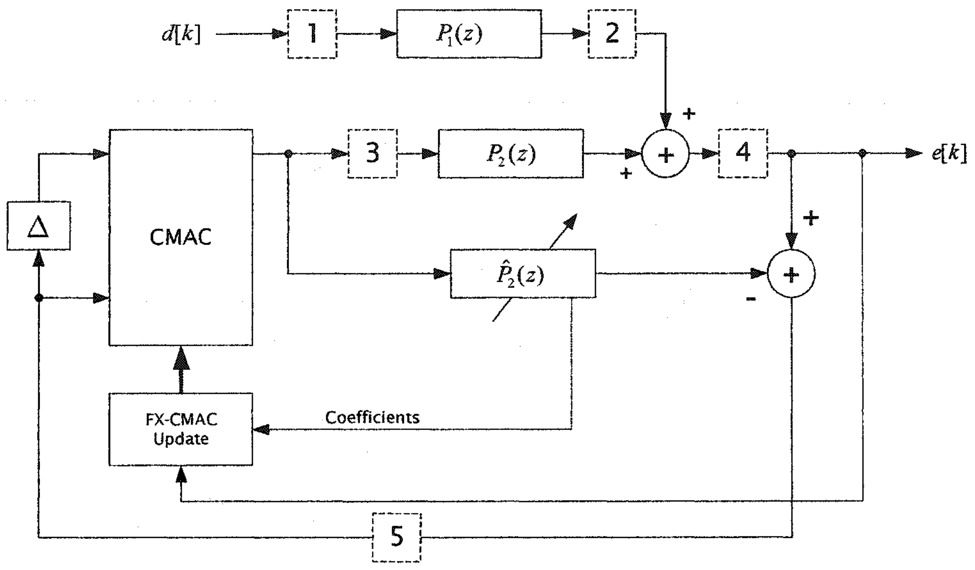


Figure 4.6 – Regenerative Filtered-X CMAC with nonlinearities.

Location one and two represent nonlinearities in the forward path. Such nonlinearities can be accommodated even in the case of regenerative cancellation using a linear adaptive compensator. A nonlinearity in locations three or four will result in two potential problems. First, this nonlinearity will result in a degradation of the regenerated signal due to the mismatch between the actual secondary path and the linear model. Second, for severe nonlinearities, this mismatch can lead to instability in the Filtered-X weight update of the compensator. However, for many soft nonlinearities which do not introduce phase distortion in the secondary path, this architecture can be utilized directly. As a particular example, figure 4.7 depicts the error signal for a simulation in the case of a deadband nonlinearity of width 1.0 at location three. All other aspects of the system are identical to that shown in figure 4.2.

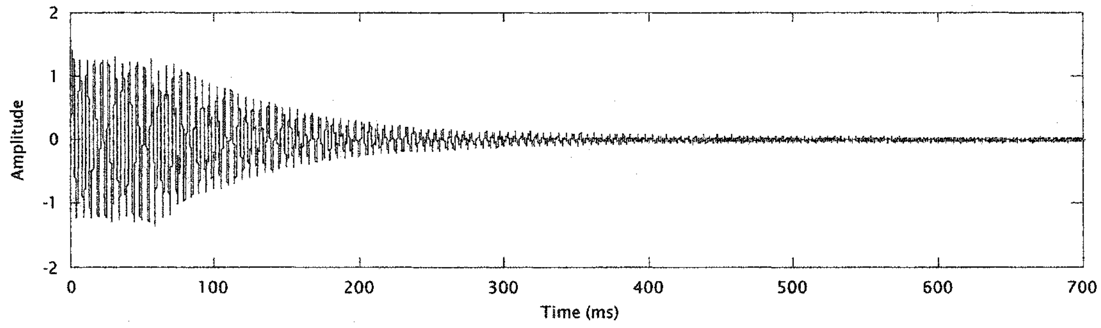


Figure 4.7 – Error signal for simulation with a deadband nonlinearity in the secondary path.

A nonlinearity in location five is generally handled adequately by the present algorithm. This nonlinearity represents a transformation of the regenerated signal. However, since this effect takes place after the removal of the compensator output, the resultant regenerated signal will remain invariant during adaptation of the compensator. Additionally, the nonlinear nature of the CMAC compensator can be utilized to reverse the effects of this nonlinearity in order to create the appropriate cancellation signal. In conclusion, the algorithm depicted in figure 4.2 is capable of compensating for a variety of system nonlinearities. However, for sufficient nonlinearity in the secondary path, it is necessary to utilize a nonlinear secondary path and implement the compensator adaptation using the full Filtered-X CMAC algorithm as shown in figure 4.8. In this case, a second time delay CMAC is adapted to model the cancellation path which is then utilized in place of $\hat{P}_2(z)$ to create the regenerative compensator input. Additionally, CMAC secondary path model is utilized to adapt the compensator using the Filtered-X CMAC approach described previously in section 3.7. As a result, this allows the algorithm to be used in cases where the secondary path is characterized by significant nonlinearity.

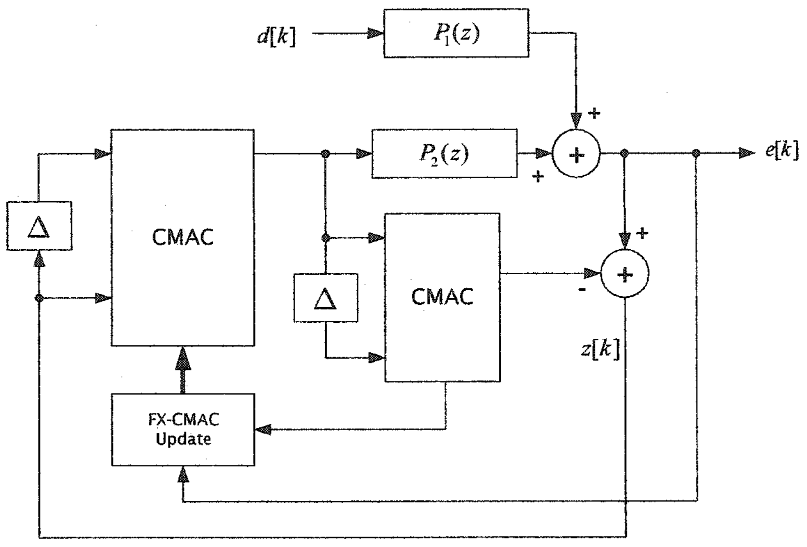


Figure 4.8 – Regenerative Time Delay CMAC Disturbance Cancellation.

4.3 Reinforcement Learning Active Disturbance Cancellation

In this section, a novel approach to nonlinear disturbance cancellation is presented. The proposed algorithm is based on the use of reinforcement learning to adapt the weights of a time delay CMAC compensator. In contrast, all of the previous techniques considered in this dissertation have utilized adaptation mechanisms based on gradient descent of the error surface. Such gradient descent algorithms are characterized by the use of a continuously available error signal in order to provide incremental updates to the CMAC weights. To ensure convergence of the weights it was necessary for these algorithms to take account of the dynamics associated with the secondary path. However, in the case of reinforcement learning, it will be shown that such *a priori* knowledge is unnecessary.

Reinforcement learning is a technique for neural network training which is based on experimentation rather than on *a priori* knowledge of the structure of the system which is being controlled. The learning takes place in a trial and error fashion. As an example, consider a hypothetical control task in which a given, finite set of parameters define the operation of the control algorithm. Design of the optimal controller consists of choosing the best set of parameters under some performance criterion. One means by which the optimal parameter settings can be found is via reinforcement learning. The reinforcement learning algorithm begins with an initial set of parameter values. A small random perturbation is made to each parameter to form a new experimental parameter set which is then utilized in performing the control task. The performance of the experimental parameter set is gauged through use of some predefined error metric. However, unlike the error metric available in gradient descent methods, the reinforcement learning error metric is only required to provide a single value as a measure of the performance of the system over the entire task. If the experimental perturbation resulted in better performance according to the error metric, then the current values of the parameter are replaced by those of the experimental parameter set. Conversely, if the experimental parameter set resulted in degraded performance, then the original parameter set is restored. After repeated trials of experimentation, it is generally possible for the learning algorithm to converge to a parameter set which optimizes the error metric over a range of control tasks.

The fundamental advantage of the reinforcement learning methodology is that it requires very little knowledge of the system being controlled. This eliminates the potential for instability resulting from modeling inaccuracies and generally simplifies the design process. The primary disadvantage of reinforcement learning is its slow rate of convergence due to the relatively small amount of information utilized by the controller. Additionally, the reinforcement learning can fail to reach the optimal parameter set due to the presence of local minima in the error metric.

Figure 4.9 depicts the reinforcement training variant of the time delay CMAC compensator in the context of a feedforward disturbance cancellation system. It is assumed that the disturbance source $u[k]$ is periodic and that the resultant trajectory in the CMAC input space is not self-intersecting in order to provide an appropriate input signal to the CMAC. The error metric used for the reinforcement algorithm is based on the average power of the error signal. This is created by the operation of low pass filtering the absolute value of the error signal, $e[k]$. The bandwidth of the error averaging filter is chosen such that the effective period of the averaging is long compared to a period of the disturbance fundamental.

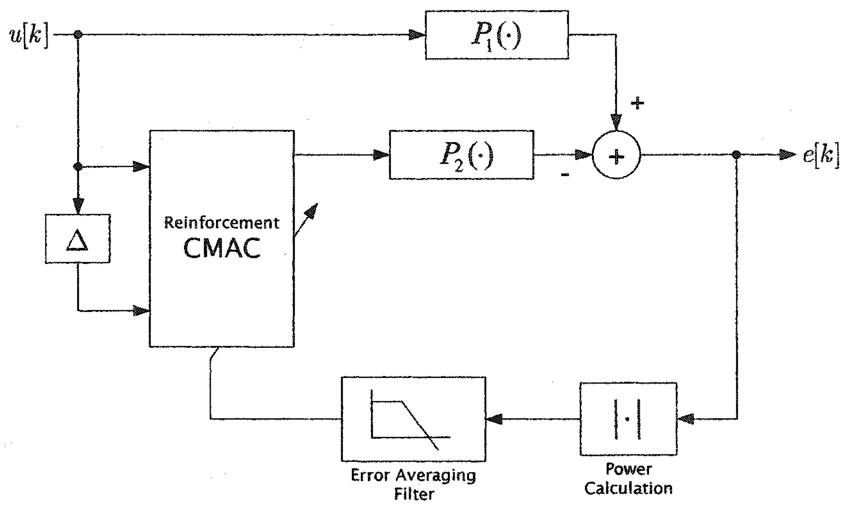


Figure 4.9 – The Reinforcement CMAC Method of Disturbance Cancellation.

Details of the reinforcement training algorithm are given in figure 4.10. At random time intervals, small experimental perturbations are performed on the active set of weights at that instant of time. After the perturbation is made, the impact of that modification is evaluated by observing the change in the error metric over a duration of time which contains many periods of the input disturbance. If the perturbation resulted in reduced power in the error signal, the weight changes are made permanent. Otherwise, the weight changes are eliminated by restoring the affected weights to their previous values. In this manner, the CMAC weights are gradually adjusted in a manner which reduces the average power of the error signal.

The error averaging filter is subject to conflicting constraints which can lead to a significant implementation tradeoff. One important consideration is that the filter must be designed to provide significant attenuation over the expected frequency range of the disturbance. In particular, the residual ripple of the disturbance must be smaller than the impact of a single experimental perturbation. If this is not the case, then the ripple will interfere with the learning process. Additionally, it is generally desirable to minimize the size of learning perturbations in order to improve the final accuracy of the solution. This results in even tighter requirements on the residual filter ripple. The error averaging filter is also subject to the opposing constraint of maximizing the speed of the transient response. This is desirable since the transient response of the filter generally determines the maximum rate at which consecutive experiments can be performed. As a result, a more rapid transient response leads to the ability to perform experiments more rapidly and therefore leads to more rapid convergence of the algorithm. These opposing constraints are best met by using a low pass filter with a very sharp transition and significant attenuation in the stop band.

Reinforcement CMAC Algorithm: Let T_S specify the maximum settling time of the error filter and let η represent the increment size parameter. Then, the following steps describe the operation of the reinforcement learning CMAC algorithm for disturbance cancellation.

1. Wait for a period of time, T_{idle}, chosen from a random uniform distribution over the range $[0, T_P]$ where T_P should be chosen to be similar to the period of the disturbances to be cancelled.
2. Store a pointer, $\mathbf{P}$, to the set of referenced weights.
3. Update all referenced weights by an increment $\pm\eta$ where the sign is chosen randomly.
4. Record error metric value, $e_{initial}$.
5. Wait time T_S for effect of the change to be realized in the error signal.
6. Record new error metric value, e_{final}.
7. If $e_{final} > e_{initial}$, reset weights pointed to by $\mathbf{P}$ to their initial values, and wait for period T_S.
8. Go to step 1.

Figure 4.10 – The Reinforcement CMAC Algorithm.

To provide an estimate of the required attenuation in the stop band of the error averaging filter, consider the simplified case where the disturbance is a single sinusoid of unity amplitude and period one as shown in figure 4.11. The influence of a single weight perturbation in the compensator output is approximated by the triangular region shown in the figure. The amplitude of the experimental perturbation must be smaller than the absolute error tolerance required at the CMAC output. In the present example, it is assumed that a single experiment results in a triangular adjustment with an amplitude equal to 2% of the disturbance signal and a duration equal to 2% of the disturbance period. As a result, the total area of the perturbation is 200×10^{-6}. This area is the same area as created by a sinusoid of period one and amplitude 314×10^{-6}. Therefore, the disturbance signal must be attenuated to an amplitude less than this in order for the impact of a single CMAC update to be evaluated. In the present example, this translates into the requirement of 70.1 dB attenuation at the frequency of the disturbance fundamental.

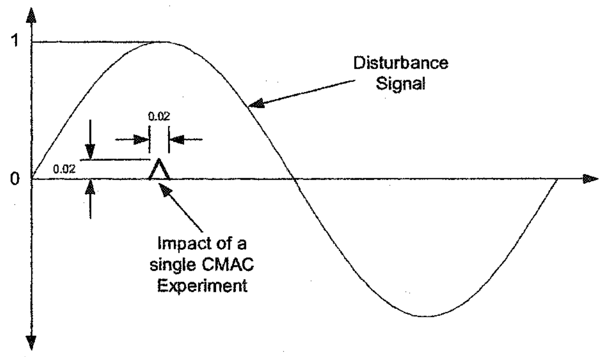


Figure 4.11 – Impact of a single experiment.

To demonstrate the basis functionality of this algorithm, simulations of the system shown in figure 4.9 are presented. The disturbance signal is represented by a single sinusoid of amplitude one and frequency 1 rad/s. The error averaging filter is implemented as a fourth order Chebychev Type II digital filter. It has a corner frequency of 0.02 radians/sample and a designed stopband attenuation of 154 dB. The filter provides 84 dB attenuation at the frequency of the disturbance fundamental (1 rad/s). The step response of the filter settles within 10 cycles of the fundamental disturbance period and therefore this value is used for the minimum time between experiments T_S. The tapped delay line of the

CMAC network has two taps with a delay of 1.5 seconds. The sampling rate of the filter and CMAC is set to 0.01 seconds. The generalization and quantization with used for the CMAC are 100 and 0.05 respectively. In this system, the forward and secondary paths are represented by the same continuous time resonant second order system as given by (4.1).

$$P_1(\cdot) = P_2(\cdot) = \frac{625}{s^2 + 10s + 625} \tag{4.1}$$

Figure 4.12 depicts the averaged error signal for this system as a function of time. The non-monotonic noise in the graph is the searching noise produced by the algorithm. The convergence is quite slow since an experimental perturbation is only applied every ten cycles of the disturbance signal. However these convergence rates are not unreasonable for may disturbance cancellation applications where the disturbance changes only slowly over time and the disturbance is of sufficiently high frequency. For example, in the case of a 10 kHz acoustic disturbance cancellation system, the convergence time of this system would correspond to approximately 2.1 seconds of training. Therefore, as long as the disturbance source changed frequency much slower than this, the algorithm would be able to track it.

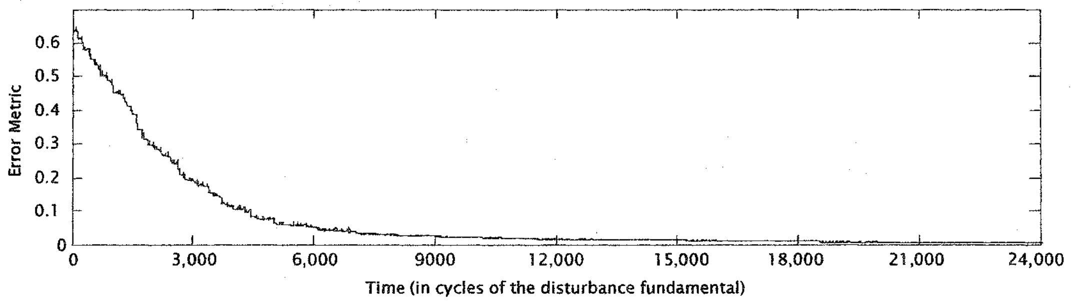


Figure 4.12 – Error signal for a typical simulation.

Figure 4.13 depicts a comparison of the disturbance source and the compensator CMAC output after convergence has been achieved. Notice that since $P_1(\cdot) = P_2(\cdot)$ the error signal is zero if these two signals are equal in value. In this case, the two signals are so similar that they can not be differentiated from one another on this graph.

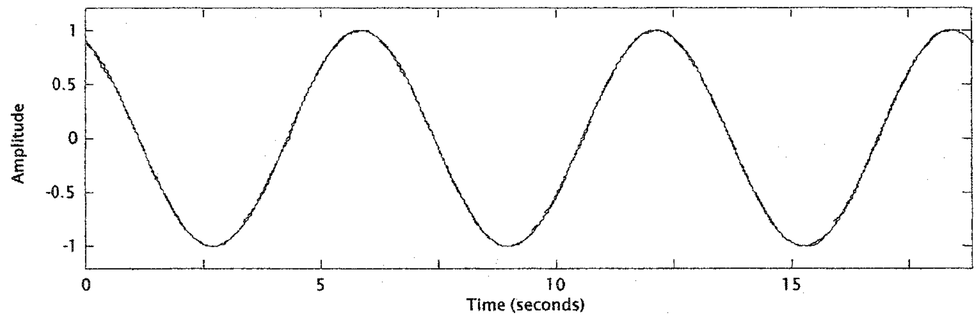


Figure 4.13 – Disturbance source (dashed line) versus cancellation signal (solid line).

