WAVELET DOMAIN ADAPTIVE FILTERING

IN SIGNAL PROCESSING

by

Su-Wei Chang

B.S.E.E., University of Colorado, 1994

A thesis submitted to the

University of Colorado at Denver

in partial fulfillment

of the requirements for the degree of

Master of Science

Electrical Engineering

1999
This thesis for the Master of Science
degree by
Su-Wei Chang
has been approved
by
Tamal Bose
Jan Bialasiewicz
Miloje Radenkovic

4/5/99
DATE
Chang, Su-Wei (M. S. Electrical Engineering)
Wavelet Domain Adaptive Filtering in Signal Processing
Thesis Directed by Associate Professor Tamal Bose

ABSTRACT

The standard time domain least-mean-square (LMS) algorithm is a widely used and standard adaptive finite impulse response (FIR) filtering algorithm. This thesis presents and studies a recently developed wavelet domain LMS adaptive FIR filtering algorithm. The proposed algorithm uses the inverse of the Cholesky factor in the wavelet domain to pre-whiten the input. The pre-whitened input has a lower eigenvalue spread than the time domain input. Therefore, the convergence rate of the LMS algorithm is improved. This thesis shows that the wavelet domain LMS algorithm can be used to improve the convergence rate for the system identification problem and for the adaptive differential pulse code modulation (ADPCM) coding/compression method, as compared to the standard time domain LMS FIR filtering algorithm.

This abstract accurately presents the content of the candidate’s thesis. I recommend its publication.

Signed
Tamal Bose
CONTENTS

Figures ........................................................................................................................................ v
Tables ........................................................................................................................................ vii

Chapter
1. Introduction ................................................................................................................................. 1
2. Wavelet Transform and Filter Banks ......................................................................................... 3
   2.1 Wavelet Transform .................................................................................................................. 3
   2.2 Wavelet Filter Banks ............................................................................................................... 6
      2.2.1 Two-Band Wavelet Filter Banks ...................................................................................... 10
           Perfect Reconstruction ......................................................................................................... 10
           Haar Wavelet Filter Banks .................................................................................................. 12
           Daubechies Compactly Supported Wavelets Filter Banks ................................................. 13
   2.2.2 M-Band Regular Perfect Reconstruction Filter Banks ..................................................... 14
3. FIR Adaptive Filtering Using Least Mean Square (LMS) Algorithm .................................. 15
   3.1 Wiener Filter .......................................................................................................................... 15
   3.2 Steepest Descent Algorithm .................................................................................................. 18
   3.3 LMS Algorithm ....................................................................................................................... 21
      3.3.1 Standard Time Domain LMS Algorithm ......................................................................... 21
      3.3.2 Self-Orthogonalizing Adaptive LMS Filtering Algorithm ............................................ 23
      3.3.3 Frequency Domain LMS Algorithm ............................................................................... 24
      3.3.4 Wavelet Domain LMS Algorithm ................................................................................... 25
4. System Identification .................................................................................................................. 29
   4.1 System Identification Using the Time Domain LMS Filtering Algorithm ......................... 29
   4.2 System Identification Using the Wavelet Domain LMS Filtering Algorithm .................... 39
   4.3 Comparisons and Discussions ............................................................................................... 49
5. Adaptive Differential Pulse Code Modulation (ADPCM) ..................................................... 51
   5.1 ADPCM Using the Time Domain LMS Filtering Algorithm ............................................... 55
   5.2 ADPCM Using the Wavelet Domain LMS Filtering Algorithm .......................................... 59
   5.3 Comparisons and Discussions ............................................................................................... 62
6. Conclusions ................................................................................................................................. 64
Appendix ......................................................................................................................................... 66
References .................................................................................................................................... 94
FIGURES

Figure

2-1 Basis Functions and Time-Frequency Resolution of the STFT ........................................ 5
2-2 Basis Functions and Time-Frequency Resolution of the WT ......................................... 6
2-3 Generic Two-Band Wavelet Analysis Filter Bank ......................................................... 7
2-4 Example of Orthonormal Wavelet Bases ........................................................................ 9
2-5 Two-Band Perfect Reconstruction Wavelet Filter Banks .............................................. 10
2-6 Two-Band Perfect Reconstruction Filter Coefficients ................................................. 12
2-7 M-Band Perfect Reconstruction Filter Banks .............................................................. 14
3-1 Block Diagram of a Generic Statistical Filtering Problem .......................................... 15
3-2 Block Diagram of a Generic Steepest Descent FIR Filtering Algorithm ..................... 20
3-3 Block Diagram of a General LMS FIR Filtering Algorithm ....................................... 23
3-4 Block Diagram of a Wavelet Domain LMS FIR Adaptive Filter .................................. 25
4-1 Block Diagram of a System Identification Model Using Time Domain LMS Filtering Algorithm ........................................................................................................... 30
4-2 The Desired Output $d(n)$ vs. the Filter Output $y(n)$ (Time Domain LMS Algorithm, $\mu = 0.05$) ........................................................................................................ 36
4-3 The Estimated Error (Time Domain LMS Algorithm, $\mu = 0.05$) ................................ 36
4-4 Adaptive Filter Coefficients $h_0$, $h_1$, $h_2$, and $h_3$ .................................................. 37
4-5 Adaptive Filter Coefficients $h_4$, $h_5$, $h_6$, and $h_7$ .................................................. 37
4-6 Adaptive Filter Coefficients $h_8$, $h_9$, $h_{10}$, and $h_{11}$ ............................................. 38
4-7 Adaptive Filter Coefficients $h_{12}$, $h_{13}$, $h_{14}$, and $h_{15}$ .............................................. 38
4-8 Adaptive Filter Impulse Response After 500 Iterations (Time Domain LMS Algorithm, $\mu = 0.05$) ........................................................................................................ 39
4-9 Block Diagram of a System Identification Model Using Wavelet Domain LMS Filtering Algorithm ........................................................................................................... 40
4-10 The Desired Output $d(n)$ vs. the Filter Output $y(n)$ (Wavelet Domain LMS Algorithm, $\mu = 0.05$) ........................................................................................................ 45
4-11 The Estimated Error (Wavelet Domain LMS Algorithm, $\mu = 0.05$) ......................... 45
4-12 Wavelet Domain Filter Impulse Response After 500 Iterations (Wavelet Domain LMS Algorithm, $\mu = 0.05$) ........................................................................................................ 46
4-13 Time Domain Filter Impulse Response After 500 Iterations (Wavelet Domain LMS Algorithm, $\mu = 0.05$) ........................................................................................................ 46
4-14 The Desired Output $d(n)$ vs. the Filter Output $y(n)$ (Wavelet Domain LMS Algorithm, $\mu = 0.3$) ........................................................................................................ 47
4-15 The Estimated Error (Wavelet Domain LMS Algorithm, $\mu = 0.3$) .......... 47
4-16 Wavelet Domain Filter Impulse Response After 500 Iterations (Wavelet Domain LMS Algorithm, $\mu = 0.3$) ..................................................................... 48
4-17 Time Domain Filter Impulse Response After 500 Iterations (Wavelet Domain LMS Algorithm, $\mu = 0.3$) .......................................................................................... 48
4-18 Mean-Squared Error of an Ensemble of 100 Runs ($\mu = 0.05$) ................. 50
4-19 Mean-Squared Error of an Ensemble of 100 Runs (Time domain LMS $\mu = 0.05$, Wavelet Domain $\mu = 0.3$) ................................................................. 50
5-1 Generic PCM Encoder .......................................................................................... 51
5-2a Generic ADPCM Encoder .................................................................................. 53
5-2b Generic ADPCM Decoder .................................................................................. 53
5-3 Generic Time Domain ADPCM System .............................................................. 55
5-4 Time Domain ADPCM Applied to a Sine Wave ............................................... 58
5-5 Time Domain ADPCM Applied to a Sine Wave with a Sudden Change .......... 59
5-6 Generic Wavelet Domain ADPCM System ....................................................... 59
5-7 Wavelet Domain ADPCM Applied to a Sine Wave .......................................... 61
5-8 Wavelet Domain ADPCM Applied to a Sine Wave with a Sudden Change .... 62
5-9 MSE of an Ensemble of 50 Runs (Input is a Sine Wave) ......................... 63
5-10 MSE of an Ensemble of 50 Runs (Sine Wave with a Sudden Change) ....... 63
TABLES

Table

2-1 The \( h_0 \) Coefficients of Daubechies Compactly Supported Wavelets .............. 13
5-1 Jayant Quantizer .................................................................................................. 54
1. **Introduction**

The adaptive filtering algorithm may be classified in two groups: linear adaptive filtering and nonlinear adaptive filtering. Simply described, the filter is considered linear if the filter output varies linearly as a function of the filter input, otherwise, the filter is nonlinear. Some of the common linear adaptive filtering algorithms are: the steepest descent algorithm, the least-mean-square (LMS) algorithm, the recursive least-squares (RLS) algorithm, and the square root (QR) adaptive algorithm. Some of the common nonlinear adaptive filtering algorithms include: the blind deconvolution, the back-propagation learning network, and the radial basis function network. Detailed descriptions of the adaptive filtering algorithms mentioned above can be found in [2].

A recently developed wavelet domain LMS adaptive FIR filtering algorithm [1] shows that the autocorrelation matrix of the wavelet domain FIR filter input has a special sparse structure. The correlation matrices of the “details” (please see Chapter 2 for the definition of “details”) of the wavelet domain FIR filter input also have special sparse structures. From [1], the autocorrelation matrix of the filter input can be estimated and updated at each recursive step using the correlation matrices of the input “details” to reduce computational complexity of the algorithm.
The purpose of this thesis is to study, simulate, and demonstrate that the wavelet domain LMS adaptive FIR filtering algorithm provides convergence rate improvement over the time domain LMS adaptive FIR filtering algorithm. This thesis also shows the wavelet domain algorithm can be applied for the system identification problem and for the ADPCM coding/compression method.

