Cascade Adaptive Filters and Applications to Acoustic Echo Cancellation

Yuan Chen
May 1, 2013
Advisor: Professor Paul Cuff

Submitted in partial fulfillment of the requirements of the degree of
Bachelor of Science in Engineering

Department of Electrical Engineering
Princeton University

Cascade Adaptive Filters and Applications to Acoustic Echo Cancellation

Yuan Chen

Abstract

Typical approaches to acoustic echo cancellation (AEC) in mobile telephones employ adaptive linear algorithms, such as the normalized least mean squares (NLMS) algorithm. Smaller, cheaper components on these devices introduce nonlinearities into the echo path, which adversely affect performance of linear AEC systems and necessitate means of nonlinear compensation. Memoryless nonlinear blocks that compute output via interpolation between a set of control points are especially attractive solutions since they are computationally inexpensive compared to full-scale Volterra approaches and can take the the shape of any arbitrary profile. We consider normalized cascade architectures of adaptive memoryless nonlinear components – in particular the cubic B-spline function and piecewise linear function – and linear, FIR adaptive filters for purposes of nonlinear acoustic echo cancellation.

Furthermore, it is well known that the NLMS algorithm converges to the optimal Wiener linear filter, which for stationary and ergodic input signals is equivalent the least squares linear filter. We apply the least squares method to the cubic spline and piecewise linear functions to compute the optimal configuration of these nonlinear components. Although least squares estimation is in general a more difficult problem to solve for cascade architectures, we introduce an iterative method which computes the solution by performing least squares estimation on each component of the cascade separately. The result of this off-line iterative scheme serves to benchmark the performance of the on-line cascade adaptive filters.

Acknowledgements

Needless to say, there have been numerous challenges, foreseen and unforeseen, throughout the entire process of completing this senior thesis, and I could not have accomplished all that which is embodied in this document without the support of many others. First and foremost, I would like to thank Professor Paul Cuff for the tremendous work he has done as my thesis adviser. From the very beginning, he has simultaneously guided me through this project and given me freedom to pursue areas pertaining to my own interests. After each of our meetings, I leave with a renewed sense of invigoration, from both having grasped once-confusing concepts and having discovered new, exciting questions to address. My achievements in this research have come in no small part from all the time he has invested.

Of course, I owe a great deal of gratitude to my parents, for their unwavering encouragement and support throughout my years at Princeton. They have instilled in me a fondness of learning and fostered a profound sense of curiosity which continue to motivate me each day. I am also grateful for all of my friends at school who have made my experience that much easier and that much more enjoyable. Mike and Nataniel have been the best roommates for whom I could possibly ask, and I'm extremely glad they have been around. During my time here, I have had the privilege of working and interacting with the outstanding members of the ELE Class of 2013. We have helped each other along this sometimes arduous academic journey, both inside and outside of the classroom. Chris Payne, especially, deserves recognition for being my Car Lab partner and a great friend throughout these years. Lastly, but certainly not least, I'd like to thank all the members of the entire Clockwork Orange Ultimate Frisbee team, who have been nothing short of family here on campus.

Contents

Chapter 1

Introduction

In telecommunication systems, acoustic echo occurs refers to the reverberation of received signals from the far-end user in the near-end environment and subsequent additive corruption of the near-end input signal. The properties of the reverberated signal depend on the impulse response of the near-end user's environment, and as the near-end user changes his communication environment, the echo path changes as well. The purpose of acoustic echo cancellation (AEC) systems is to estimate the echo path of the near-end environment and using the far-end input signal, remove the reverberation from the near-end input. As the echo path is initially unknown and possibly changing, AEC systems must adapt to their operation environment.

Modern hands-free telephones and mobile phones use linear adaptive filtering to perform AEC [1]. The classic approach to the AEC problem is the employ the least mean squares (LMS) algorithm to form a linear estimate of the echo path [2]. As telecommunication devices employ smaller and cheaper components, the response of the loudspeaker-echo path-microphone enclosure gains nonlinear characteristics. In particular, the use of low-grade loudspeakers introduces a significant degree of nonlinear distortion, and linear AEC schemes are inadequate for complete removal of the reverberation [3]. Nonlinear acoustic echo cancellation (NAEC) systems seek to simultaneously adapt to the environment impulse response, which we consider to be linear [4], and nonlinearities associated with device components.

The full Volterra filter can represent a wide class of nonlinearities, but due to its high computational complexity and slow convergence characteristics, its use is impractical for real system implementations [1]. Using knowledge of the echo path structure, we construct a series cascade architecture of nonlinear and linear adaptive filters to model the response of the loudspeaker-echo path-microphone enclosure. The cascade architecture is a subclass of the Volterra filter but has fewer parameters and thus lower computational complexity and faster convergence [1]. The cascade of adaptive filters improves the performance of AEC systems in the presence of distorting loudspeakers [3]. The authors of [5] demonstrate the improved performance of empirically tuned filters in specific architectures for acoustic cancellation of sigmoid-distorted echos as compared to standard LMS schemes.

We implement generalized cascade adaptive filter architectures using the spline function and piecewise linear function as memoryless nonlinear preprocessing blocks. This choice of nonlinearity allows the cascade of filters to adapt to arbitrary memoryless nonlinear distortions, not just those associated loudspeaker saturation. Furthermore, by employing the method normalized nonlinear adaptive filters [6], we implement systems which do not require empirical tuning of filter parameters to fit specific operating environments. Moreover, while the authors of [3] and [5] discuss the use of the spline function in series cascade, the piecewise linear function is a novel component in how pertains to adaptive filtering and AEC.

LMS and LMS-based filters perform adaptation in an on-line approach; that is, the filter adapts with each sample of input and eventually converges to the desired configuration. Alternatively, off-line approaches use a large block of samples to directly compute the optimal configuration of filter parameters. On-line approaches perform in real time while there is a delay associated with off-line approaches. It is useful to perform off-line adaptation to compute the optimal filtering scheme as a means to evaluate the function of on-line adaptive algorithms. The method of least squares is a well know off-line fitting procedure for linear, FIR filters, and indeed, the performance of the least squares filter serves a benchmark for the performance of LMS adaptive filter. We extend least squares estimation to our proposed nonlinear components and solve the off-line adaptation problem for cascaded architectures. The premise of AEC provides a rubric to more rigorously benchmark the performance of these filter architectures, but the algorithms we examine are by no means limited to use only in AEC systems.

Chapter 2

Background

2.1 Problem Definition

Let $x(n)$ be the far-end input of a telecommunication system. The signal $x(n)$ travels through the loudspeaker-echo path-microphone enclosure, which we denote as the system $\mathcal{H}$, and let $\hat{x}(n) = \mathcal{H}[x(n)]$ be the resulting signal. Let $d(n) = \hat{x}(n) + b(n)$ be the additive combination of the reverberated far-end input and the near end input $b(n)$. The goal of an acoustic echo cancellation (AEC) system is to form an estimate, $\hat{\mathcal{H}}$, of the echo path given $x(n)$ and $d(n)$. Using this estimate of the echo path, the AEC system computes the reverberated far-end input as $y(n) = \hat{\mathcal{H}}[x(n)]$ and forms the echo-canceled output $e(n) = d(n) - y(n)$. A perfect AEC system outputs $e(n) = b(n)$. The signal $e(n)$ is also an error signal which updates the system's estimate of the echo path.

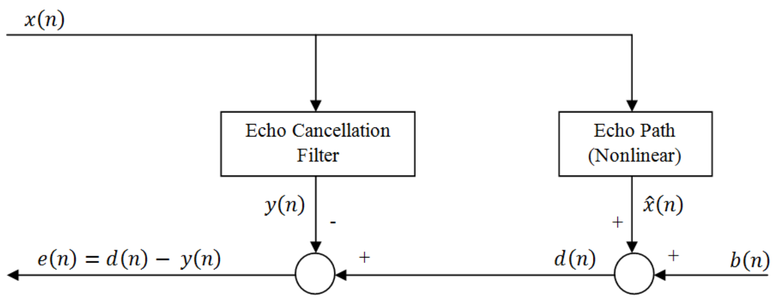


Figure 2.1: Diagram of a typical AEC system

We focus on the single-talk case, where $b(n) = 0$. The double-talk case (when both the far-end and near-end inputs are non-zero) degrades the performance of typical echo cancellation schemes since the error signal used to obtain filter tap gradients is artificially large [7]. We proceed without loss of generality, as we can extend the algorithms developed for single-talk echo cancellation into the double-talk case with the implementation of double-talk detection schemes. Under these schemes, the AEC system pauses the update process to the estimate of $\mathcal{H}$ when it detects double-talk occurring. The algorithms for single-talk echo cancellation can also be incorporated into more complex detection schemes where the update process does not pause during double-talk [8].

2.2 Least Mean Squares

The classical approach to AEC is to use the Least Mean Squares (LMS) algorithm to adaptively define a linear FIR filter as an estimate of the echo path [2]. In this approach, we define the echo cancellation filter to have the M-length taps vector

$$\mathbf{w}(n) = \begin{bmatrix} w_0(n) & w_1(n) & w_2(n) & \cdots & w_{M-1}(n) \end{bmatrix}^T$$

at the n^{th} iteration. The tap weights are updated in each iteration according to

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{x}(n), \tag{2.1}$$

where $e(n)$ is the error signal, and

$$\mathbf{x}(n) = \begin{bmatrix} x(n) & x(n-1) & x(n-2) & \cdots & x(n-M+1) \end{bmatrix}^T.$$

The output of the echo cancellation filter is defined as

$$y(n) = \sum_{i=0}^{M-1} w(n)x(n-i) = \mathbf{w}^T(n)\mathbf{x}(n). \tag{2.2}$$

μ is a positive constant value known as the step size parameter which controls the amount by which the filter weights are updated in each iteration. A value of μ

that is too small results in slow convergence while a value that is too large results in unstable behavior [2]. Proper selection of the step size parameter requires prior knowledge of input signal and echo path statistics, which vary for different inputs and echo paths. Practical implementations of AEC systems must be able to function in novel and changing environments where such statistics are unknown and difficult to estimate. One technique for the selection of the step size parameter is to normalize this value against the total energy of the input vector $\mathbf{x}(n)$. The normalized LMS (NLMS) algorithm updates the filter weights according to

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\mu_a}{\gamma + \mathbf{x}^T(n)\mathbf{x}(n)} e(n)\mathbf{x}(n), \tag{2.3}$$

where γ is a small constant to prevent division by zero, and μ_a is chosen in the range $0 < \mu_a < 2$ to guarantee convergence [9]. The output of the echo cancellation filter is also defined by equation (2.2) for the NLMS algorithm.

2.3 Nonlinearity and Cascade of Filters

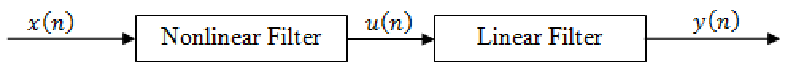
The LMS and NLMS algorithms can only provide a linear, FIR estimate of the echo path. Typically however, the echo path in the AEC problem for telecommunications applications contains nonlinearities resulting from the use of smaller, cheaper speakers and higher signal levels in hands-free operation [4]. The use of low-cost speakers in commercial devices results in poorer performance of the NLMS algorithm in AEC applications [3]. There is also an additional nonlinearity associated with the microphone which affects the signal after it has traveled through the echo path, though this effect is typically less significant than speaker distortion [4].

The Volterra series adaptive filter can represent a large class of nonlinear systems but entails a high computational complexity. We compute the output of the discrete-time Volterra filter with nonlinearity order D according to [1]

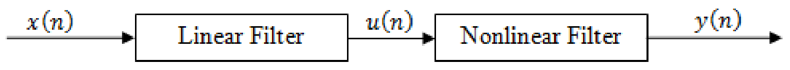
$$y(n) = \sum_{i=1}^{D} \sum_{j_1,\cdots,j_i=0}^{M-1} \theta_i(j_1,\cdots,j_i)x(n-j_1)...x(n-j_i). \tag{2.4}$$

That is, the output of the filter is the sum of products of the elements of the input vector $\mathbf{x}(n)$, and the parameters $\theta_i(j_1, \cdots, j_i)$ are weighting terms for these multiplicative combinations. We can adaptively update these parameters using a variation of LMS [1]. Because of its large number of parameters, adaptation schemes for the Volterra filter are slow to converge [4]. Though the Volterra filter improves the performance of AEC systems in the presence of nonlinear distortions, its slow convergence and high computational load render it impractical to implement on real devices.

Another method of introducing nonlinear compensation capabilities into the AEC system is by the concatenation of adaptive memoryless nonlinear filters as both preprocessing and post-processing blocks to the standard adaptive FIR filter. Two common architectures for the cascade combination of filters are the Hammerstein model, which adds a nonlinear preprocessing block to the linear filter, and the Wiener model, which adds a nonlinear post-processing block to the linear filter [5].



(a) Hammerstein Architecture



(b) Wiener Architecture

Figure 2.2: Cascade nonlinear systems

Given knowledge of the echo path and nonlinearity, we can use a model-based approach to choose a specific form of the nonlinear filter function. For example, the sigmoid function approximates the saturation and clipping associated with speaker distortion [1]. Choosing a specific adaptive nonlinearity reduces the number of parameters needed to define the filter and improves the convergence rate of the overall system [5]. Alternatively, we can choose a more generic adaptive nonlinearity such as a neural network [4] or spline function [5]. While such functions demonstrate slower convergence due to having a higher number of parameters, they can model a wider class of nonlinearities.

In general, the Hammerstein and Wiener architectures have fewer parameters to update than the Volterra filter. More specifically, we view the cascade architectures as a subclass of the Volterra filter (where many of the parameters $\theta_i = 0$) [1]. We therefore expect the cascade architectures to have faster convergence than the Volterra filter. A drawback of the cascade architecture is that we must update all filters in the cascade using only a single joint error signal $e(n)$ [10]. Furthermore, the updates to the parameters of each constituent filter are dependent, which leads to possible estimation errors [4].

2.4 General Normalized Adaptive Filters

The authors of [5] implement the Hammerstein and Wiener architectures using both the sigmoid and spline as nonlinear processing functions. The implementation updates the parameters of all the filters through a modified version of LMS. The authors, however, empirically select separate step sizes for updating individual parameters to guarantee filter convergence. Such a method of selecting step sizes again requires knowledge of signal and echo path statistics and is inappropriate for the implementation of an AEC system which operates in unknown environments. Just as NLMS provides guarantees of convergence for linear, FIR adaptive models, we seek a method to provide such guarantees to cascade filter architectures.

Given a nonlinear function of the form $y \equiv y(\mathbf{w}(n), \mathbf{x}(n))$, where $\mathbf{w}(n)$ is a vector of parameters and $\mathbf{x}(n)$ is a vector of input history, the mean squared error minimization of the LMS algorithm generalizes to [6]

$$w_i(n+1) = w_i(n) + \mu e(n) \frac{\partial y}{\partial w_i}(n), \tag{2.5}$$

where w_i is the i^{th} element of $\mathbf{w}$. We observe that when y is a linear, FIR filter, that is $y(n) = \mathbf{w}(n)^T \mathbf{x}(n)$, $\frac{\partial y}{\partial w_i} = x_i$, and equation (2.5) reduces to (2.1). Further, by optimizing the step size to minimize the expected magnitude of squared error in the next iteration, the authors of [6] arrive at the normalized update rule for general nonlinear adaptive filters:

$$\mu = \frac{\mu_a}{\sum_{i=0}^{M-1}\left(\frac{\partial y}{\partial w_i}(n)\right)^2}, \tag{2.6}$$

$$w_i(n+1) = w_i(n) + \frac{\mu_a}{\sum_{i=0}^{M-1}\left(\frac{\partial y}{\partial w_i}(n)\right)^2} e(n)\frac{\partial y}{\partial w_i}(n). \tag{2.7}$$

Again we observe that when y is a linear, FIR filter, equation (2.7) reduces to the NLMS update of equation (2.3).

As with the standard NLMS algorithm, the generalized nonlinear normalized adaptive filter requires a selection of the auxiliary step size μ_a. Unlike the NLMS algorithm, there is not a fixed interval for μ_a on which there is guaranteed convergence for all nonlinearities; rather the stability interval must be empirically determined. While empirical determination of filter parameters limits the practicality of the non-normalized LMS algorithm, the selection of this auxiliary step size does not encounter the same drawback, since the region of stability for a specific nonlinear function does not depend on signal statistics [6]. We can empirically select a known working value for μ_a, and the adaptive filter is always guaranteed to converge regardless of the input signal and operation environment.

We represent the output of a general Hammerstein cascade architecture in the form

$$\begin{aligned} y(n) &= g(f(\mathbf{x}(n), \mathbf{v}(n)), \mathbf{w}(n)) \\ &= \mathbf{w}(n)^T f(\mathbf{x}(n)), \end{aligned} \tag{2.8}$$

where $g(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T\mathbf{x}$ is a linear, FIR filter, $f(\mathbf{x}(n)) \equiv f(\mathbf{x}(n), \mathbf{v}(n))$ is a nonlinear function defined by parameters $\mathbf{v}$, and

$$f(\mathbf{x}(n)) = \begin{bmatrix} f(x(n)) & f(x(n-1)) & \cdots & f(x(n-M+1)) \end{bmatrix}^T.$$

The output of the overall filter structure is simply a linear combination of some finite output history from the nonlinear preprocessing block. The update rules for the filter weights are

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n) f(\mathbf{x}(n)), \tag{2.9}$$

$$v_i(n+1) = v_i(n) + \mu e(n)\mathbf{w}^T(n)\frac{\partial f}{\partial v_i}(\mathbf{x}(n)). \tag{2.10}$$

The term $\frac{\partial f}{\partial v_i}(\mathbf{x}(n))$ represents the vector

$$\begin{bmatrix} \frac{\partial f}{\partial v_i}(x(n)) & \frac{\partial f}{\partial v_i}(x(n-1)) & \cdots & \frac{\partial f}{\partial v_i}(x(n-M+1)) \end{bmatrix}^T.$$

In the case that $f(\mathbf{x}(n)) = \mathbf{x}(n)$, then equation (2.10) reduces to equation (2.3), the NLMS update for a linear, FIR filter. The output and update rules for the general normalized Wiener cascade architecture have a similar derivation.