The reinforcement learning algorithm can be viewed as a stochastic gradient descent of the error surface. As a result, if there are local minima in the error surface, the algorithm may not converge to zero error. Such an occurrence is common in the system of figure 4.9 and the presence and extent of the problem depends on the nature of the input signal and the parameters of the CMAC. A typical demonstration of the local minima problem is shown in the time domain waveforms of figure 4.14. In this case, the primary and secondary plant are implemented as unity gains and therefore the error signal is equal to the difference between the disturbance and the CMAC output. The CMAC output has converged to the point where its output straddles the desired waveform but will converge no further given any amount of additional training.

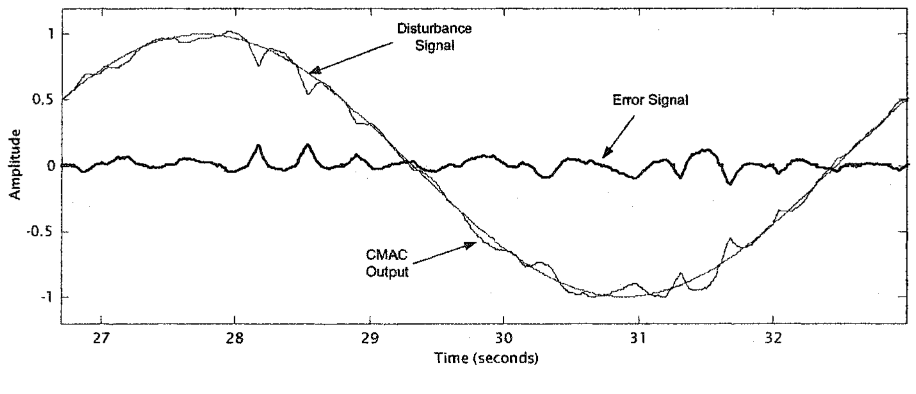


Figure 4.14 – Illustration of the local minima problem.

An intuitive explanation for this phenomenon is provided by the representative drawing of figure 4.15 where the dashed line represents the desired output and the solid line represents the actual CMAC output at some instant of time. Suppose an upward increment is made at the location indicated by arrow 1. This clearly would lead to the eventual optimal solution. However, due to the generalization behavior of the CMAC, this increment produces an increased output at points 2 and 3 as well. Often, the increase in error at points 2 and 3 will be more significant than the decrease in error at point 1. As a result the perturbation will not be accepted even though it would have ultimately lead to a better overall solution.

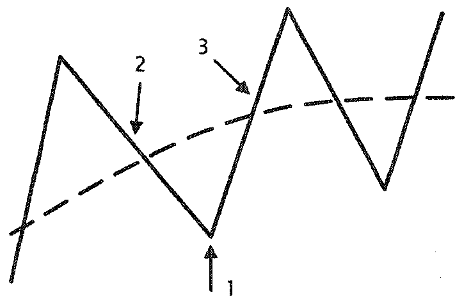


Figure 4.15 – Representative drawing showing local minima problem.

One technique commonly utilized to elude local minima is to add noise to the learning. This can be implemented in the reinforcement learning by occasionally keeping an experiment even if the experiment was not beneficial. The drawback of this solution is that it further slows the convergence of the network. Another possibility is to utilize multiple CMAC networks of decreasing generalization width operating in parallel. As the generalization of the CMAC is reduced, the periodicity and amplitude of the residual noise is generally decreased. However, a CMAC with finer generalization width generally takes longer to converge. By utilizing multiple CMAC networks with varied generalization, the coarse generalization CMAC allows for rapid convergence for the bulk of the signal while the fine generalization CMAC reduces the amplitude of the residual noise caused by the local minima problem. Additionally, the residual noise can be shifted higher in frequency to the point where it is filtered out by the secondary path.

A second refinement of the reinforcement learning CMAC algorithm is to utilize a momentum term to speed the convergence rate. Specifically, the momentum term is implemented by repeating successful experiments until that particular perturbation is no longer useful. Figure 4.16 depicts the result of adding the momentum learning to the simulation of the previously described system.

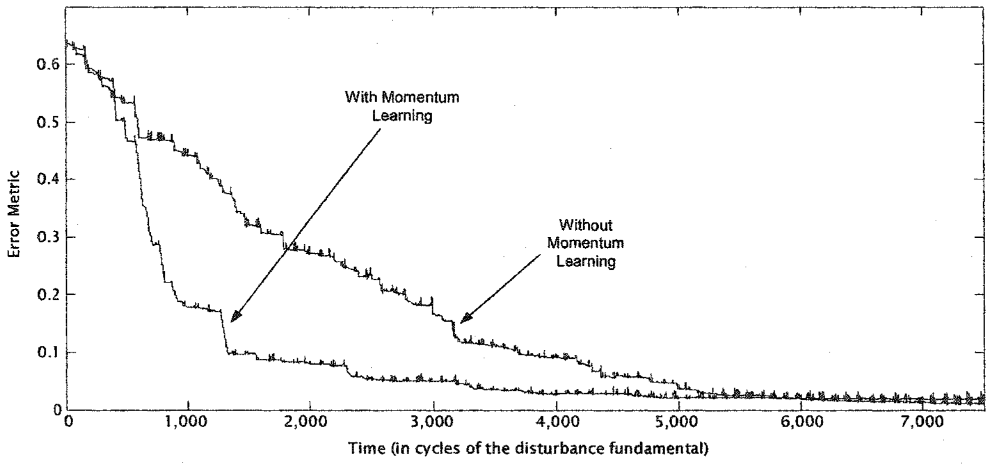


Figure 4.16 – Error metric with and without momentum learning.

4.4 Narrowband Cancellation using Recurrent Neural Networks

In section 2.3, it was shown that the time delay neural network could be configured within a feedback loop in order to function as a nonlinear oscillator. It was also shown that this capability can be viewed as an extension of the marginally stable IIR oscillator. In this section, a recurrent CMAC network will be utilized for active disturbance cancellation in the case of a narrowband periodic disturbance signal. A fundamental advantage of this approach is that it allows cancellation to be attained using only the observed error signal. The basis architecture to be considered is shown in figure 4.17.

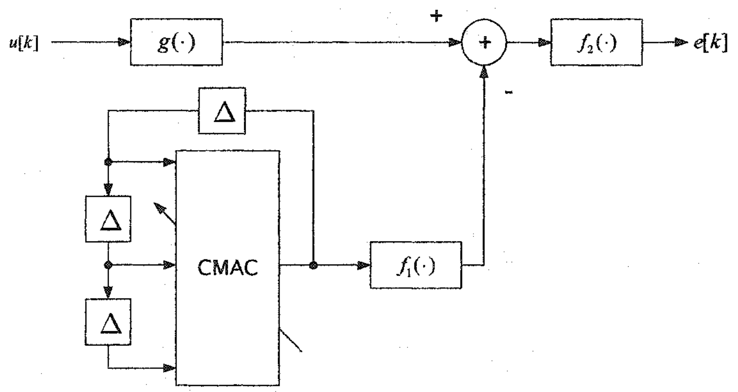


Figure 4.17 – Recurrent CMAC disturbance cancellation architecture.

A periodic disturbance source denoted as the signal $u[k]$ propagates through the forward path represented by $g(\cdot)$ and $f_2(\cdot)$. The recurrent CMAC compensator is implemented using a time delay CMAC with a feedback loop consisting of a single delay element. Cancellation of the feedforward disturbance is achieved by training the CMAC to produce the limit cycle which produces an appropriate cancellation signal. Under the assumption that the secondary path is represented by a linear gain or odd static nonlinearity, the relatively simple adaptation algorithm of figure 4.18 is feasible. Since there is no phase delay in the secondary path, the observed error signal, $e[k]$, is proportional to the error in the CMAC output. As a result, the CMAC weight update can be performed utilizing the observed error signal. The feedback loop filter represented by the Z domain transfer function, $L(z)$, improves convergence properties by attenuating high frequencies induced by the CMAC learning. The filter must have low pass characteristics and the bandwidth must be approximately equal to the upper frequency limit of the disturbance signal. However, the precise nature of the filter response and its bandwidth are not critical to convergence of the algorithm.

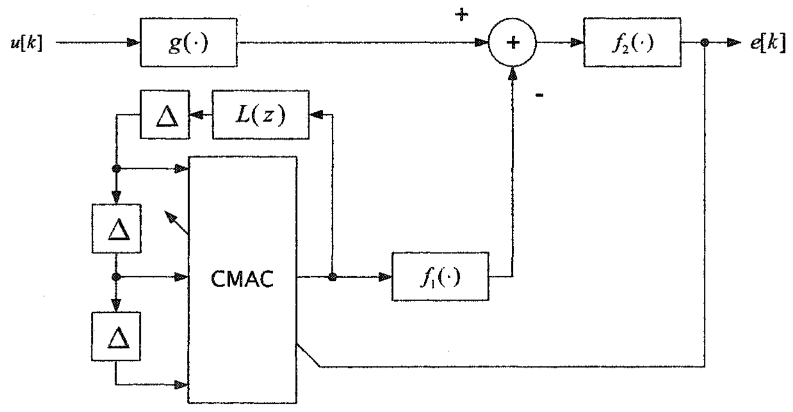


Figure 4.18 – Recurrent NN cancellation for a static secondary path.

The simulation results of figures 4.19 and 4.20 attest to the basic functionality of the recurrent neural network cancellation algorithm. These results are for the simplified case of a single tone disturbance source with all paths represented as unity gains. That is, $f_1(\cdot) = 1$,

$f_2(\cdot) = 1$, and $g(\cdot) = 1$. The tapped delay line contains two taps and the CMAC parameters are as follow: 10 μs sample rate, learning gain $\alpha = 0.01$, generalization width $\beta = 40$, and quantization width $\Delta = 0.05$. The feedback loop filter is implemented as a first order continuous time transfer function with a single pole located at a frequency of 50 Hz. Figure 4.19 depicts the error signal, $e[k]$, as a function of time for the simulation of the system of figure 4.18. Adaptation of the CMAC compensator is enabled at time 0.1 seconds, after which a rapid decrease in the error signal magnitude is apparent as the CMAC approximates the limit cycle required to attain cancellation. The disturbance signal is comprised of a single sinusoid of frequency 50 Hz and unity amplitude. At time 0.5 seconds in this simulation the disturbance signal undergoes a step change in frequency to 100 Hz. As result, the error signal increases momentarily and then is gradually reduced as the CMAC adapts to the new limit cycle. This simulation demonstrates the basic convergence of the algorithm and its ability to track changes in the disturbance signal. Additionally, it reveals that the loop filter parameters are not critical since, in this case, the same loop filter provides convergent adaptation of the CMAC at both disturbance frequencies.

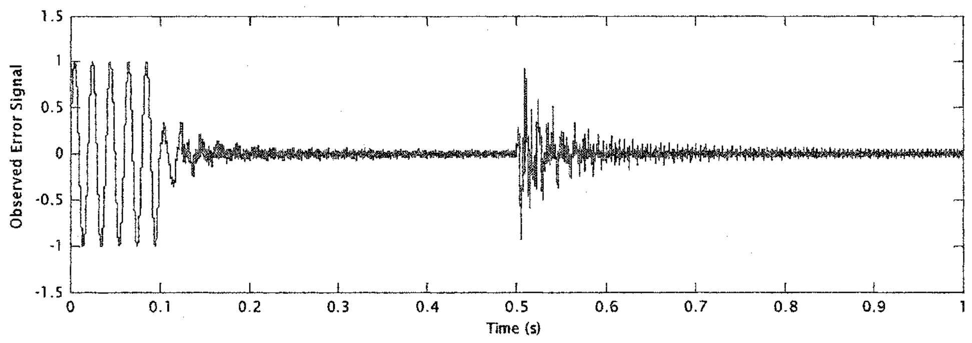


Figure 4.19 – Error Signal for the recurrent Neural Network Cancellation System.

Figure 4.20 depicts the original disturbance signal and the cancellation signal over a small region of time about the transition in the disturbance frequency. Given the unity gain nature of the signal paths, perfect cancellation occurs when these two wave forms are equal in value. This illustration shows the transient response as the recurrent CMAC adapts to the new limit cycle required for the higher frequency disturbance.

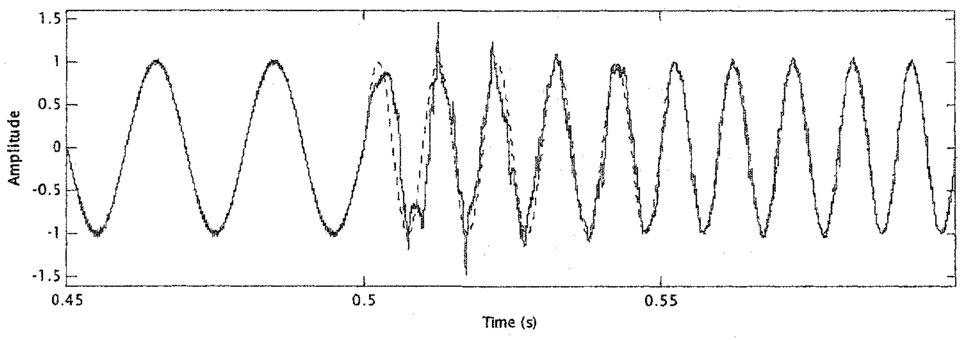


Figure 4.20 – Disturbance and cancellation signals near the frequency step.

Figure 4.21 depicts the trajectory of the CMAC input vector in the delayed coordinate space for the same simulation. That is, x_1 and x_2 represent the CMAC inputs created by the tapped delay line. The presence of two limit cycles corresponding to sinusoidal oscillations at the two distinct disturbance frequencies is clearly visible.

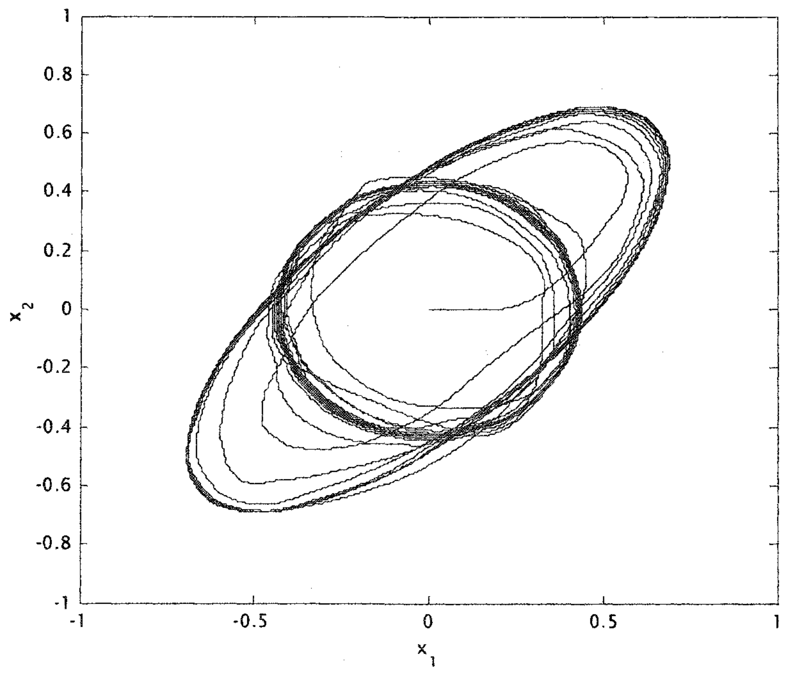


Figure 4.21 – Embedding space pointer trajectory.

The stability of the limit cycle implemented by the recurrent CMAC is dependent upon the nature of the trajectory and the CMAC parameters. However, the stability is also a function of the rate of convergence during training. Consider the simulation result depicted in figure 4.23. In this case, a two input recurrent CMAC is trained to produce a periodic waveform which is the sum of two harmonically related sinusoids. Training is initiated at time zero. In this simulation a relatively large learning gain ($\alpha = 0.2$) was utilized. As a result, the convergence is very rapid. At time 125 ms., the adaptation is disabled and it can be seen that the CMAC output rapidly drifts away from the trained limit cycle. This shows that the limit cycle is not very stable and continuous adaptation would be required to maintain operation within the periodic orbit.

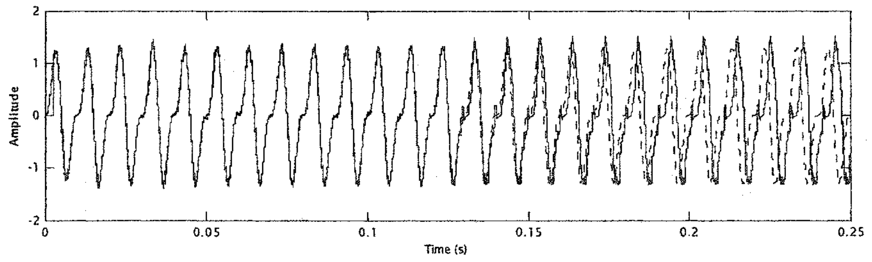


Figure 4.22 – Recurrent CMAC output (solid) and reference (dashed) for a learning gain of 0.2.

Figure 4.23 depicts the same simulation except for a reduction in learning gain to $\alpha = 0.03$. In this case, the same period of training results in a much more stable limit cycle as evidenced by the fact that the CMAC output remains closely correlated to the reference signal even after the CMAC adaptation is disabled at time 125 ms. This is a counterintuitive outcome since it would be reasonable to expect that the higher learning gain should produce more rapid training and therefore result in a more stable limit cycle.

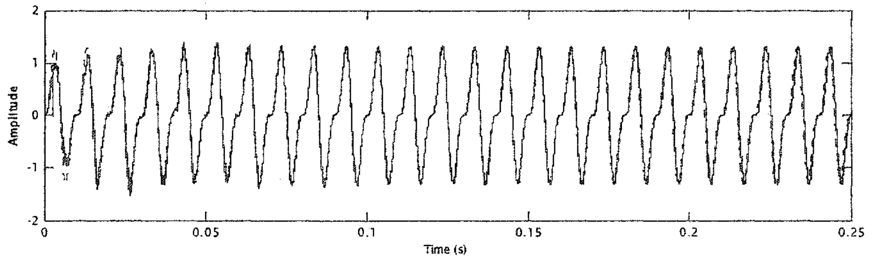


Figure 4.23 – Recurrent CMAC output (solid) and reference (dashed) for a learning gain of 0.03.