Chapter 2 provides an introduction of the theory of wavelet transformation and filter banks. This chapter is focused on describing the details of 2-Band perfect reconstruction filter bank theory which are used in the simulations of subsequent chapters. The M-Band perfect reconstruction filter bank theory is discussed topically in this chapter to introduce the reader to alternate techniques.

Chapter 3 presents the derivation of the LMS FIR filtering algorithm from the steepest descent algorithm. Time domain, frequency domain, and wavelet domain LMS FIR filtering algorithms are included.

Chapters 4 and 5 describe two applications of the LMS adaptive algorithms: system identification and ADPCM. Simulations of these applications using time domain and wavelet domain LMS adaptive filtering algorithms are also presented.

The thesis concludes with Chapter 6, which summarizes and compares the results of the simulations. Also, advantages and disadvantages of the time domain and wavelet domain LMS FIR filtering algorithms are presented.
2. Wavelet Transform and Filter Banks

An overview of comparisons between Fourier transform and wavelet transform is given in section 2.1. The traditional Fourier transform bases come from differential equations, which does not have a dilation equation. Wavelets include the scaling function, which is the solution of the dilation equation. The dilation equation includes two time scales, therefore, the solution of a dilation equation \( \phi(t) \) is finitely bounded. The scaling function \( \phi(t) \) is nonzero in a finite interval, and therefore \( \phi(2t) \) is zero outside of half of the interval. This unique bounded characteristic leads the wavelet bases to be localized. To best describe the relationships between the low pass filter \( h_0 \) and the scaling function, and the high pass filter \( h_1 \) and the wavelets, section 2.2 shows the dilation and wavelet equations and the derivation of the solutions to these equations.

2.1 Wavelet Transform

The Fourier transform is a traditional way of presenting a signal in the frequency domain. The standard Fourier transform of a continuous time signal \( x(t) \) is

\[
X(w) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} x(t)e^{-j\omega t} dt.
\]  

(2-1)
The Fourier transform described above works fine for stationary signals (e.g., sinusoidal waves), where the frequency of the signal does not vary with time. However, in many applications (e.g., audio and video signal processing), the signals are nonstationary and one is interested to obtain localized frequency information in a specific period of time. Another difficulty with the standard Fourier transform, even with a stationary signal, is when there is a sudden change in the signal (e.g., noise transients). The result is seen as broadband energy over the entire frequency axis because of the infinite extent of the basis functions [3]. To overcome these shortcomings of the standard Fourier transformation, a windowed or short-time Fourier transform (STFT) was developed. The STFT obtains the time-localized interval of the $x(n)$-of-interest and takes the Fourier transform of it as

$$X_{\text{window}}(w, \tau) = \int_{-\infty}^{+\infty} x(t)g(t - \tau)e^{-jwt}dt.$$  \hspace{1cm} (2.2)

Another alternative is to use the wavelet transform to localize the time-frequency characteristics of a signal. The wavelet transform of $x(t)$ is

$$X_{\text{wavelet}}(a, b) = |a|^{-1/2} \int_{-\infty}^{+\infty} x(t) \psi\left(\frac{t - b}{a}\right)dt,$$  \hspace{1cm} (2.3)

where $a$ is the scaling parameter. Small values of the absolute value of the scaling parameter ($|a|$) represent high frequencies and large values of $|a|$ represent low frequencies. The time localization is controlled by the translation parameter $b$. The functions $\psi^{a, b}$ are “wavelets” and the function $\psi$ is the “mother wave”.

Figure 2-1 shows the comparisons between the windowed Fourier transform and the wavelet transform [4]. According to Figure 2-1, the windowed Fourier transform only provides frequency information of the signal since the time-axis bandwidth is a constant and only limited time domain information is available [3]. Figure 2-2 shows that for the wavelet transform, a higher frequency corresponds to a wider bandwidth in frequency axis but a narrower bandwidth in time-axis. A lower frequency corresponds to a narrower bandwidth in frequency axis but a wider bandwidth in time-axis [3]. Therefore, the wavelet transform (WT) is more capable of presenting multiresolution of time-frequency characteristics of the signal compared to STFT.
2.2 Wavelet Filter Banks

Filter banks are at the heart of wavelet domain algorithm simulations. The signal is decomposed or transformed to wavelet domain using the analysis filter bank and reconstructed using the synthesis filter bank. Figure 2-3 shows the block diagram of a generic 2-Band analysis filter bank. In Figure 2-3, $H_0$ represents the low pass filter and $H_1$ represents the high pass filter in the analysis bank [7]. The output signal of each filtering stage is downsampled by a factor of two.
From Figure 2-3, the outputs of the filters (a3, d3, a2, d2, a1, d1, etc.) are called "details" of the input signal x(t). To describe the relations between the highpass and lowpass filters in Figure 2-3 and the scaling and wavelet functions, the dilation equation (equation 2.4) and the wavelet equation (equation 2.5) are introduced as

\[ \phi(t) = 2 \sum_{k}^{N} h_{0}(k) \phi(2t - k), \]  

(2.4)

\[ \psi(t) = 2 \sum_{k}^{N} h_{1}(k) \phi(2t - k). \]  

(2.5)

The dilation equation can be solved by taking the continuous-time Fourier transform of both sides of the equation [5]. The results (equations 2.6-2.9) show that the solution of the dilation equation is a product of cascaded filters.
\[ \Phi(\omega) = \Phi\left(\frac{\omega}{2}\right) H_0\left(\frac{\omega}{2}\right) \]  

(2.6)

\[ \Phi(\omega) = \Phi\left(\frac{\omega}{4}\right) H_0\left(\frac{\omega}{4}\right) H_0\left(\frac{\omega}{2}\right) \]  

(2.7)

\[ \Phi(\omega) = \Phi\left(\frac{\omega}{8}\right) H_0\left(\frac{\omega}{8}\right) H_0\left(\frac{\omega}{4}\right) H_0\left(\frac{\omega}{2}\right) \]  

(2.8)

\[ \Phi(\omega) = \prod_{j=1}^{\infty} H_0\left(\frac{\omega}{2^j}\right) \Phi(0) \]  

(2.9)

To obtain \( \phi(t) \), perform the inverse Fourier transform of \( \Phi(\omega) \) shown in equation 2.9. In general, continuous filtering and downsampling of \( x(t) \) until the output of the low pass filter is out of resolution will lead to the scaling function \( \phi(t) \). The scaling function is dependent on the values of the low pass filter coefficients. The wavelet equation (equation 2.5) can be solved after obtaining \( \phi(t) \). Figure 2-4 shows some examples of orthonormal wavelet bases [4]. A wavelet basis is orthonormal when the scaling and translation parameters are non-overlapping. The finite-interval-bounded characteristics of \( \phi(t) \) and \( \psi(t) \) are shown clearly in Figure 2-4 [4].
Figure 2-4  Example of Orthonormal Wavelet Bases
2.2.1 Two-Band Wavelet Filter Banks

Two-Band wavelet filter banks are used for simulations in this thesis because of their simplicities. The Haar filter bank is used for the system identification problem in Chapter 4 to verify the convergence rate improvement of the wavelet domain LMS FIR filtering algorithm. The Haar basis function is orthonormal, but it is not continuous. Therefore, the Haar basis is in general not adequate for signal processing [3]. The Daubechies compactly supported wavelets filter banks have continuous basis functions and are widely used. The Daubechies compactly supported wavelets filters with length three and six were used for the ADPCM simulations in this thesis.

2.2.1.1 Perfect Reconstruction

Figure 2-5 shows a block diagram of a generic Two-Band perfect reconstruction filter bank [6]. From Figure 2-5, the gap between the analysis bank and synthesis bank indicates the downsampld signal can be transmitted, compressed, coded, or stored with a lower computational complexity. The signal is subsequently reconstructed using the synthesis filter bank.

![Figure 2-5 Two-Band Perfect Reconstruction Wavelet Filter Banks](image-url)
Two crucial requirements make the perfect reconstruction possible: the alias cancellation condition (equation 2.10) and the no distortion condition (equation 2.11), given by

\[ F_0(z)H_0(-z) + F_1(z)H_1(-z) = 0, \quad (2.10) \]

\[ F_0(z)H_0(z) + F_1(z)H_1(z) = 2z^{-1}. \quad (2.11) \]

The filter designs for the filters \( H_0(z), H_1(z), F_0(z), \) and \( F_1(z) \) are as follows:

1) To satisfy equation 2.10, choose \( F_0(z) = H_1(-z) \) and \( F_1(z) = -H_0(-z). \)

2) Let \( P_0(z) = F_0(z)H_0(z), \) \( P_0(z) \) is a low pass filter since both \( F_0(z) \) and \( H_0(z) \) are low pass filters.