2.5 Least Squares Estimation

Given the input vector $\mathbf{x}$ and associated output vector $\mathbf{y}$ of length N, the method of least squares estimation computes the filter taps vector $\mathbf{w}$ of length M which minimizes the mean squared error

$$\xi = \frac{1}{N}\sum_{j=0}^{N-1}\left(y(j) - \sum_{k=0}^{M-1} w(k)x(j-k)\right)^2. \tag{2.11}$$

In the AEC problem statement, we assume that the unknown impulse response is causal, and we further assume that $x(n) = 0, \forall n < 0$. For stationary and ergodic input signals, the least squares solution tends toward the Wiener solution for filter estimation [9]. Since the LMS algorithm converges toward the optimal Wiener simulation, we use the least squares estimate as a benchmark for adaptation performance.

The computation of the least squares solution requires the generation of the data matrix $\mathbf{A}$ from the input vector $\mathbf{x}$. Following the assumption that $x(n) = 0, \forall n < 0$, we use the pre-windowing method to generate $\mathbf{A}$,

$$\mathbf{A} = \begin{bmatrix} x(0) & 0 & 0 & \cdots & 0 \\ x(1) & x(2) & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x(N-1) & x(N-2) & x(N-3) & \cdots & x(N-M) \end{bmatrix}. \tag{2.12}$$

Rewriting (2.11) in terms of $\mathbf{A}$, the least squares estimate computes $\mathbf{w}$ such to minimize the parameter $\xi = |\mathbf{y} - \mathbf{A}\mathbf{w}|^2$. From the derivation provided in [11], we compute the least squares solution as

$$\mathbf{w} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}\mathbf{y}. \tag{2.13}$$

For linear, FIR filters and stationary, ergodic input signals, this solution $\mathbf{w}$ provides the optimal configuration of filter taps to which the LMS algorithm converges. The least squares estimation is an off-line learning method since we cannot perform adaptation instantaneously. Just as the authors of [6] extend the LMS algorithm to generalized nonlinear adaptive filters, we use the linear least squares problem statement to motivate the derivation of a least squares solution for nonlinear and cascaded filters. The purpose of the solutions to these nonlinear estimation problems is also to benchmark the performance of the cascade adaptive filter architectures.

Chapter 3

Design and Implementation

We implement Hammerstein cascade architecture adaptive filters using the spline and piecewise linear functions as nonlinear preprocessing blocks. These choices of nonlinearities allows the cascade system to adapt to a wider class of distortions than a more specific, model-based choice such as the sigmoid function. Extending the method of least squares, we propose an off-line fitting scheme for these nonlinear components and their associated Hammerstein cascade architecture systems.

3.1 Spline Function Hammerstein System

A spline function consists of a series of L control points $\left[\begin{array}{cccc} \mathbf{Q}_1 & \mathbf{Q}_2 & \cdots & \mathbf{Q}_L \end{array} \right]$, where $\mathbf{Q}_i = \left[\begin{array}{cc} x_i & q_i \end{array} \right]^T$, $x_i < x_{i+1}, \forall i$, and the spline output $F(x)$ for an input x is determined as the interpolation between these points [12] [13]. The mathematical formulation of the cubic spline ensures continuity between control points as well as the existence of its first and second derivatives [12]. The existence of the first derivative makes the spline function suitable for adaptation via gradient methods (e.g. LMS) [3]. Because the control points can take any coordinate value, the spline has the capability to take the approximate form of any given function. This characteristic is especially useful for adapting to distortions which do not have a known model.

In our implementation, there is an even space Δx between each x_i and x_{i+1}, and in addition, the position of each x_i remains the same in each update. That is, adaptation of the spline function consists only of updates the points q_i. We determine the output of the cubic spline function for a given input using only four local control points. We decompose an input x into local and global parameters ν and i. This decomposition process requires the computation of the intermediate parameters $z\prime$ and z [13]:

$$z\prime = \frac{x}{\Delta x} - \frac{L-2}{2}, \tag{3.1}$$

$$z = \begin{cases} 2 & z\prime < 2 \\ z\prime & 2 \leq z\prime \leq L-2 \\ L-2 & z\prime > L-2 \end{cases}. \tag{3.2}$$

The parameter z determines the complete index position of of an input x relative to the control points x_i. We introduce a modified, more generalized version of equation (3.1) to provide greater versatility in the spline definition:

$$z\prime = \frac{x - x_{\lceil \frac{L-2}{2} \rceil}}{\Delta x} - \left\lceil \frac{L-2}{2} \right\rceil. \tag{3.3}$$

Under the definition of equation (3.1), the spline must have an odd number of control points, the x-coordinate of its middle control point, $x_{\frac{L-2}{2}}$, must be 0, and the control points must be spread on an interval of the form $[-k, k]$. On the other hand, there are no such restrictions under the definition of equation (3.1). We also note that the floor function is an acceptable substitute to the ceiling function for calculating the "middle" index, so long as we remain consistent in our convention. Although an ideal implementation of the spline function uses control points which span the entire interval of possible inputs, equation (3.2) accounts for the case that an input falls outside the interval $[x_1, x_L]$. In particular, we map any input $x < x_1$ to the index 2 and any input $x > x_{L-2}$ to x_{L-2}. Equation (3.2) guarantees the proper handling of extreme inputs and ensures that every input has a mapping to an control point.

The decomposition of an input into global and local index components continues [13]:

$$i = \lfloor z \rfloor, \tag{3.4}$$

$$\nu = z - \lfloor z \rfloor . \tag{3.5}$$

For an input $x \in [x_1, x_L]$, i is the index of the nearest control point which is no larger than x (i.e. i satisfies $x_i \leq x < x_{i+1}$). ν is a parameter which describes the relative position of the input x between the control points x_i and x_{i+1}. One useful property of the spline function is that we can compute the output at a point x as the composition of functions, determined by the global index i and the parameter ν [12]:

$$F(x) = \sum_{k=0}^{3} q_{i+k-1} c_k(\nu), \tag{3.6}$$

where each $c_i(\nu)$ is a third order polynimal of ν.

The exact choices of c_i depend on the specific type of interpolation method. We implement a B-spline function, which uses the polynomials [13]

$$\begin{aligned} c_0(\nu) &= \frac{1}{6}\left(-\nu^3 + 3\nu^2 - 3\nu + 1\right), \\ c_1(\nu) &= \frac{1}{6}\left(3\nu^3 - 6\nu^2 + 4\right), \\ c_2(\nu) &= \frac{1}{6}\left(-3\nu^3 + 3\nu^2 + 3\nu + 1\right), \\ c_3(\nu) &= \frac{1}{6}\nu^3. \end{aligned} \tag{3.7}$$

We express the computation of equation (3.6) for a B-spline in matrix form as

$$F(x) = \begin{bmatrix} \nu^3 & \nu^2 & \nu & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} q_{i-1} \\ q_i \\ q_{i+1} \\ q_{i+2} \end{bmatrix}, \tag{3.8}$$

and we express the general computation of a spline output as [13]

$$F(x) = \mathbf{TMQ}, \tag{3.9}$$

where $\mathbf{T} = \begin{bmatrix} \nu^3 & \nu^2 & \nu & 1 \end{bmatrix}$, $\mathbf{Q} = \begin{bmatrix} q_{i-1} & q_i & q_{i+1} & q_{i+2} \end{bmatrix}^T$, and $\mathbf{M}$ is a 4×4 matrix determined by the choice of spline.

We also implement the B-spline function in a Hammerstein cascade architecture. The generalized method of normalized nonlinear adaptive filters requires the computation of the filter function's partial derivatives with respect to the updated parameters. In this case, we seek the partial derivative of the spline with respect to the control points' q coordinates. From equation (3.6), we derive this computation as

$$\frac{\partial F}{\partial q_j} = \frac{\partial}{\partial q_j}\left(\sum_{k=0}^{3} q_{i+k-1} c_k(\nu)\right), \tag{3.10}$$

$$= \begin{cases} c_k(\nu) & j = i+k-1, 0 \le k \le 3 \\ 0 & otherwise \end{cases}. \tag{3.11}$$

The partial derivative with respect to a control point q_j of the spline output for an input x is the value of the polynomial associated with the input-control point pair. Further, we express the calculation of the spline's partial derivatives in matrix form:

$$\frac{\partial F}{\partial q_j} = \begin{cases} \mathbf{T}\mathbf{M}_{j-i+2} & i-1 \le j \le i+2 \\ 0 & otherwise \end{cases}, \tag{3.12}$$

where $\mathbf{M}_k$ refers to the k^{th} column of $\mathbf{M}$. We note that in equation (3.12), $\mathbf{T}$ is a function of ν, and both i and ν are functions of x.

The general update rule for filters of this form derived in equations (2.9) and (2.10) also requires a history vector of both the input x and the output from the spline $F(x)$. To reduce the number of computations, we only calculate the spline output for the most recent input $x(n)$, and we generate the history vector

$$\mathbf{F}(n) = \begin{bmatrix} F_n(x(n)) & \cdots & F_{n-M+1}(x(n-M+1)) \end{bmatrix}^T,$$

where $F_n(x(n)) = F(x(n), \mathbf{q}(n))$. That is, the history vector of spline output depends both on the history of input and the adaptation history of the spline control points. We compute the output of the cascade of filters as

$$y(n) = \mathbf{w}^T(n)\mathbf{F}(n). \tag{3.13}$$

Additionally, it is necessary to keep a history of the spline's partial derivatives with respect to $\mathbf{q}$ to avoid excess computation. In each update iteration, the algorithm only calculates the partial derivatives of the spline at the most recent input. Unlike the calculation of spline output history, the partial derivative history depends only on the input history, since the calculation of equation (3.12) does not involve the control points.

Substituting the spline output history vector into equation (2.9), we derive the update rule for the linear filter taps of B-spline Hammerstein architecture:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{F}(n). \tag{3.14}$$

We define a $M \times L$ matrix $\Psi(n)$ which describes the partial derivative history of the function with respect to the input history:

$$\Psi(n) = \begin{bmatrix} \frac{\partial F_n}{\partial q_1} & \frac{\partial F_n}{\partial q_2} & \cdots & \frac{\partial F_n}{\partial q_L} \\ \frac{\partial F_{n-1}}{\partial q_1} & \frac{\partial F_{n-1}}{\partial q_2} & \cdots & \frac{\partial F_{n-1}}{\partial q_L} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_{n-M+1}}{\partial q_1} & \frac{\partial F_{n-M+1}}{\partial q_2} & \cdots & \frac{\partial F_{n-M+1}}{\partial q_L} \end{bmatrix}, \tag{3.15}$$

where the partial derivative term $\frac{\partial F_m}{\partial q_j}$ is evaluated at the input point $x(m)$. The update rule for the nonlinear filter parameters of this cascade filter system then becomes

$$\mathbf{q}(n+1) = \mathbf{q}(n) + \mu e(n)\mathbf{w}^T(n)\Psi(n). \tag{3.16}$$

Ψ is a sparse matrix, since in each row there are at most four non-zero terms. Even though equation (3.16) requires matrix multiplication, we can reduce the number of required computations by employing sparse matrix multiplication algorithms.

What remains is the calculation of the normalized step size μ. From equation (3.13), we derive that the partial derivative of the output y with respect to the j^{th}

linear filter tap is the j^{th} element of $\mathbf{F}$:

$$\frac{\partial y}{\partial w_j} = \mathbf{F}_j. \tag{3.17}$$

We compute the partial derivative of the output with respect to the j^{th} nonlinear filter parameter as

$$\frac{\partial y}{\partial q_j} = \mathbf{w}^T(n)\frac{\partial \mathbf{F}}{\partial q_j}(n). \tag{3.18}$$

Alternatively, we can rewrite equation (3.18) as

$$\frac{\partial y}{\partial q_j} = \mathbf{w}^T(n)\Psi_j(n), \tag{3.19}$$

where $\Psi_j(n)$ denotes the j^{th} column of $\Psi(n)$. Substituting the results of equations (3.17) and (3.19) into equation (2.6), we arrive at the normalized step size for the B-spline Hammerstein cascade architecture:

$$\mu = \frac{\mu_a}{\gamma + \mathbf{F}^T\mathbf{F} + (\mathbf{w}^T\Psi)(\mathbf{w}^T\Psi)^T}. \tag{3.20}$$

Note that in our convention, $\mathbf{F}(n)$ is a column vector, and $\mathbf{w}^T(n)\Psi(n)$ is a row vector. We empirically determine $\mu_a = 1$ as a value of the auxiliary step size for which the cascade of filters converges.

3.2 Piecewise Linear Function Hammerstein System

The piecewise linear function has a similar structure to the spline. We define the piecewise linear function through a series of L control points $\begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 & \cdots & \mathbf{Q}_L \end{bmatrix}$, where $\mathbf{Q}_i = \begin{bmatrix} x_i & q_i \end{bmatrix}^T$, $x_i < x_{i+1}, \forall i$; this is identical to the construction of the spline definition. Similar to the adaptive spline, the adaptive piecewise linear function has fixed x_i coordinates of the control points, and there is an even spacing Δx between consecutive control points. Again, the control points can adapt to any unknown linearity, and the output $G(x)$ for an arbitrary input x is computed as the interpolation between control points. The piecewise linear function only interpolates between two consecutive control points (compared to four control points required for the spline), which simplifies the required computation.

The output computation still requires the intermediate step of decomposing the input x into the global parameter i and the local parameter ν. Similar to the decomposition process with spline functions, the decomposition process for piecewise linear functions requires the computation of the intermediate parameters $z\prime$ and z:

$$z\prime = \frac{x - x_{\lceil \frac{L-2}{2} \rceil}}{\Delta x} - \left\lceil \frac{L-2}{2} \right\rceil , \tag{3.21}$$

$$z = \begin{cases} 1 & z\prime < 1 \\ z\prime & 1 \leq z\prime \leq L-1 \\ L-1 & z\prime > L-1 \end{cases} . \tag{3.22}$$

Equation (3.22) serves a similar role as equation (3.2), which is to account for input which falls outside the interval $[x_1, x_L]$, but has a different upper bound case than since the piecewise linear function only needs two control points to calculate output.

The decomposition process proceeds for the piecewise linear function in the exact same fashion as for the spline function, and we compute i and ν:

$$i = \lfloor z \rfloor , \tag{3.23}$$

$$\nu = z - \lfloor z \rfloor . \tag{3.24}$$

Recall that for an input $x \in [x_1, x_{L-1}]$, the parameter i is the index of the largest valued control point no greater than x (i.e. $x_i \leq x < x_{i+1}$), and accordingly, we refer to i as a global parameter. Also recall that ν is a local parameter which describes the relative position of x between x_i and x_{i+1}. Whereas for a spline the output for an input x is the composition of four functions determined according to i and ν, in the piecewise linear case, the output is simply the linear interpolation between q_i and q_{i+1} with ν acting as a weighting term. That is, we compute the output of the function for an input x as

$$G(x) = (1-\nu)\, q_i + \nu q_{i+1}. \tag{3.25}$$

Equation (3.25) replaces the set of polynomials in equation (3.7). Furthermore, unlike splines, there is no choice in the specific "type" of piecewise linear function as equation (3.25) is the only interpolation scheme between two consecutive control points. We express the computation of function output in matrix form as

$$G(x) = \begin{bmatrix} \nu & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q_i \\ q_{i+1} \end{bmatrix}. \tag{3.26}$$

As with the spline, the output computation for the piecewise linear function has a more general form:

$$G(x) = \mathbf{TMQ}, \tag{3.27}$$

where $\mathbf{T} = \begin{bmatrix} \nu & 1 \end{bmatrix}$, $\mathbf{Q} = \begin{bmatrix} q_i & q_{i+1} \end{bmatrix}^T$, and $\mathbf{M}$ is a fixed 2×2 matrix as shown in equation (3.26). The piecewise linear function requires fewer computations since the calculation of its output requires the multiplication of a matrix that is one-quarter of the size of the matrix used in calculating the spline output.

From equation (3.25), we derive the expressions for the function's partial derivatives with respect to the control points:

$$\frac{\partial G}{\partial q_j} = \begin{cases} 1-\nu & j = i \\ \nu & j = i+1 \\ 0 & otherwise \end{cases}. \tag{3.28}$$

We can express equation (3.25) in the form of matrix operations as

$$\frac{\partial G}{\partial q_j} = \begin{cases} \mathbf{TM}_{j-i+1} & j \in \{i, i+1\} \\ 0 & otherwise \end{cases}, \tag{3.29}$$

where $\mathbf{M}_k$ denotes the k^{th} column of $\mathbf{M}$, and i is a function of the input x. This is exactly analogous to the partial derivative calculation for the spline function.

Just as for the spline function, the implementation of a Hammerstein cascade architecture using a piecewise linear preprocessing block requires the history of both the input x and the output of the nonlinear function $G(x)$. Again, to reduce the amount of computation, we only calculate the nonlinear function output for the most recent input $x(n)$ in each iteration. This process gives rise to the piecewise linear output history vector

$$\mathbf{G}(n) = \begin{bmatrix} G_n(x(n)) & G_{n-1}(x(n-1)) & \cdots & G_{n-M+1}(x(n-M+1)) \end{bmatrix}^T,$$

where $G_n(x(n)) = G(x(n), \mathbf{q}(n))$. We then express the output of the whole cascade of filters as

$$y(n) = \mathbf{w}^T(n)\mathbf{G}(n). \tag{3.30}$$

For adaptation purposes, we also keep a history matrix of the function's partial derivatives with respect to control points, and this history matrix has a very similar structure to its analog in the spline Hammerstein implementation.

We derive the update rule for the linear filter taps by substituting the the function output history vector into equation (2.9), and we note that it an identical structure to (3.14).