The explanation for this phenomenon is due to the different areas of the CMAC input space which are visited in each case. Figure 4.24 depicts the input space trajectory of the CMAC in the previous simulations. Figure 4.24a is for the case of the lower learning rate of 0.03 while figure 4.24b is for the higher learning gain of 0.2. Notice that the exercised region of the map is greater in the case of the lower learning gain since there is more of a transient response before the limit cycle is reached. As a result, the CMAC mapping is trained along a wider region about the trajectories in this case. Therefore, slight perturbations from the ideal trajectory will be corrected. In contrast, the faster adaptation rate results in training only very close to the actual limit cycle. As a result, perturbations from the limit cycle can result in inappropriate CMAC outputs due to a lack of training at nearby points. As a result, slight perturbations can lead to a significant drift away from the limit cycle. More robust training can also be provided through the utilization of additive noise during the adaptation process.

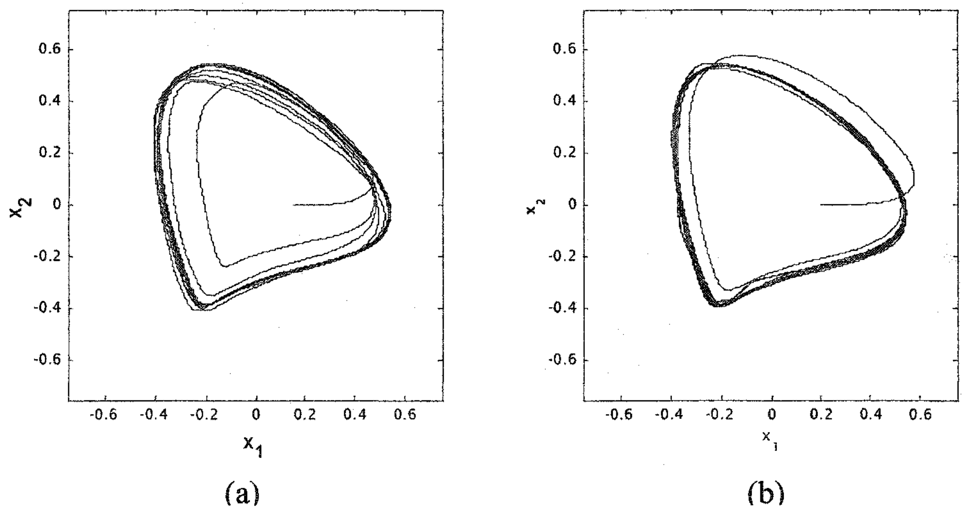


Figure 4.24 – Limit cycle trajectories for learning gain of 0.03 (a) and 0.2 (b).

In the general case where the secondary path has dynamical properties, the recurrent CMAC algorithm can be combined with the Filtered-X CMAC or reduced Filtered-X CMAC algorithms in order to provide convergent training. For example, figure 4.25 depicts the case of a secondary path which can be modeled by a linear dynamical system. In this case, a linear FIR filter is adapted to represent the characteristics of the secondary path. The resultant model is then utilized to perform the reduced Filtered-X CMAC updates on the recurrent CMAC compensator.

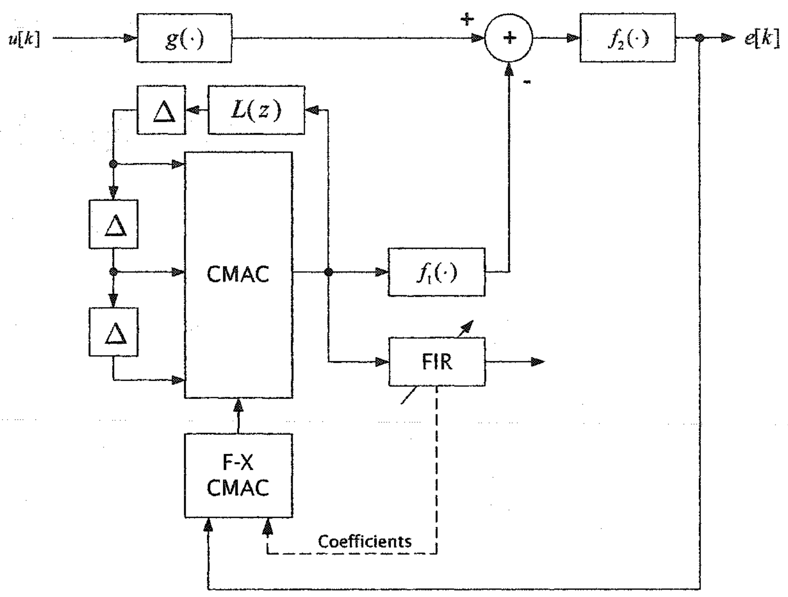


Figure 4.25 – Recurrent CMAC cancellation using the reduced Filtered-X CMAC algorithm.

This approach requires a model of the secondary path dynamics and if such a model exists, then it is also possible to provide cancellation via the regenerative technique described in section 4.2. However, the recurrent CMAC approach has the advantage that a relatively crude model of the secondary path can be utilized since it is only utilize to invert the phase characteristics introduced in the error signal. In contrast, the secondary plant model required in the regenerative approach must be a very precise model of the secondary path in order to accurate estimate the exact value of the compensator contribution in the error signal.

CHAPTER 5

ACOUSTIC CANCELLATION LABORATORY PLATFORM

5.1 Introduction

This chapter describes a practical implementation of the Reduced Filtered-X CMAC algorithm on an acoustic duct laboratory model. The primary objective of this effort is to confirm that the proposed algorithm can provide effective performance even when subject to the constraints of a physical implementation. In typical active disturbance cancellation applications, these constraints include: limited computation time, limited precision math operations, limited sensor ranges, sensor noise, and the presence of unexpected higher order vibrational modes. As in any acoustic cancellation problem, the efficacy of the solution depends in large part on the nature of the acoustic environment. In the case of an ideal cylindrical duct it will be shown in section 5.2 that there is a fundamental cut-on frequency below which the duct is only able to sustain a fundamental mode. The fundamental mode is characterized by equipressure across the duct cross-section and therefore fundamental mode disturbances can be attenuated using a single speaker source. The present model considers only the case of fundamental mode cancellation and the disturbance signal is limited in frequency to ensure that higher order modes are not excited.

An additional outcome in development of the experimental model was the generation of several refinements to the algorithm in order to make it more amenable to hardware implementation. These modifications include a linear variant of the algorithm introduced in section 5.3 and the use of partial updates to reduce the computational overhead as described in section 5.2. A detailed description of the actual hardware implementation and an evaluation of the cancellation performance is provided in section 5.4.

5.2 Cylindrical Duct Acoustics

In order to understand the capabilities and limitations of single source active noise cancellation in a cylindrical duct, it is necessary to have a basic understanding of the behavior of acoustical disturbances in such an environment. For that reason, this section reviews the fundamentals of acoustics in cylindrical ducts, focusing primarily on deriving the modal responses of such systems and the cut-on frequencies associated with the higher order acoustic modes.

An acoustic disturbance, or sound, is simply a minute fluctuation in air pressure superimposed on the much greater background atmospheric pressure. When such a disturbance is created in free space it travels outward in spherical wavefronts at a speed of approximately 350 meters per second, although the exact velocity varies with temperature and relative humidity. Air is unable to support a transverse wave due to insufficient binding force between air molecules and therefore sound waves propagate as longitudinal disturbances with alternating regions of rarefaction (regions of lower than background pressure) and compression (regions with greater than background pressure). Acoustical waves are completely specified by a single scalar function representing the differential air pressure as a function of space and time as given in equation (5.1). The pressure distribution must obey the scalar wave equation or Helmholtz equation given in (5.2) in addition to any boundary conditions applicable to the region under study.

$$p(x,y,z,t) \tag{5.1}$$

$$\nabla^2 p(x,y,z,t) = \frac{1}{c^2}\frac{\partial^2 p(x,y,z,t)}{\partial t^2} \tag{5.2}$$

In this section, the propagation of sound waves in a cylindrical duct of radius a and infinite length will be considered. The derivation will utilize the cylindrical coordinate system depicted in figure 5.1. The speed of sound, c, is taken to be uniform and only time harmonic solutions to the wave equation are considered. It is assumed that the walls are perfectly rigid and the effect of friction along the walls is neglected.

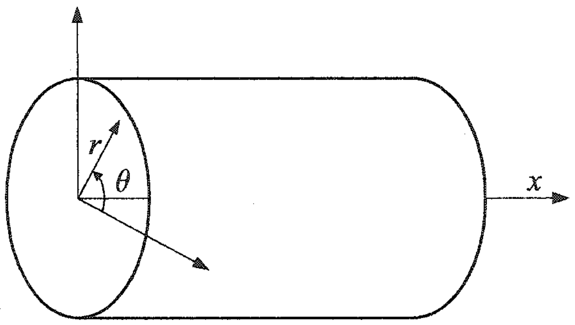


Figure 5.1 – Acoustic duct in cylindrical coordinate system.

The time varying distribution of pressure within the cylindrical duct is found via solution of the scalar wave equation given in (5.2). For the special case of time harmonic solutions the wave equation reduces to the expression given in (5.3). Additionally, given the cylindrical geometry of the problem, it is natural to express the Laplacian in cylindrical coordinates which produces the Helmholtz equation given in equation (5.4). Finally, care must be taken to enforce the boundary conditions presented by the duct walls on the solution of this partial differential equation.

$$\nabla^2 p + \omega^2 p = 0 \tag{5.3}$$

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial r^2} + \frac{1}{r}\frac{\partial p}{\partial r} + \frac{1}{r^2}\frac{\partial^2 p}{\partial \theta^2} + \omega^2 p = 0 \tag{5.4}$$

Analytical solution of this partial differential equation can be obtained using the separation of variables technique to yield the complete solution given in equations (5.5) [94]. The solution is time harmonic as assumed and has radial variation in the form of a varying order Bessel function of the first kind. The constants $\alpha_{m\mu}$, k_x are specified by the boundary conditions and frequency of the harmonic solution while the parameters m and μ are integers related to particular modes where $m = 0,1,2,\cdots$ and $\mu = 0,1,2,\cdots$.

$$p(x,r,\theta) = AJ_m\left(\frac{\alpha_{m\mu} r}{a}\right)\cos(m\theta)e^{-jk_x x}e^{j\omega t} \tag{5.5}$$

The present analysis assumes an infinitely long duct and therefore the only boundary condition in effect is that presented by the cylindrical duct wall. This boundary condition requires that the radial partial derivative of the pressure go to zero at the duct radius as given by equation (5.6) [14].

$$\left.\frac{\partial p}{\partial r}\right|_{r=a} = J_m'(\alpha_{m\mu}) = 0 \tag{5.6}$$

In order to satisfy this boundary condition, $\alpha_{m\mu}$ must be chosen as the μth root of the derivative of the Bessel function of the first kind and order m. These roots can be numerically computed and are given in table 5.2 for several lower order modes.

	$m=0$	$m=1$	$m=2$
$\mu=0$	0	1.84	3.05
$\mu=1$	3.83	5.33	6.71
$\mu=2$	7.02	8.54	9.97

Figure 5.2 – Roots $\alpha_{m\mu}$ of $J_m'(\alpha_{m\mu})$.

The wave number in the direction of propagation is given by equation (5.7) which can be found by substituting the solution (5.5) into the wave equation (5.4). If the wave number is real for a given frequency ω and mode m, μ then the respective mode will be able to propagate. If the wave number is imaginary, the solution will be an evanescent wave which will decay rapidly.

$$k_x = \sqrt{k^2 - k_r^2} = \sqrt{\left(\frac{\omega}{c}\right)^2 - \left(\frac{\alpha_{m\mu}}{a}\right)^2} \tag{5.7}$$

The frequency above which a given mode will sustain a propagating wave front is known as the cut-on frequency of the mode. The cut-on frequency can be determined from solution of equation (5.7) with $k_x = 0$ given that this is the boundary at which the wave number transitions from real to imaginary. This results in equation (5.8) for the cut-on frequency f_c in Hertz.

$$f_c = \frac{c\alpha_{m\mu}}{2\pi a} \tag{5.8}$$

It is common to refer to the various possible modes by the designation $P_{m\mu}$. Notice that the mode number m indicates the number of half wavelengths contained in the circumference of the duct while both wave numbers help specify the more complicated behavior of the wave in the radial direction. The fundamental duct mode is the plane wave mode P_{00}. Notice that $\alpha_{m\mu} = 0$ and therefore the cut-on frequency for this mode is zero indicating that any frequency acoustic wave can be propagated in this mode. Spatially, this mode represents equipressure wave fronts across the duct cross section and time harmonic traveling waves in the x direction.

	$m=0$	$m=1$	$m=2$
$\mu=0$	0 Hz	2.02 kHz	3.34 kHz
$\mu=1$	4.20 kHz	5.85 kHz	7.36 kHz
$\mu=2$	7.70 Khz	9.36 kHz	10.93 kHz

Figure 5.3 – Cut-on Frequencies for Lower Order Duct Modes.

Cut-on frequencies for several of the lower order cylindrical duct modes are presented in figure 5.3. The first higher order mode is P_{01} which begins propagation at about 2 kHz. This mode is characterized by having no pressure variation in the theta direction but variation in the radial direction. In order to visualize the pressure distribution for the different possible modes, consider the plane cross section of the duct shown in figure 5.4. Figures 5.5 – 5.7 depict the pressure distribution on this cross section as surfaces where the height of the surface represents the differential pressure. Notice that for the plane wave the pressure is uniform across the duct and represents a traveling wave in the x direction. For higher order modes, there is substantial variation in the pressure across the duct.

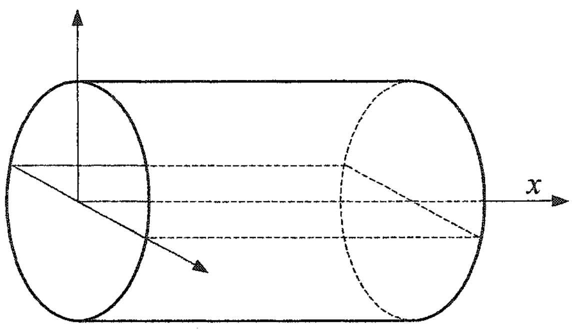


Figure 5.4 – Planar cross section of duct.

Figure 5.5 depicts the pressure distribution associated with the fundamental mode of the duct. Notice that the pressure is uniform across the duct and varies sinusoidal in the direction of travel. To a first order approximation, a speaker located inside the duct with a diameter approximately equal to the duct diameter will produce a uniform pressure disturbance across the duct cross-section. Additionally, the uniformity of the pressure distribution means that a single sensor located anywhere across the duct cross-section is adequate for sensing the error signal.

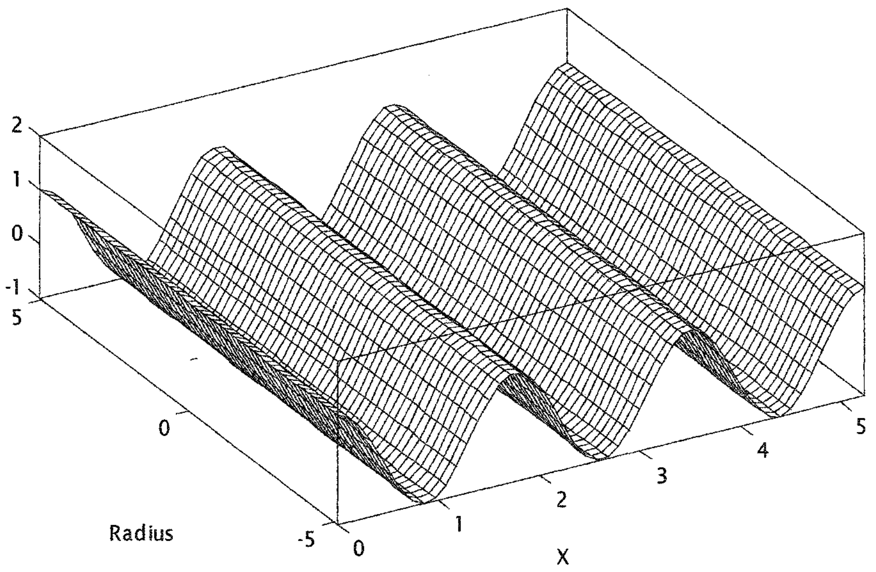


Figure 5.5– Plane wave (P_{00} mode) pressure distribution.

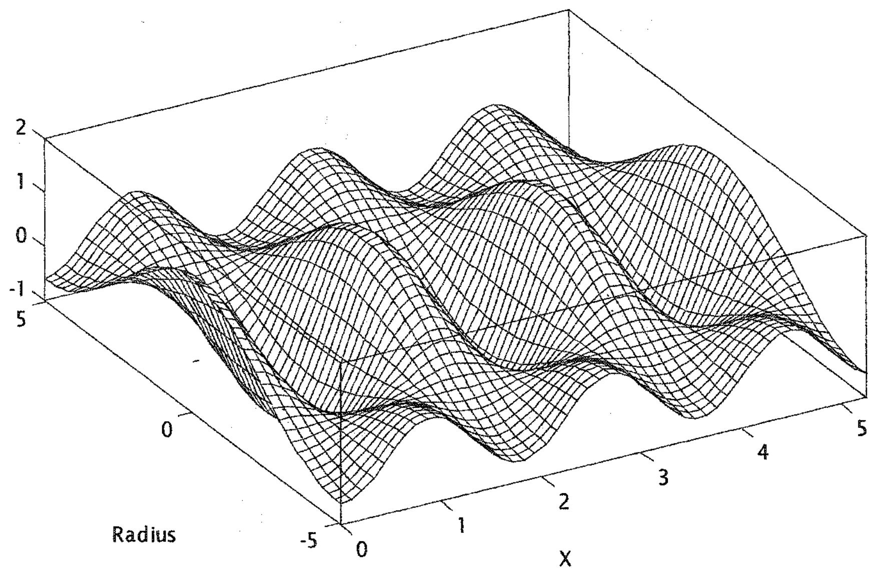


Figure 5.6 – Mode P_{01} pressure distribution.

Figures 5.6 and 5.7 depict the pressure distribution associated with two higher order duct modes. Notice that in both of these cases, the pressure is not uniform across the duct cross-section. In general, cancellation of these higher order modes requires an additional cancellation source per mode in a location that is not located near a null of the mode to be cancelled.