3) \( P_1(z) = F_1(z)H_1(z), \) substitute values of \( F_1(z) \) and \( H_1(z) \) from step 1).

\[ P_1(z) = -H_0(-z)F_0(-z) = -P_0(-z). \]

4) Now, the equation 2.11 is simplified to

\[ P_0(z) - P_0(-z) = 2z^{-1} \quad (2.12) \]

5) Design a low pass filter that satisfies equation (2.12) and use step 1) to derive other filters. Figure 2-6 shows the relations between filter coefficients.
2.2.1.2 Haar Wavelet Filter Banks

The Haar filter banks consist of four simple filters: The low pass (equation 2.13) and high pass (equation 2.14) filters in the analysis filter bank, and the low pass (equation 2.15) and high pass (equation 2.16) filters in the synthesis bank.

\[
h_0 = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}] \quad (2.13)
\]

\[
h_1 = [\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}] \quad (2.14)
\]

\[
f_0 = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}] \quad (2.15)
\]

\[
f_1 = [-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}] \quad (2.16)
\]
2.2.1.3 Daubechies Compactly Supported Wavelets Filter Banks

The $h_0$ (low pass filter) coefficients of Daubechies compactly supported wavelet filter banks is shown in Table 2-1 [4]. Filter coefficients for $h_1$, $f_0$, and $f_1$ can be derived using Figure 2-6.

Table 2-1 The $h_0$ (Low Pass) Coefficients of Daubechies Compactly Supported Wavelets

<table>
<thead>
<tr>
<th>N = 2</th>
<th>n</th>
<th>$h_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.0836130607382057</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>.8065160013164321</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>-.2241428690402344</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>-.1380150200405656</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N = 3</th>
<th>n</th>
<th>$h_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.2028751852325858</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>.5068516003190219</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>-.4987770211840314</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>-.1380150200405656</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N = 4</th>
<th>n</th>
<th>$h_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.2323778127084594</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>.7146467075232013</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>-.4238948904605013</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>-.0772715936408769</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N = 5</th>
<th>n</th>
<th>$h_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.1601227170419275</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>.6128292670718927</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>.7328882383737725</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>.1346248400112063</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>-.3364863864883282</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N = 6</th>
<th>n</th>
<th>$h_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.1114374733353035</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>.4964223500384533</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>.7211224840211459</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>.3122807570150165</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>-.2226564305545406</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>-.1256760376376309</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N = 7</th>
<th>n</th>
<th>$h_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.0957516453782252</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>.7072286500000000</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>.0335423031748000</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>.0035538422101151</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>.0011772575000145</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N = 8</th>
<th>n</th>
<th>$h_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.0677320450820317</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>.3957320450820317</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>.7251320450820317</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>.4077320450820317</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>.0330603203032012</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>-.2240503203032012</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>.0710002192002702</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>-.0080101931101274</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>-.0380101931101274</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>.0149700166086858</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>.0122590493506566</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>.0019597379295413</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>.0010184070344723</td>
<td>13</td>
</tr>
</tbody>
</table>
2.2.2 M-Band Regular Perfect Reconstruction
Filter Banks

Section 2.1 shows the Two-Band Wavelet transform works great to represent the localized time-frequency characterizations of transients in a low frequency signal. However, a long-duration high frequency signal (wide bandwidth in frequency-axis and narrow bandwidth in time-axis) is not well decomposed using a Two-Band wavelet transform described in previous sections. M-Band wavelets provide the capability of zoom in into narrow bandwidth (in time-axis) high frequency components of a signal.

Figure 2-7 shows a generic M-Band wavelet filter bank [8]. The design of M-Band wavelet filters for a perfect reconstruction is not straightforward as for the Two-Band wavelet filter bank. Please refer to [8] for details of M-Band K-Regular perfect reconstruction filter banks.

![Figure 2-7 M-Band Perfect Reconstruction Filter Banks](image-url)
3. **FIR Adaptive Filtering Using Least Mean Square (LMS) Algorithm**

The LMS algorithm is a widely used linear adaptive filtering algorithm. In order to understand the derivation of the LMS algorithm, the Wiener filter and the steepest descent algorithm are introduced in sections 3.1 and 3.2. Section 3.3 describes the time domain LMS algorithm, the frequency domain LMS algorithm, and the wavelet domain LMS algorithm. Please note that only real-valued systems are considered in this thesis. Therefore, the conjugate parts are eliminated from the equations. In order to distinguish between scalar, vector and matrix variables, the vector or matrix variables will be presented in boldface characters.

### 3.1 Wiener Filter

Figure 3-1 shows the block diagram of a general statistical filtering problem [2]. The purpose of the linear discrete filter is to estimate the desired response \( d(n) \).

![Figure 3-1 Block Diagram of a Generic Statistical Filtering Problem](image)
The filter output $y(n)$ and the estimated error $e(n)$ of the system shown in Figure 3-1 are defined as

$$y(n) = \sum_{k=0}^{\infty} w_k u(n-k), \text{ and}$$  \hfill (3.1)

$$e(n) = d(n) - y(n).$$ \hfill (3.2)

For the optimum filter design shown in Figure 3-1, the mean-square error of the system needs to be minimized. The cost function $J$ is the function that needs to be minimized for the system to reach the optimum solution. Therefore, the cost function of the system shown in Figure 3-1 can be defined as the mean-square error of the system. Equation (3.3) shows the definition of the mean-square error of the system, where $E[\cdot]$ is the expectation operator.

$$J = E[|e(n)|^2]$$ \hfill (3.3)

The optimum solution of the problem shown in Figure 3-1 occurs when the output of the linear filter, $y(n)$, becomes very close to the desired response $d(n)$. Or in other words, the minimum mean-squared error of the system is obtained. Let $y_o(n)$ be the output of the optimized filter $w_o(n)$, the minimum mean-squared error $J_{\text{min}}$ is defined as

$$y_o(n) = \sum_{k=0}^{\infty} w_o(n) u(n-k), \hfill (3.4)$$
\[ e_o(n) = d(n) - y_o(n) = d(n) - \text{Estimated } d(n), \quad \text{(3.5)} \]

\[ J_{\text{min}} = E[|e_o(n)|^2], \quad \text{(3.6)} \]

where \( e_o(n) \) is the optimum estimation error and \( y_o(n) \) is the optimum filter output.

The principle of orthogonality may be summarized as “The necessary and sufficient condition for the cost function \( J \) to attain its minimum value is that the corresponding value of the estimated error \( e_o(n) \) is orthogonal to each input sample that enters into the estimation of the desired response at time \( n \) [2]”. The principle of orthogonality provides a method to verify that the optimum filter design is reached. Equation (3.7) defines the principle of orthogonality.

\[ E[u(n-k)e_o(n)] = 0 \quad \text{(3.7)} \]

Substituting equations (3.4) and (3.5) into equation (3.7), the Wiener-Hopf equations are

\[ E[u(n-k)(d(n) - \sum_{i=0}^{\infty} w_{oi} u(n-i))] = 0, \quad k = 0, 1, 2, \ldots \quad \text{(3.8)} \]

Rearranging equation (3.8) yields

\[ \sum_{i=0}^{\infty} w_{oi} E[u(n-k)u(n-i)] = E[u(n-k)d(n)], \quad k = 0, 1, 2, \ldots \quad \text{(3.9)} \]

Let \( R \) be the autocorrelation matrix of the filter input and \( P \) be the cross-correlation matrix of the filter input and desired response, where \( R \) and \( P \) are defined as
\[
\mathbf{R} = E[\mathbf{u}(n)\mathbf{u}^T(n)], \text{ and} \tag{3.10}
\]
\[
\mathbf{p} = E[\mathbf{u}(n)d(n)]. \tag{3.11}
\]
Combining equations (3.9), (3.10), and (3.11), the Wiener-Hopf equations are reduced to a compact matrix form,
\[
\mathbf{R}\mathbf{w}_o = \mathbf{p}. \tag{3.12}
\]
The solution to the Wiener-Hopf equations is
\[
\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p}. \tag{3.13}
\]
Please note that the filter input of the Wiener filter is assumed to be zero mean. The solution shown in equation (3.13) is straightforward, but is computationally expensive, especially for a large number of tap weights (inverse of \( \mathbf{R} \) has to be computed). Another way to obtain the minimum mean-squared error of the system is using the steepest descent algorithm.

### 3.2 Steepest Descent Algorithm

The method of steepest descent is a gradient and iterative algorithm. The step size can either be chosen to accomplish the maximum decrease of the cost function in each iterative step or can be chosen to be a constant (fixed step-size) [10]. For the purpose of this thesis, the fixed step-size algorithm will be used. This algorithm uses the negative gradient as the direction vector of the cost function to reach the
minimum point of the cost function. The algorithm starts with some initial condition assumptions, computes the gradient vector using the initial conditions, and updates filter coefficients based on the negative gradient as the direction vector. The recursive process continues until the minimum of the cost function is reached.

The cost function $J$ of the system shown in Figure 3-1 can be expressed as a function of filter tap weights. Rearranging equation (3.3), using equations (3.1) and (3.2), yields [9]

$$J(w) = E[e(n)]^2 = E[e(n)e(n)] = E[d(n)d(n)] - 2E[d(n)\sum_{k=0}^{M-1} w(n-k)] + E[\sum_{k=0}^{M-1} \sum_{i=0}^{M-1} w(n-k)w(n-i)]$$

$$= \sigma_d^2 - 2\sum_{k=0}^{M-1} w E[d(n)u(n-k)] + \sum_{k=0}^{M-1} \sum_{i=0}^{M-1} w E[u(n-k)u(n-i)]$$

$$= \sigma_d^2 - 2\sum_{k=0}^{M-1} w E[d(n)u(n-k)] + \sum_{k=0}^{M-1} \sum_{i=0}^{M-1} w E[u(n-k)u(n-i)]$$

$$= \sigma_d^2 - 2\sum_{k=0}^{M-1} w T w(n) + w(n)Rw(n),$$

(3.14)

where $\sigma_d^2$ is the variance of the desired response, assuming that the desired response $d(n)$ is zero mean. The minimum mean-squared error of the system, or the minimum point of the cost function $J$, can be determined by obtaining the gradient of $J$ and setting it to zero. From equation (3.14) the gradient of $J$ is

$$\nabla J(n) = -2p + 2Rw(n).$$

(3.15)

Setting $\nabla J(n) = 0$ to find the minimum point of $J(n)$,

$$\nabla J(n) = -2p + 2Rw(n) = 0,$$

(3.16)
\[ w_o = R^{-1}p. \] 

(3.17)

This yields the same solution found in equation (3.13), which is the solution of the Wiener-Hopf equations based on the principle of orthogonality.