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{G}(n). \tag{3.31}$$

The history matrix of function partial derivatives $\Psi(n)$ is a $M \times L$ matrix with the same structure as the matrix in equation (3.15):

$$\Psi(n) = \begin{bmatrix} \frac{\partial G_n}{\partial q_1} & \frac{\partial G_n}{\partial q_2} & \cdots & \frac{\partial G_n}{\partial q_L} \\ \frac{\partial G_{n-1}}{\partial q_1} & \frac{\partial G_{n-1}}{\partial q_2} & \cdots & \frac{\partial G_{n-1}}{\partial q_L} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial G_{n-M+1}}{\partial q_1} & \frac{\partial G_{n-M+1}}{\partial q_2} & \cdots & \frac{\partial G_{n-M+1}}{\partial q_L} \end{bmatrix}, \tag{3.32}$$

where the partial derivative term $\frac{\partial G_m}{\partial q_j}$ is evaluated at the input point $x(m)$. The update rule for the nonlinear filter parameters of this cascade filter system is then

$$\mathbf{q}(n+1) = \mathbf{q}(n) + \mu e(n)\mathbf{w}^T(n)\Psi(n). \tag{3.33}$$

The determination of the normalized step size μ has the same derivation for the piecewise linear Hammerstein system as the spline Hammerstein system. Accordingly, we omit the derivation and only report the final result:

$$\mu = \frac{\mu_a}{\gamma + \mathbf{G}^T\mathbf{G} + (\mathbf{w}^T\Psi)(\mathbf{w}^T\Psi)^T}. \tag{3.34}$$

We empirically determine $\mu_a = 1$ as a choice of the auxiliary step size for which the filter converges. The piecewise linear function is a simplification of the spline function, which can approximately attain similar results but requires fewer computations in both the output calculation and parameter updates. The partial derivative history matrix $\Psi(n)$ is again sparse, but for the piecewise linear function, each row of this matrix contains only two nonzero terms (compared to four for the spline function).

3.3 Least Squares Nonlinear Estimation

For both the adaptive spline and adaptive piecewise linear functions, we can compute a least squares estimate of the optimal configuration of control points. Given the input signal $\mathbf{x}$ and output signal $\mathbf{u}$, both of length N, the least squares estimate computes the control points vector $\mathbf{q}$ of length L to minimize the mean squared error

$$\xi = \frac{1}{N}\sum_{j=0}^{N-1}\left(u(j) - F(x(j),\mathbf{q})\right)^2. \tag{3.35}$$

From the construction of the spline and piecewise linear function, we can represent the output of $F(x(j),\mathbf{q})$ as a linear combination of the control points:

$$F(x(j),\mathbf{q}) = \sum_{i=1}^{L}\beta(j,i)q(i). \tag{3.36}$$

There exists a mapping from each input $x(j)$ to the vector

$$\beta(j) = \begin{bmatrix} \beta(j,1) & \cdots & \beta(j,L) \end{bmatrix}.$$

β is a sparse vector, since the output of the cubic B-spline is the linear combination of four control points (similarly, the output of the piecewise linear function is the linear combination of two control points), and we can compute the nonzero terms as

$$\left[\begin{array}{ccc} \beta(j, i(x(j)) - 1) & \cdots & \beta(j, i(x(j)) + 2) \end{array}\right] = \mathbf{TM}, \tag{3.37}$$

where $\mathbf{T}$ and $\mathbf{M}$ are as defined according to the construction of the spline function. Note that in equation (3.37) i refers to the index computation for spline input (as found in equation (3.4)) and not to a variable of summation (as found in equation (3.36)). The computation of the nonzero terms in β for the piecewise linear function has similar structure to to equation (3.37) but has different indexing values.

Substituting the result from equation (3.36) into equation (3.35), we can rewrite the mean squared error as

$$\xi = \frac{1}{N} \sum_{j=0}^{N-1} \left(u(j) - \sum_{i=1}^{L} \beta(j, i) q(i) \right)^2. \tag{3.38}$$

At the optimal configuration of $\mathbf{q}$, the partial derivative $\frac{\partial \xi}{\partial q_k} = 0, \forall k \in \{1, 2, \dots, L\}$ (note that we use the notation $q_k \equiv q(k)$). Solving for the partial derivative of MSE with respect to a single control point, we have

$$\frac{\partial \xi}{\partial q_k} = \frac{1}{N} \sum_{j=0}^{N-1} \left(2\beta(j, k) \left(\sum_{i=1}^{L} \beta(j, i) q(i) \right) - 2\beta(j, k) u(j) \right). \tag{3.39}$$

Setting the result from equation (3.39) to 0, we have the optimality condition

$$\sum_{j=0}^{N-1} \beta(j, k) u(j) = \sum_{j=0}^{N-1} \beta(j, k) \sum_{i=1}^{L} \beta(j, i) q(i). \tag{3.40}$$

The solution to least squares nonlinear estimation of cubic B-spline functions satisfies the constraint presented in equation (3.40) for all k simultaneously.

As a preliminary for expressing the constraint for all k in matrix form, we define the matrix

$$\Omega = \begin{bmatrix} \beta(1,1) & \beta(1,2) & \cdots & \beta(1,L) \\ \beta(2,1) & \beta(2,2) & \cdots & \beta(2,L) \\ \vdots & \vdots & \ddots & \vdots \\ \beta(N,1) & \beta(N,2) & \cdots & \beta(N,L) \end{bmatrix}. \tag{3.41}$$

Ω has similar structure to the matrix Ψ defined in equation (3.15) since $\beta(j,k) = \frac{\partial F_j}{\partial q_k}$. Whereas Ψ is a $M \times L$ matrix, where M is the number of filter taps in the linear portion of a Hammerstein cascade system, Ω is a $N \times L$ matrix, where N is the number of samples of input and output. Enumerating all L constraints of the form described in equation (3.40), we have

$$\begin{bmatrix} \sum_{j=0}^{N-1} \beta(j,1)u(j) \\ \sum_{j=0}^{N-1} \beta(j,2)u(j) \\ \vdots \\ \sum_{j=0}^{N-1} \beta(j,L)u(j) \end{bmatrix} = \begin{bmatrix} \sum_{j=0}^{N-1} \beta(j,1) \sum_{i=1}^{L} \beta(j,i)q(i) \\ \sum_{j=0}^{N-1} \beta(j,2) \sum_{i=1}^{L} \beta(j,i)q(i) \\ \vdots \\ \sum_{j=0}^{N-1} \beta(j,L) \sum_{i=1}^{L} \beta(j,i)q(i) \end{bmatrix}. \tag{3.42}$$

Using the definition of Ω, equation (3.42) becomes

$$\Omega^T \mathbf{u} = \Omega^T \Omega \mathbf{q}, \tag{3.43}$$

and solving for $\mathbf{q}$, we have

$$\mathbf{q} = \left(\Omega^T \Omega\right)^{-1} \Omega^T \mathbf{u}. \tag{3.44}$$

Similar to the least squares solution for the linear, FIR filter, the computation of the least squares estimate for the cubic B-spline and piecewise nonlinear requires solving a set of normal equations. Again, the least squares estimate is an off-line learning algorithm used to benchmark the performance of the on-line adaptive filters. Based on the relationship between the linear least squares filter and the LMS algorithm, we expect the nonlinear adaptive filters to converge to the least squares nonlinear estimate for stationary and ergodic inputs. Additionally, the derivation of the least squares estimate for standalone filters leads to an off-line learning scheme for cascaded filters.

The solution to the least squares estimate of a Hammerstein cascade of filters is the M-length taps vector $\mathbf{w}$ and L-length control points vector $\mathbf{q}$ which minimizes the parameter

$$\xi = \frac{1}{N} \sum_{j=0}^{N-1} \left(y(j) - \sum_{k=0}^{M-1} w(k) \sum_{i=1}^{L} q(i)\beta(j-k+1, i) \right)^2, \tag{3.45}$$

given the N-length input $\mathbf{x}$ and output $\mathbf{y}$ vectors. Just as in the linear least squares case, we assume that $x(n) = 0, \forall n < 0$ and consequently $\beta(n, i) = 0, \forall i \in \{1, 2, \ldots, L\}$, $n < 0$. Because the solution requires solving for both the filter taps and control points simultaneously, this problem is harder to solve directly than the least squares estimate for standalone filters. As an alternative, we propose an iterative approach to computing the least squares estimate of cascade filter architectures which solves for the least squares estimate of each standalone filter individually.

In the n^{th} iteration of this process, we first calculate the output of the spline preprocessing block

$$\mathbf{u}_n = F(\mathbf{x}, \mathbf{q}_{n-1}), \tag{3.46}$$

using the input vector and the control points vector of the previous iteration. The initial configuration of the control points $\mathbf{q}_0$ forms an identity mapping (i.e. $\mathbf{u}_1 = \mathbf{x}$). Given the spline output vector, we compute the least squares fit for the standalone linear, FIR filter and update the filter taps

$$\mathbf{w}_n = \left(\mathbf{A}_n^T \mathbf{A}_n\right)^{-1} \mathbf{A}_n^T \mathbf{y}, \tag{3.47}$$

where we use $\mathbf{u}_n$ to construct the data matrix $\mathbf{A}_n$.

We then update the estimate of the control points vector $\mathbf{q}_n$ to minimize the mean squared error given the updated filter taps $\mathbf{w}_n$. Changing the order of summation in equation (3.45), we have

$$\xi = \frac{1}{N}\sum_{j=0}^{N-1}\left(y(j) - \sum_{i=1}^{L} q(i) \sum_{k=0}^{M-1} w(k)\beta(j-k+1,i)\right)^2 . \tag{3.48}$$

Equation (3.48) collects all terms indexed by k under one summation, and as a result, we can solve for the final term separately. That is, we perform the summation and define

$$\phi(j,i) = \sum_{k=0}^{M-1} w(k)\beta(j-k+1,i), \tag{3.49}$$

which yields a term that is dependent only on indices and i and j. Substituting this term into equation (3.48), we arrive at the following expressing for MSE:

$$\xi = \frac{1}{N}\sum_{j=0}^{N-1}\left(y(j) - \sum_{i=1}^{L} \phi(j,i)q(i)\right)^2 . \tag{3.50}$$

Note that equation (3.50) has the exact same structure as equation (3.38), and the rest of the least squares computation proceeds identically.

Taking the partial derivative of MSE with respect to each of the control points and setting the result to 0, we obtain L optimality constraints

$$\begin{bmatrix} \sum_{j=0}^{N-1} \phi(j,1)u(j) \\ \sum_{j=0}^{N-1} \phi(j,2)u(j) \\ \vdots \\ \sum_{j=0}^{N-1} \phi(j,L)u(j) \end{bmatrix} = \begin{bmatrix} \sum_{j=0}^{N-1} \phi(j,1) \sum_{i=1}^{L} \phi(j,i)q(i) \\ \sum_{j=0}^{N-1} \phi(j,2) \sum_{i=1}^{L} \phi(j,i)q(i) \\ \vdots \\ \sum_{j=0}^{N-1} \phi(j,L) \sum_{i=1}^{L} \phi(j,i)q(i) \end{bmatrix} . \tag{3.51}$$

In a similar fashion to the standalone spline least squares estimation, we define the matrix

$$\Phi = \begin{bmatrix} \phi(1,1) & \phi(1,2) & \cdots & \phi(1,L) \\ \phi(2,1) & \phi(2,2) & \cdots & \phi(2,L) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(N,1) & \phi(N,2) & \cdots & \phi(N,L) \end{bmatrix} . \tag{3.52}$$

Taking into account the computation for $\phi(j,i)$, we observe that the m^{th} column of Φ is the convolution of the m^{th} column of Ω with the taps vector $\mathbf{w}$. Substituting the definition of Φ into equation (3.51), we have

$$\Phi_n^T \mathbf{y} = \Phi_n^T \Phi_n \mathbf{q}_n, \tag{3.53}$$

where Φ_n is the Φ matrix computed using the n^{th} iteration of filter taps $\mathbf{w}_n$. Solving this set of normal equations yields the least squares estimate

$$\mathbf{q}_n = \left(\Phi_n^T \Phi_n\right)^{-1} \Phi_n^T \mathbf{y}. \tag{3.54}$$

We perform this process for a predetermined number of iterations and examine the performance of the block-based off-line learning algorithm. Alternatively, it is possible to establish a convergence condition: for example, when the change in MSE of the training input signal between iterations moves below some threshold ϵ. Just as in the linear least squares case, we again require the input signal to be stationary and ergodic. In general, we expect this iterative process to converge to some minimum in MSE, but we do not rule out the possibility of convergence toward a local minimum instead of a global minimum.

Chapter 4

Results

We implement the NLMS adaptive filter, the spline Hammerstein system and the piecewise linear Hammerstein system as MATLAB objects. We examine the performance of these filter structures in an AEC simulation using both randomly and independently drawn Gaussian input and voice audio input, and we modify an existing MATLAB demonstration to obtain a suitable room impulse response [14]. Finally, we implement the least squares off-line block adaptation algorithm in MATLAB to obtain an optimal echo cancellation scheme against which we benchmark the performance of the on-line adaptation schemes. In each of these experiments, the length of the environment impulse response is a known parameter for the adaptive filters.

The magnitude of the error signal $e(n)$ provides a means to quantify the performance of each system. Since we limit our simulations to the single-talk scenario, we expect the error signal to have zero magnitude in the steady state. A common value used to measure the performance of an AEC system is the echo reduction loss enhancement (ERLE), which is expressed in dBs as [2]:

$$ERLE = 10 \log_{10} \frac{\mathbb{E}\left[d^2(n)\right]}{\mathbb{E}\left[e^2(n)\right]}. \tag{4.1}$$

We estimate the expected value of $x(n)$ as the windowed mean of the signal to the time n:

$$\mathbb{E}\left[x(n)\right] = \begin{cases} \frac{1}{n+1}\sum_{i=0}^{n} x(i), & n < W \\ \frac{1}{W}\sum_{i=n-W+1}^{n} x(i), & n \geq W \end{cases}, \tag{4.2}$$

where W is the window size. For complete cancellation of echo (i.e. $e(n) = 0$), the ERLE of the system increases without bound based on this definition of expected value.

4.1 Acoustic Impulse Response Model

All AEC simulations use a MATLAB example project on non-normalized LMS AEC techniques to generate an appropriate model for the room impulse response shown in figure 4.1 [14]. The impulse response is 4000 samples in length at a sampling rate

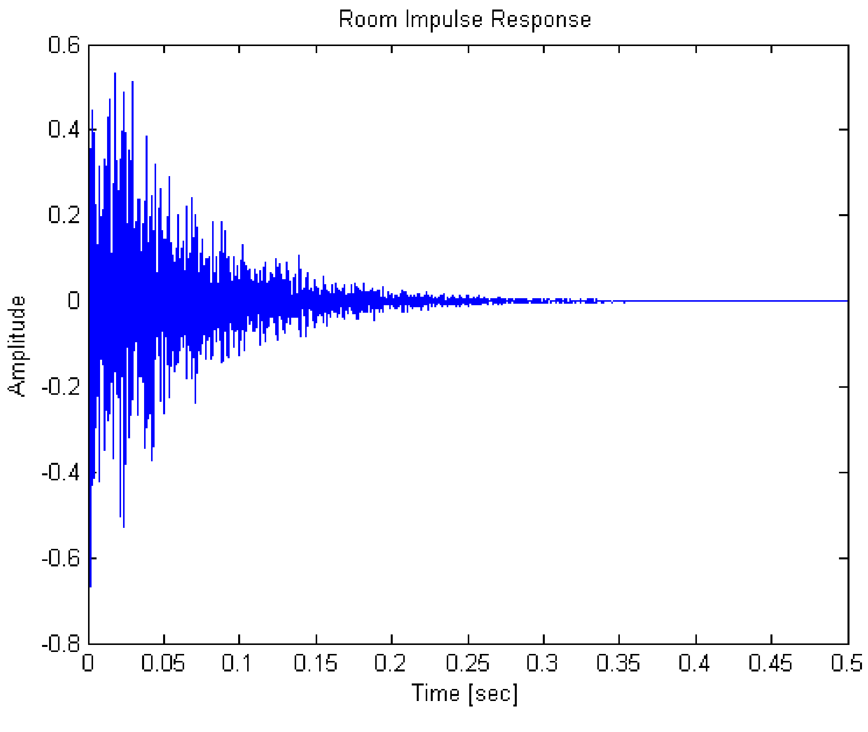


Figure 4.1: Model of Room Impulse Response for AEC Simulation

of 8 kHz, and the true length of impulse response is closer to 2000 samples. That is, samples of the impulse response after index 2000 are approximately zero-magnitude.

For purposes of examining the performance of the adaptive systems with linear filter components of different lengths, we truncate the model impulse response to the desired number of filter taps. Truncated signals of shorter length are lower in energy than those of longer length, and thus to minimize the effect of impulse response scaling, we ensure that each response is passive. Specifically, we normalize the DFT of each impulse response such that the modeled system does not amplify any frequency. This serves as an additional control in the simulation experiment so that the results more reliably indicate the effect of linear filter length on system performance.

4.2 Random Input Performance

The first performance test uses samples drawn from independently from a Gaussian distribution with mean $\bar{x} = 0$ and variance $\sigma^2 = 1$ as far end input $x(n)$. The linear portion of the echo path has impulse response equal to the frequency normalized version of the first M samples of the model impulse response. In this test, there is a sigmoid nonlinear prefix to the echo path path which has the form $f(x) = \frac{1-e^{-4x}}{1+e^{-4x}}$. Both the spline Hammerstein system and piecewise linear Hammerstein system have L control points which span the interval $[\bar{x} - \sigma^2, \bar{x} + \sigma^2]$. To avoid effects resulting from mismatched filter length, we assign each of the adaptive filter systems to have the a linear, FIR portion of with M filter taps. Although a filter with greater than M filter taps should ideally achieve the same performance as one with exactly M taps, having a larger number of taps adds additional variables to the learning processes of these systems, and thus the constraint on filter length is a control in this experiment.