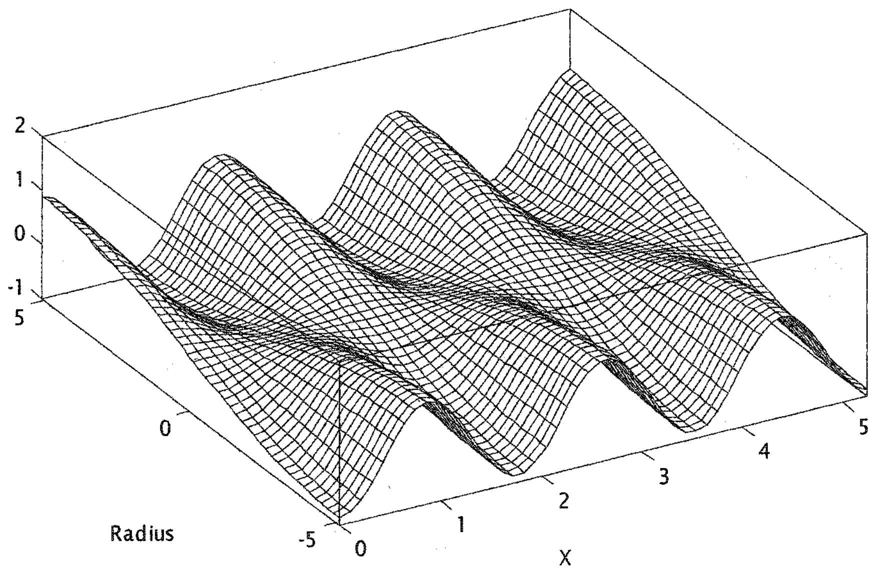


Figure 5.7– Mode P_{10} pressure distribution.

5.3 Partial Updates in the Reduced Filtered-X CMAC Algorithm

One of the fundamental limiting factors in real-time disturbance cancellation implementations is the computational overhead of the algorithm. Even though the computational overhead of the Filtered-X CMAC algorithm is modest in comparison to other approaches, it is still a limiting factor at high frequencies as well as in the case of high order compensators and secondary path models. For example, in the duct system described in section 5.4, the secondary path FIR filter is required to have 500 taps in order to adequately model the time delay and resonant structure of the secondary acoustic path. Consequently, this results in the storage of 500 delayed sets of CMAC pointers and requires the updating of 500 sets of CMAC weights on each control cycle as can be seen from the Reduced Filtered-X CMAC weight update shown in (5.9). This represents a substantial computational burden.

$$\mathbf{w}[k+1] = \mathbf{w}[k] + \frac{\alpha}{\beta} e[k] \sum_{i=0}^{M-1} b_i \mathbf{a}[k-i] \tag{5.9}$$

At the expense of reducing the convergence rate of the algorithm, the computational burden per control cycle can be reduced by performing only a subset of the weight updates on each cycle. Such partial update algorithms are commonly utilized to reduce the overhead of LMS coefficient updates in high order FIR filters as described in section 1.5. The differentiating factoring in the different partial update algorithms is the means by which the subset of weights to be updated is chosen. In the remainder of this section, three different partial update algorithms will be considered. The first two are inspired by commonly utilized approaches in adaptive FIR filters. It will be shown that these are generally not appropriate for use with the Filtered-X CMAC algorithm. Finally, a random partial update algorithm is proposed which is effective even in the case where the error signal is highly correlated with the update pointer.

The first partial update technique is given in equations (5.10) and (5.11). This algorithm is based on the sequential LMS partial update method described in section 1.5. On each iteration, a single delayed set of weights is updated using the appropriate secondary path filter coefficient and the current observed error. The index of the weight set to be updated is incremented on each discrete time step so that the algorithm linearly cycles through a single Filtered-X CMAC update in M iterations.

$$\mathbf{w}[k+1] = \mathbf{w}[k] + \alpha e[k] b_{i[k]} \mathbf{a}[k - i[k]] \tag{5.10}$$

$$i[k+1] = \begin{cases} i[k]+1 & \text{if } i[k] < M-1 \\ 0 & \text{otherwise} \end{cases} \tag{5.11}$$

Simulation and laboratory experiments reveal that this approach often results in undesirable performance. In particular, this partial update technique can lead to instability in cases where the standard update algorithm is stable. Additionally, in many cases this update technique results in extremely slow rates of convergence. These problems occur due to correlation between the error signal and the pointer update signal $i[k]$. As a result, the problems are most noticeable when the period of the pointer update signal is comparable to the fundamental period of the error signal. This in turn depends on the length of the secondary filter, the update rate, and the nature of the error signal. To illustrate the problem, consider the extreme case where the disturbance is a single tone with a period that happens to be exactly equal to the length of the secondary path impulse response. In this case, the pointer update $i[k]$ will be exactly synchronized with the error signal. As a result, each set of lagged values will only ever get updated with a single value of the error signal.

A second method which is based on the periodic update LMS algorithm is shown in equations (5.12)-(5.14). In this case, the error signal is subsampled at the rate of the entire filter update and each value of the subsampled error signal is used to update the entire set of weights.

$$e_s[k+1] = \begin{cases} e[k] & \text{if } k\%M = 0 \\ e_s[k] & \text{otherwise} \end{cases} \tag{5.12}$$

$$\mathbf{w}[k+1] = \mathbf{w}[k] + \alpha e_s[k] b_{i[k]} \mathbf{a}[k - i[k]] \tag{5.13}$$

$$i[k] = k\%M \tag{5.14}$$

In the case where the subsampled error signal has similar properties to the original error signal, then this algorithm will have stability requirements which are the same as the full algorithm. For example, if the filter is fed by a white noise source, this might be a good approximation. However, in the Filtered-X CMAC algorithm, the error signal is often highly periodic and therefore, the subsampled version of the error signal can have much different properties than the full signal. Thus, this algorithm also suffers from the problem of correlation between the updating and the error signal.

Due to the problems with the traditional approaches to partial updates, the recommended partial update algorithm for use with the Filtered-X CMAC algorithm is given in equations (5.15) and (5.16).

$$i[k] = \text{random selection from } \{0,1,\cdots,M-1\} \tag{5.15}$$

$$\mathbf{w}[k+1] = \mathbf{w}[k] + \alpha e[k] b_{i[k]} \mathbf{a}[k-i[k]] \tag{5.16}$$

In this case, the index of the weight to be updated is selected randomly on each iteration. As a result, every value of the error signal is utilized and given the random selection process, each set of delayed pointers will on average get trained utilizing values of the error signal evenly distributed across the period. As shown in (5.15) and (5.16), the algorithm provides the largest possible decrease in computations by performing only a single update per cycle. However, in general, any desired number of updates could be made per cycle thus allowing a tradeoff between computational overhead and rate of convergence.

Figure 5.8 provides a comparison of the convergence rate of the Filtered-X CMAC algorithm for different size updates using the random partial update algorithm. Each trace shows the average power in the error signal as a function of time. The parameter R indicates the number of randomly selected weights which are updated on each interval. These results are from simulation with a resonant secondary path modeled by a 300 tap FIR filter and a 300 Hz sinusoidal disturbance signal. Notice that the time required to reach a certain error threshold is inversely proportional to the number of pointer sets updated at each cycle. For example, with R=150, the algorithm requires 0.5 seconds to converge to less than 0.001. In comparison, with R=50, the algorithm takes 1.3 seconds which is approximately three times longer.

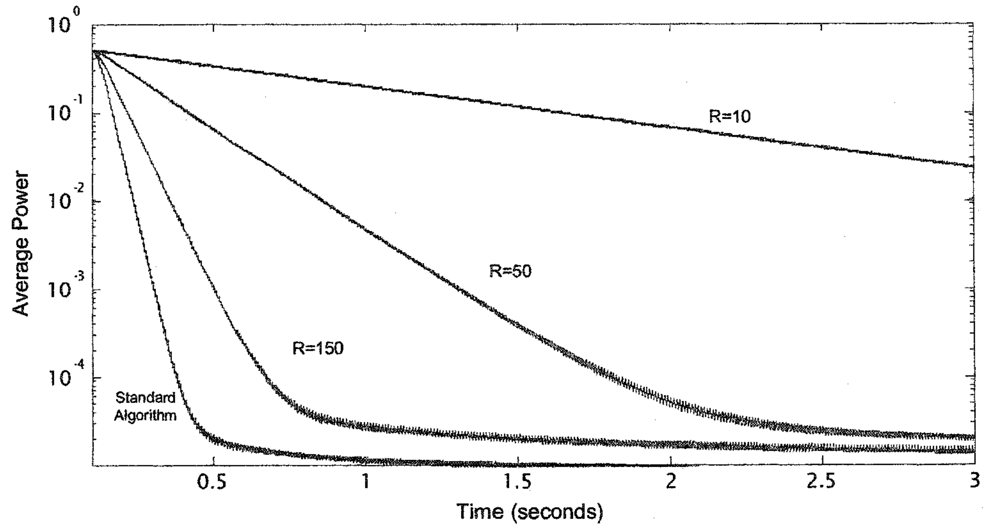


Figure 5.8 – Convergence rate versus partial update size.

5.4 The Linear Filtered-X CMAC Algorithm

In chapter two, it was shown that the complexity of a linear model can be reduced by utilizing a narrowband approximation which is accurate only over a limited range of frequencies. In the case of a single sinusoid, a suitable model at any given frequency is simply a two input FIR filter. Analogously, in the case of a time delay CMAC, a two input Time Delay CMAC can represent a mapping between two periodic signals given that certain conditions on the input are met. This capability is well suited for narrowband cancellation since in such cases the models are only required to be accurate over a small range of frequencies. In such cases, the time delay CMAC has a single closed orbit trajectory in its input space as represented by figure 5.9. For a time invariant input, the CMAC is constrained to this trajectory. As a result, the input space can be represented as a single dimension thereby permitting the use of a single input CMAC for a compensator.

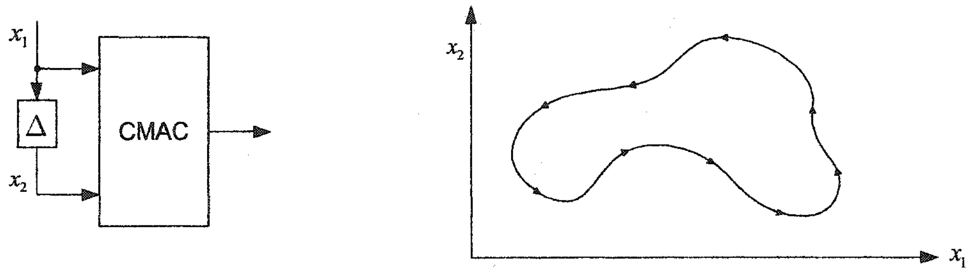


Figure 5.9 – Typical input space trajectory.

This reduction in the input space dimension can be accomplished via the technique demonstrated in figure 5.10. Figure 5.10a depicts a typical periodic input signal and its corresponding input space trajectory. Suppose a repetitive ramp waveform can be generated from the disturbance signal as shown in figure 5.10b. It is assumed that this sawtooth waveform is synchronized to the fundamental period of the actual disturbance signal. For example, such a waveform can be easily generated from a synchronization pulse provided by a tachometer. The sawtooth waveform can be used as input to a single dimension CMAC as shown. As a result, an entire period of the disturbance signal is mapped into a single pass through the one dimensional CMAC input space.

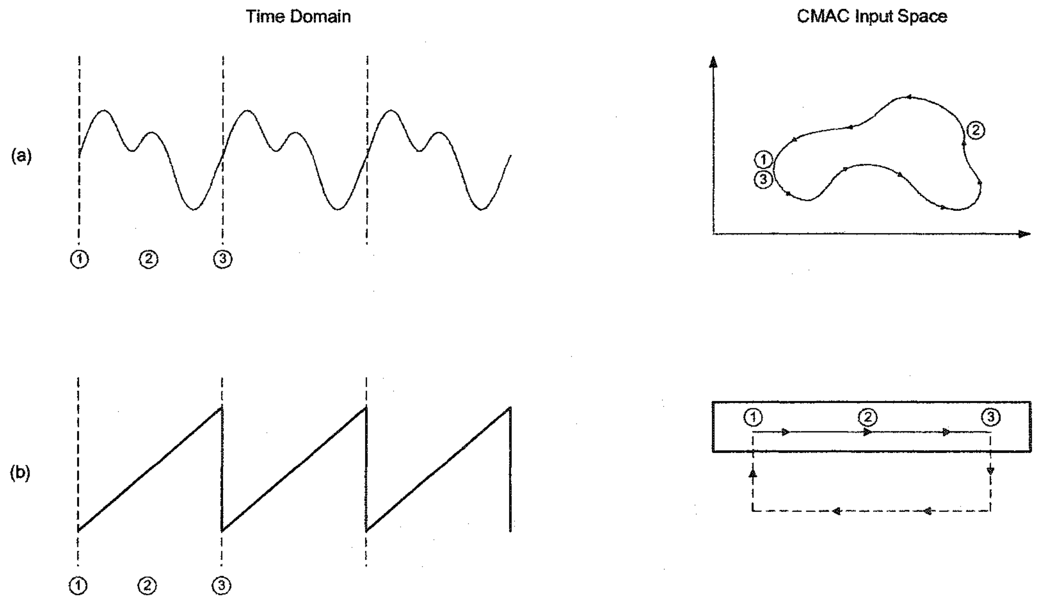


Figure 5.10 – Two and One dimensional input space trajectories.

Figure 5.11 depicts a possible hardware implementation of the linear Reduced Filtered-X CMAC algorithm. A synchronization pulse is provided by the disturbance source. This pulse is used to reset an integrator which produces a repetitive ramp, $r(t)$, that is phase locked to the disturbance signal. Meanwhile, a counter generates an estimate of the disturbance signal period which is then used to normalize the ramp signal. As a result, $u(t)$ represents a fixed amplitude sawtooth waveform independent of the period of the input signal. This provides two advantages. First, this allows for full utilization of the CMAC input space at all disturbance frequencies. Second, this allows the generalization to scale with the period of the input signal. Specifically, the generalization width remains a constant percentage of the disturbance period even as the frequency of the signal varies.

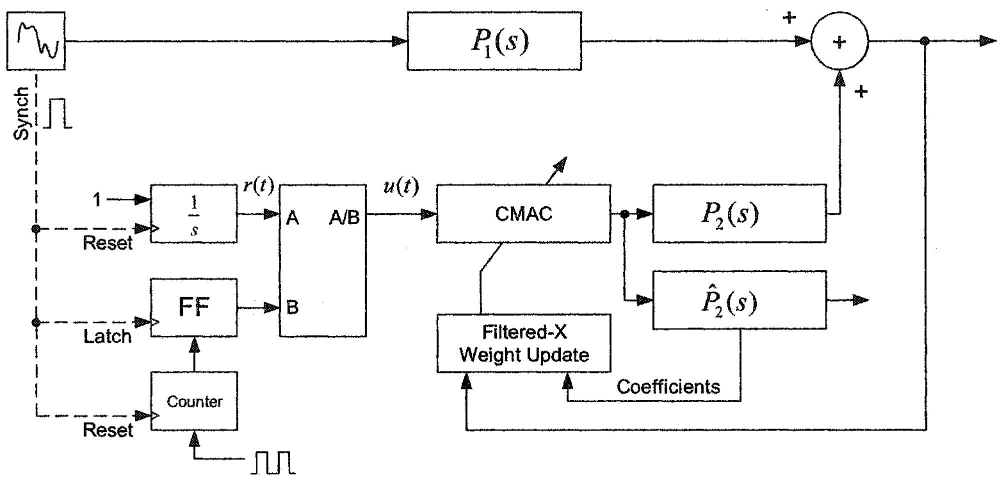


Figure 5.11 – The linear Filtered-X CMAC Algorithm.

There are several advantages of the linear CMAC algorithm described in this section. First, the reduced dimension of the CMAC compensator leads to decreased memory requirements and reduced computational overhead. Second, this technique eliminates the need to choose an appropriate delay for the tapped delay line in order to ensure an appropriate input space trajectory under various conditions. Additionally, this approach does not require an actual estimate of the reference input, but only a digital start-of-period pulse, as could be produced by a tachometer. Additionally, this approach does not allow for interference produced by nearby trajectories. Finally, this technique provides for a generalization width which automatically scales with the period of the disturbance.

However, there are disadvantages associated with this technique as well. First, it is only applicable to the case of periodic input to periodic output relationship. Additionally, the CMAC must be retrained for any change in the relationship. Finally, the end points have slightly different generalization effects then the rest of the network (i.e. there is no generalization from the end of the cycle to the beginning of the cycle). However, this effect can be eliminated through use of a circular CMAC structure.

5.5 Implementation Details and Results

This section presents results for application of the Reduced Filtered-X CMAC algorithm to an acoustic duct laboratory model. This experimental platform is intended to validate the basic functionality of the algorithm in the presence of practical limitations presented by a realistic physical setting. Some common non-ideal features include quantization effects, limited sensor range and resolution, limited computational capability, and the presence of higher order dynamics. The results presented in this section reveal that the Filtered-X CMAC algorithm is capable of providing substantial levels of attenuation even in the presence of such factors.

The laboratory system used in this research is shown in the block diagram of figure 5.12. The duct is constructed from Schedule 40 PVC pipe with an inner diameter of four inches. A midrange four inch polypropylene cone 40 Watt speaker is utilized to generate the disturbance signal in one branch of the duct work while a second speaker of the same type is utilized as the cancellation actuator. The speakers have a frequency response of approximately 85 Hz to 10 kHz and have a significant resonance located at approximately 130 Hz. A wide bandwidth uni-directional condenser microphone element is used as the error sensor and is located at the center of the duct cross-section. The microphone has a sensitivity of -64 dB and a typical frequency response of 30 Hz to 10 kHz.