Figure 3-2 shows a generic block diagram of the steepest descent FIR filtering algorithm.

![Block Diagram of a Generic Steepest Descent FIR Filtering Algorithm](image)

Equation (3.19) is the update equation of the algorithm shown in Figure 3-2, where \( \mu \) is the fixed step size of the algorithm. The update equation is used to update filter coefficients of the subsequent recursive step of the algorithm.

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

(3.18)

\[ w(n+1) = w(n) + \mu (p - Rw(n)), \quad n = 0, 1, 2, ... \] 

(3.19)
3.3 LMS Algorithm

To proceed with the steepest descent algorithm described in section 3.2, statistical characteristics of the filter input $u(n)$ and the desired response $d(n)$ have to be known to determine $R$ and $p$. In reality, a priori knowledge of statistical characteristics of the system are generally not available. The simplest solution is to estimate $R$ and $p$ using available data. The estimated $R$ and $p$ are shown in equations (3.20) and (3.21).

\[ \hat{R} = u(n)u^T(n) \]  
(3.20)

\[ \hat{p} = u(n)d(n) \]  
(3.21)

3.3.1 Standard Time Domain LMS Algorithm

Substituting estimated $R$ and $p$ from equations (3.20) and (3.21) into equation (3.19) results in

\[ w(n+1) = w(n) + \mu (u(n)d(n) - u(n)u^T(n)w(n)), \]  
(3.22)

\[ w(n+1) = w(n) + \mu u(n)(d(n) - u^T(n)w(n)), \]  
(3.23)

\[ w(n+1) = w(n) + \mu u(n)(d(n) - y(n)), \]  
and  
(3.24)

\[ w(n+1) = w(n) + \mu u(n)e(n), \quad n=0,1,2,... \]  
(3.25)
Equation (3.25) shows the standard time domain LMS FIR filtering algorithm. The simplicity of the LMS algorithm is that the estimated values of $R$ and $p$ are used.

For stability of the system, the fixed step size $\mu$ ranges between 0 and $\frac{2}{\lambda_{\text{max}}}$, where $\lambda_{\text{max}}$ is the maximum eigenvalue of the autocorrelation matrix of the filter input.

The difference between the final value of the cost function $J$ obtained by the LMS algorithm $J(\infty)$ and the minimum value of $J$ obtained by Wiener filter $J_{\text{min}}$ is called the excess mean-squared error. The ratio of the excess mean-squared error to the minimum mean-squared error obtained by Wiener filter is called the misadjustment. The misadjustment can be controlled by the value assigned to the fixed step size $\mu$.

A large value of $\mu$ leads the system to converge faster but also leads to a larger misadjustment. This is because the LMS algorithm uses estimate values of $R$ and $p$ at each recursive step to compute the estimated gradient vector, and the filter tap weights suffer from a gradient noise. With a large step size $\mu$, the system progress faster, but the gradient noise is not largely filtered out as when a small value of $\mu$ is assigned to the system. A small value of $\mu$ makes the system converge slower, but reduce the misadjustment. Figure 3-3 shows a generic block diagram of the LMS FIR filtering algorithm.
Please note that to be able to distinguish between the time domain adaptive filter and the wavelet domain adaptive filter to be discussed in section 3.3.4, the time domain adaptive filter in equations (3.22) through (3.25) will be denoted as $h(n)$.

### 3.3.2 Self-Orthogonalizing Adaptive LMS Filtering Algorithm

The self-orthogonalizing adaptive LMS algorithm is a modified LMS algorithm to improve the convergence rate of the LMS algorithm. The filter input is prewhitened and transformed into a less correlated (or uncorrelated) vector. This transformed vector is then used as the LMS algorithm filter input. The convergence rate of the LMS algorithm is improved because the eigenvalue spread of the filter input is reduced. The update equation for this algorithm is [2]
\[ w(n+1) = w(n) + \alpha \mathbf{R}^{-1} \mathbf{u}(n)e(n), \quad n = 0,1,2,\ldots, \quad (3.26) \]

where \( \alpha \) is a constant and ranges between 0 and 1. According to [4], \( \alpha \) can set to be \( \frac{1}{M} \), where M is the filter length. A statement in [2] makes this modified LMS algorithm very powerful: "An important property of the self-orthogonalizing filtering algorithm is that, in theory, it guarantees a constant rate of convergence, irrespective of input statistics". For detailed information of this algorithm, please refer to Reference 2.

3.3.3 Frequency Domain LMS Algorithm

The self-orthogonalizing filtering algorithm described in section 3.3.1 provides convergence rate improvement to the LMS algorithm. The disadvantage of this algorithm is that it is computationally expensive. To reduce computation complexity and take advantage of the fast convergence speed of the self-orthogonalizing filtering algorithm, the frequency domain LMS algorithm was developed. The frequency domain LMS algorithm uses discrete-cosine transform (DCT) to transform the input vector to frequency domain, estimates the eigenvalues using estimated autocorrelation of the input, and updates the filter coefficients. The outcome of this algorithm is a more efficient algorithm than the time domain
algorithm. The frequency domain LMS algorithm is out of the scope of this thesis, for detailed description of this algorithm, please refer to Reference [2].

3.3.4 Wavelet Domain LMS Algorithm

The main idea of the wavelet domain LMS algorithm is similar to the frequency domain LMS algorithm: to reduce computational complexity. A recently developed wavelet domain LMS adaptive filtering algorithm [1] uses sparse structures of matrices in wavelet domain to estimate and update autocorrelation matrix of the filter input. Figure 3-4 shows block diagram of a wavelet domain LMS FIR adaptive filter.

![Block Diagram of a Wavelet Domain LMS FIR Adaptive Filter](image)

Figure 3-4   Block Diagram of a Wavelet Domain LMS FIR Adaptive Filter
Let $y(n)$ be the wavelet transform of the filter input $x(n)$. Let $Q$ be the wavelet transformation matrix, $y(n) = Qx(n)$. The matrix $Q$ contains high pass and low pass filters and decimators described in Chapter 2. For example, let a two-band, one level wavelet be used to decompose the filter input $x(n)$. Let $a$ be the output of the low pass filter and the decimator of the analysis filter bank and let $b$ be the output of the high pass filter and the decimator of the analysis filter bank. The wavelet transform of the filter input $x(n)$ is $y(n) = [a \ b]$, where $a$ and $b$ are vectors of half of the size of $x(n)$.

According to [1], the update equation for the wavelet domain adaptive filter coefficients is

$$w(n+1) = w(n) - \mu \hat{g}(n), \quad n = 0, 1, 2, \ldots, \quad (3.26)$$

$$\hat{R}_y(n) \hat{g}(n) = -2e(n)y(n), \quad (3.27)$$

where $w(n)$ is the wavelet transform of the time domain adaptive filter $h(n)$, that is, $w(n) = Qh(n)$. $\hat{R}_y$ is the estimated autocorrelation matrix of the wavelet domain filter input, $\hat{R}_y(n) = y^T(n)y(n)$. The estimated error $e(n)$ has the similar definition as for the time domain LMS algorithm,

$$e(n) = d(n) - z(n), \quad (3.28)$$
\[ z(n) = w^T(n)y(n). \] 

(3.29)

Expanding (3.29) yields

\[ z(n) = (Qh(n))^T Qx(n), \] 

(3.30)

and

\[ z(n) = h^T(n)Q^T Qx(n). \] 

(3.31)

From Figure 3-4, the wavelet transformation is a unitary transformation. Therefore,

\[ Q^T Q = I, \] 

(3.32)

\[ z(n) = h^T(n)x(n). \] 

(3.33)

Equation (3.33) shows the output of the wavelet domain LMS adaptive filter is actually the same as the output of the time domain LMS adaptive filter. This implies that the time domain desired response \( d(n) \) is used for the wavelet domain LMS algorithm. Combining and rearranging equations (3.26) and (3.27) yields the final update equation used for the simulations in this thesis,

\[ w(n+1) = w(n) + 2\mu \hat{R}_y^{-1} e(n)y(n), \quad n = 0, 1, 2, \ldots \] 

(3.34)

A difficulty was encountered when using equation (3.34) for simulations. When the autocorrelation matrix \( \hat{R}_y \) is either singular or badly scaled (in other words, the eigenvalue spread of \( \hat{R}_y \) is large), the inverse of \( \hat{R}_y \) cannot be computed, an error occurred, and the recursive procedure stopped. According to [1], when situations
like this occurs, a “diagonal loading” methodology and be used to ensure the $\mathbf{R}_y$ is not ill-conditioned. Let $i$ be iteration step of the algorithm, $c$ be a small positive constant, the diagonal loading can be implemented as

$$\hat{\mathbf{R}}_y = \frac{c}{i} \mathbf{I} + \mathbf{R}_y. \quad (3.35)$$

The emphasis of this thesis is to study, simulate, and apply the wavelet domain LMS FIR adaptive algorithm to different applications. More efficient way of running the wavelet domain LMS algorithm that requires more complex estimation methodologies using the sparse structure of the wavelet domain matrices is documented in [1].
4. System Identification

System identification is used to provide a linear mathematical model of an unknown dynamic process or plant. There are many ways to obtain the mathematical model of an unknown system. The adaptive controller using pole placement design with or without zero cancellation (deterministic self-tuning regulators) [11], and adaptive filtering using LMS or recursive least-squares (RLS) algorithms, are examples of commonly used methodologies. The system identification using the adaptive LMS filtering algorithm is described in subsequent sections.