For comparison purposes, we perform this test without nonlinear pre-distortion, and we report the results in figure 4.2. The NLMS system outperforms the cascade-of-filters systems in adapting to a purely linear unknown system as it has significantly lower magnitude error. The prefix block profiles for the purely linear case shown in figure 4.3 indicate that the cascade architectures have difficulty adapting in the absence of nonlinear distortion. We do not report ERLE values for this simulation because the error for the NLMS system reaches 0, resulting in an infinite ERLE value. Ideally, we expect the Hammerstein architecture systems to achieve the same level

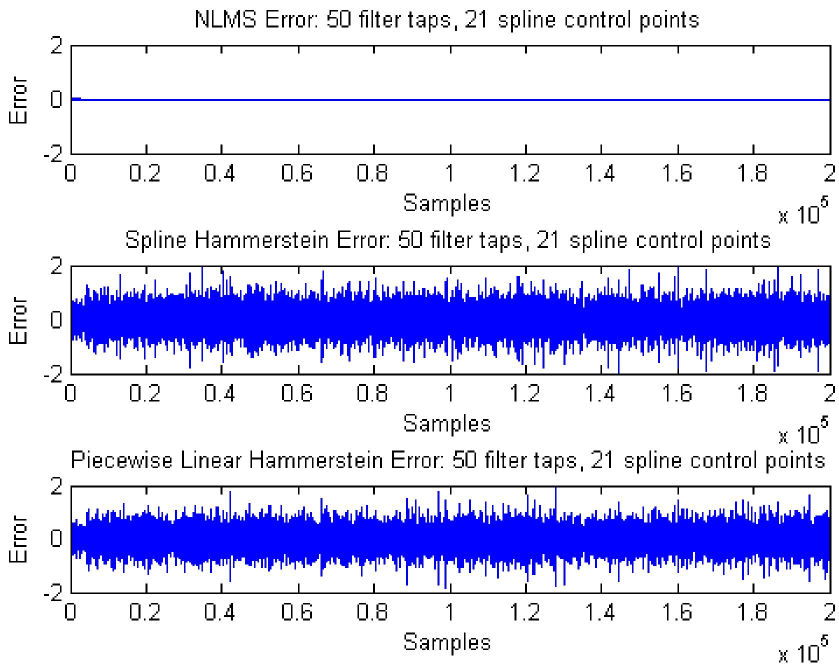


Figure 4.2: ERLE Measurements for Unknown Linear System

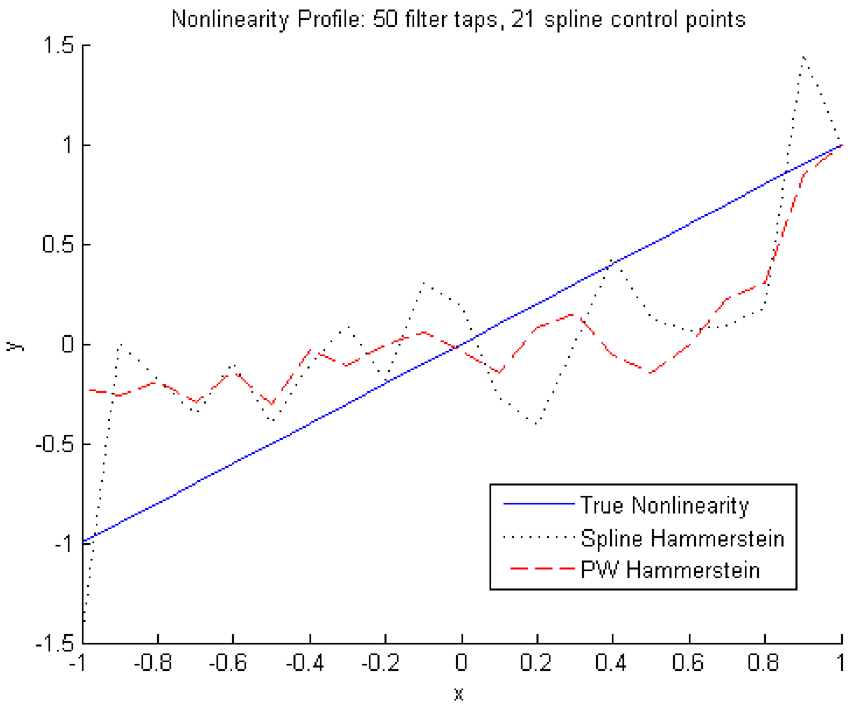


Figure 4.3: Nonlinearity Profiles in Absence of Nonlinearity

of performance as NLMS system. The concatenation of an adaptive prefix block increases the class of systems that the Hammerstein architecture can mimic. The optimal configuration of the cascade of filters in the case where there is no nonlinear distortion is an adaptive prefix block which retains an identity mapping (i.e. $f(x) = x$) and a linear, FIR filter which has the same taps as the converged NLMS filter. The poor performance of the cascade systems in the absence of nonlinear distortion could result from slow convergence of control points or the presence of local mimima in the expected error signal (with respect to filter parameters) which inhibit adaptation and convergence toward the optimal NLMS solution.

We begin with the performance test in the presence of nonlinear distortion with adaptive systems of $M = 50$ filter taps and $L = 21$ control points, and we report the results in figure 4.4. The calculation of ERLE uses a window size of $W = 50000$. The

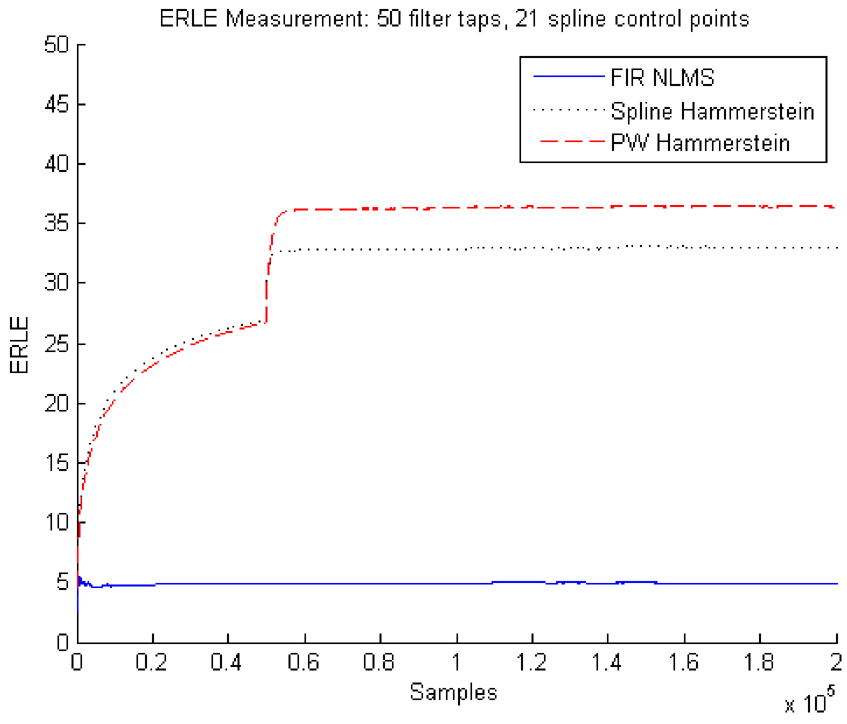


Figure 4.4: ERLE Measurements for $[M, L] = [50, 21]$

standard NLMS filter quickly reaches its peak performance and maintains a constant ERLE for the remaining samples. This indicates that the presence of the nonlinearity adversely affects the NLMS adaptive filter system, since the NLMS system has significantly lower ERLE than in the purely linear case. Conversely, with the sigmoid distortion of $x(n)$, the Hammerstein architecture systems have higher ERLE than the NLMS system and higher ERLE than the Hammerstein architecture systems in the purely linear case. The Hammerstein systems converge to peak performance in approximately the first 60000 samples. The piecewise appearance of the plot is an artifact of the windowed averaging procedure: the point at which the ERLE plot appears to have an undefined first derivative corresponds to the sample number at which the scheme for estimating expectation switches from a purely running average to a windowed average. The prefix profiles in this scenario indicate that the adaptation process yields a profile which is qualitatively more accurate than the profiles in the purely linear case. From figure 4.6, we see that NLMS mean absolute magnitude

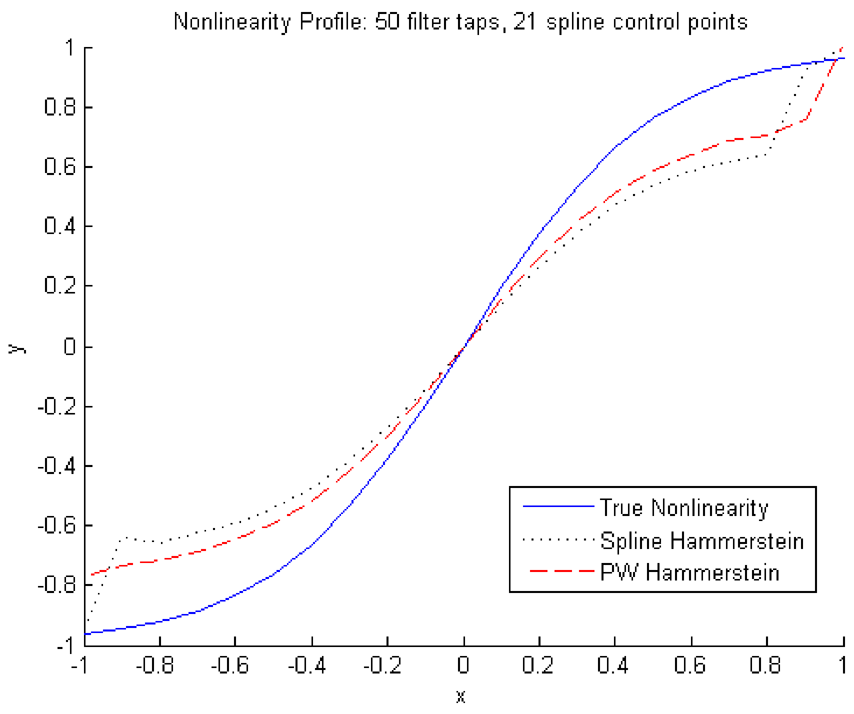


Figure 4.5: Nonlinearity Profiles for $[M, L] = [50, 21]$

.

of error seems to remain constant with respect to time (number of samples) and does not respond to the adaptation process. Unlike the purely linear case, the NLMS system cannot completely eliminate the estimation error $e(n)$ when there is nonlinear distortion. We observe that the ERLE for the piecewise linear function Hammer-

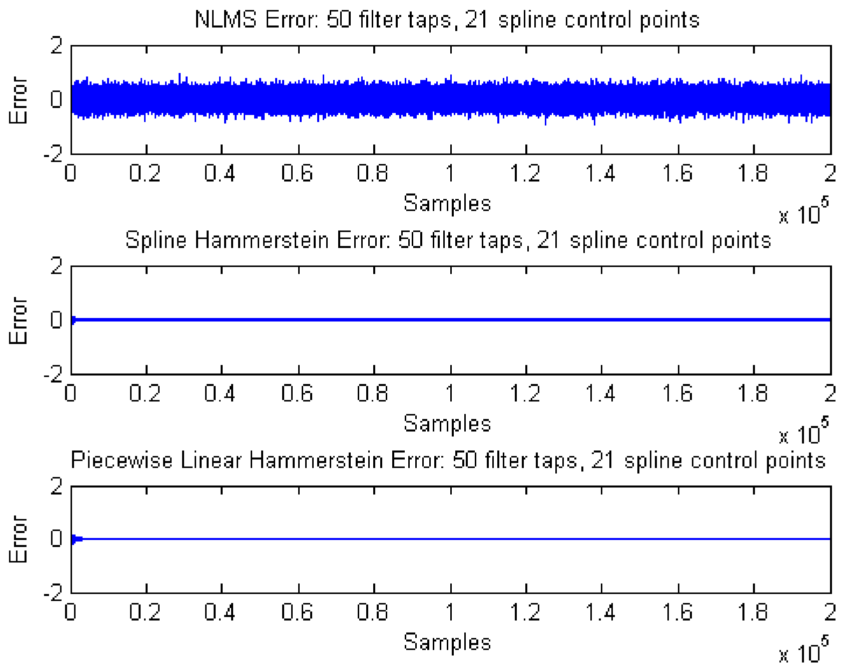


Figure 4.6: Error Signal $e(n)$ for $[M, L] = [50, 21]$

stein system is higher than the ERLE for the spline function Hammerstein system. Based on the profile of nonlinearities from figure 4.5, the piecewise linear function does not converge significantly faster than the spline function to the true nonlinear profile. Furthermore, by construction the spline function can more smoothly adapt to continuous nonlinearities.

One question of interest is the effect of filter length M on the performance of these filter architectures. Specifically, AEC systems on mobile phones must be able to adapt to impulse responses with lengths on the magnitude of tens to hundreds of milliseconds at a sampling rate around 8 kHz for phones [5], which implies that M must on the order of magnitude of thousands of samples. Unlike the filter length, there are no explicit constraints or requirements for the number or range of control points related to the physical acoustic environment. This is because we can choose the convention by which physical sound is converted into the far end input $x(n)$ simply by scaling to some known value. The only requirement on L is that there must be enough points to perform interpolation (at least four for spline, at least two for piecewise linear). Accordingly, we repeat the performance test on random input for $M = 100$ and $M = 1000$ filter taps while maintaining $L = 21$ control points.

At $M = 100$, the cascaded filter systems still outperform the NLMS in terms of ERLE and error magnitude. While the ERLE for the NLMS system remains nearly

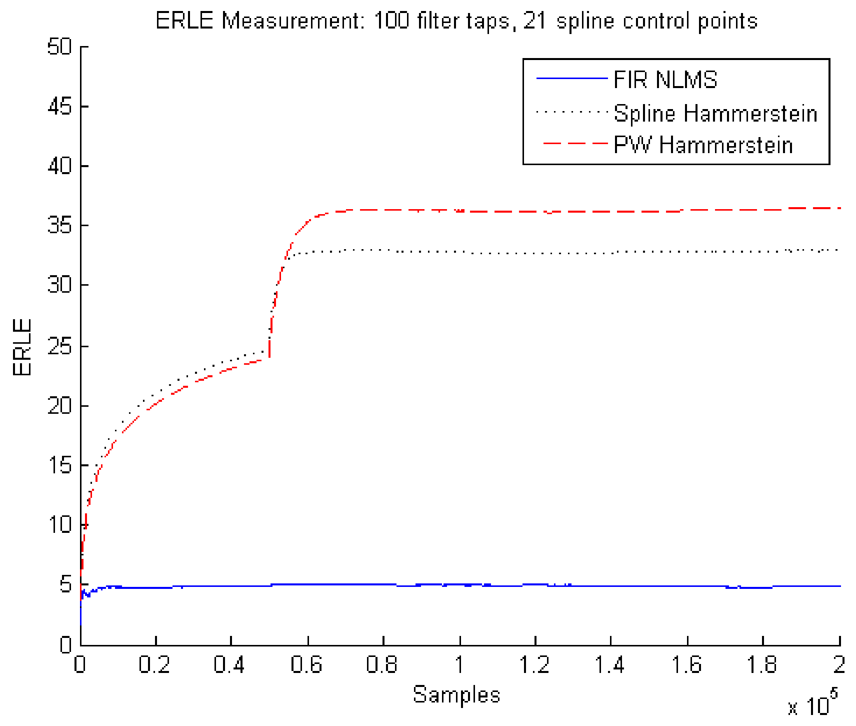


Figure 4.7: ERLE Measurment for $[M, L] = [100, 21]$

the same, the ERLE values for the spline function Hammerstein and piecewise linear function Hammerstein systems both converge to a smaller value. Again, the piecewise linear Hammerstein system noticeably outperforms the spline Hammerstein system. Although the steady state ERLE values are similar for $M = 100$ and $M = 50$ filter taps, figure 4.7 shows that the system with the longer linear filter length converges to its peak performance slightly more slowly than one with shorter length. Figures 4.8 and 4.6 also show that the Hammerstein systems with longer length take more input samples to converge to the steady state error output.

The $M = 1000$ filter taps systems continue the observed trend and demonstrate much slower convergence to the steady state. Figure 4.9 shows that neither the spline nor piecewise linear Hammerstein systems reach peak performance by the end of the

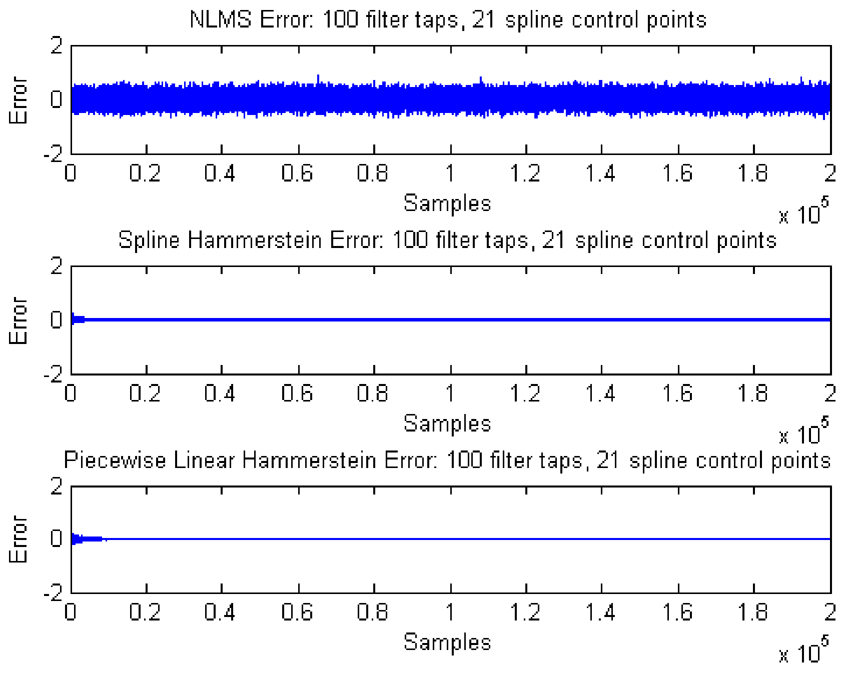


Figure 4.8: Error Signal $e(n)$ for $[M, L] = [100, 21]$

200000 sample input. The ERLE for the system with spline preprocessing appears to reach its peak performance more quickly than the system with piecewise linear preprocessing. On the other hand, we extrapolate the results from figure 4.9 to infer that the piecewise linear cascaded architecture reaches a higher steady state ERLE value. This remains consistent with the results observed for systems of lower linear filter length. Error measurements in figure 4.10 indicate that the spline Hammerstein system adapts more quickly to reduce high magnitude error but converges to a higher steady state error. Conversely, the piecewise linear Hammerstein system has higher magnitude error for a greater number of input samples but eventually converges to a lower steady state error. These observations confirm the inference regarding better peak performance for the piecewise linear architecture.