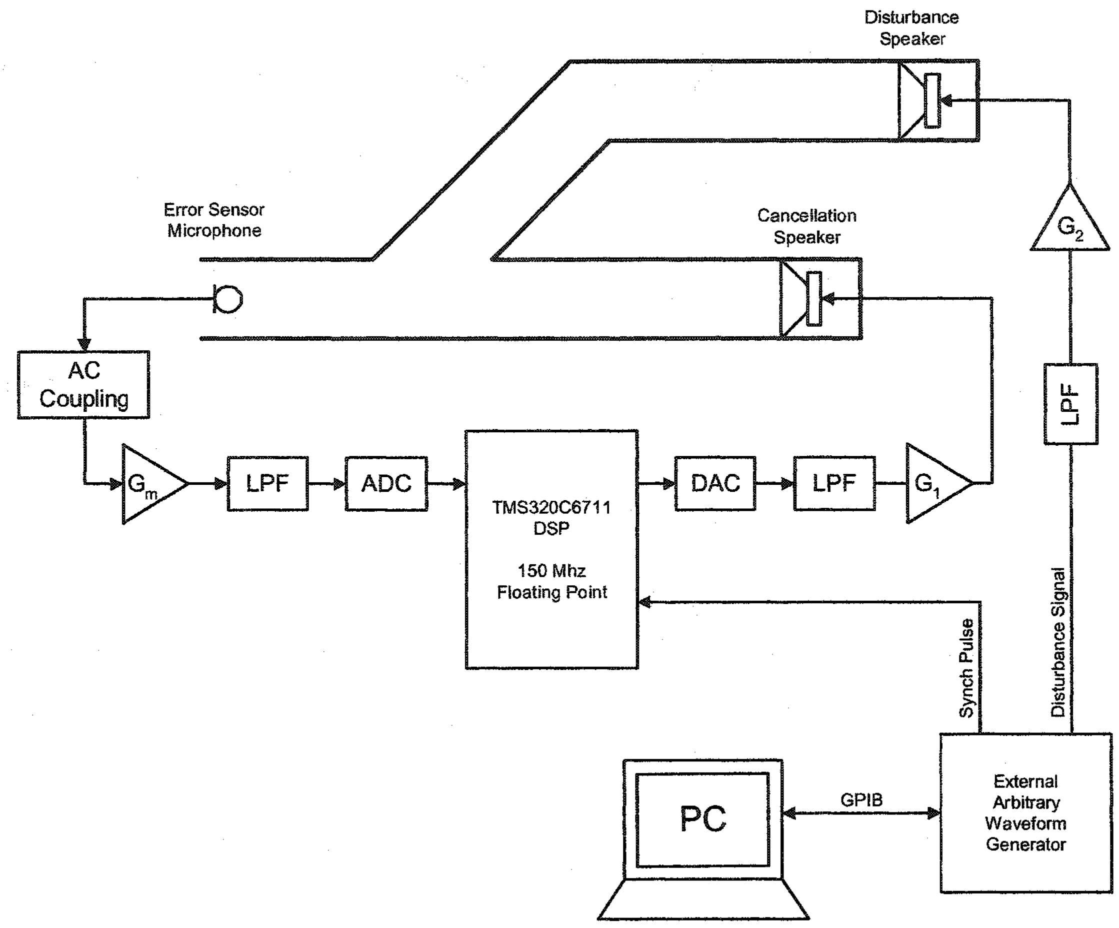


Figure 5.12 - Block Diagram of Experimental Acoustic Environment.

The signal processing hardware is designed to restrict the frequency of the disturbance and cancellation signals such that only the fundamental acoustic mode can be excited. To accomplish this goal, both of these signal paths are filtered utilizing fourth order active analog Bessel filters with a cut-off frequency of 600 Hz. The cancellation signal is generated using a 16 bit Digital to Analog converter with a maximum sample rate of 150 k samples per second. The cancellation and disturbance speaker are driven using high power operational amplifiers with maximum output current of 2 amps and a gain-bandwidth product of 1.4 Mhz. The error microphone signal is AC coupled and amplified by gain G_m. A fourth order low pass Bessel filter is utilized as an anti-aliasing filter. The amplified error signal is converted to the digital domain using a 16 bit successive approximation analog to digital converter. The high resolution of this converter eliminates the need for an automatic gain control on the error signal.

The control algorithm is implemented on the Texas Instruments Digital Signal Processor (DSP) TMS320C6711. The 'C6711 is a floating point processor with a clock rate of 150 Mhz in this application circuit. However, it superscalar architecture allows for simultaneous execution of up to eight instructions in a single clock period. As a result, under ideal circumstances, the 'C6711 can operate at 1200 MIPS. The 'C6711 variant used in this research has 72 kB of on-chip memory which can be allocated to serve either as cache or RAM. Additionally, the DSP is interfaced to an external synchronous dynamic RAM with 16 megabytes capacity. The on-chip memory has an access time of one to four CPU clocks depending on whether it is located in the level 1 or level 2 cache memory. However, the external memory can have up to a 16 clock cycle latency for single address accesses. This overhead is a significant consideration in implementation of the memory intensive CMAC algorithm.

The results presented in this section pertain to the linear Reduced Filtered-X CMAC algorithm as described in section 5.4. This variant of the Filtered-X CMAC algorithm is designed solely for cancellation of periodic disturbance signals. In this experimental platform the periodic disturbance is generated by an arbitrary waveform generator which is controlled by a PC through a GPIB interface. The only reference signal provided to the DSP is a synchronization pulse which is phase locked to the fundamental of the disturbance signal. In practical applications, such a signal is often generated by a tachometer on a piece of rotational equipment.

Figure 5.13 depicts the interrupt based execution flow of the algorithm as implemented in the DSP. The implementation is based on the single dimension version of the Filtered-X CMAC algorithm as described in section 5.4. In this case, the compensator input is a ramp which is generated by an integrator. The integrator update loop increments the integrator on 10 μs intervals while the integrator reset loop resets the integrator in response to receiving a synchronization pulse. The control loop is timer initiated at a 100 ms period. The majority of the algorithm is performed in this loop and as a result, this execution thread occupies most of the DSP execution time. The interrupt priorities are established such that all other interrupts are capable of interrupting the control loop. During normal operation, the control loop performs all of the functionality necessary to create the cancellation signal and to update the CMAC compensator. During initialization, the control loop drives the cancellation speaker with a random noise sequence and performs identification of the linear FIR secondary path model. In this implementation, the secondary path is represented by a 500th order FIR filter. This filter length allows the modeling of an impulse response of 50 ms duration. In this application, the measured transient response is approximately 42 ms.

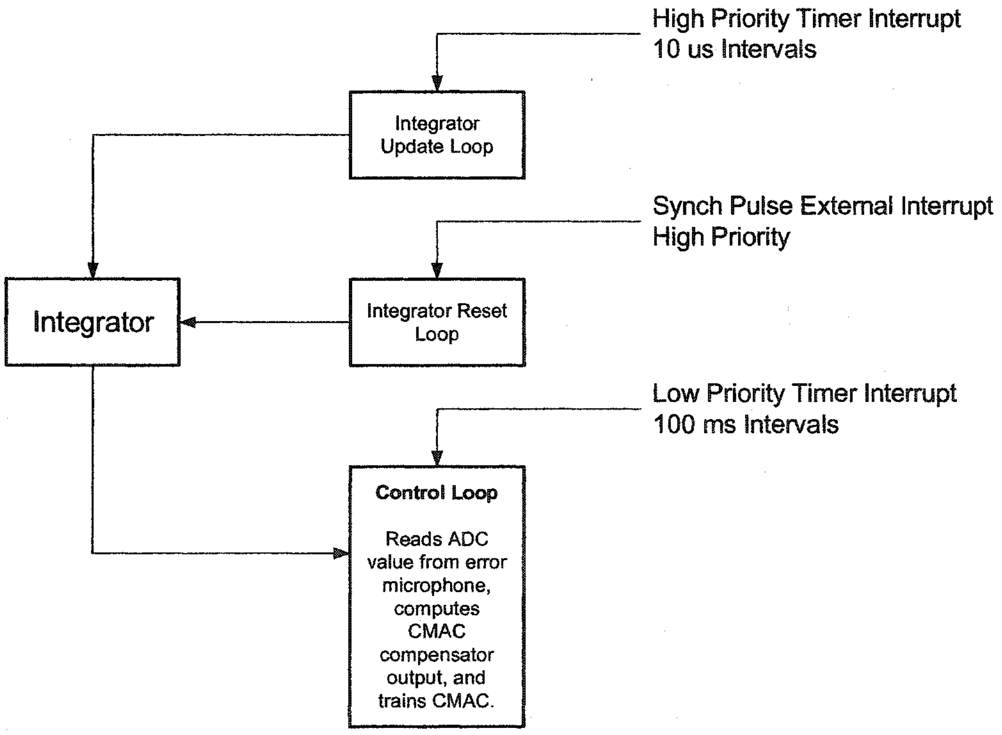

High Priority Timer Interrupt
10 us Intervals
Integrator Update Loop
Synch Pulse External Interrupt
High Priority
Integrator
Integrator Reset Loop
Low Priority Timer Interrupt
100 ms Intervals
Control Loop
Reads ADC value from error microphone, computes CMAC compensator output, and trains CMAC.

Figure 5.13 – Execution loops in the DSP implementation.

Figure 5.14 depicts the cancellation performance of the system as a function of frequency. In this experiment, the disturbance signal is a single sinusoid which was swept in frequency from 50 Hz to 800 Hz. At each frequency, a 100 ms pause was implemented to allow the CMAC to converge and the average power at each frequency was recorded. This experiment was repeated with the cancellation algorithm disabled. Comparison of the two plots indicates that the algorithm provides over 30 dB attenuation over most of the frequency range. The low frequency resonant peak in the uncancelled power spectrum is produced by the speaker resonance and the gain roll-off above 600 Hz is due to the presence of the low pass filter in the disturbance signal path.

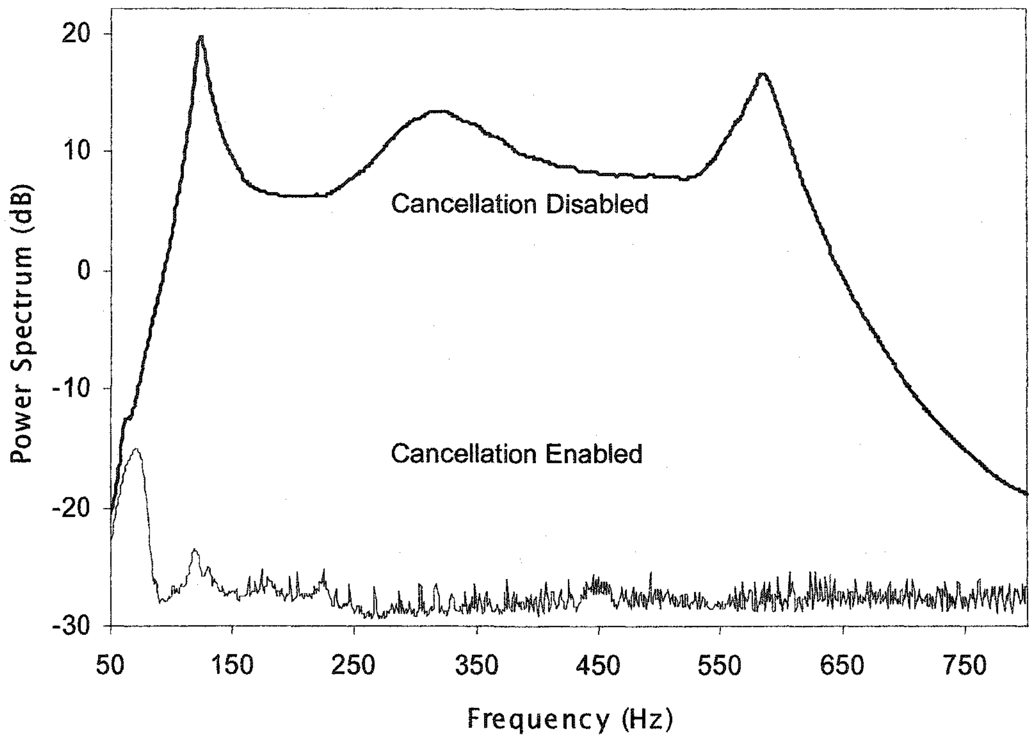


Figure 5.14 – Power Spectrum with and without cancellation enabled.

One principle advantage of the Time Delay CMAC compensator is its ability to provide cancellation in nonlinear systems. This capability is demonstrated by the present system given that the disturbance signal can be chosen such that it is nonlinearly related to the reference signal. In practical applications, this situation often occurs when the disturbance source contains harmonics of some fundamental frequency and yet the harmonics are not represented in the reference signal. To illustrate the performance of the present system in this case, the external disturbance was configured as a square wave as shown in figure 5.15. The presence of ringing in this signal is a result of the low pass filter in the disturbance signal path which eliminates some of the high frequency content of the square wave. In this figure, the compensator is disabled and therefore the cancellation signal is zero. The resultant error signal as measured by the error microphone is depicted in the bottom trace.

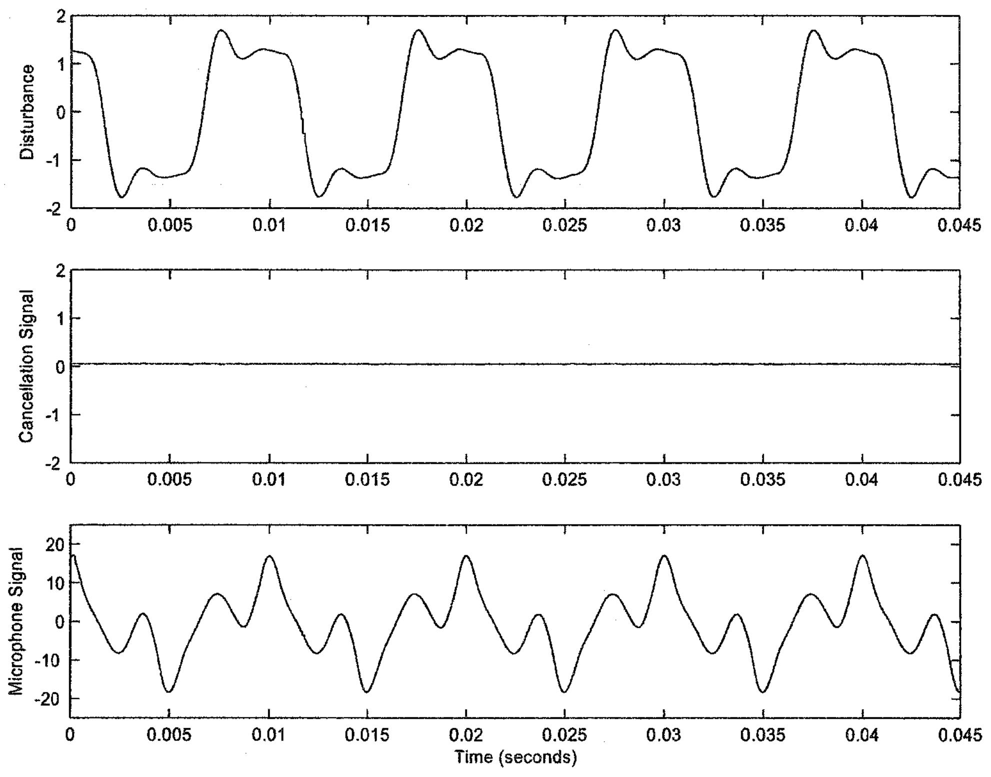


Figure 5.15 – Cancellation for a nonlinear disturbance signal.

Figure 5.16 depicts the same set of signals as shown in figure 5.15 after the CMAC compensator has been enabled and given sufficient time to converge. Notice that the error signal has been reduced to near zero indicating that the compensator output is producing an appropriate cancellation signal. In particular, this reveals that the compensator is producing energy at frequencies which are not present in its input signal. This particular example is for the case of a square wave with frequency 100Hz.

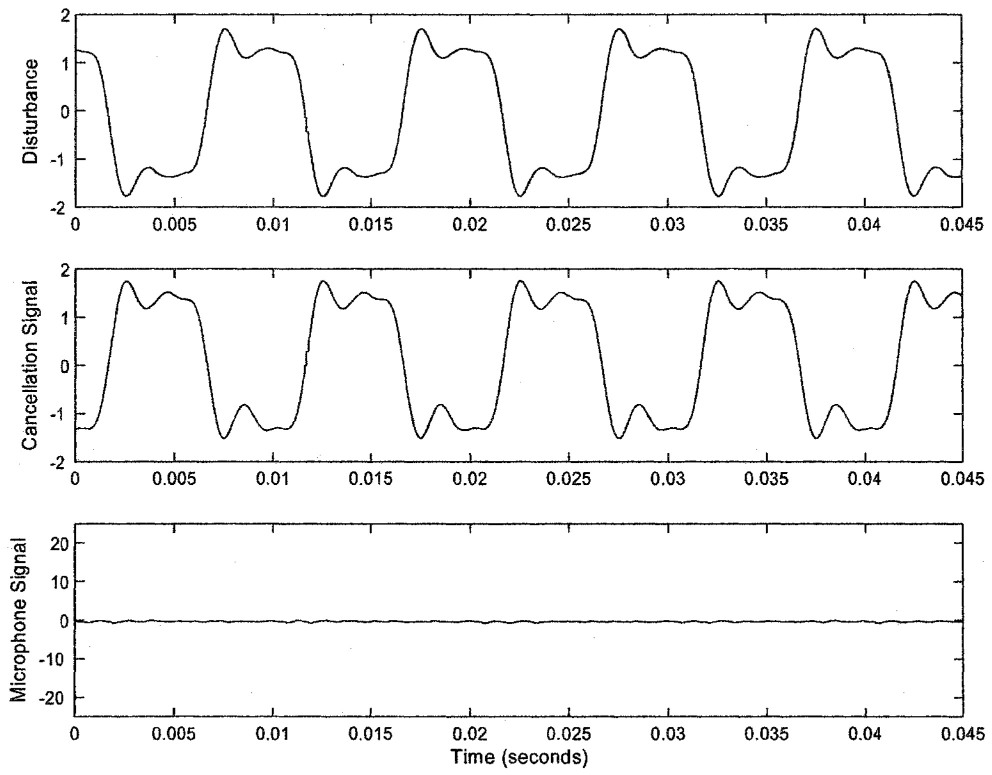


Figure 5.16 – Cancellation for a nonlinear disturbance signal.

CHAPTER 6

IMPLEMENTATION ISSUES

6.1 Introduction

This concluding chapter addresses several topics of concern to practical implementation of the CMAC compensator in active disturbance cancellation systems. To that end, section 6.2 provides an estimate of the computational requirements of the algorithm. This topic is of particular importance since the computational efficiency is the key factor in determining the upper operating frequency for the algorithm. The presented results indicate that the time delay CMAC compensator requires only marginally greater computation per control cycle then the conventional adaptive linear techniques.

In the development of the Filtered-X CMAC algorithms, it was assumed that a secondary path model was developed off-line prior to use of the CMAC compensator. Since the precision required of the secondary path model is quite minimal, this approach is suitable for most systems. However, in cases where there is great temporal variation in the secondary path, a means for continuously adapting the secondary path model during operation must be utilized. In the case of linear adaptive cancellation, several techniques for continuous secondary path modeling have been developed. In section 6.3, these techniques are reviewed in order to consider their applicability in the case of a time delay CMAC compensator.

The time delay CMAC compensator was developed in the context of single input, single output systems. However, in many practical problems, effective cancellation requires the use of multiple actuators. For example, in the case of acoustic cancellation within a cavity, at least one additional cancellation source will be required for each modal response to be attenuated. In addition, it is often necessary to minimize the disturbance at an array of error sensors rather than at a single location. Therefore, most practical disturbance cancellation systems utilize an array of cancellation sources and minimize the error over a set of error sensors. In general, it is possible for any cancellation source to influence any error sensor. In section 6.4, a multi-channel extension of the Reduced Filtered-X CMAC algorithm is presented which is suitable for such applications.