4.1 System Identification Using the Time Domain LMS Filtering Algorithm

Figure 4-1 shows the block diagram of a system identification model using a standard time domain adaptive LMS algorithm. The unknown system and the adaptive filter share the same input $u(n)$. The error $e(n)$ is the difference between the output of the unknown system $d(n)$ and the output of the adaptive filter $y(n)$. The ideal mathematical model of the unknown system is obtained when $e(n) = 0$. Therefore, the error $e(n)$ is used to control the adaptive filter. The cost function that needs to be minimized is $J = E[|e(n)|^2]$, where $e(n) = d(n) - y(n)$. Please note that in reality the error $e(n)$ will never reach zero because the LMS algorithm uses
estimated values of $\mathbf{R}$ and $\mathbf{p}$. $\mathbf{R}$ is the autocorrelation matrix of the adaptive filter input and $\mathbf{p}$ is the cross correlation of the filter input and the unknown system output.

![Block Diagram of a System Identification Model Using Time Domain LMS Filtering Algorithm](image)

Figure 4-1 Block Diagram of a System Identification Model Using Time Domain LMS Filtering Algorithm

The procedures of system identification using a standard time domain LMS filtering algorithm are described below.

1. Determining the unknown system

The unknown system to be modeled by system identification has to be determined, so the input $\mathbf{u}$ can be applied to the unknown system and the output of the unknown system $d(n)$ can be obtained. The purpose of the system identification model shown in this thesis is to demonstrate that the adaptive FIR filter correctly modeled the "unknown" system. The "unknown" system is chosen to be a 16-tap
FIR filter (we will assume that we do not know the system). The filter coefficients are

\[ g(n) = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1] \]  \hspace{1cm} (4.1)

where \( n \) is the steps of iteration. After the appropriate steps of iteration, the adaptive FIR filter coefficients should converge to the values shown in equation (4.1).

2. Choosing an FIR filter tap length for system identification.

   Generally, for real applications, the system is unknown. Therefore, the filter tap length is normally assumed to be \( 2^k \), where \( k = 1, 2, 3, \ldots \). In this thesis, the adaptive filter tap length is chosen to be 16. Let \( h(n) \) be the adaptive filter impulse response, then the initial condition of this filter is

\[ h(n) = 0. \]  \hspace{1cm} (4.2)

When the system identification works properly, \( h(n) \) will identify with \( g(n) \) as the system converges (when error \( e(n) \) is close to zero and maintained as a constant value).

3. Defining an input \( u \) and assuming a total iteration number.
The input \( u \) used for the system identification simulation in this thesis is a random signal. The random signal is chosen because it can model a non-stationary input, and the tracking capability of the adaptive filter can be verified. The input \( u \) is a vector, and the length of the input \( u \) has to be equal or greater than the sum of the total iteration steps plus the filter tap length minus one. For example, if a 16-tap FIR filter is used and 500 iterations are planned for the simulation, the length of vector \( u \) must be equal or greater than 515, i.e.,

\[
\mathbf{u} = [u(499), u(498), u(497), \ldots, u(0), u(-1), u(-2), \ldots, u(-15)].
\]  

(4.3)

Let \( n = 500 \) be the iteration steps of the system identification model. Let 16 be the adaptive filter tap length, \( \mathbf{u}(n) \) is determined as

\[
\mathbf{u}(0) = [u(0), u(-1), u(-2), u(-3), \ldots, u(-15)]  
\]  

(4.4)

\[
\mathbf{u}(1) = [u(1), u(0), u(-1), u(-2), \ldots, u(-14)]  
\]  

(4.5)

\[
\mathbf{u}(2) = [u(2), u(1), u(0), u(-1), \ldots, u(-13)]  
\]  

(4.6)

\[
\vdots 
\]

\[
\mathbf{u}(499) = [u(499), u(498), u(497), u(496), \ldots, u(484)]  
\]  

(4.7)

4. Using LMS algorithm to update the adaptive filter tap weights and choosing a fixed step size \( \mu \) for the LMS algorithm.

From equation 3.25, the update equation for a standard time domain LMS filtering algorithm is

\[
\mathbf{h}(n+1) = \mathbf{h}(n) + \mu \mathbf{u}(n) e(n), n = 0, 1, 2, \ldots
\]

where \( \mu \) is the step size of the LMS algorithm and \( e(n) = d(n) - y(n) \). As discussed earlier in section
system stability requires that the fixed step size $\mu$ ranges between 0 and $\frac{2}{\lambda_{\text{max}}}$.

For the fastest LMS algorithm convergence rate, the ideal way is to use a fixed step size of $\frac{2}{\lambda_{\text{max}}} - \epsilon$, where $\epsilon$ is a small positive number. The problem is that a priori information of the filter input is generally not available, and the maximum eigenvalue of the autocorrelation matrix of the filter input $\lambda_{\text{max}}$ cannot be determined. In general, the system is evaluated with various $\mu$ values, e.g., 0.01, 0.05, 0.1, 0.5 etc. If the value of $\mu$ is too large, the system will become unstable which means the $\mu$ value used probably exceeds $\frac{2}{\lambda_{\text{max}}}$. Try smaller $\mu$ values until the system becomes stable and converges nicely.

5. Recursively updating the adaptive filter tap weights using step 4, and adjusting iteration steps if necessary.

Depending on the convergence rate of the system, adjust iteration numbers. It is desirable to choose a number that is beyond the convergence of the system to show that the convergence is stable.

Now, following the above 5 steps, an example of the time domain LMS algorithm is shown below.
1. Unknown system is a 16-tap FIR filter, and the impulse response of this FIR filter is \( g(n) = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1] \).

2. The adaptive FIR filter is a 16-tap FIR filter, and the initial condition of this filter is \( h(n) = 0 \).

3. Assume an iteration number of two, the length of \( u \) has to be equal or greater than \( 2 + 16 - 1 = 17 \). Let \( u \) be a random signal and a vector of length 17, then
\[
u = [1, 2, 3, -1, 4, 7, -5, 3, 9, 20, -9, 1, -3, 8, 0, 2, 10]
\]

4. Since the iteration number is two, \( n = 0, 1 \). Let the fixed step size \( \mu \) be 0.05.

\( n = 0 \),
\[
d(0) = g(0)u^T(0) = (0.1*1+0.2*2+0.3*3+0.4*-1+...0.1*2) = 23.8
\]
\[
y(0) = h(0)u^T(0) = (0*1+0*2+0*3+0*-1+...0*2) = 0
\]
\[
e(0) = d(0) - y(0) = 23.8 - 0 = 23.8
\]
\[
h(1) = h(0) + \mu u(0)e(0)
\]
\[
= [0, 0, 0, ...] + 0.05*[1,2,3,-1,4,7,-5,3,9,20,-9,1,-3,8,0,2]*23.8
\]
\[
= [1.19, 2.38, 3.57, -1.19, 4.76, 8.33, -5.95, 3.57, 10.71, 23.8, -10.71, 1.19, -3.57, 9.52, 0.238]
\]

\( n = 1 \),
\[
d(1) = g(1)u^T(1) = (0.1*2+0.2*3+0.3*-1+0.4*4+...0.1*10) = 25.3
\]
\[
y(1) = h(1)u^T(1) = (1.19*2+2.38*3+3.57*-1+...-1.19*4+...2.38*10) = -11.9
\]
\[
e(1) = d(1) - y(1) = 25.3 + 11.9 = 37.2
\]
\[ h(2) = h(1) + \mu u(1)e(1) \]
\[ = [2.38, 4.76, \ldots, 4.76] + 0.1 \times [2, 3, -1, 4, 7, -5, 3, 9, 20, -9, 1, -3, 8, 0, 2, 10] \times 49.1 \] (4.15)
\[ = [4.91, 7.96, 1.71, 6.25, 17.78, -0.97, -0.37, 2031, 47.91, 7.06, -8.85, -4.39, 1131, 952, 3.72, 20.98] \]

5. Obviously the system does not converge within two iterations, so the iteration number and the step size \( \mu \) will be adjusted. The recursive process described above continues until the tap weights of the adaptive filter converge to the tap weights of the “unknown” system.