Figures 4.11 and 4.12 shows that there is not an observable, direct effect of linear filter length on the steady state configuration of the nonlinear preprocessing blocks. There are few qualitative differences between the nonlinear profile for systems with $M = 100$ and $M = 1000$ filter taps, and both profiles appear nearly identical to the

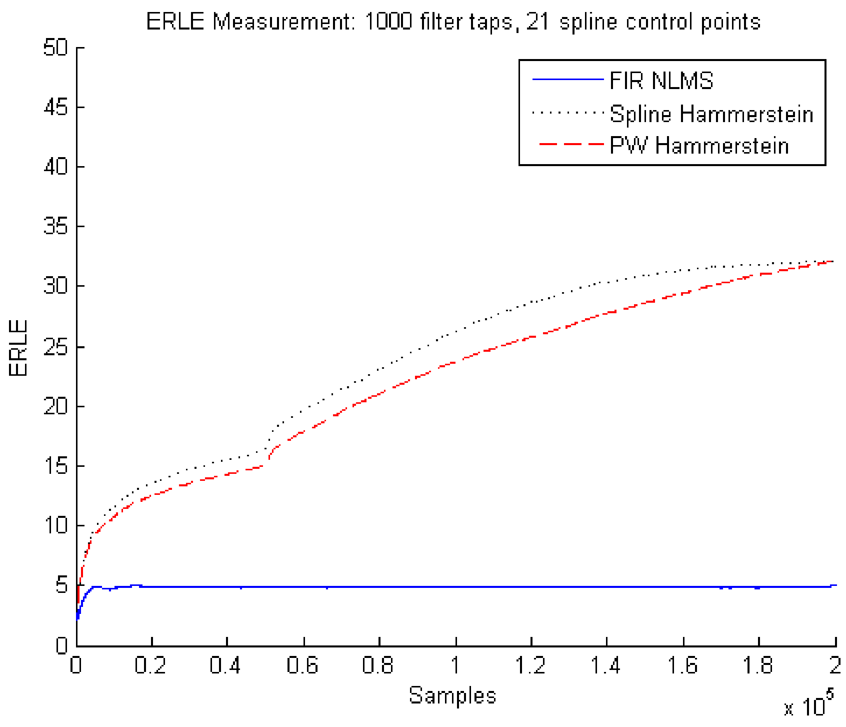


Figure 4.9: ERLE Measurement for $[M, L] = [1000, 21]$

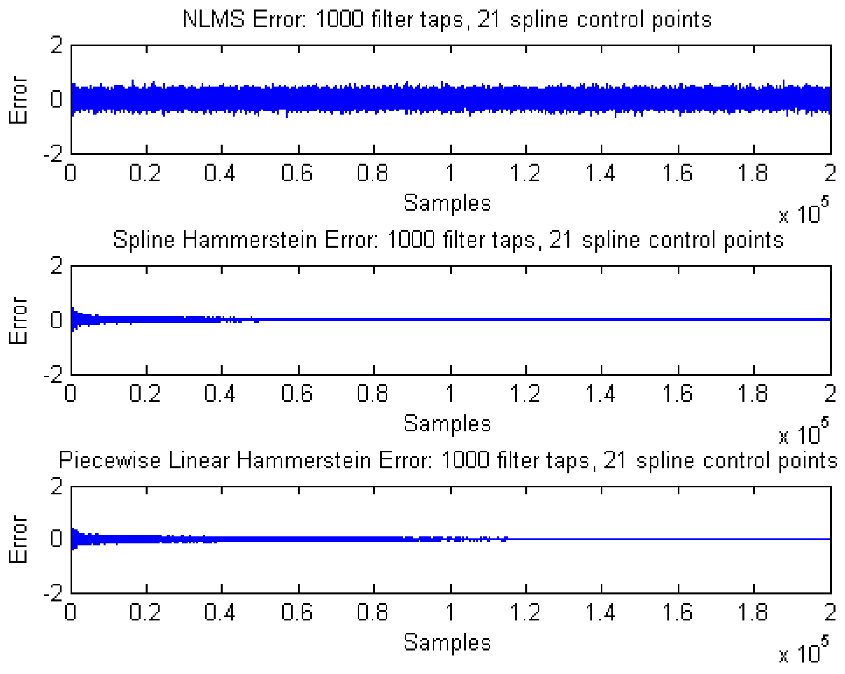


Figure 4.10: Error Signal $e(n)$ for $[M, L] = [1000, 21]$

nonlinearity profile for the system with $M = 50$ filter taps. Note that the nonlinearity

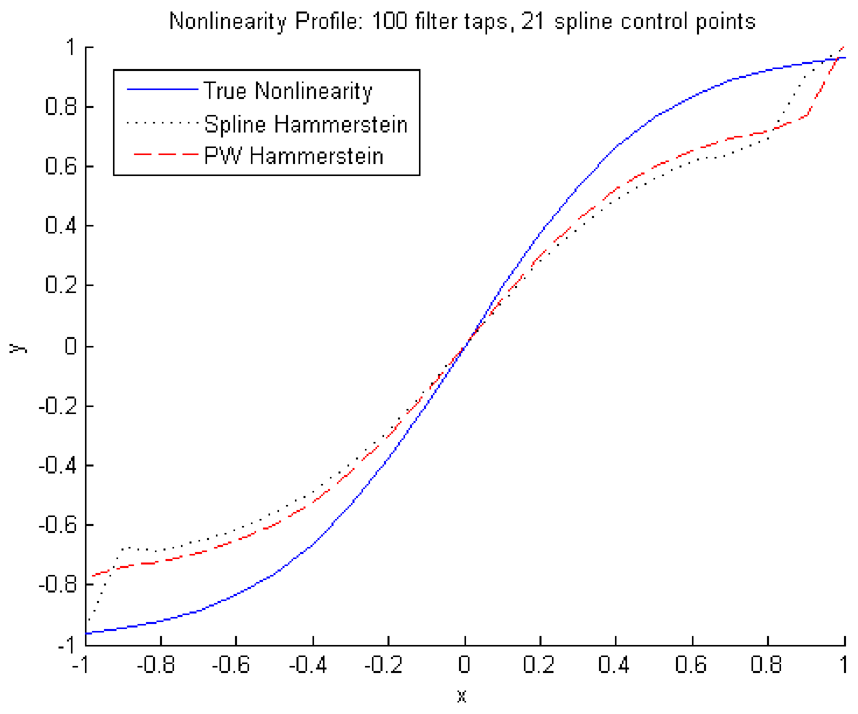


Figure 4.11: Nonlinearity Profile for $[M, L] = [100, 21]$

profiles do not necessarily indicate anything regarding the adaptation dynamics of the preprocessing blocks. Recall that for the $M = 50$ and $M = 100$ simulations, the systems reach steady state configuration before the final sample of input, and even in the $M = 1000$ simulation, the systems neared their final configuration. It is thus possible for the longer linear filter to hinder the adaptation of the nonlinear component, resulting in slower convergence. Alternatively, the slow convergence could also result directly from the longer linear filters. That is, even if the nonlinear block of Hammerstein architectures adapted with a rate that is not dependent on the length of the linear component, the slower convergence of the linear component itself slows down the convergence of the system as a whole.

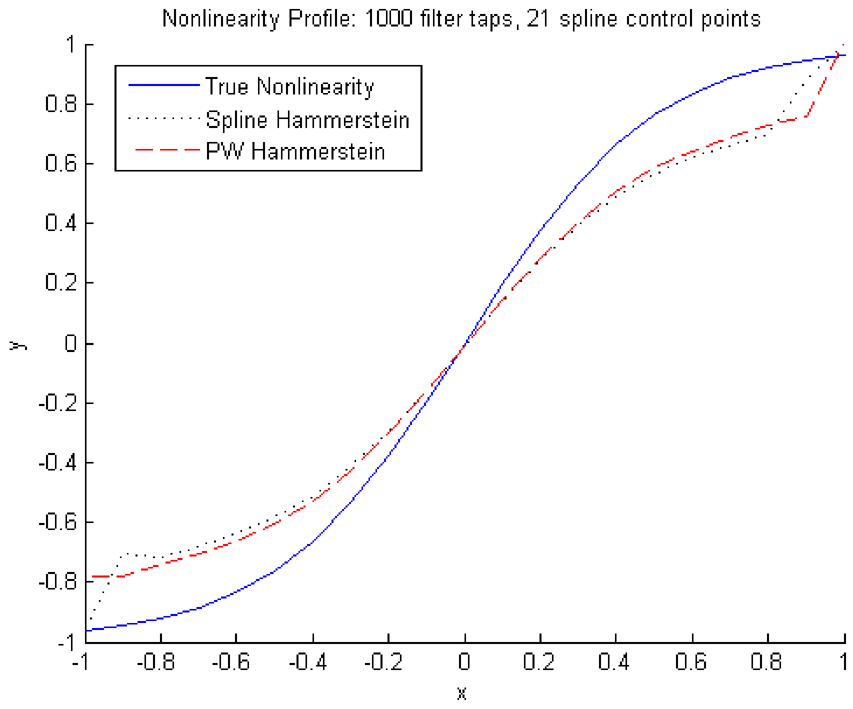


Figure 4.12: Nonlinearity Profile for $[M, L] = [1000, 21]$

4.3 Nonlinear Profile Rescaling

Consider the spline function (or piecewise-linear function) where the control points $\mathbf{q}$ have a scaling factor α. The output of such a function is

$$\begin{aligned} F(x, \alpha\mathbf{q}) &= \sum_{k=0}^{3} \alpha q_{i+k-1} c_k(\nu), \\ &= \alpha \sum_{k=0}^{3} q_{i+k-1} c_k(\nu), \\ &= \alpha F(x, \mathbf{q}), \end{aligned} \tag{4.3}$$

where $\nu \equiv \nu(x)$, and thus the output of the spline function is homogeneous with respect to control points. One property of linear systems is that the output is also homogeneous with respect to the impulse response, and consequently the Hammerstein system remains unaltered when we apply a scaling factor to one component and the inverse scaling factor to the other component. As the learning scheme adapts both components of the cascade system simultaneously, there is no guarantee that the resulting configurations scale exactly to the model configurations. In fact, the nonlinearity profiles for Gaussian input AEC simulation (figures 4.5, 4.11 and 4.12) appear to be mismatched from the model nonlinearity by some fixed constant. To provide further analysis on the performance of these cascade filters, we seek to compute this mismatched scaling factor in the adaptation process.

One method to choose the scaling factor is to minimize the squared error between the control points and the model nonlinearity. To minimize the effect of boundary cases, we consider only the squared error for non-boundary control points. Specifically, we choose a parameter α to minimize the error term

$$\xi = \sum_{i=2}^{L-1} \left(\alpha q_i - f(x_i)\right)^2, \tag{4.4}$$

where $\mathbf{Q}_i = \left[\begin{array}{cc} x_i & q_i \end{array} \right]^T$ is a control point as defined in the construction of the spline, and f is the model nonlinearity. The optimality condition requires the partial derivative of ξ with respect to α, computed as

$$\frac{\partial \xi}{\partial \alpha} = \sum_{i=2}^{L-1} 2q_i(\alpha q_i - f(x_i)), \tag{4.5}$$

to be equal to zero. This leads to the condition

$$\sum_{i=2}^{L-1} q_i f(x_i) = \sum_{i=2}^{L-1} \alpha q_i^2. \tag{4.6}$$

Defining the vector $\mathbf{f} = \left[\begin{array}{cccc} f(x_1) & f(x_2) & \cdots & f(x_L) \end{array} \right]$ and denoting $\tilde{\mathbf{f}}$ and $\tilde{\mathbf{q}}$ to be the appropriately truncated representations of $\mathbf{f}$ and $\mathbf{q}$ respectively, we solve for the optimal choice of α:

$$\alpha = \frac{\tilde{\mathbf{q}}\tilde{\mathbf{f}}^T}{\tilde{\mathbf{q}}\tilde{\mathbf{q}}^T}. \tag{4.7}$$

This rescaling technique yields a nonlinear profile which more accurately represents the true nonlinear configuration for random input AEC simulations of all considered filter lengths. After rescaling, we confirm that there is little qualitative

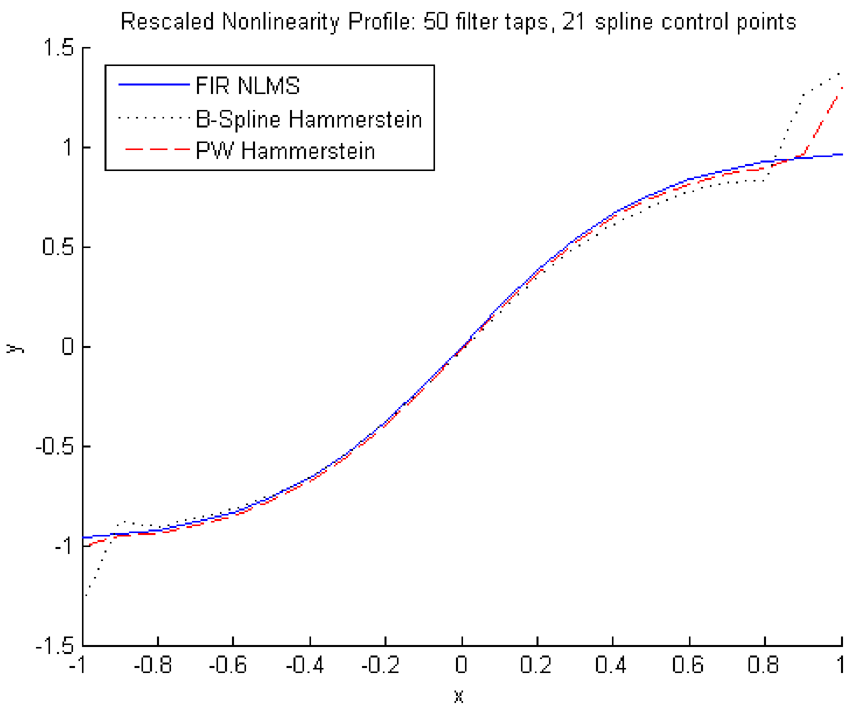


Figure 4.13: Rescaled Nonlinearity Profile for $[M, L] = [50, 21]$

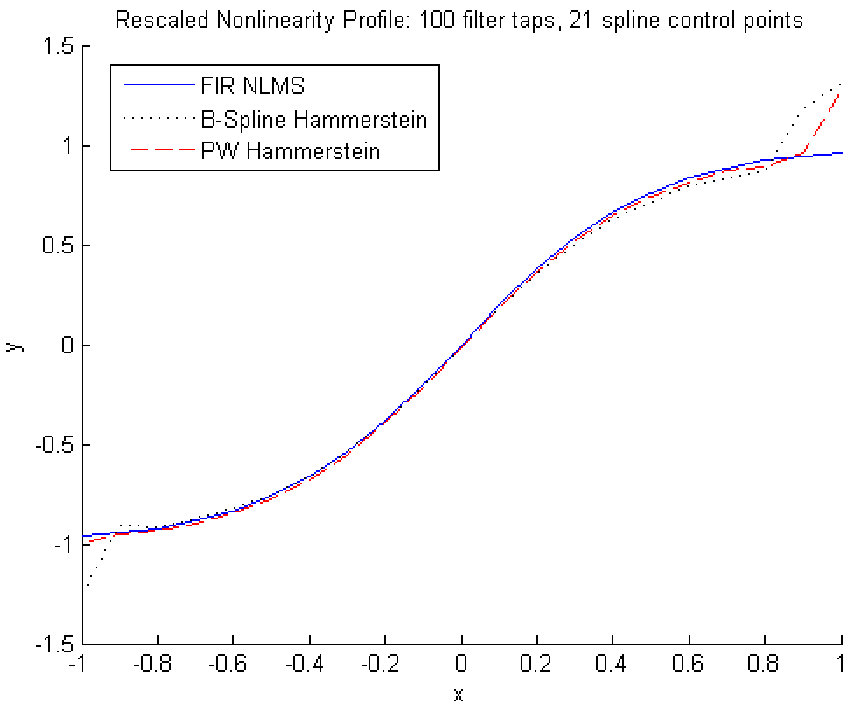


Figure 4.14: Rescaled Nonlinearity Profile for $[M, L] = [100, 21]$

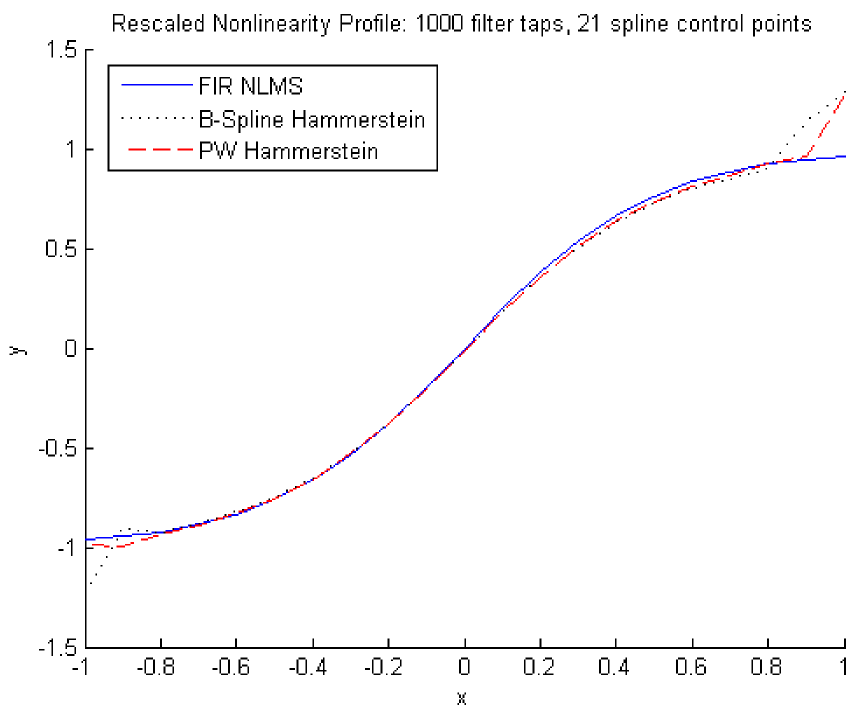


Figure 4.15: Rescaled Nonlinearity Profile for $[M, L] = [1000, 21]$

difference in the final configuration of the nonlinear component for Hammerstein systems of different linear filter length. We also observe that there is little qualitative difference in the location internal control points between the adaptation result of the spline function cascade and the adaptation result of the piecewise linear function cascade. In general, the rescaled profiles of Hammerstein systems show that these cascade architectures in fact do adapt well to this memoryless nonlinearity, a fact which we cannot as easily infer from the original nonlinearity profiles.