6.2 Computational Requirements of the Filtered-X CMAC Algorithms

The computational overhead associated with the CMAC neural network can be divided into that required by three separate tasks: active weight addressing, output calculation, and weight updating. Referring to the binary CMAC algorithm described in section 2.2, the active weight addressing requires the computation of β indices. It is assumed that the coarse-coding is performed via dedicated hardware and therefore, each index is created via a single subtraction and division as shown in equation (2.3). Often, the division is performed using a binary right shift operation, but in the present derivation it will be considered as an additional computation. It is assumed that the floor operation in (2.3) occurs intrinsically within the integer divide operation and therefore does not require a separate operation. Therefore, for a CMAC with T inputs and generalization β, the weight addressing operation requires $2\beta T$ computations. The output calculation involves a summation of the β active weights which requires $\beta - 1$ operations. Finally, the standard CMAC error-based weight update requires a single multiplication and addition per weight resulting in a total of 2β operations. Often, the learning gain is implemented as a binary shift operation to reduce the computational overhead, but will be considered here as a separate multiplication operation. Figure 5.1 provides a summary of the computational requirements for the binary CMAC algorithm.

Operation	Computations
Active Weight Computation	$2\beta T$
Output Calculation	$\beta - 1$
Weight Updates	2β

Figure 6.1 – Computational requirements of the CMAC algorithm.

In the Filtered-X CMAC algorithm, there are two separate time delay CMAC networks. One is used as the compensator while the other models the secondary path. Let N represent the number of inputs to the compensator network and M represent the number of inputs to the secondary path model. The compensator weights are updated via the Filtered-X CMAC weight update which is repeated in (6.1).

$$\mathbf{w}[k+1] = \mathbf{w}[k] + \frac{\alpha}{\beta} e[k] \sum_{i=0}^{M-1} \nabla_i \mathbf{a}[k-i] \tag{6.1}$$

The secondary path gradients, ∇_i, are numerically estimated using the secondary path CMAC. The simplest method for attaining these estimates is via the single sided estimate given in (6.2). In this equation, $f_{cmac}(\cdot)$ represents the recall operation of the secondary path model CMAC and $y_{\Delta i}$ represents the present input vector to the secondary path with a perturbation Δy applied to the i th element. Since the perturbation magnitude Δy is constant, a division operation is not required since this can be merged into the learning gain.

$$\nabla_i = \frac{f_{cmac}(y_{\Delta i}) - f_{cmac}(y)}{\Delta y} \tag{6.2}$$

From equation (6.2), computation of each gradient requires two evaluations of the secondary path model CMAC. However, one evaluation is common to the entire set of gradients. Therefore, computation of all M gradients requires $M+1$ evaluations of the secondary path CMAC in addition to M subtraction operations. Using the results from figure 6.1, the total computational requirement for calculation of the gradients is given in (6.3).

$$M + (M+1)(2\beta M + \beta - 1) = 2\beta M^2 + 3\beta M + \beta - 1 \qquad (6.3)$$

Computation of the current compensator output simply requires a standard CMAC output calculation comprising $2\beta N + (\beta - 1)$ computations. It is assumed that past values of the weight selection vector are stored in a tapped delay line and therefore do not require additional computation. As a result, the compensator weight updates requires $2M\beta + 1$ operations. The additional operation is the multiplication of the error signal, $e[k]$, by the learning gain which needs to be performed only a single time since that term is common to all weight updates. The total computational requirements of the algorithm are given in figure 6.2. Notice that the total requirements of the algorithm are proportional to the generalization width and also increase with the square of the input dimension of the secondary plant model.

Operation	**Computations**
Gradient Calculations	$2\beta M^2 + 3\beta M + \beta - 1$
Compensator Addressing	$2\beta N$
Compensator Output	$\beta - 1$
Weight Updates	$2M\beta + 1$
Total	$2\beta M^2 + 5\beta M + 2\beta N + 2\beta - 1$

Figure 6.2 – Computational requirements of the Filtered-X CMAC algorithm.

The Reduced Filtered-X CMAC algorithm utilizes an FIR model of the secondary path. The computational overhead is similar to the standard Filtered-X CMAC algorithm except that the gradient calculations are not necessary. This is due to the fact that the gradients are simply equal to the coefficients of the secondary path model and therefore require no additional computation. As a result, the total computation for the reduced Filtered-X CMAC algorithm are given in Figure 6.3. Notice that the reduced variant of the algorithm has substantially less computational overhead than the full version. In particular this is the case for long secondary paths since the computational in the reduced algorithm only increases linearly with this dimension.

Operation	**Computations**
Compensator Addressing	$2\beta N$
Compensator Output	$\beta - 1$
Weight Updates	$2M\beta + 1$
Total	$2\beta M + 2\beta N + \beta$

Figure 6.3 – Computational requirements of the Reduced Filtered-X CMAC algorithm.

To provide a context for these results, it is instructive to compute the corresponding computational cost of the Filtered-X LMS algorithm. Consider the analogous case where the secondary path filter has M coefficients and the linear compensator has N coefficients. The compensator output is simply an FIR computation requiring $2N$ operations. The weight updates are performed using the standard LMS update ($2N$ operations) using a filtered version of the input signal. The filtering is performed via the secondary path model which requires an additional $2M$ computations. The total number of computations for the Filtered-X LMS algorithm is tabulated in figure 6.3. Thus, the computational cost of the Reduced Filtered-X CMAC algorithm is approximately a factor of β greater then the conventional Filtered-X LMS algorithm. In the limit as $\beta \to 1$, the computational requirements of the Reduced Filtered-X CMAC algorithm are on the same order of magnitude as the Filtered-X LMS algorithm.

Operation	**Computations**
Compensator Output	$2N$
LMS Weight Updates	$2N$
Secondary Path Filter Output	$2M$
Total	$2M + 4N$

Figure 6.4 – Computational requirements of the Filtered-X LMS algorithm.

6.3 Continuous Secondary Path Adaptation

In the CMAC active disturbance cancellation algorithms presented previously in chapters three and four, it was assumed that a model of the secondary path was trained off-line prior to activation of the cancellation algorithm. Given that the accuracy requirements of the secondary path are not critical, this methodology is appropriate in many practical applications. For example, typical results for the Filtered-X LMS algorithm indicate that the secondary path model can have up to 90 degrees of phase error and the algorithm remains convergent [62]. Additionally, in the majority of applications, there is more flexibility in the design of the cancellation path than in the forward path and the physical environment can be designed so as to reduce the time variability of the secondary path. In some applications, it is also possible to provide occasional updates to the secondary path model in order to accommodate long term changes in the environment. For example, a cancellation system can be designed to provide identification of the secondary path on each initialization. However, there are other applications in which the secondary path can vary drastically during operation and the only feasible solution is for the secondary path to be adapted during cancellation in order to track these changes and maintain stability of the algorithm. In the case of a linear adaptive compensator, several techniques to accomplish this have been developed [62]. In this section, a brief overview of the most popular approaches is presented and the applicability of each to the Filtered-X CMAC algorithm is addressed.

Consider the standard feedforward cancellation architecture depicted in figure 6.5. Off-line adaptation of the secondary path model, $\hat{P}_2(z)$, requires only a straightforward application of linear system identification techniques. In particular, it is assumed that the disturbance is not present, and therefore $d[k] = 0$. Additionally, it is assumed that the compensator, $C(z)$, is disabled and has zero output. A white noise source, represented by $N(z)$, is used as excitation for the secondary path. It is important that this excitation have spectral content which extends over the entire frequency range over which cancellation is to be required. Since the disturbance signal is zero, the error signal, $e[k]$, provides a measure of the error in the secondary path model. As a result, this error signal can be utilized directly to provide LMS adaptation of the secondary path model.

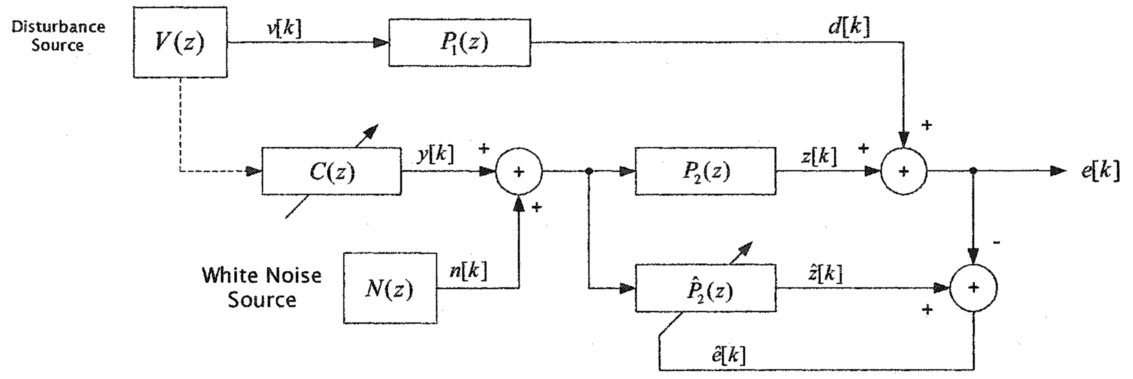


Figure 6.5 – Off-line adaptation of the secondary path model.

This algorithm is not well suited for continuous adaptation of the secondary path due to the fact that during cancellation the assumptions of zero disturbance and zero compensator output are obviously invalid. Furthermore, the compensator and disturbance signals will generally have magnitudes that are much larger then the white noise source in order to ensure that the identification noise source contributes little to the overall error. In terms of the secondary path identification, the disturbance signal represents a measurement error in the output of the secondary path. If this additive error is uncorrelated with the secondary path input, the secondary path model can still be adapted to the optimal solution in a statistically average sense. However, in this architecture, the disturbance is always highly correlated with the secondary path input whenever effective cancellation is achieved. In fact, if the error signal has been driven to zero for completely cancellation, the disturbance and cancellation signals are directly related by the secondary path transfer function. In this case, the secondary path appears to take an input signal and produce zero output from the perspective of the adaptation algorithm. This is obviously not at all indicative of the actual characteristics of the secondary path. This problem generally renders the use of this algorithm inadequate for continuous on-line adaptation of the secondary path.

In acoustic cancellation applications, on-line adaptation of the secondary path can be achieved using the third sensor method illustrated in figure 6.6 [62]. This approach is only applicable in cases where the secondary path can be divided into two segments. The first is represented by a static model while the second is modeled by a continuously adapted filter. As its name implies, this method requires an additional sensor located at the summing junction as marked by an asterisk in figure 6.6. The secondary path model is a series connection of two separate transfer functions, $P_2(z)$ and $P_3(z)$. The first transfer function, $P_2(z)$, models the cancellation actuator, the transmission path to the third sensor, and the dynamics of the third sensor. This portion of the secondary path is modeled off-line using the previously described approach and is then held constant during cancellation. The second portion of the cancellation path model, $P_3(z)$, represents the transmission path from the third sensor to the error sensor and the dynamics associated with the error sensor. This portion of the secondary path is adapted continuously during operation. Notice that there is no correlated disturbance associated with continuous adaptation of $\hat{P}_3(z)$ since, in this case, the disturbance is added at the input to the model rather than the output. The third sensor method is most applicable to cases with long secondary paths in which case, the time variability of the secondary path can be substantially reduced by using an adaptive model for the majority of the signal path.

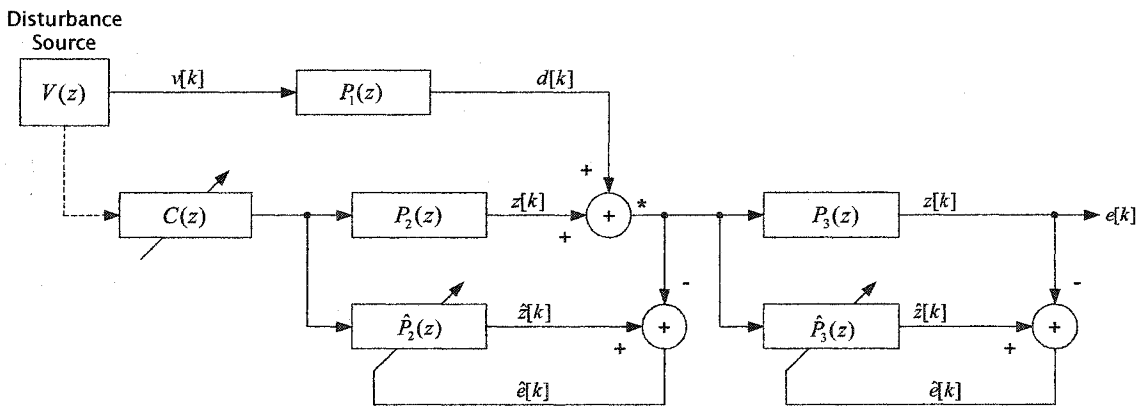


Figure 6.6 – Three sensor algorithm for on-line secondary path modeling.

The primary disadvantages of the third sensor method are its inability to provide continuous adaptation of the entire secondary path and its requirement of an additional sensor. Additionally, notice that the excitation signal used in adapting $\hat{P}_3(z)$ is dependent upon the nature of the disturbance signal and the forward path dynamics. For example, in a linear system with a single tone disturbance, this excitation source will be a single sinusoid as well. As a result, the secondary path model will be narrowband and will require additional adaptation to function at other frequencies. Additionally, the minimal spectral content generally dictates use of the leaky LMS algorithm in order to maintain stability of the coefficient updates.

It is possible to utilize the third sensor method for continuous adaptation with the Reduced Filtered-X CMAC algorithm. Let $p_i \; \forall i \in \{0,1,\cdots,P-1\}$ represent the coefficients of the continuously adapted portion of the secondary path and let $b_i \; \forall i \in \{0,1,\cdots,L-1\}$ represent the coefficients of the static portion of the secondary path. With these definitions, the compensator output, $z[k]$, can be represented by the sequential filtering operation given in equation (6.4) where $y[k]$ represents the output of the compensator.

$$z[k] = \sum_{n=0}^{P-1}\sum_{m=0}^{L-1} p_n b_m y[k-n-m] \tag{6.4}$$

From (6.4), the gradients through the complete secondary path model can be calculated with the result shown in equation (6.5).

$$\nabla_i[k] = \frac{\partial z[k]}{\partial y[k-i]} = \sum_{n=0}^{\min\{P-1,i\}} p_n b_{i-n} \tag{6.5}$$

Using the gradients given in (6.5), the reduced Filtered-X CMAC weight update rule for use with the third sensor method is given in equation (6.6).

$$\mathbf{w}[k+1] = \mathbf{w}[k] + \eta \hat{e}[k] \sum_{i=0}^{M-1} \left\{ \sum_{n=0}^{\min(P-1,i)} p_n b_{i-n} \right\} \mathbf{a}[k-i] \tag{6.6}$$

Notice that the same result can be obtained by multiplying the Z domain polynomials of the two secondary path models to obtain a single FIR filter with $P+L$ taps and then using that filter in the standard Filtered-X CMAC algorithm. However, the multiplication must be performed on every iteration to ensure that the most recent secondary path model is utilized. Notice also that this algorithm has more computational overhead then a fixed length secondary path model of the same overall length.

In the case of a periodic disturbance source, a technique known as the time difference algorithm can be utilized for continuous adaptation of the secondary path model [62]. It is assumed that the period of the disturbance, T, can be estimated directly from the reference signal. The time difference algorithm is depicted in the block diagram of figure 6.7.

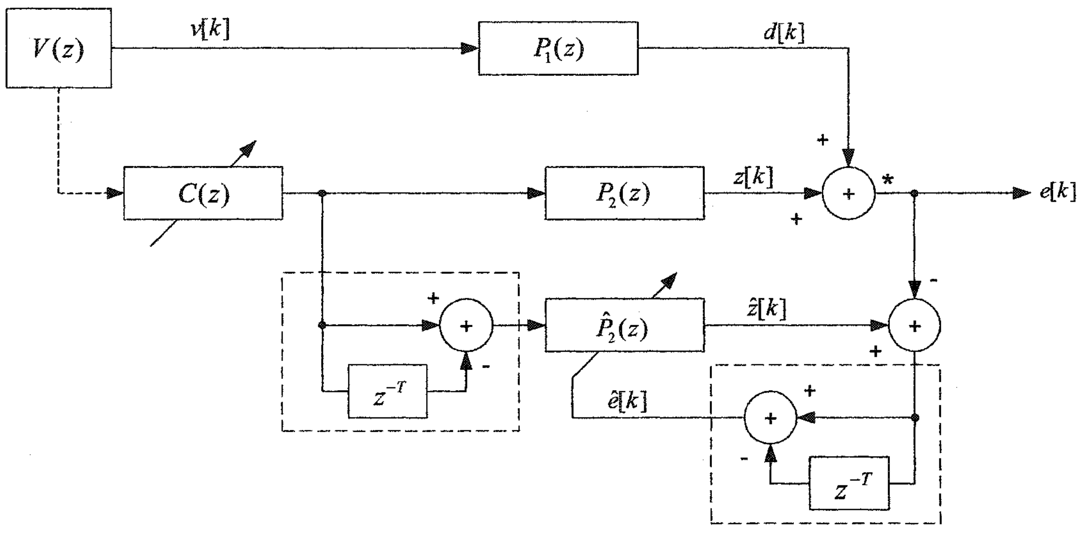


Figure 6.7 – Time Difference Algorithm for continuous secondary path modeling.

The time difference algorithm utilizes the *a priori* knowledge of periodicity in the disturbance signal to remove the effect of the cancellation signals from the input and output of the identification model. This is accomplished through the use of two comb filters as denoted by the dashed lines in figure 6.7. The comb filter results in the elimination of any signal content with period T. Such filters are simple to implement but are only effective given an accurate measurement of the fundamental period of the disturbance. By removing the periodic portion of the secondary model input and output signals, the remaining portion is uncorrelated with the disturbance and cancellation signals. Therefore, the standard LMS algorithm can be utilized for adaptation of the secondary path model. However, since most of the input signal has been eliminated, there is a concern as to whether the residual noise has sufficient spectral content to ensure convergent adaptation of the secondary path model. Use of an additive white noise excitation and the leaky LMS algorithm can alleviate this problem. The time difference algorithm can be used directly with the Reduced Filtered-X CMAC algorithm. In that case, the time difference algorithm is utilized to adapt the secondary model and the resultant coefficients can be used directly in the Reduced Filtered-X CMAC weight updates.