The results of the above example simulated with an iteration number of 500, and a fixed step size of 0.05, is shown in Figures 4-2 through 4-8. For detailed Matlab codes of simulations, please see the Appendix. Other simulations (not documented herein) show that a step size greater than 0.05 makes the system unstable, therefore, \( \mu = 0.05 \) is the largest step size that the system can use. Figure 4-2 shows the output of the adaptive FIR filter \( y(n) \) identifies with the “unknown” system \( d(n) \). From Figure 4-2, after a transition period about 100 iterations, \( y(n) \) converges to \( d(n) \), and the two signals overlap with each other. Figure 4-3 shows the estimation error \( e(n) \) vs. iterations. Figures 4-4 through 4-7 show each of the 16 coefficients of the adaptive filter converge to the “unknown” system \( g(n) \). Figure 4-8 shows the adaptive filter’s impulse response after 500 iterations, which identified with \( g(n) \).
Figure 4-2 The Desired Output $d(n)$ vs. the Filter Output $y(n)$
(Time Domain LMS Algorithm, $\mu = 0.05$)

Figure 4-3 The Estimated Error (Time Domain LMS Algorithm, $\mu = 0.05$)
Figure 4-4  Adaptive Filter Coefficients $h_0$, $h_1$, $h_2$, and $h_3$

Figure 4-5  Adaptive Filter Coefficients $h_4$, $h_5$, $h_6$, and $h_7$
Figure 4-6  Adaptive Filter Coefficients $h_8$, $h_9$, $h_{10}$, and $h_{11}$

Figure 4-7  Adaptive Filter Coefficients $h_{12}$, $h_{13}$, $h_{14}$, and $h_{15}$
4.2 System Identification Using the Wavelet Domain LMS Filtering Algorithm

Figure 4-9 shows the block diagram of a system identification model using wavelet domain LMS filtering algorithm. Note that the only difference between Figures 4-1 and 4-9 is the LMS algorithm portion. The wavelet domain transformation of $u(n)$ is $u_w(n)$ and is used as the adaptive filter input. As described in section 3.3.4, the adaptive filter is a wavelet transform of the time domain adaptive filter that is denoted by $w(n)$.
Figure 4-9  Block Diagram of a System Identification Model Using Wavelet Domain LMS Filtering Algorithm

The procedures of system identification using a wavelet domain LMS filtering algorithm are described below.

1. Determining the unknown system (same as section 4.1).

2. Choosing an FIR filter tap length for system identification.

   In this thesis, the adaptive filter tap length is chosen to be 16. Note that the adaptive filter is in the wavelet domain. Let \( w(n) \) be the wavelet domain adaptive filter, where the initial condition of this filter is

   \[
   w(n) = 0. \quad (4.16)
   \]

3. Defining an input \( u \) and assuming an iteration number.

   The input \( u \) is generated the same way as described in section 4-1, equation (4.3). Let \( n=500 \) be the iteration steps of the system identification model. Let 16
be the adaptive filter tap length. \( u(n) \) is determined as shown in equations (4.4) through (4.6).

The 2-Band Haar analysis filter banks are used to transform \( u(n) \) from the time domain to the wavelet domain. The Haar analysis filter banks have a low pass filter

\[
Haarh_0 = \left[ \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} \right]
\]

and a high pass filter \( Haarh_1 = \left[ -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right] \). Using the Haar analysis filter banks for the wavelet domain transformation, \( u_w(n) \) is defined as

\[
u_w(0) = \left[ Dec(\text{conv}(u(0), Haarh_0)), Dec(\text{conv}(u(0), Haarh_1)) \right] \tag{4.17}
\]

\[
u_w(499) = \left[ dec(\text{conv}(u(499), Haarh_0)), dec(\text{conv}(u(499), Haarh_1)) \right], \tag{4.18}
\]

where \( dec \) is the decimation of factor two, and \( \text{conv} \) is the time domain convolution.

4. Using wavelet domain LMS algorithm to update the adaptive filter tap weights and choosing a fixed step size \( \mu \) for the LMS algorithm.

From equation 3.26, the update equation for a wavelet domain LMS filtering algorithm is

\[
w(n+1) = w(n) + 2\mu R_w e(n)y(n), \quad n = 0, 1, 2, \ldots
\]

where the step size of the LMS algorithm and \( e(n) = d(n) - y(n) \) and \( R_w \) is the estimated autocorrelation matrix of the wavelet domain filter input \( u_w(n) \).

5. Recursively updating the adaptive filter tap weights using step 4 and adjusting
iteration steps if necessary.

6. Using Haar synthesis filter banks to convert the filter tap weights back to time domain.

Following the above 6 steps, an example of the wavelet domain LMS algorithm is given, where one iteration is shown.

1. The unknown system is a 16-tap FIR filter, where the impulse response is 
g(n)=[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1].

2. The adaptive FIR filter is a 16-tap FIR filter, where the initial condition of this filter is \( w(n) = 0 \).

3. Assume an iteration number of one, the length of \( u \) has to be equal or greater than \( 1 + 16 - 1 = 16 \). Let \( u \) be a random signal and \( u \) is a vector of length 16, \( u(0) = [1, 2, 3, -1, 4, 7, -5, 3, 9, 20, -9, 1, -3, 8, 0, 2] \). The time domain convolution of the low pass Haar filter and \( u(0) \) is

\[
\text{conv}(u(0), \text{Haar}_0) = [0.71, 2.12, 3.56, 1.41, 2.12, 7.78, 1.41, -1.41, 8.48, 20.5, 7.78, -5.66, -1.41, 3.54, 5.66, 1.41, 1.41]. \tag{4.19}
\]

The decimation of factor two of the equation (4.19) is

\[
\text{dec(conv}}(u(0), \text{Haar}_0) = [2.12, 1.41, 7.78, -1.41, 20.5, -5.66, 3.54, 1.41]. \tag{4.20}
\]

The time domain convolution of the high pass Haar filter and \( u(0) \) is

\[
\text{conv}(u(0), \text{Haar}_1) = [0.71, 0.71, 0.71, -2.83, 3.54, 2.12, -8.48, 5.66, 4.24, 7.78, -20.5, 7.07, -2.83, 7.78, -5.66, 1.41, -1.41]. \tag{4.21}
\]
The decimation of factor two of the equation (4.21) is

\[ \text{dec}(\text{conv}(u(0), Haar h_1)) = [0.71, -2.83, 2.12, 5.66, 7.78, 7.07, 7.78, 1.41]. \tag{4.22} \]

The wavelet domain input \( u_w(0) \) is the combination of equations (4.20) and (4.22),

\[ u_w(0) = [2.12, 1.41, 7.78, -1.41, 20.5, -5.66, 7.07, 7.78, 7.78, 1.41]. \tag{4.23} \]

4. Since the iteration number is one, \( n = 0 \). Let the fixed step size \( \mu \) be 0.05.

\[ n = 0, \]

\[ d(0) = g(0)u^T(0) = (0.1*1 + 0.2*2 + 0.3*3 + 0.4*(-1) + \ldots + 0.1*2) = 23.8 \tag{4.24} \]

\[ y(0) = w(0)u_w^T(0) = (0*1.12 + 0*1.41 + 0*7.78 + 0*(-1.41) + \ldots + 0*1.41) = 0 \tag{4.25} \]

\[ e(0) = d(0) - y(0) = 23.8 - 0 = 23.8 \tag{4.26} \]

\[ \hat{R_w}(0) = \hat{R}_w^T(0)\hat{R}_w(0) + (c / n)I \tag{4.27} \]

\[ w(n+1) = w(n) + 2\mu\hat{R}_w e(n)y(n), \quad n = 0, 1, 2, \ldots \tag{4.28} \]

Note that sometimes the estimate of \( \hat{R}_w \) is singular and the inverse cannot be determined. In order to make sure the estimated \( \hat{R}_w \) is non-singular, diagonal loading shown in equation (4.27) is used, where \( n \) is the number of iterations and \( c \) is a small positive constant. A value of \( c = 0.0001 \) is used for simulations.

5. The recursive process described above continues until the tap weights of the adaptive filter identify with the tap weights of the “unknown” system.
6. Convert the converged adaptive filter tap weights to the time domain using Haar synthesis filter banks. The low pass filter is $Haar f_0 = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right]$ and the high pass filter is $Haar f_1 = \left[-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right]$.

The results of the above example simulated with an iteration number of 500 and a fixed step size of 0.05 is shown in Figures 4-10 through 4-13. Figure 4-10 shows that the output of the adaptive FIR filter $y(n)$ converges to the desired signal $d(n)$. Figure 4-11 shows the estimated error of the wavelet domain LMS algorithm. Figure 4-12 shows the wavelet domain adaptive filter’s impulse response after 500 iterations. Figure 4-13 shows the time domain adaptive filter’s impulse response (converted from the wavelet domain back to the time domain) after 500 iterations, which identified with $g(n)$. The fixed step size of $\mu = 0.05$ was used for comparison purposes (compare with the time domain LMS algorithm results). Figures 4-14 through 4-18 show the maximum step size of $\mu = 0.3$ can be used for the wavelet domain LMS algorithm simulated in this section.
Figure 4-10  The Desired Output $d(n)$ vs. the Filter Output $y(n)$
(Wavelet Domain LMS Algorithm, $\mu = 0.05$)

Figure 4-11  The Estimated Error (Wavelet Domain LMS Algorithm, $\mu = 0.05$)
Figure 4-12  Wavelet Domain Filter Impulse Response After 500 Iterations (Wavelet Domain LMS Algorithm, $\mu = 0.05$)

Figure 4-13  Time Domain Filter Impulse Response After 500 Iterations (Wavelet Domain LMS Algorithm, $\mu = 0.05$)
Figure 4-14 The Desired Output $d(n)$ vs. the Filter Output $y(n)$ (Wavelet Domain LMS Algorithm, $\mu = 0.3$)

Figure 4-15 The Estimated Error (Wavelet Domain LMS Algorithm, $\mu = 0.3$)
Figure 4-16  Wavelet Domain Filter Impulse Response After 500 Iterations  
(Wavelet Domain LMS Algorithm, $\mu = 0.3$)

Figure 4-17  Time Domain Filter Impulse Response After 500 Iterations  
(Wavelet Domain LMS Algorithm, $\mu = 0.3$)
4.3 Comparisons and Discussions

The system identification used in this thesis proves that the wavelet domain LMS filtering algorithm converged properly. Figures 4-8, 4-13, and 4-17 demonstrated that like the time domain LMS algorithm, the wavelet domain LMS algorithm also successfully identified the "unknown" system. Figure 4-18 shows comparisons of MSE of an ensemble of 100 runs for both time domain and wavelet domain LMS algorithms simulated with $\mu = 0.05$. From Figure 4-18, the wavelet domain LMS algorithm shows a slower convergence rate. Is this a true statement? Further investigation demonstrates that the comparison between the time domain LMS algorithm with the wavelet domain algorithm using the same fixed step size is not valid. Different update equations are used and the estimation of the gradient of the cost are different for the time domain LMS algorithm and the wavelet domain algorithm. Therefore, for comparison purposes, the "best system performance" should be used. The convergence rate using the largest possible fixed step size before the system becomes unstable for each algorithm may be compared.