An alternative method to compute the scaling of the nonlinear profile is to minimize the squared error between estimated linear impulse response and the true echo path impulse response. Again, because of the system's homogeneity, it is irrelevant which component of Hammerstein architecture is affected by the scaling factor. In the profile squared error minimization scheme, we compute the "optimal" scaling factor directly and assume the inverse scaling factor on the linear impulse response. The alternative proposed here is to the nonlinear scaling factor by computing the "optimal" inverse scaling factor for the linear component. Given the true linear impulse response $\mathbf{w}$ and the estimate $\hat{\mathbf{w}}$, the nonlinear profile rescaling factor is

$$\alpha' = \frac{\hat{\mathbf{w}}^T\hat{\mathbf{w}}}{\hat{\mathbf{w}}^T\mathbf{w}}. \tag{4.8}$$

The derivation of equation (4.8) is exactly analogous to the derivation of equation (4.7).

A comparison between α and α' provides some intuition on the validity of the rescaling process. We report the results in table 4.1. The rescaling factors are close

M	Spline α	Spline α'	Piecewise Linear α	Piecewise Linear α'
50	1.3507	1.4149	1.2850	1.2911
100	1.3172	1.3658	1.2737	1.2775
1000	1.2897	1.3322	1.2656	1.2743

Table 4.1: Comparison of Rescaling Factors for On-Line Adaptation

in magnitude for the piecewise linear function cascaded architecture, which suggests that the scheme of minimizing squared error in the nonlinear profile indeed yields an appropriate scaling factor. On the other hand, the spline function cascaded systems have noticeably higher magnitude in α' than in α. This observation indicates that appropriate scaling factor for such systems may not be the one which minimizes the squared error in the nonlinear profile. One possible explanation for this phenomenon is that the boundary control points for the adaptive B-spline exhibit behavior that is less smooth than the behavior of boundary control points for adaptive piecewise linear functions. The spline boundary control points take on values which are more extreme, which results in a calculation of α which is too low in magnitude.

4.4 Simulated Echo Cancellation Performance

To perform a more realistic AEC test of the cascade adaptive filters, we use a segment of human speech, with range normalized to $[-1, 1]$, as the input signal. The signal has a distribution given in figure 4.16. Although the maximum and minimum values of the input signal are 1 and -1 respectively, nearly all the mass of the distribution lies on the interval $[-.4, .4]$. Accordingly, we define the cascade systems with nonlinear

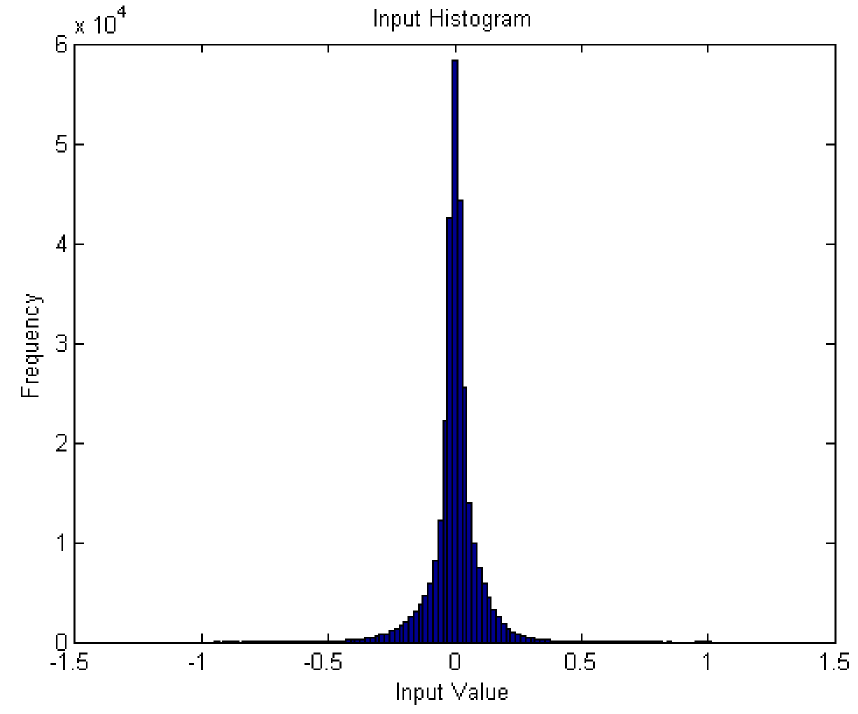


Figure 4.16: Histogram/Distribution Estimate of Speech Input

components whose control points span the smaller, inner interval while keeping the number of total control points constant as before. This allows greater precision in nonlinear adaptation. Both the B-spline function and piecewise linear function are capable of handling input which is outside the span of its control points. The distribution of input also necessitates a modified choice of model nonlinearity to perform an appropriate simulation. The choice of nonlinear distortion for the random input simulations, $f(x) = \frac{1-e^{-4x}}{1+e^{-4x}}$, is approximately linear on the interval $[-.4, .4]$. In order to introduce an adequate degree of nonlinearity into the simulation, we use a modified version of this function, $f(x) = \frac{1-e^{-12x}}{1+e^{-12x}}$. We specify all systems to have $M = 4000$ filter taps and $L = 21$ control points.

The ERLE measurements for the AEC simulation shown in figure 4.17 indicate the cascade-of-filters architecture outperforms the standalone NLMS system in this test. This is consistent with the results from random input simulation, as the cascade components successfully adapt to the nonlinear distortion present in the echo path. The Hammerstein systems initially converge more slowly than the linear adaptive

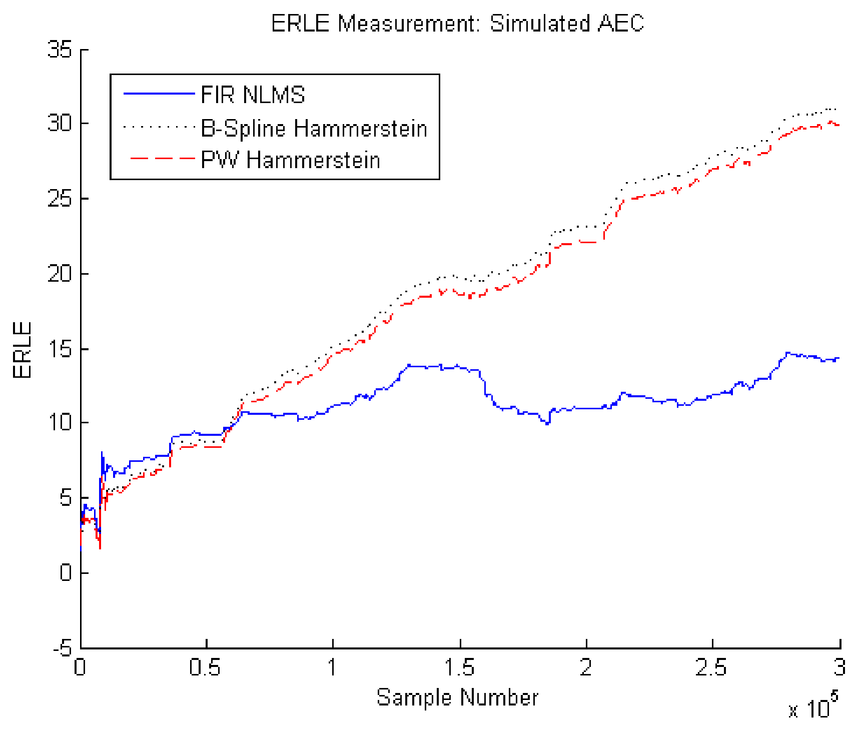


Figure 4.17: ERLE Measurements for AEC Simulation

filter, as the standalone NLMS system outperforms the nonlinear cascade systems in ERLE measurement during early input samples. The error signal measurements from the AEC simulation suggest that the spline cascade system converges to a lower magnitude error than the piecewise linear cascade system. Both the spline cascade system and piecewise linear cascade system nearly completely cancel the echo, and by the final sample of input, the Hammerstein systems continue to adapt to improve performance. On the other hand, the standalone NLMS AEC system reaches peak performance before the final sample of input and cannot completely cancel the echo. Based on figures 4.17 and 4.18, the cascade systems outperform the linear system in the presence of memoryless, nonlinear distortion.

The nonlinearity profiles for the Hammerstein systems also suggest that these filter architectures successfully adapt to memoryless nonlinear distortions. The rescaled profiles (using the nonlinearity error minimization scheme) give further evidence for these systems' adaptive capabilities. With the proper choice of the scaling factor α, both the adaptive B-spline and piecewise linear function form the model nonlinear

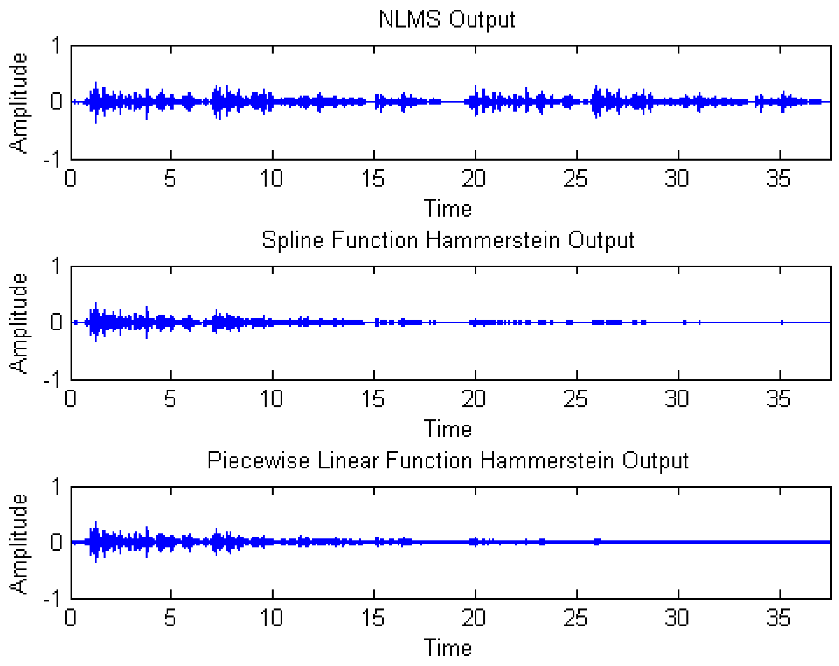


Figure 4.18: Error Signal/System Output $e(n)$ for AEC Simulation

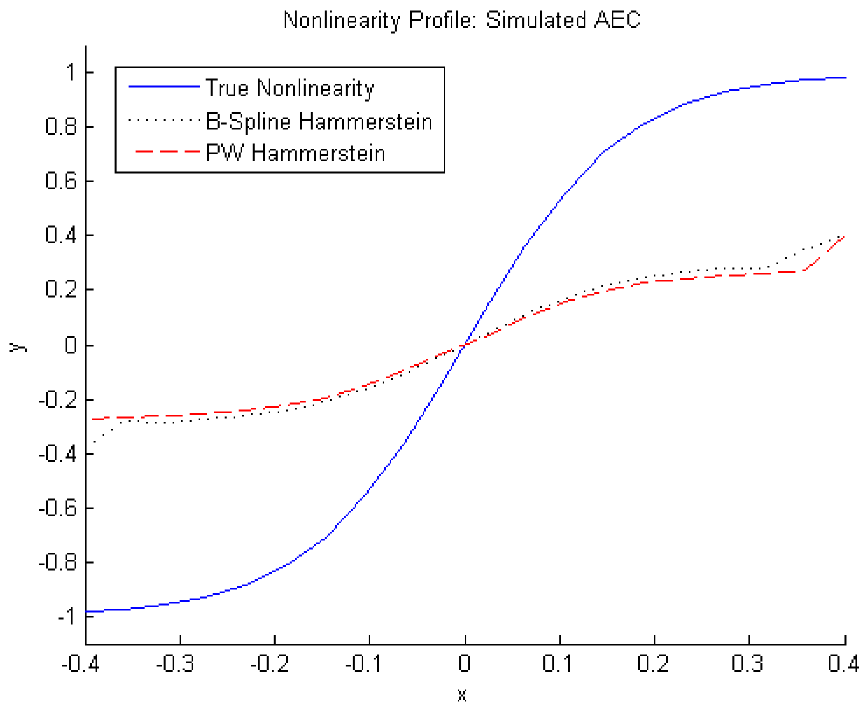


Figure 4.19: Nonlinearity Profile for AEC Simulation

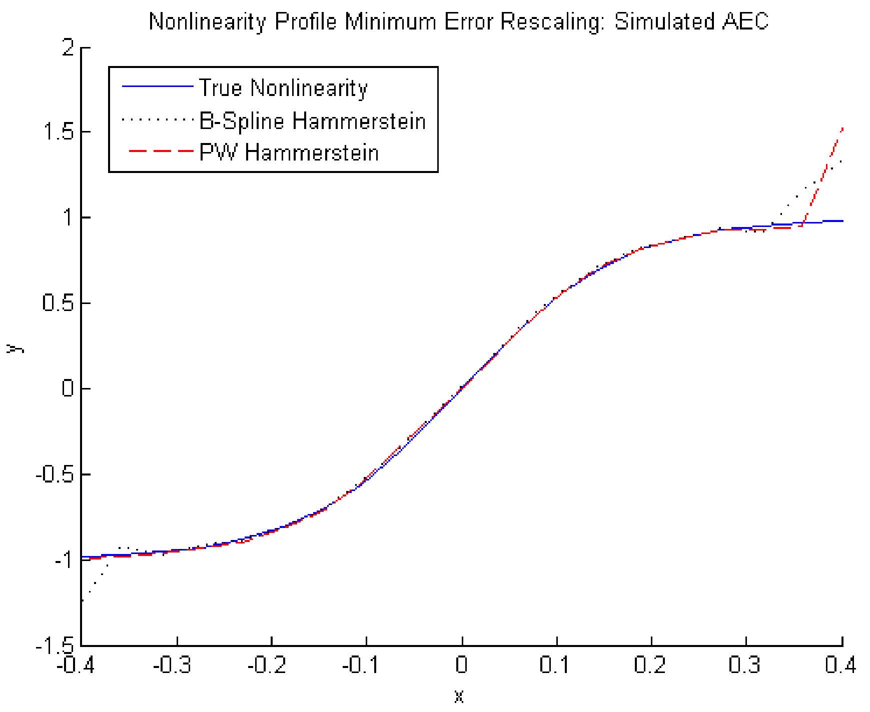


Figure 4.20: Rescaled Nonlinearity Profile (Nonlinearity Error Minimization) for AEC Simulation

distortion nearly perfectly for the non-boundary control points. To confirm the validity of such a rescaling scheme, we also compute the scaling factor α' by minimizing the squared error between the adaptive linear impulse responses and the true echo path impulse response.

Figures 4.20 and 4.21 indicate that $\alpha' > \alpha$ and that the profile error minimization rescaling scheme may not be valid in this simulation. Recall that results from random input simulation show that the scaling factor computed by minimizing the squared error between impulse responses is consistently higher than the scaling factor computed using the nonlinearity error minimization scheme for the B-spline, but these two parameters are approximately equal for the piecewise linear function. For voice audio input AEC simulation, the scaling factor computed using error minimization for impulse responses is higher in magnitude for both choices of adaptive nonlinear filters.

For the most part however, the results from this AEC simulation are consistent

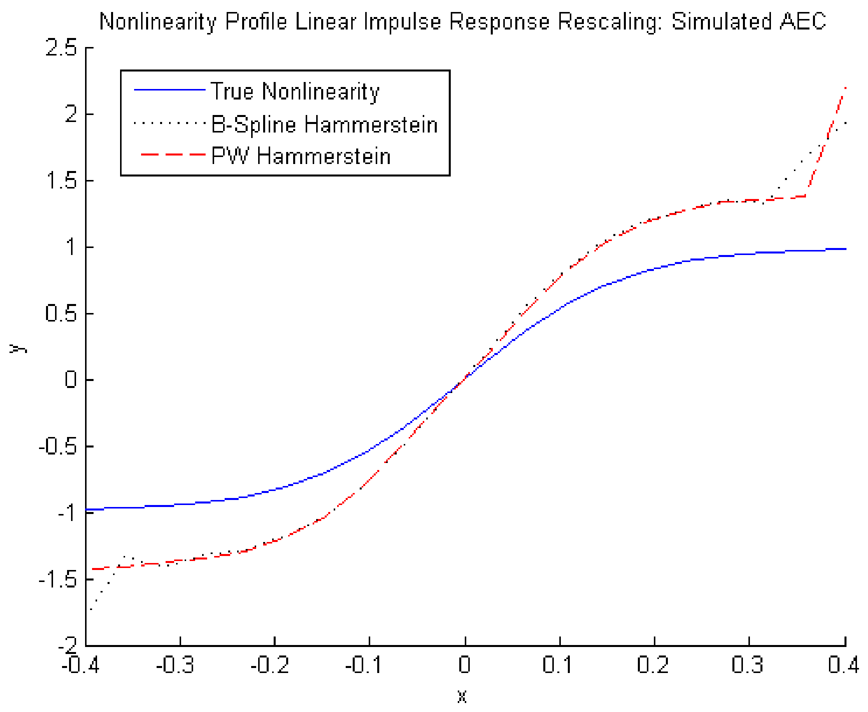


Figure 4.21: Rescaled Nonlinearity Profile (Impulse Response Error Minimization) for AEC Simulation

with the results from the random input simulation and reaffirm the performance capabilities of these Hammerstein adaptive systems. In the presence of nonlinear predistortion, the systems with adaptive nonlinear preprocessing capabilities consistently outperform the standard NLMS linear adaptive system. We know from properties of the NLMS algorithm that the performance of the linear components of Hammerstein systems does not directly depend on the input statistics. The adaptation of the nonlinear filters however do depend on signal statistics in the sense that the range of the input affects their performance. With proper selection of the x coordinates of the control points, the normalized adaptation algorithm for the nonlinear components performs independently from other statistical properties of the input. Thus we expect to achieve the observed results of consistency in performance between the random input and vocal audio input AEC simulations.