The final on-line secondary path modeling technique to be presented in this section is known as the additive noise method [62]. Figure 6.8 depicts an implementation of this algorithm. In this approach, an additive small amplitude noise source is continuously added to the compensator output and the secondary path model is driven solely by the additive noise source. Notice that the secondary path model does not receive the compensator output. As a result, the disturbance signal, $d[k]$, is uncorrelated with the excitation signal and the resultant adaptation is convergent in a statistical sense. In the steady state case where the disturbance is completely cancelled, the error signal, $e[k]$, will reflect only the error due to the white noise source. In this case, the algorithm reduces to the standard noiseless system identification model.

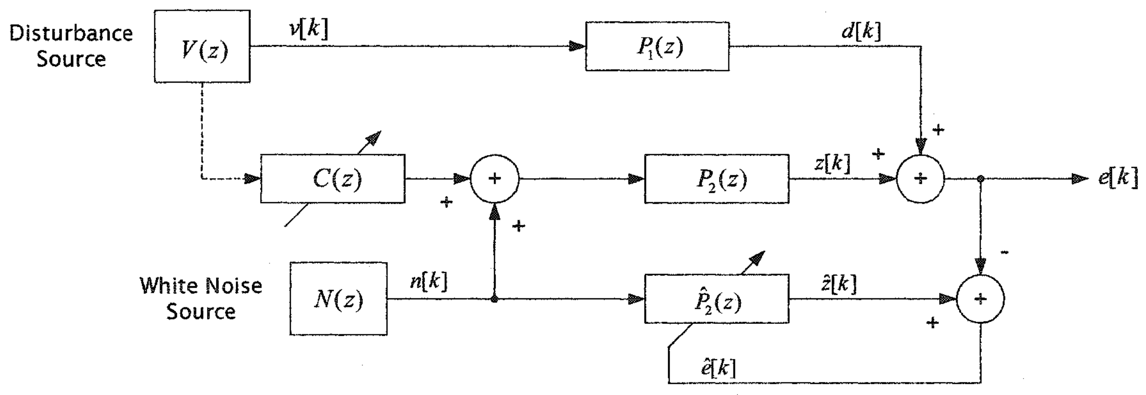


Figure 6.8 – Additive noise technique for continuous secondary path adaptation.

There are several shortcomings associated with the additive noise technique. First, the additive noise source places a limit on the maximum attainable cancellation. Second, the disturbance signal, although now uncorrelated with the model excitation, is always much larger than the excitation signal and therefore represents a very significant interference in the adaptation of the filter. This can result in a significant reduction in the rate of convergence. However, even with these limitations, this algorithm can be used without modification for adaptation of the secondary path model in the reduced Filtered-X CMAC algorithm.

The methods reviewed in this section provide the capability for continuous adaptation of the secondary path model. Each of these techniques is applicable for training of the secondary path model for use in the Reduced Filtered-X CMAC algorithm. However, these techniques cannot be utilized for continuous adaptation of a nonlinear secondary path model as required by the full Filtered-X CMAC algorithm. The reason for their failure in this case is that each of these techniques relies on the linearity of the secondary path model. In particular, each of these approaches is based on utilizing a linearly related set of training signals. However, the greater flexibility provided by a nonlinear model requires that it be trained with the actual signals which will present during operation. For example, a linear model can be trained using a low signal level white noise source and the corresponding secondary path output. Once the model converges, it will be accurate for any input signal. However, training of a time delay neural network in the same fashion will not produce a global model of the path. For example, the network would not produce the correct response for a larger amplitude signal for example. As a consequence, none of the small signal continuous identification techniques presented in this section are suited for on-line adaptation of a nonlinear secondary path model.

6.4 A Multi-Channel Filtered-X CMAC Algorithm

In most active noise and vibration cancellation applications it is not possible to achieve significant cancellation using a single actuator and single error sensor. For example, any physical system which possesses a modal response will generally require the use of an additionally actuator per mode to be cancelled. Additionally, it is often desirable to provide attenuation of the disturbance at multiple spatial locations. In such systems, each actuator is generally able to influence each error sensor through a unique dynamical transmission path. The disturbance cancellation algorithm must be capable of adapting all of the cancellation signals in a manner which reduces the collective error over the set of error sensors. In this section, an extension of the reduced Filtered-X CMAC algorithm is developed for use in such multi-channel cancellation applications. The block diagram for this system is shown in figure 6.9.

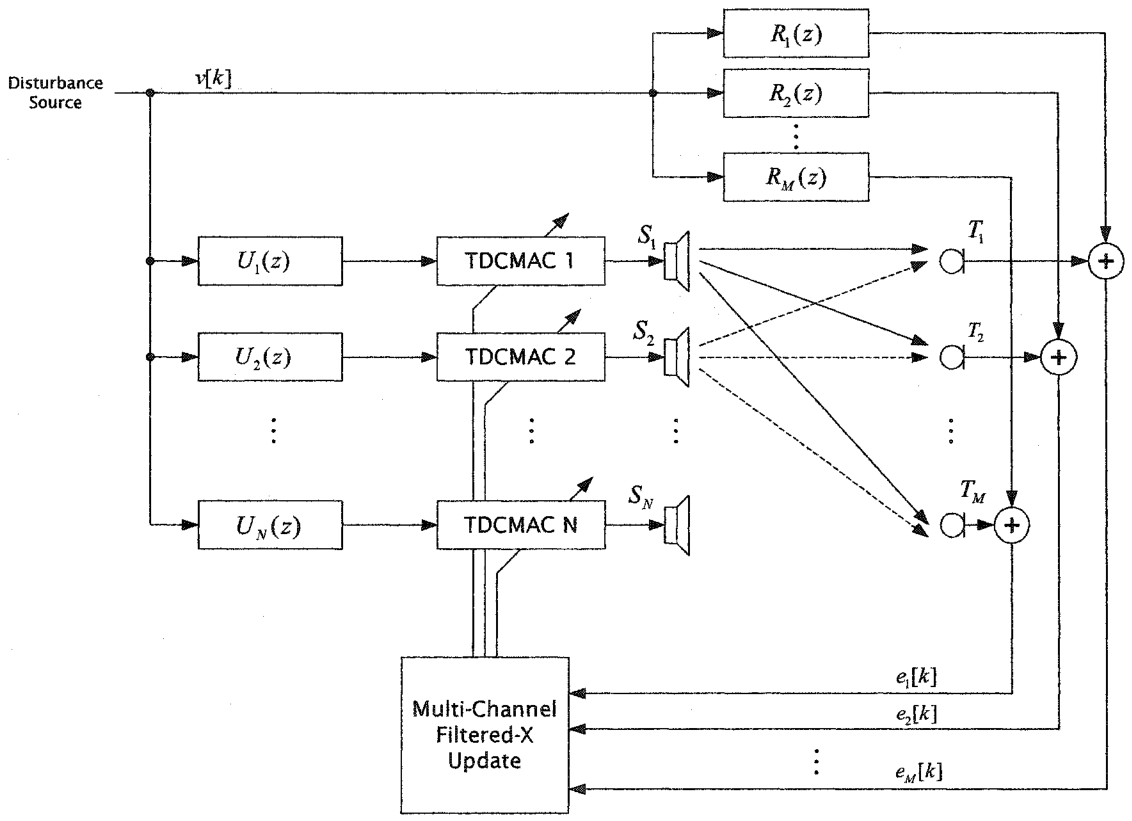


Figure 6.9 – The Multi-Channel Reduced Filtered-X CMAC Algorithm

The multi-channel cancellation system of figure 6.9 has N actuators denoted as $S_1, S_2, \cdots, S_N$. Each actuator is driven by an independent time delay CMAC network. The objective of the cancellation system is the minimization of the error signal at the set of M

error sensors denoted $T_1, T_2, \cdots, T_M$. In general, the forward path between the disturbance source and each sensor is unique. These forward paths are represented in figure 6.9 by the transfer functions $R_1(z), R_2(z), \cdots, R_M(z)$. The secondary paths consist of the transfer functions relating each actuator to each error sensor. Assuming that each of these paths can be approximated by a linear transfer function, the aggregate secondary path model can be described in the matrix equation of (6.30). The element $B^{ij}(z)$ represents the linear transfer function from the output of time delay CMAC i to the error sensor T_j.

$$\begin{bmatrix} E_1(z) \\ E_2(z) \\ \vdots \\ E_M(z) \end{bmatrix} = \begin{bmatrix} D_1(z) \\ D_2(z) \\ \vdots \\ D_M(z) \end{bmatrix} - \begin{bmatrix} B^{11}(z) & B^{12}(z) & \cdots & B^{1N}(z) \\ B^{21}(z) & B^{22}(z) & & B^{2N}(z) \\ \vdots & & \ddots & \\ B^{M1}(z) & B^{M2}(z) & & B^{MN}(z) \end{bmatrix} \begin{bmatrix} S_1(z) \\ S_2(z) \\ \vdots \\ S_N(z) \end{bmatrix} \tag{6.7}$$

The transfer function $B^{ij}(z)$ is modeled as an FIR filter with coefficients b_l^{nm} $\forall l \in \{0,1,\cdots,P-1\}$. For a simplification of the notation, it has been assumed that each of the secondary path transfer functions $B^{ij}(z)$ can be modeled by the same length filter. In a practical application, the actual filters could be chosen of different lengths if that suited the application. It is assumed that the matrix secondary path model is adapted off-line and is then held constant during adaptation of the compensators. The off-line adaptation must be performed in N phases. On phase i an excitation signal is presented at actuator S_i and transfer functions $B^{ij}(z)$ $\forall j \in \{1,2,\cdots,M\}$ are adapted.

The objective in adapting the compensator is to minimize the instantaneous sum squared error, E, as defined in (6.8).

$$E = \sum_{m=1}^{M} e_m^2[k] \tag{6.8}$$

The weight updates for training of the N CMAC compensators are based on a gradient descent of the sum of the instantaneous sum square error. Thus, the weight update for CMAC i takes the form of equation (6.9).

$$\mathbf{w}_i[k+1] = \mathbf{w}_i[k] - \alpha \frac{\partial}{\partial \mathbf{w}_i[k]} \sum_{m=1}^{M} e_m^2[k] = \mathbf{w}_i[k] - \alpha \sum_{m=1}^{M} \frac{\partial e_m^2[k]}{\partial \mathbf{w}_i[k]} \tag{6.9}$$

The error signal at sensor M is given by equation (6.10) where $d_m[k]$ represents the contribution of the original disturbance sense in the respective error signal.

$$e_m[k] = d_m[k] - \sum_{n=1}^{N} \sum_{l=0}^{P-1} b_l^{nm} y_n[k-l] \tag{6.10}$$

The partial derivative term of equation (6.9) can be simplified using this expression for the error signal. The result is given in (6.11).

$$\begin{aligned} \frac{\partial e_m^2[k]}{\partial \mathbf{w}_i[k]} &= 2e_m[k] \frac{\partial e_m[k]}{\partial \mathbf{w}_i[k]} = 2e_m[k] \frac{\partial}{\partial \mathbf{w}_i[k]} \left\{ d_m[k] - \sum_{n=1}^{N} \sum_{l=0}^{P-1} b_l^{nm} y_n[k-l] \right\} \\ &= -2e_m[k] \frac{\partial}{\partial \mathbf{w}_i[k]} \sum_{n=1}^{N} \sum_{l=0}^{P-1} b_l^{nm} \mathbf{a}_n^T[k-l] \mathbf{w}_n[k-l] \end{aligned} \tag{6.11}$$

If the compensator weights vary slowly with respect to the secondary path filter length P, then the approximation given in (6.12) is justified.

$$\frac{\partial \mathbf{w}_i[k-l]}{\partial \mathbf{w}_i[k]} \cong 1 \quad \forall\, l \in \{0,1,\cdots,P-1\} \tag{6.12}$$

With assumption (6.12), the partial derivative term of (6.9) can be further reduced producing the result given in (6.13).

$$\frac{\partial e_m^2[k]}{\partial \mathbf{w}_i[k]} = 2e_m[k] \sum_{l=0}^{P-1} b_l^{nm} \mathbf{a}_i^T[k-l] \tag{6.13}$$

Finally, substitution of this result into the weight update equation of (6.9) yields the final form of the weight update equation for the multi-channel Reduced Filtered-X CMAC algorithm as shown in (6.14).

$$\mathbf{w}_i[k+1] = \mathbf{w}_i[k] - \alpha \frac{\partial}{\partial \mathbf{w}_i[k]} \sum_{m=1}^{M} e_m[k] \sum_{l=0}^{P-1} b_l^{nm} \mathbf{a}_l^T[k-l] \quad \forall\, i \in \{1,2,\cdots,N\} \tag{6.14}$$

This weight update is valid for each of the N CMAC compensators. The overall computational demand of the multi-channel algorithm is a factor of MN times the computational requirements of the single channel variant with the same secondary path filter length.

CHAPTER 7

CONCLUSION

Neural network based active disturbance cancellation methods offer the ability to provide effective cancellation in systems characterized by significant nonlinear dynamics. This is a considerable advantage in comparison to adaptive linear approaches given that such nonlinear dynamics are commonplace in many practical applications. In addition, neural network approaches maintain the ability to adapt to changes in the structure of the system and are able to function with little *a priori* information regarding the nature of the system.

Of the many neural network architectures which have been developed, the CMAC neural network has particular advantages in the application of active disturbance cancellation. The primary strengths of the CMAC are its computational efficiency and robust convergence properties which are generally offset by its limited generalization capabilities. However, the typical active disturbance cancellation system has available an abundance of training data and therefore, additional training can be used to compensate for the limited generalization capabilities of the CMAC. Additionally, the high operational speed of the CMAC network allows the algorithm to be utilized to attain cancellation at frequencies which are not achievable via other neural network algorithms.

This dissertation has presented means by which conventional linear adaptive disturbance cancellation architectures can be extended for use with the Time Delay CMAC neural networks. In general, the CMAC weight adaptation must be modified to take account of the secondary path characteristics. For a general nonlinear secondary path, this resulted in derivation of the Filtered-X CMAC algorithm while for a linear dynamical secondary path the Reduced Filtered-X CMAC algorithm is appropriate. It was shown that these algorithms could be viewed as an extension of the Filtered-X LMS algorithm which is used in the case of linear adaptive compensators. Convergence of the Time Delay CMAC compensator was shown analytically for cases where the secondary path is comprised of a linear gain, a static nonlinearity, a time delay, or a linear dynamical system. Additionally, in each of these cases, a bound on the maximum stable learning gain was derived. These bounds provide guidance as to the necessary reduction in learning gain as a function of the system parameters.

Chapter 5 described a laboratory implementation of the Filtered-X CMAC algorithm. This demonstrates the basic feasibility of the algorithm for application to practical systems. Additionally, several practical refinements of the algorithm were introduced including the partial update schemes of section 5.1, the single dimension variant of the algorithm in section 5.3, and the multi-channel extension of the algorithm in section 6.2.

There are numerous directions in which the present research could be extended. One potential research area concerns the representational capabilities of the Time Delay CMAC network. The convergence proofs established in this research were based on the assumption that the nonlinear system could be represented by a fully trained Time Delay CMAC. However, it is evident that the Time Delay CMAC is not capable of representing any arbitrary nonlinear system. Some preliminary results concerning the types of signals which can be represented by the Time Delay CMAC were presented in section 2.3. Future work is necessary to provide a more general answer to this problem. For example, it may be possible to define certain class of periodic signals which can be represented by the Time Delay CMAC. In addition to their theoretical significance, such results would be useful in guiding practical selection of appropriate CMAC parameters for representing certain types of signals.

Another potential research direction is the extension of the alternative neural network architectures presented in chapter 4. The preliminary results presented indicate the basic feasibility of these ideas, but no attempt was made toward theoretical analysis or even rigorous empirical studies of these architectures. For example, in the case of the recurrent CMAC approach, it would be beneficial to gain insight as to the requirements of the loop filter in order to maintain stability of the algorithm.

The learning gain bounds derived for the cases of a secondary path time delay and the Reduced Filtered-X CMAC algorithm are quite conservative in nature. This was demonstrated via simulation in section 5.4. It is possible to refine these proofs so that they are tighter bounds on the actual learning gain requirements. One possibility in this regard is to implement more realistic assumptions on the overlap between nearby points. The present derivation is based on allowing a fixed amount of overlap between all nearby data points. However, in most practical applications, the degree of overlap is a function of the time difference between the data points. Therefore, one could assume a linear decrease in the degree of overlap as a function of temporal separation between the points. This should lead to a less conservative bound on the learning gain.

In conclusion, there are a variety of directions in which the present research can be extended to both increase the theoretical understanding as well as to improve the practical application of CMAC based active disturbance cancellation. It is hoped that this dissertation will provide useful assistance for such endeavors.

REFERENCES

[1] Albertos, P., Strietzel, R., and Mort, N., Control Engineering Solutions, The Institution of Electrical Engineers, London, 1997.

[2] Albus, J.S., *Brains, Behavior, & Robotics*, BYTE Publications, Inc., Peterborough, NH, 1981.

[3] An, P.-C.E., "An Improved Multi-Dimensional CMAC Neural Network: Receptive Field Function and Placement," Ph. D. Dissertation, University of New Hampshire, Durham, NH, September 1991.

[4] Annema, A.–J., *Feed-forward Neural Networks*, Kluwer Academic Publishers, Norwell, MA, 1995.

[5] Åström, K.J., and Wittenmark, B., *Adaptive Control, Second Edition*, Addison-Wesley, Reading, MA, 1995.

[6] Åström, K.J., and Wittenmark, B., *Computer-Controlled Systems, Third Edition*, Prentice-Hall, New Jersey, 1997.

[7] Barth, D.R., and Kashani, R., "Engine Noise Reduction Using Narrowband Feedback Control," *Proceedings of the SAE Noise and Vibration Conference*, Traverse City MI, May 15-18, 1995.

[8] Bell, L.H., and Bell, D.H., *Industrial Noise Control*, Marcel Dekker, Inc., New York, 1994.

[9] Bodson, M., and Douglas, S.C., "Adaptive Algorithms for the Rejection of Sinusoidal Disturbances with Unknown Frequency," Automatica, vol. 33, no. 12, pp. 2213-2221, December 1997.

[10] Bouchard, M., "New Recursive-Least-Squares Algorithms for Nonlinear Active Control of Sound and Vibration Using Neural Networks," IEEE Transactions on Neural Networks, vol. 12., no. 1, January 2001. pp. 135-147.

[11] Bouchard, M., Paillard, B., and Dinh C.T.L., "Improved Training of Neural Networks for the Nonlinear Active Control of Sound and Vibration," IEEE Transactions on Neural Networks, vol. 10, no. 2, March 1999.