Figure 4-19 shows that the largest step size for the time domain is $\mu = 0.05$ and the largest step size for the wavelet domain is $\mu = 0.3$. Note that there is no improvement of convergence rate using the wavelet domain LMS algorithm in this case. This is because the input used was a random signal, where no correlation of
any kind exists between each inputs. Therefore, the prewhitening of input in the wavelet domain has no effect on the convergence rate.

Figure 4-18 Mean-Squared Error of an Ensemble of 100 Runs ($\mu = 0.05$)

Figure 4-19 Mean-Squared Error of an Ensemble of 100 Runs (Time domain LMS $\mu = 0.05$, Wavelet Domain $\mu = 0.3$)
5. Adaptive Differential Pulse Code Modulation (ADPCM)

Adaptive Differential Pulse-Code Modulation (ADPCM) is a generic data compression algorithm that is widely used in speech applications. The speech signal is compressed before it is transmitted to the receiver and is decompressed after it is received by the receiver. ADPCM uses an adaptive predictor and quantizer to account for sudden changes and the nonstationary nature of the speech signal.

ADPCM is a modified algorithm based on the pulse-code modulation (PCM) scheme. The PCM encoder typically consists of three processes: sampling, quantization, and coding. The sampling process converts a continuous-time signal into a discrete-time signal. The quantization process converts a continuous-amplitude signal into a discrete-amplitude signal. And the coding provides appropriate digital presentations for the samples of the discrete signal. Figure 5-1 shows a generic PCM encoder.

In a typical telephony system, the speech signal is generally coded with the PCM algorithm. A sample rate of 8 kHz and a nonlinear quantizer is used and the
quantized speech signal is coded into 8-bit words. The required bit rate for this system is 64 kbps.

Instead of transmitting the PCM waveform, the ADPCM transmits the quantized estimation error. The ADPCM encoders use an adaptive predictor to estimate the next audio sample from previous samples obtained from the input signal. This predicted sample is compared with the previous sample; the difference between the two is the estimation error. The estimation error is quantized using an adaptive quantizer before transmitting. When the adaptive predictor is optimized, the variance of the estimation error should be smaller than the variance of the input signal. Therefore, a smaller number of quantization levels is required for the estimation error than for the PCM waveform. A common implementation takes 16-bit PCM samples and converts them into 4-bit samples using ADPCM, giving a compression rate of 4:1. For a radio frequency (RF) transmitting system, a compression rate of 4:1 of bit rate yields a good 6 dB link margin improvement. The improvement of the link budget leads to less restricted hardware requirements (required transmitter power, antenna gain, etc.) and therefore, lower cost. Figures 5-2a and 5-2b show the simplified ADPCM encoder and decoder.
Figure 5-2a Generic ADPCM Encoder

Figure 5-2b Generic ADPCM Decoder
There are two types of adaptive quantizers, they are the feedforward and the feedback adaptive quantizers [9]. When the feedforward adaptive quantizer is used, the quantization step sizes, $\Delta(n)$, have to be transmitted together with the sample bits [9]. When the feedback adaptive quantizer is used, only the sample bits have to be transmitted. The Jayant quantizer that is used for simulations in this thesis is a feedback quantizer. Table 5-1 shows the Jayant quantizer [9].

Table 5-1 Jayant Quantizer

<table>
<thead>
<tr>
<th>No. Bits</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>M(1)</td>
<td>0.8</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>M(2)</td>
<td>1.6</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>M(3)</td>
<td></td>
<td>0.25</td>
<td>0.9</td>
</tr>
<tr>
<td>M(4)</td>
<td></td>
<td>1.7</td>
<td>0.9</td>
</tr>
<tr>
<td>M(5)</td>
<td></td>
<td></td>
<td>1.2</td>
</tr>
<tr>
<td>M(6)</td>
<td></td>
<td></td>
<td>1.6</td>
</tr>
<tr>
<td>M(7)</td>
<td></td>
<td></td>
<td>2.0</td>
</tr>
<tr>
<td>M(8)</td>
<td></td>
<td></td>
<td>2.4</td>
</tr>
</tbody>
</table>

The $M(\eta)$ are used to update the quantizer steps,

$$\Delta_{n+1} = \Delta_n M(\eta),$$

where

$$\eta = \text{floor} \left( \frac{|e(n)|}{\Delta_n} \right).$$

The initial quantizer step size was obtained using the bit length (b) and the full magnitude (Fs) of the signal,

$$\Delta_o = 2^{b-1} F_s.$$
5.1 ADPCM Using the Time Domain LMS Filtering Algorithm

Figure 5-3 shows the generic block diagram of a time domain ADPCM encoder/decoder. The quantized estimation error is used to control the adaptive algorithm of the system because the quantized estimation error is transmitted to the receiver.

The time domain ADPCM method consists of following procedures:

1. Defining an input signal. The ADPCM algorithm is tested on two signals for simulation purposes. The first signal is a 100 Hz sine wave with a sampling frequency of 8 kHz. This input signal is used to test the convergence rate the time domain ADPCM algorithm. The second signal is the same sine wave (frequency is 100 Hz, sampling frequency is 8 kHz) but with a sudden change after the signal converged. This second input signal is used to test the behavior.
of the time domain ADPCM algorithm in the event that a sudden input signal change occurs.

2. Picking FIR filter tap weights \( h(n) \) and setting initial conditions. A 16-tap FIR filter is used for simulations, the initial FIR filter coefficients are zeros.

3. Defining the quantizer initial step size using equation (5.3).

4. Letting the reconstructed signal \( \tilde{s}(n) \) be zeros (see Figure 5-3). The signal \( \tilde{s}(n) \) is the input of the adaptive FIR filter, therefore, the length of \( \tilde{s}(n) \) is the same as the FIR filter tap length. The updated \( \tilde{s}(n) \) are generated at each iteration of the algorithm.

5. Obtaining the filter output, \( \tilde{s}(n) \) estimated = \( \tilde{s}(n) \) \( \ast \) \( h^T(n) \).

6. Obtaining \( e(n) \), \( e(n) = s(n) - \tilde{s}(n) \) estimated.

7. Updating quantizer step using equations (5.1) and (5.2).

8. Obtaining the quantized estimation error \( \tilde{e}(n) \).

9. Obtaining the input of the FIR filter for next iteration, \( \tilde{s}(n) = \tilde{e}(n) + \tilde{s}(n) \) estimated.

10. Using LMS algorithm to update the adaptive filter tap weights and choosing a fixed step size \( \mu \) for the LMS algorithm. From equation 3.25, the update
equation for a standard time domain LMS filtering algorithm is
\[ h(n+1) = h(n) + \mu u(n)e(n), \quad n=0,1,2,..., \]
where \( \mu \) is the step size of the LMS algorithm and \( e(n) = d(n) - y(n) \). After many tests and iterations, the results show the best fixed step size \( \mu \) is 0.8 for the normal sine wave and 0.5 for the sine wave with a sudden change.

11. Repeat from step 5.

12. Calculate the signal to noise ratio (SNR) after the signal converges.

\[
\text{SNR} = 10 \log \left( \frac{\text{var(original signal)}}{\text{var(original signal} - \text{new signal})} \right) \\
= 10 \log \left( \frac{\text{var(original signal)}}{\text{var(noise)}} \right) \tag{5.4}
\]

Figure 5-4 shows results of the time domain ADPCM applied to a 100 Hz sine wave with a sampling frequency of 8 kHz. The original signal, the predicted signal, the reconstructed signal, and the estimation error are shown in Figure 5-4. From Figure 5-4, it can be seen that the estimation error converges to zero at approximately 300 iterations. The SNR of the reconstructed signal (44.04 dB) is 17.06 dB better than the SNR of the estimated signal (26.98 dB). The SNR is calculated after estimation error converges to zero, between approximately 400 to 500 iterations.
Figure 5-4 Time Domain ADPCM Applied to a Sine Wave

Figure 5-5, shows results of the time domain ADPCM applied to a 100 Hz sine wave with a sampling frequency of 8 kHz and a sudden change at approximately 500 iterations. The original signal, the estimated signal (estimated by the adaptive predictor), the reconstructed signal, and the estimation error are shown in Figure 5-5. From Figure 5-5, the estimation error converges to zero at approximately 300 iterations. A sudden input change occurred at about 500 iteration, the estimation error re-converges to zero at about 800 iterations. The SNR of the reconstructed signal (59.46 dB) is 19.43 dB better than the SNR of the estimated signal (40.03 dB). The SNR is calculated after the estimation error re-converges to about zero, between approximately 900 to 1000 iterations.
5.2 ADPCM Using the Wavelet Domain LMS Filtering Algorithm

Figure 5-6 shows the generic block diagram of a wavelet domain ADPCM encoder/decoder.