4.5 Iterative Least Squares Performance

The method of least squares estimation provides an additional means to benchmark the performance of generalized the adaptive filter systems. Using the iterative least squares approach for Hammerstein systems, we estimate the optimal ERLE measure and provide the convergence characteristics of the iterative algorithm. We first perform the least squares fit using random input samples drawn independently from a Gaussian distribution with mean $\bar{x} = 0$ and variance $\sigma^2 = 1$. The input vector $\mathbf{x}$ consists of 2000 such samples, and the output vector $\mathbf{y}$ is the echoed version of $\mathbf{x}$ using an appropriately sampled and truncated impulse response as the echo path with nonlinear pre-distortion $f(x) = \frac{1-e^{-4x}}{1+e^{-4x}}$. We perform the least squares estimation for systems with $L = 21$ nonlinear function control points on the interval $[\bar{x} - \sigma^2, \bar{x} + \sigma^2]$ and varying linear filter lengths. These conditions are identical to those used for the random input AEC simulation.

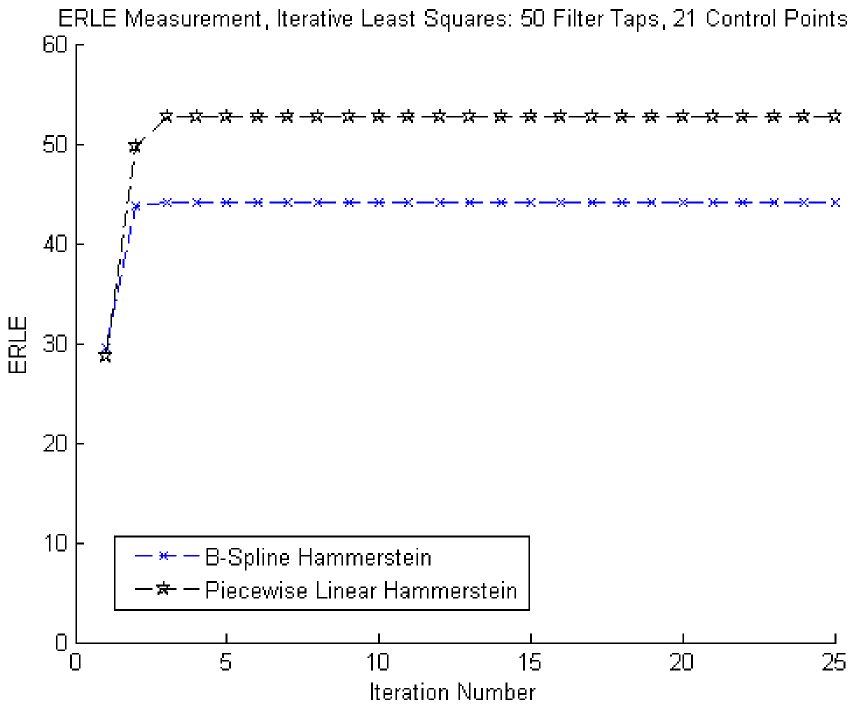


Figure 4.22: ERLE Measurement, Iterative Least Squares, for $[M, L] = [50, 21]$

Figure 4.22 shows that the iterative least squares scheme quickly converges to an optimal ERLE value for the systems with $M = 50$ filter taps. Comparing against

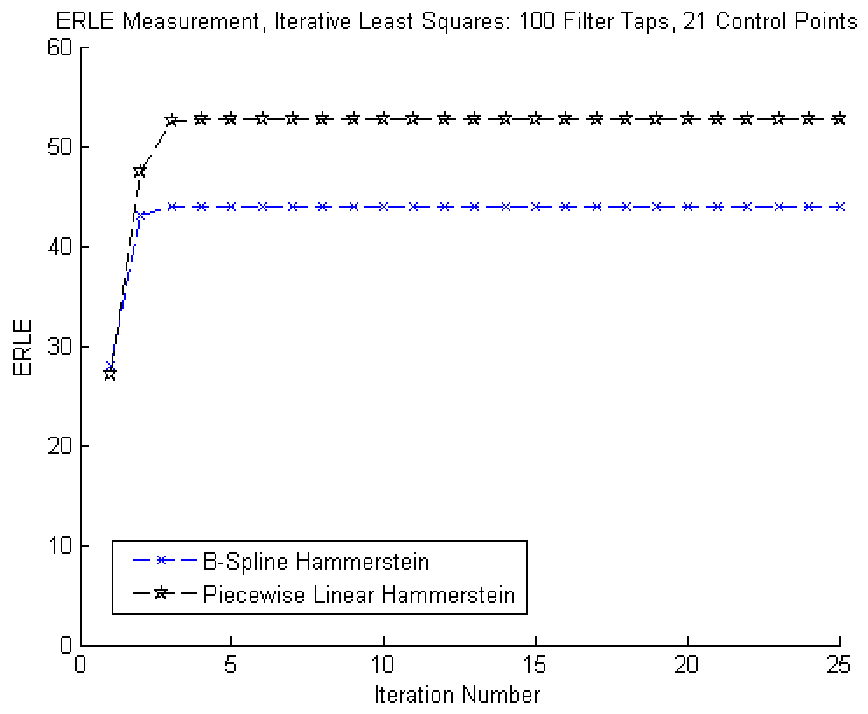


Figure 4.23: ERLE Measurement, Iterative Least Squares, for $[M, L] = [100, 21]$

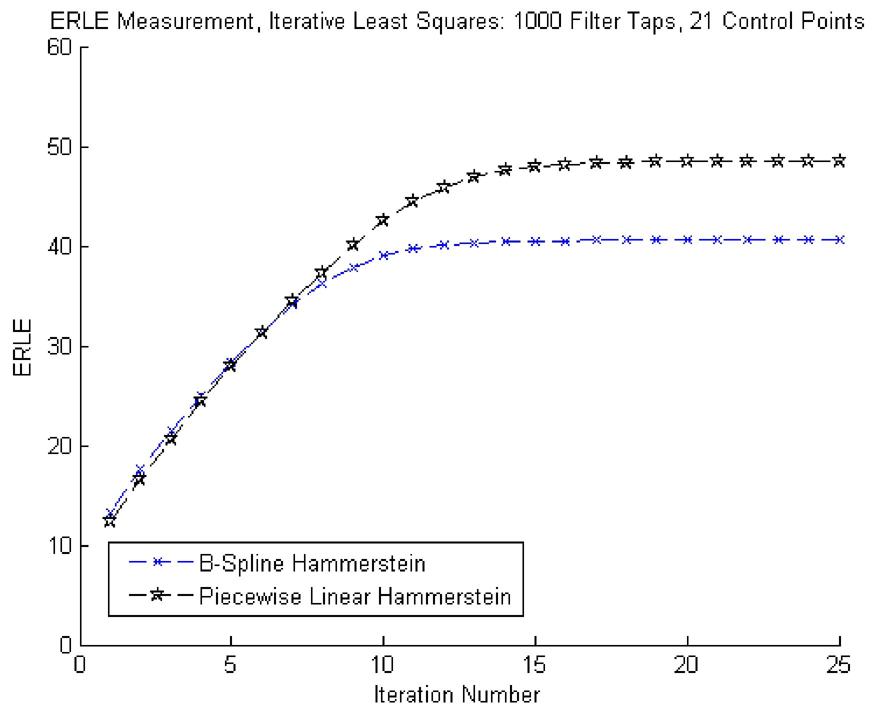


Figure 4.24: ERLE Measurement, Iterative Least Squares, for $[M, L] = [1000, 21]$

the results of on-line adaptation, we note that the least squares estimate achieves an ERLE measurement which is approximately 10 dBs higher for the spline cascade system and approximately 15 dBs higher for the piecewise linear cascade system. We also observe similar results for systems with $M = 100$ and $M = 1000$ filter taps. As the number of filter taps increase, the iterative least squares scheme converges more slowly and achieves a lower steady state ERLE. One possible intuitive explanation for the decrease in ERLE with increasing filter lengths is that the nonlinear components cannot exactly adapt to the given distortion. As we increase the linear filter length, the mismatch in the nonlinear component compounds over an increasing number of samples which adversely affects the overall performance of the system as a whole. In general, the ERLE measurements from the iterative least squares estimation shows that the on-line adaptation schemes for random input AEC converge to a suboptimal configuration.

The rescaled nonlinearity profiles for the Hammerstein systems suggest that there is no significant effect from the linear component filter length on the optimal adaptation of the nonlinear components in the steady state. We rescale all provided profiles using the nonlinearity error minimization scheme. Following scaling modification, profiles show that in the optimal configuration, the internal control points match the model nonlinearity nearly perfectly for all three tested systems. For cascade architectures with the piecewise linear function as the preprocessing component, the boundary control points match the model nonlinearity as well. On the other hand, boundary control points for spline processing blocks do not conform exactly to the true shape of distortion.

Once again, to validate the rescaling scheme for the nonlinearity profiles, we also compute the scaling factor using the impulse response error minimization scheme. The

M	Spline α	Spline α'	Piecewise Linear α	Piecewise Linear α'
50	1.2428	1.2768	1.2333	1.2365
100	1.2374	1.2741	1.2351	1.2382
1000	1.2584	1.2930	1.2551	1.2582

Table 4.2: Comparison of Rescaling Factors for Iterative Least Squares Estimation

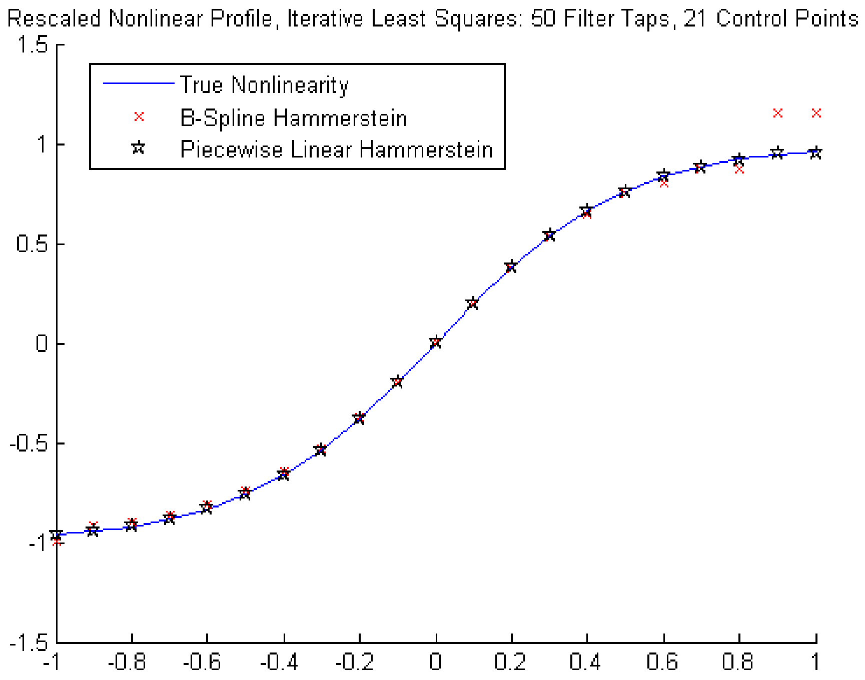


Figure 4.25: Rescaled Nonlinearity Profile, Iterative Least Squares, for $[M, L] = [50, 21]$

results provided in table 4.2 indicate that the scaling factor computed using impulse response error minimization is consistently higher in magnitude than the scaling factor computed using nonlinearity error minimization for B-spline cascaded systems, while the two scaling factors are approximately equal in magnitude for piecewise linear function cascaded systems. These observations are in congruence with the rescaling results for the on-line adaptation simulations. As in the on-line case, one possible explanation for the disparity in scaling factors for B-spline Hammerstein in iterative least squares estimation is the behavior of the boundary control points.

The off-line estimation approach requires far fewer samples of input and output to operate than the on-line adaptation schemes. There are however, certain restrictions on the minimum size of input. In general, to ensure convergence to the optimal level of performance, the input and output signals should be representative of source statistics. This intuitively means that the off-line fitting performs better when there are more samples of input. Another necessary, but by no means sufficient, condition is

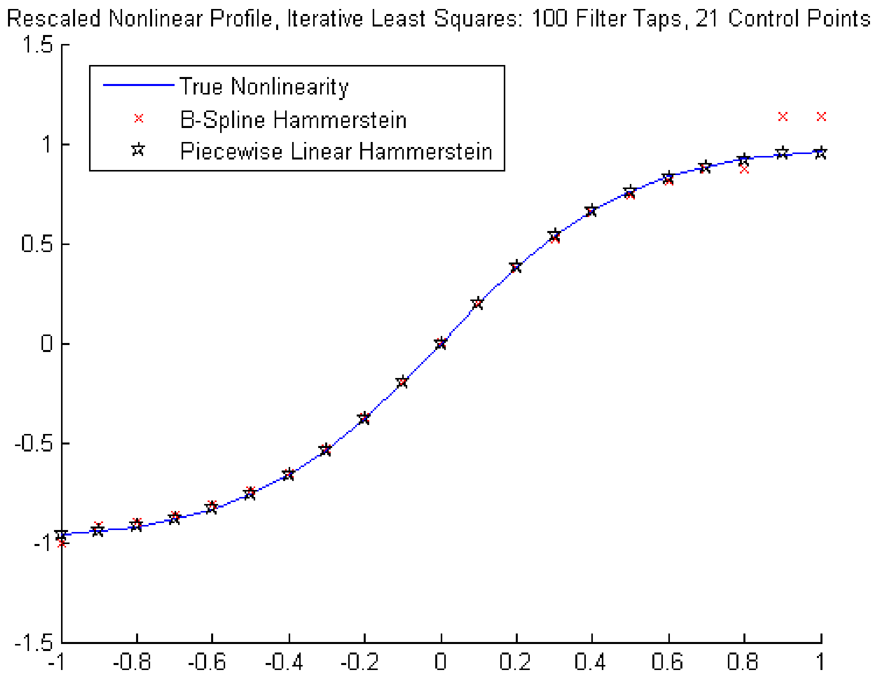


Figure 4.26: Rescaled Nonlinearity Profile, Iterative Least Squares, for $[M, L] = [100, 21]$

that the number of samples in the input signal must exceed the number of filter taps in the linear impulse response. The method of least squares fitting can estimate as many filter taps as the number of input samples. Using larger size input signals requires more memory and computation time and presents the trade-off between performance and complexity.

Iterative least squares estimation for the voice audio input AEC simulation requires signals of longer length to achieve the desired optimal level of performance. Unlike the random input simulation, a vocal source is not stationary. As an extreme example of this, consider the case when the speaker produces a long segment of silence followed by a segment of speech: the source distribution certainly changes between the silent segment and the spoken segment. Moreover, the adaptation of the nonlinear components depends on the range of amplitudes in the training input data. Consequently, we select the input $\mathbf{x}$ and output $\mathbf{y}$ to be vectors with 12000 samples which encompass a range of signal amplitudes representative of the entirety of the

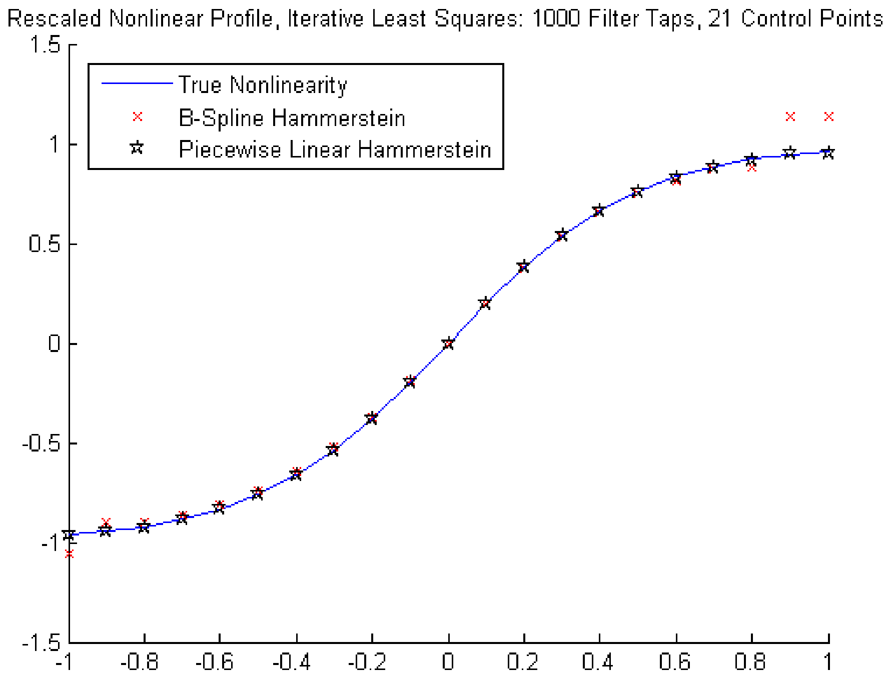


Figure 4.27: Rescaled Nonlinearity Profile, Iterative Least Squares, for $[M, L] = [1000, 21]$

input and output signals. We modify experimental conditions for off-line estimation such that they are identical to the conditions in the on-line voice audio input AEC simulation. Specifically, we use a Hammerstein cascade of a nonlinear function with $L = 21$ control points spanning the interval $[-.4, .4]$ and a linear filter with $M = 2000$ filter taps. In addition, we use the function $f(x) = \frac{1-e^{-12x}}{1+e^{-12x}}$ as the model nonlinearity.