[12] Bouchard, M., and Stephan Quednau, "Multichanel Recursive-Least-Squares Algorithms and Fast-Transversal-Filter Algorithms for Active Noise Control and Sound Reproduction Systems," IEEE Transactions on Speech and Audio Processing, vol. 8, no. 5, pp. 606-618, September 2000.

[13] Brogan, W.L., *Modern Control Theory*, Third Edition, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[14] Brown, M. and C. J. Harris, "Comments on 'Learning Convergence in the Cerebellar Model Articulation Controller'," IEEE Transactions on Neural Networks, vol. 6, no. 4, July, 1995.

[15] Brown, M., and Harris, C. J., "The Modelling Abilities of the Binary CMAC," Proceedings of the IEEE International Conference on Neural Networks, Orlando, Florida, 1994. pp 1335-1339.

[16] Brown, M. and Harris, C., *Neurofuzzy Adaptive Modelling and Control*. New York: Prentice Hall. 1994.

[17] Brown, M., Harris, C. J., and Parks, P. C., "The Interpolation Capabilities of the Binary CMAC," Neural Networks, vol. 6, pp. 429-440, 1993.

[18] Browne, E. T., "Limits to the Characteristic Roots of a Matrix," American Mathematical Monthly, vol. 46, no. 5, May 1939. pp. 252-265.

[19] Brualdi, R. A., and Mellendorf, S., "Regions in the Complex Plane Containing the Eigenvalues of a Matrix," American Mathematical Monthly, vol. 101, no. 10, December 1994, pp. 975-985.

[20] Burgess, J. C., "Active Adaptive Sound Control in a Duct: A Computer Simulation," J. Acoust. Soc. Am., Vol. 70, No. 3, September 1981, pp. 715-726.

[21] Campagna, D. P. "Stability and Weight Smoothing in CMAC Neural Networks." Ph. D. Dissertation, University of New Hampshire, Durham, NH, May 1998.

[22] Chassaing, Rulph, *DSP Applications Using C and the TMS320C6x DSK*, John Wiley & Sons, Inc., New York, 2002.

[23] Chen, F. –C., and Chang, C.-H., "Practical Stability Issues in CMAC Neural Network Control System," IEEE Transactions on Control Systems Technology, vol. 4, no. 1, January, 1996. pp. 86-91.

[24] Chiang, C. –T., and Lin, C. –S., "CMAC with General Basis Functions," Neural Networks, vol. 9, no. 7, pp. 1199-1211, 1996.

[25] Cotter, Neil E., and T. J. Guillerm, "The CMAC and a Theorem of Kolmogorov," Neural Networks, vol. 5, pp. 221-228, 1992.

[26] Couche, J., and Fuller, C., "Active Control of Power Train and Road Noise in the Cabin of a Sport Utility Vehicle with Advanced Speakers," 1999 International Symposium on Active Control of Sound and Vibration, Fort Lauderdale, Florida, USA.

[27] Cowan, C. F. N., and Grant, P. M. (Editors), *Adaptive Filters*, Prentice-Hall, Englewood Cliffs, NJ, 1985.

[28] D'Angelo, H., *Linear Time-Varying Systems: Analysis and Synthesis*, Allyn and Bacon, Boston, 1970.

[29] Dias, V. F., and Sommerfeldt, S. D., "Active Vibration Control of a Structure Using Virtual Sensors," *The Journal of the Acoustical Society of America*, vol. 107, no. 5, pp. 28, May 2000.

[30] Donnerstein, E., and Wilson, D. W., "Effects of Noise and Perceived Control on Ongoing and Subsequent Aggressive Behavior," Journal of Personality and Social Psychology, vol. 34., pp. 774-781, 1976.

[31] Douglas, Scott C., "Fast Implementations of the Filtered-X LMS and LMS Algorithms for Multichannel Active Noise Control," IEEE Transactions on Speech and Audio Processing, vol. 7, no. 4, July 1999, pp. 454-465.

[32] Douglas, Scott C., "An efficient implementation of the modified filtered-X LMS algorithm," IEEE Signal Processing Letters, vol. 4, no. 10, October 1997, pp. 286-288.

[33] Douglas, Scott C., "Adaptive Filters employing Partial Updates," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 44, no. 3, March 1997, pp. 209-216.

[34] Eldracher, M., Staller, A., and Pompl, R., "Adaptive Encoding Strongly Improves Function Approximation with CMAC," Neural Computation, vol. 9, pp. 403-417, 1997.

[35] Eldracher, M., Staller, A., and Pompl, R., "Function Approximation with Continuous Valued Activation Functions in CMAC," Forschungsberichte Kunstliche Intelligenz FKI-199-94, Institut fur Informatik, Technische Universitat Munchen, 1994.

[36] Elliott, S. J., Nelson, P. A., "Active Noise Control," IEEE Signal Processing Magazine, vol. 10, no. 4, pp. 12-35, 1993.

[37] Foreman, John E. K., *Sound Analysis and Noise Control*, Van Nostrand Reinhold, New York, 1990.

[38] Fowler, L. P., "Application of the Filtered-X LMS Algorithm for Disturbance Rejection in Time-Periodic Systems," Masters Thesis, Virginia Polytechnic Institute, May, 1996.

[39] Fuller, C. R., Elliot, S.J., and Nelson, P.A., *Active Control of Vibration*, Harcourt Brace & Company, New York, 1997.

[40] Fuller, C. R., and von Flotow, A. H., "Active Control of Sound and Vibration," IEEE Control Systems Journal, vol. 15, no. 6, December 1995.

[41] González-Serrano, F. J., Figueiras-Vidal, A. R., and Artes-Rodriguez, A., "Generalized CMAC Architecture and Training," IEEE Transactions on Neural Networks, vol. 9, no. 6, November 1998. pp. 1509-1513.

[42] González-Serrano, F. J., Figueiras-Vidal, A. R., and Artes-Rodriguez, A., "Fourier analysis of the generalized CMAC neural network," Neural Networks, vol. 11, pp. 391-396, 1998.

[43] Håkansson, L., Claesson, I., and Sturesson, P-O., "Adaptive Feedback Control of Machine-Tool Vibration Based on the Filtered-X LMS Algorithm," *International Journal of Low Frequency Noise, Vibration, and Active Control*, 1997.

[44] Haykin, Simon, *Neural Networks, A Comprehensive Foundation*, Macmillan College Publishing Company, Inc., New York, 1994.

[45] Haykin, Simon, *Introduction Adaptive Filters*, Macmillan Publishing Company, New York, 1984.

[46] Haykin, Simon, *Adaptive Filter Theory, Fourth Edition*, Prentice-Hall, New Jersey, 2002.

[47] Hori, N., Christenson, R. E., Seto, K., and Spencer, B. F., "Active Vibration Control of Coupled Buildings Using Relative Movement," Proceedings of the Fifth Motion and Vibration Conference, Sydney, Australia, December, 2000.

[48] Irwin, G. W., Warwick, K., and Hunt, K. J., *Neural Network Applications in Control*, The Institution of Electrical Engineers, London, UK, 1995.

[49] Jacobson, C. A., Johnson, C. R., McCormick, D. C., and Sethares, W. A., "Stability of Active Noise Control Algorithms," IEEE Signal Processing Letters, vol. 8, no. 3, March, 2001, pp. 74-76.

[50] Jagannathan, S., "Discrete-Time CMAC NN Control of Feedback Linearizable Nonlinear Systems Under a Persistence of Excitation," IEEE Transactions on Neural Networks, vol. 10, . 1, January 1999. pp. 128 – 137.

[51] Jen, H.C., Tian, Z., Smith, R., and Coghill, G.G., "Hardware implementation of a CMAC neural network using PLDs," Proceedings of the 8th Australian Conference on Neural Networks, pp. 40-44, June 1997.

[52] Kashani, R., and Sutherland, J. W., "Adaptive Feedforward Control for Periodic Disturbance Rejection with Application to Machining Processes," Transactions of The North American Manufacturing Research Conf./SME., May 1996.

[53] Ker, J. S., Kuo, Y. H., Wen, R. C., and Liu, B. D., "Hardware Implementation of CMAC Neural Network with Reduced Storage Requirement," IEEE Transactions on Neural Networks, vol. 8, no. 6, November 1997.

[54] Khalil, H. K., *Nonlinear Systems, Third Edition*, Prentice-Hall, Upper Saddle River, NJ, 2002.

[55] Kim, J. J., and Amick, H., "Active Vibration Control in Fabs," *Semiconductor International*, July, 1997.

[56] Kosko, B., *Neural Networks for Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1992.

[57] Kraft, L. G., "Optimized Weight Smoothing for CMAC Neural Networks," NIPS 97.

[58] Kraft, L. G., and Campagna, D. P., "A Comparison of CMAC Neural Networks and Traditional Adaptive Control System," Proceedings of the 1991 American Controls Conference, Pittsburgh, PA, May 1989.

[49] Kraft, L.G., and Liu, K., "Stability of CMAC Neural Networks on Closed Loop Vibration Control Systems," IASTED Controls and Applications Conference, Cancun, Mexico, 2000.

[60] Kraft, L. G., and Pallotta, J., "Vibration Control Using CMAC Neural Networks with Optimized Weight Smoothing," Proceedings of the 1999 American Controls Conference, San Diego.

[61] Kuo, S. M., Panahi, I, Chung, K., Horner, T., Nadeski, M., and Chyan, J. "Design of Active Noise Control Systems with the TMS320 Family," Texas Instruments Application Report, SPRA042, 1996.

[62] Kuo, S. M., and Morgan, D. R., *Active Noise Control Systems*. New York: John Wiley & Sons, Inc., 1996.

[63] Kuo, S. M., Panahi, I., Chung, K., Horner, T., Nadeski, M., and Chyan, J., "Design of Active Noise Control Systems with the TMS320 Family," Texas Instruments Application Note, SPRA042, June, 1996.

[64] Lane, S. H., Handleman, D. A., and Gelfand, J. J., "Theory and Development of Higher Order CMAC Neural Networks," *IEEE Control Systems Magazine*, April 1992.

[65] Lay, Steven R., *Analysis with an Introduction to Proof*, Prentice Hall, New Jersey, 1990.

[66] Levine, W. S. (Editor), *The Control Handbook*, CRC Press, Boca Raton, Florida, 1996.

[67] Lin, C. –S., Chiang, C. –T., "Learning Convergence of CMAC Technique," IEEE Transactions on Neural Networks, vol. 8., no. 6, November 1997. pp. 1281-1292.

[68] Lueg, P., "Process of Silencing Sound Oscillations," U.S. Patent No. 2,043,416, June, 1936.

[69] Miller, S. A., "Efficient Computation of the Maximum Eigenvalue of Large Symmetric Matrices," Center for Control Engineering and Computation, Technical Report CCEC-00-0817, August, 2000.

[70] Miller, W. T., "Real-Time Application of Neural Networks for Sensor-Based Control of Robots with Vision," IEEE Transactions on Systems, Man, and Cybernetics, vol. 19, no. 4, July/August 1989. pp. 825-831

[71] Miller, W. T., "Real-Time Neural Network Control of a Biped Walking Robot," Control Systems Magazine, February, 1994, pp. 41-48.

[72] Miller, W. T., Sutton, R. S., and Werbos, P. J., (Editors) *Neural Networks for Control.* Cambridge, MA, MIT Press, December 1990.

[73] Miller, W. T., "Real-time neural network control of a biped walking robot," IEEE Control Systems, pp. 41-48, February 1994.

[74] Miller, W.T., Box, B.A., Whitney, E.C., and Glynn, J.M., "Design and implementation of a high speed CMAC neural network using programmable logic cell arrays," in *Advances in Neural Information Processing Systems 3*, edited by R.P. Lippmann, J.E. Moody, and D.S.Touretzky. Morgan Kaufmann, San Mateo, CA, pp. 1022-1027, 1991.

[75] Miller, W. T., Glanz, F. H., and Kraft, L. G., "CMAC: An associative neural network alternative to backpropagation," Proceedings of the IEEE, Special Issue on Neural Networks, II, vol. 78, pp. 1561-1567, October, 1990.

[76] Miller, W. T., "Real-time neural network control of a biped walking robot," IEEE Transactions on Automatic Control, vol. 14, pp. 41-48, Feb. 1994.

[77] Miller, W. T., Hewes, R. P., Glanz, F. H., and Carter, M. J., "Real time dynamic control of an industrial manipulator using a neural network based learning controller," IEEE Journal of Robotics and Automation, vol. 6, pp. 1-9, 1990. .

[78] Miller, W. T., Hewes, R. P., Glanz, F. H., and Kraft, L. G., "Real-Time Dynamic Control of an Industrial Manipulator Using a Neural-Network-Based Learning Controller," IEEE Transactions on Robotics and Automation, vol. 6, no. 1, February 1990. pp. 1-9.

[79] Miller, W. T., and Glanz, F. H., "The University of New Hampshire Implementation of the Cerebellar Model Arithmetic Computer – CMAC," Robotics Laboratory, University of New Hampshire, Durham, NH. www.ece.unh.edu/robots/cmac.htm

[80] Minguez, A., Recuero, M., Ulin, V., and Gil, J. S., "Engine Noise Cancellation Inside An Automobile," 1999 International Symposium on Active Control of Sound and Vibration, Fort Lauderdale, Florida, USA.

[81] Mosquera, C., Gómez, J. A., Pérez, F. and Sobreira, M., "Adaptive IIR Filters for Active Noise Control," Sixth International Congress Sound and Vibration, Copenhagen, Denmark, July 5-8, 1999.

[82] Nayfeh, A., and Balachandran, B., *Applied Nonlinear Dynamics*, John Wiley & Sons, New York, 1995.

[83] Nelson, J., "Real-Time Control of a Robot Pole Balancing System Using Complementary Neural Network and Optimal Techniques," Masters Thesis, University of New Hampshire, Durham, NH, 1994.

[84] Nelson, D. S., Douglas, S. C., and Bodson, M., "Fast Exact Filtered-X LMS and LMS Algorithms for Multichannel Active Noise Control," Proceedings of the IEEE

International Conference on Acoustics, Speech, and Signal Processing, Munich, Germany, April 1997.

[85] Nelson, P. A., and S. J. Elliott, *Active Control of Sound*, Academic Press, San Diego, CA, 1992.

[86] Parks, P. C., and Militzer, J., "A Comparison of Five Algorithms for the Training of CMAC Memories for Learning Control System," Automatica, vol. 28, no. 5 pp 1027-103, 1992.

[87] Pallota, J., "Vibration Control Using Weight Smoothing CMAC Neural Networks," Masters Thesis, University of New Hampshire, Durham, NH, May 1999.

[88] Pallota, J. and Kraft, L. G., "Two Dimensional Function Learning Using CMAC Neural Network with Optimized Weight Smoothing," Proceedings of 1999 American Control Conference, San Diego, CA.

[89] Peterson, W.H., and Northwood, T. D., "Noise Raised Blood Pressure Without Impairing Auditory Sensitivity," Science, vol. 211, pp. 1450-1452, 1981.

[90] Principe, José C., Euliano, Neil R., and Lefebvre, W. Curt, *Neural and Adaptive Systems*, John Wiley & Sons, Inc., New York, 2000.

[91] Sedgewick, J., "Cut Out That Racket," *The Atlantic Monthly*, November, 1991.

[92] Shelton, R. O., and Peterson, J. K., "Controlling a Truck with an Adaptive Critic CMAC Design," Simulation, vol. 58, no. 5, pp. 319-326, May, 1992.

[93] Shou, S., and Shi, J., "Active Balancing and Vibration Control of Rotating Machinery: A Survey," *The Shock and Vibration Digest*, vol. 33, no. 4, July 2001, pp. 361-371.

[94] Skudrzy, Eugen, *The Foundations of Acoustics*, Springer-Verlag, New York, 1971.

[95] Smith, R. L., *Intelligent Motion Control with an Artificial Cerebellum*, Ph. D. Dissertation, University of Auckland, New Zealand, July 1998.

[96] Snyder, Scott D., *Active Noise Control Primer*, Springer-Verlag, New York, 2000.

[97] Snyder, S. D., and Tanaka, N., "Active Control of Vibration Using a Neural Network," IEEE Transactions on Neural Networks, vol. 6, no. 4, July 1995.

[98] Spangler, R., Vipperman, J., Pletner, B., and Brown, D., "Enabling advanced lithography through active vibration control," Micro Magazine, October, 2001.

[99] Strang, G., *Linear Algebra and Its Applications*, Third Ed., Harcourt Brace Jovanovich, Fort Worth, 1988.

[100] Thompson, D. E., and Kwon, S., "Neighborhood Sequential and Random Training Techniques for CMAC," IEEE Transactions on Neural Networks, vol. 6, no. 1, January 1995. pp. 196-205.

[101] Tokhi, M. O., and Leitch, R. R., *Active Noise Control*, Clarendon Press, Oxford, 1992.

[102] Varga, R. S., "Gerschgorin-Type Eigenvalue Inclusion Theorems and Their Sharpness," Electronic Transactions on Numerical Analysis, vol. 12, pp. 113-133, 2001.

[103] Voss, H., "Bounds for the Minimum Eigenvalue of a Symmetric Toeplitz Matrix," Electronic Transactions on Numerical Analysis, vol. 8, pp. 127-137, 1999.

[104] Wang, A. K., and Tse, B., "Adaptive Active Noise Control for Headphones Using the TMS320C30 DSP," Texas Instruments Application Note, SPRA160, January, 1997.

[105] Wang, X. G., Sun, J. C., and Xi, F. J., "Active Vibration Control of Circular Saws for Wood Cutting," NSF Design and Manufacturing Research Conference, Vancouver, British Columbia, Canada, January, 2000.

[106] Widrow, B., Shur, D., and Shaffer, S., "On Adaptive Inverse Control," Proceedings of the 15th ASILOMAR Conference on Circuits, Systems, and Computers, pp. 185-195, 1981.

[107] Widrow, Bernard, and Stearns, Samuel D., *Adaptive Signal Processing*, Prentice-Hall, New Jersey, 1985.

[108] Wong, Y., and Sideris, A., "Learning Convergence in the Cerebellar Model Articulation Controller," IEEE Transactions on Neural Networks, vol. 3, no. 1, January, 1992. pp. 115-121.

[109] Xi, F., Wang, X., and Qin, Z., "Modeling and Analysis for Active Control of Circular Saw Vibration," *Journal of Vibration and Control*, vol. 6, no. 8, pp. 1225-1241, 2000.

[110] Yoon, D. B., and Y.H. Kim, "Active Control of Radiated Duct Noise with an Insufficient Number of Sensors and Actuators," Journal of Sound and Vibration, vol 3., pp 351-369, 1999.