Figure 4.28 shows the convergence of ERLE values with increasing number of iterations. For the voice input AEC simulation, the iterative least squares estimation scheme for Hammerstein systems with $M = 2000$ filter taps converges in fewer iterations than the least squares estimation of Hammerstein systems with $M = 1000$ filter taps using Gaussian input samples. Recall that least squares results from Gaussian input simulations indicate that systems with more filter taps require a greater number of iterations to converge. One possible explanation for observing faster convergence for the system of longer length is that the least squares computation for voice input uses training signals with a greater number of samples. The larger training set could

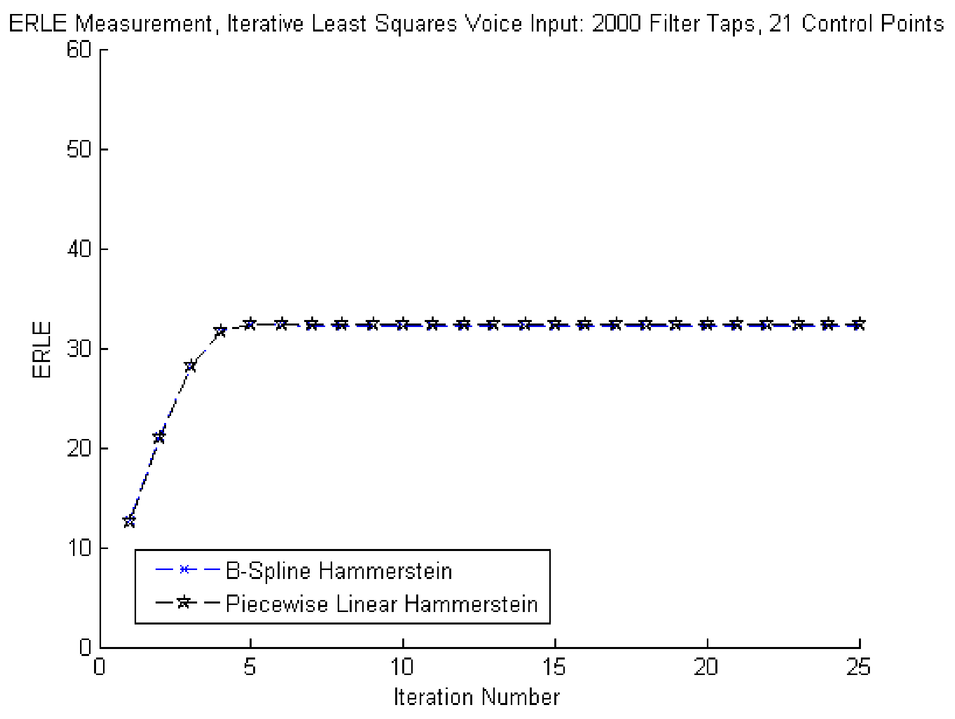


Figure 4.28: ERLE Measurements, Iterative Least Squares, for AEC Simulation

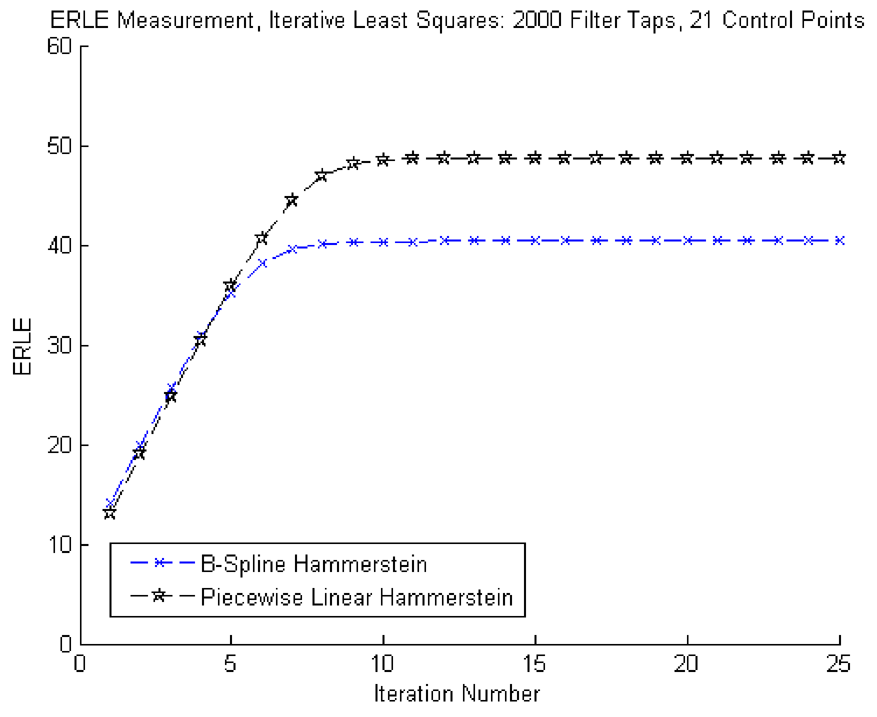


Figure 4.29: ERLE Measurements, Iterative Least Squares, for Gaussian Input Simulation for $[M, L] = [2000, 21]$

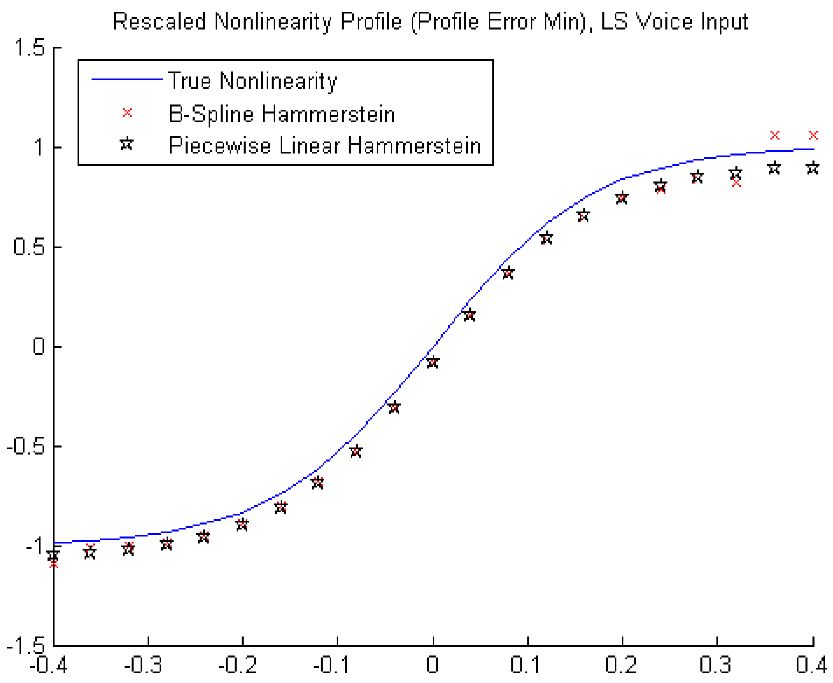


Figure 4.30: Rescaled Nonlinearity Profile (Nonlinearity Error Minimization), Iterative Least Squares, for AEC Simulation

result in quicker iterative adaptation and overcome the consequences of the longer linear filter length.

The optimal ERLE is approximately the same for the B-spline Hammerstein and piecewise linear function Hammerstein systems. This observation is not consistent with the results from the random input simulation, which show that the piecewise linear function cascade outperforms the B-spline cascade. Additionally, the least squares estimate of the Hammerstein systems using voice audio input converges to a lower ERLE value than the least squares estimate using random input. For more direct comparison, we provide the ERLE plot for the simulation using random input for Hammerstein systems with $M = 2000$ filter taps and $L = 21$ control points.

The rescaled nonlinearity profiles using the nonlinearity error minimization scheme from the voice input AEC simulation show that in the “optimal” least squares configuration both the spline function and piecewise linear function have control points which are slightly mismatched from the true nonlinearity. This mismatch could account for

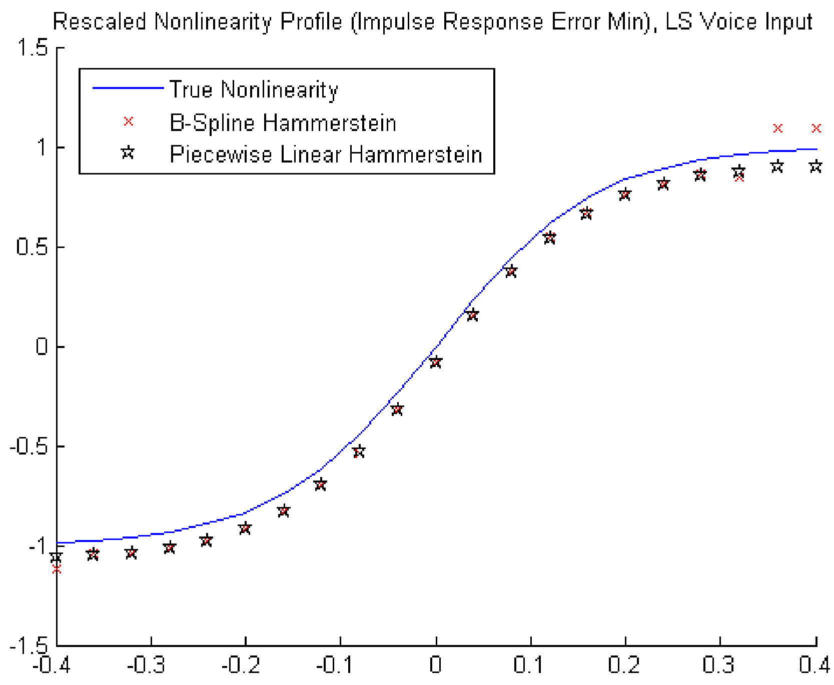


Figure 4.31: Rescaled Nonlinearity Profile (Impulse Response Error Minimization, Iterative Least Squares, for AEC Simulation

the lower optimal ERLE value. One additional difference between the least squares results for voice audio simulation and those for random input simulation is that in the voice audio AEC simulation, the rescaling factors α (nonlinearity error minimization) and α' (impulse response error minimization) are approximately equal in magnitude for both the spline cascade and piecewise linear function cascade systems.

The results from least squares estimation for voice input AEC simulation suggest that the optimal level of performance for Hammerstein systems is significantly lower than in the random input simulation. Comparing these results against the results from on-line learning simulation, we observe that the on-line adaptive filters reach an ERLE value that is near the "optimal" value obtained from the iterative least squares scheme. It is possible that the value obtained in off-line adaptation is not a true "optimal" measurement of performance. Intuitively, we treat voice as a nonstationary source, and in order to perform the least squares estimation, we choose a training set which aims to represent the average statistics of the source. It is likely that this averaging process causes the inferior performance of off-line adaptation. Whereas the LMS-based algorithms function with nonstationary sources and environments, the least squares scheme requires stationarity for proper performance.

Overall, results from these simulations demonstrate that the proposed iterative scheme successfully solves the least squares estimation problem for Hammerstein cascade architectures which use B-spline and piecewise linear function preprocessing blocks. The iterative algorithm approaches the more difficult problem of solving for optimal configurations of linear, FIR components and memoryless nonlinear components simultaneously by solving for the optimal configurations of each block individually. As with standard least squares off-line fitting techniques, we impose stationarity and ergodicity constraints on the input signal to ensure the desired performance. Given that the signals meet these conditions, the proposed iterative least squares scheme provides the optimal configuration for the discussed Hammerstein systems.

Chapter 5

Conclusions and Further Considerations

Cascade-of-filters architectures allow for acoustic echo cancellation systems to compensate for nonlinear distortions in the echo path. The need for such compensation arises from the use of smaller and cheaper components – loud speakers and, to a smaller extent, microphones – in telecommunication systems. Additionally, high signal levels from hands-free operation expose the nonlinear features in the loudspeaker and adversely affect the performance of standard linear echo cancellation schemes. The Volterra adaptive filters can represent a wide class of nonlinearities, but its large number of filter parameters increase computational complexity and cause slow convergence. Cascade-of-filters architectures using memoryless nonlinear blocks belong to a subclass of Volterra filters and have significantly fewer parameters, thus making them less computationally complex while possessing faster convergence characteristics.

By employing the method of general normalized nonlinear adaptive filters, we design and implement the cubic B-spline function-linear, FIR cascade and piecewise linear function-linear, FIR cascade architectures capable of adapting to any model of memoryless nonlinearity. The piecewise linear function is a simplification of the cubic B-spline which linearly interpolates between its consecutive control points. The step size normalization process provides guarantees of stability and filter convergence independent of signal and environment statistics. The standard NLMS algorithm for linear, FIR filters converges to the optimal least squares filter. We extend the method of least squares to solve the optimal configuration of standalone B-spline and piecewise linear functions. Using the classic least squares solutions for both the linear, FIR filter and the given nonlinear functions, we devise an iterative method to compute the least squares estimate for Hammerstein cascaded filters.

Although theoretically the cascade architectures can always attain a level of performance which is no worse than that of the standalone NLMS filter, experimental results indicate that the adaptation dynamics of the nonlinear portions of these systems may inhibit this optimal behavior. In the absence of nonlinearity, we observe superior performance for the standalone NLMS filter, which may be the result of slower convergence of the cascade filters toward the optimal configuration. In the presence of nonlinear distortion, the Hammerstein architectures consistently outperform the linear adaptive filter. For independently drawn Gaussian input samples, the simplified piecewise linear function cascaded system performs better in echo cancellation than the spline function cascaded system, in part because the piecewise linear function component has better handling of input samples which fall outside the interval of control points. In the voice audio input AEC simulation, we observe that both Hammerstein systems again outperform the linear adaptive filter in the presence of nonlinear distortion and that both achieve a similar measure of performance.

Based on results from iterative least squares estimation, the on-line adaptation schemes converge to a suboptimal level of performance for independently drawn Gaussian input samples. The optimal ERLE measurement computed from least squares scheme is consistently higher than the steady state convergence ERLE value for the nonlinear normalized LMS algorithms. In addition, the least squares configuration for piecewise linear function cascade systems achieve a higher level of optimal performance than the spline cascade systems. This result is consistent with the performance results from the on-line adaptation simulations. Erroneous behavior occurs in the off-line adaptation technique when the input signal and echo path do not maintain stationarity and ergodicity. Consequently, the “optimal” configuration computed using this method for voice input AEC simulations may not reflect the true optimal configuration for the Hammerstein systems.

It is important to note that these cascade filters are just one component of the AEC system. In our simulations, we carefully choose experimental conditions so that we may test the performance of the adaptive filter systems in isolation. One key component of a standard AEC system which we do not implement is a method for filter length selection. In truly novel acoustic environments, the length of the impulse response is unknown to the system, and an external component must detect this length and select an appropriate number of linear filter taps. A linear filter that is too short results in erroneous behavior and the general inability to perform echo cancellation while a linear filter that is too long results in poor convergence speed. Our AEC simulations considered the the length of the acoustic impulse response to be known to proceed with testing. In general, we do not intend for these cascade filters to function by themselves as comprehensive AEC systems, and for this reason, we do not include real-world results. Rather, these cascade architecture filters offer an improvement to standalone linear NLMS filters in the presence of nonlinear distortion. We demonstrate the capability of the normalized Hammerstein architecture systems to outperform the standard linear approaches in well-controlled experimental settings. When implemented as part of a comprehensive AEC system, we expect the nonlinear cascade filters to outperform NLMS linear filters in real-world environments as well.

References

[1] J.-P. Costa and A. Arliaud, "Acoustic echo cancellation using nonlinear cascade filters," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. V, Hong Kong, 2003, pp. 389–392.

[2] R. Chinaboina *et al.*, "Adaptive algorithms for acoustic echo cancellation in speech processing," *International J. of Research and Reviews in Appl. Sciences*, vol. 7, no. 1, pp. 38–42, Apr. 2011.

[3] A. Uncini *et al.*, "Acoustic echo cancellation in the presence of distorting loudspeakers," in *Proc. of Eusipco*, vol. 1, Tolouse, Sep. 2002, pp. 535–538.

[4] M. Mossi *et al.*, "Robust and low-cost cascaded non-linear acoustic echo cancellation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2011, pp. 89–92.

[5] M. Scarpiniti *et al.*, "Comparison of hammerstein and wiener systems for nonlinear acoustic echo cancelers in reverberant environments," in *17th International Conference on Digital Signal Processing*, 2011, pp. 1–6.

[6] S. Kalluri and G. R. Arce, "A general class of nonlinear normalized adaptive filtering algorithms," *IEEE Transactions on Signal Processing*, vol. 47, no. 8, pp. 2262–2272, Aug. 1999.

[7] M. Asharif, K. Yamashita, and H. Miyagi, "Double-talk echo cancelling by non-freezing adaptive algorithm," in *International Workshop on Acoustic Signal Enhancement*, Sep. 2001.

[8] K. Fujii, N. Saitoh, R. Oka, and M. Muneyasu, "Acoustic echo cancellation algorithm tolerable for double talk," in *Hands-Free Speech Communication and Microphone Arrays*, May 2008, pp. 200–203.

[9] P. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation.* New York, NY: Springer Science, 2008.

[10] V. Hedge, C. Radhakrishnan, D. Krusienski, and W. K. Jenkins, "Series-cascade nonlinear adaptive filters," in *45th Midwest Symposium on Circuits and Systems*, vol. 3, 2002.

[11] S. Haykin, *Adaptive Filter Theory.* Up Saddle River, NJ: Prentice Hall, 1996.

[12] S. Guanieri, F. Piazza, and A. Uncini, "Multilayer feedforward networks with adaptive spline activation function," *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 672–683, May 1999.

[13] M. Solazzi, F. Piazza, and A. Uncini, "An adaptive spline nonlinear function for blind signal processing," in *Proc. IEEE Signal Processing Society Workshop, Neural Networks for Signal Processing*, vol. 1, 2000, pp. 396–404.

[14] S. C. Douglas. Acoustic echo cancellation (aec). Math Works. [Online]. Available: http://bit.ly/ZEXVvVl