![Wavelet Domain ADPCM System Diagram]

Figure 5-6 Generic Wavelet Domain ADPCM System
The procedures for the wavelet domain ADPCM system are similar to the time domain ADPCM system procedures. The differences are that the input of the FIR filter is in the wavelet domain, and the update equation for the wavelet domain FIR filter is
\[ w(n+1) = w(n) + 2\mu R_w e(n)y(n), \quad n=0,1,2,..., \]
where \( \mu \) is the step size of the LMS algorithm and \( e(n) = d(n) - y(n) \) and \( R_w \) is the estimated autocorrelation matrix of the wavelet domain filter input \( u_w(n) \).

Figure 5-7 shows the results of the wavelet domain ADPCM applied to a 100 Hz sine wave with a sampling frequency of 8 kHz. The original signal, the estimated signal (estimated by the adaptive predictor), the reconstructed signal, and the estimation error are shown in Figure 5-7. From Figure 5-7, the estimation error converges to zero at approximately 100 iterations. The SNR of the reconstructed signal (126.09 dB) is 15.41 dB better than the SNR of the estimated signal (110.68 dB). The SNR is calculated after the estimation error converges to zero, between approximately 400 to 500 iterations.
Figure 5-7 Wavelet Domain ADPCM Applied to a Sine Wave

Figure 5-8 shows the results of the time domain ADPCM applied to a 100 Hz sine wave with a sampling frequency of 8 kHz and a sudden change at approximately 500 iterations. The original signal, the estimated signal (estimated by the adaptive predictor), the reconstructed signal, and the estimation error are shown in Figure 5-8. From Figure 5-8, the estimation error converges to zero at approximately 100 iterations. A sudden input change occurred at about 500 iteration, the estimation error re-converges to zero at about 550 iterations. The SNR of the reconstructed signal (165.54 dB) is 12.67 dB better than the SNR of the estimated signal (152.87 dB). The SNR is calculated after estimation error re-converges to zero, between approximately 900 to 1000 iterations.
5.3 Comparisons and Discussions

Figures 5-9 and 5-10 show the MSE of an ensemble of 50 runs of the time and wavelet domain ADPCM applied to a sine wave and a sine wave with a sudden change. Figures 5-9 and 5-10 show the wavelet domain LMS algorithm converges at least 100 iterations faster than the time domain LMS algorithm. According to results shown in section 5.2, the wavelet domain algorithm also provides a better SNR than the time domain algorithm. The drawback of the wavelet domain algorithm is the computational complexity. Reference [1] contains a proposed algorithm to minimize the computational complexity.
Figure 5-9 MSE of an Ensemble of 50 Runs (Input is a Sine Wave)

Figure 5-10 MSE of Ensemble of 50 Runs (Sine Wave with Sudden Change)
6. Conclusions

The simulations in chapters 4 and 5 show that the wavelet domain LMS filtering algorithm can be used for system identification models and also for the ADPCM encoding/decoding. Lessons learned from performing simulations are the following:

1. The convergence rate depends on the nature of the input signal, the fixed step size, and the adaptive FIR filter length.

2. The diagonal loading for the autocorrelation of the input signal has to be relatively small (the value varies with the nature of the input), otherwise, the ability to converge will be affected.

3. The wavelet domain LMS filtering algorithm provides a faster convergence rate when the components of the input are correlated.

4. A faster convergence rate is desirable for the ADPCM applications. For example, during a real time signal transmission, if a sudden input change occurs, the adaptive filter has to be re-optimized. An algorithm with a faster convergence rate provides better data quality and less risk of data loss.

5. There is not an “absolute” superior algorithm for signal processing. The algorithms should be evaluated and selected depending on different applications, hardware/software design and cost factors.
The advantage of the standard time domain LMS filtering algorithm is its simplicity. But the convergence rate of the standard LMS algorithm depends heavily on the input characteristics and therefore the designer has very little control over the convergence rate. The advantage of the wavelet domain LMS algorithm is its faster convergence rate. In addition, the autocorrelation matrix of the input in wavelet domain has a sparse structure [1], which can be used to minimize computational complexity.
APPENDIX

The Appendix contains Matlab codes used for simulations. The Matlab codes are arranged based on the order of plots that appeared in the thesis.

System Identification with Time Domain LMS Filtering Algorithm

% Time Domain LMS Adaptive Filtering Algorithm
% Identify with a 16-tap FIR Filter

clear all;
u=0.05; % Selecting the fixed step size for the LMS algorithm
n=200; % Setting the iteration length
S=n+16;
X=randn(S); % Generating input signal
X=X(l,:);
h=[0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1];
% h is the system that the adaptive FIR filter will converge to

% Initial conditions
w(l,:)=zeros(1,16); % w is the impulse response of the adaptive FIR filter

for i=1:n; % Starting the recursive LMS algorithm
x=X(i:i+15);
x=fliplr(x);
d(i)=x*h';
y(i)=x*w(i,:');
e(i)=d(i)-y(i);
w(i+1,:)=w(i,:)+u*e(i)*x; % Updating the FIR filter coefficients
end;

N=1:n;
figure; % Plotting the desired signal vs. output of the adaptive FIR filter
plot(N,d,'-m',N,y,'--b');
xlabel('Number of Iterations');
ylabel('d (-) vs. y (--)');
grid;
figure;  % Plotting the system error
for k=1:n;
    error(k)=e(k)^2;
end;
plot(error);
xlabel('Number of Iterations');
ylabel('Error');
grid;

figure; % Plotting the impulse response of the adaptive FIR filter after convergence
stem(w(501,:));
xlabel('Number of Iterations');
ylabel('Impulse Response of The FIR Adaptive Filter');
grid;

figure;  % Plotting System parameters
subplot(4,1,1);
plot(w(:,1));
ylabel('h0');
grid;
subplot(4,1,2);
plot(w(:,2));
ylabel('h1');
grid;
subplot(4,1,3);
plot(w(:,3));
ylabel('h2');
grid;
subplot(4,1,4);
plot(w(:,4));
xlabel('Number of Iterations');
ylabel('h3');
grid;
```matlab
figure;  % Plotting System parameters
subplot(4,1,1);
plot(w(:,5));
ylabel('h4');
grid;
subplot(4,1,2);
plot(w(:,6));
ylabel('h5');
grid;
subplot(4,1,3);
plot(w(:,7));
ylabel('h6');
grid;
subplot(4,1,4);
plot(w(:,8));
xlabel('Number of Iterations');
ylabel('h7');
grid;

figure;  % Plotting System parameters
subplot(4,1,1);
plot(w(:,9));
ylabel('h8');
grid;
subplot(4,1,2);
plot(w(:,10));
ylabel('h9');
grid;
subplot(4,1,3);
plot(w(:,11));
ylabel('h10');
grid;
subplot(4,1,4);
plot(w(:,12));
xlabel('Number of Iterations');
ylabel('h11');
grid;
```
figure; %Plotting System parameters
subplot(4,1,1);
plot(w(:,13));
ylabel('h12');
grid;
subplot(4,1,2);
plot(w(:,14));
ylabel('h13');
grid;
subplot(4,1,3);
plot(w(:,15));
ylabel('h14');
grid;
subplot(4,1,4);
plot(w(:,16));
xlabel('Number of Iterations');
ylabel('h15');
grid;
System Identification with Wavelet Domain LMS Filtering Algorithm

% Wavelet Domain LMS Adaptive Filtering Algorithm
% Identify with a 16-tap FIR Filter

clear all;
u=0.3; % Selecting the fixed step size for the LMS algorithm
n=500; % Setting the iteration length
S=n+50;
X=randn(S); % Generating input signal
X=X(1,:);
h=[0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1];
% h is the system that the adaptive FIR filter will converge to

% Initial conditions
w(1,:)=zeros(1,16);

for i=1:n; % Starting the recursive LMS algorithm
x=X(i:i+15);
x=fliplr(x);
d(i)=x*h';

[A11,D11]=dwt(x,'haar'); % Wavelet decomposition of the input signal
xdwt=[A11 D11];

y(i)=xdwt*w(i,:);
e(i)=d(i)-y(i);
R=xdwt'*xdwt; % Computing the autocorrelation matrix of the input signal
RR=R+(0.0001/i)*eye(length(R)); % Diagonal Loading
Rinv=inv(RR);
g=-2*e(i)*xdwt*Rinv;
w(i+1,:)=w(i,:)-u*g; % Updating the FIR filter coefficients
end;

N=1:n;
figure; % Plotting the desired signal vs. output of the adaptive FIR filter
plot(N,d,'-m',N,y,'--b');
xlabel('Number of Iterations');
ylabel('d (-) vs. y (--)');
grid;
figure; % Plotting the system error
for k=1:n;
    error(k)=e(k)^2;
end;
plot(error);
xlabel('Number of Iterations');
ylabel('Error');
grid;

figure; % Plotting the impulse response of the wavelet domain adaptive FIR filter after convergence
stem(w(501,:));
xlabel('Number of Iterations');
ylabel('DWT Filter Impulse Response');
grid;

x=w(501,:);
CA=x(1:8);
CD=x(9:16);
ww=idwt(CA,CD,'haar'); % Convert the wavelet domain FIR filter coefficients to time domain
figure; % Plotting the time domain FIR impulse response
stem(ww);
xlabel('Number of Iterations');
ylabel('Time Domain Filter Impulse Response');
grid;
MSE of Ensemble of 100 Runs, Time/Wavelet Domain System ID

% MSE of Ensemble of 100 Runs, Time Domain System Identification
% Identify with a 16-tap FIR Filter, u=0.05

clear all;
for k=1:100
u=0.05;
N=500;
S=N+50;
X=randn(S);
X=X(1,:);
h=[0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1];

% Initial conditions
w(1,:)=zeros(1,16);

for i=1:N;
x=X(i:i+15);
x=fliplr(x);
d(i)=x*h';
y(i)=x*w(i,:);'
e(i)=d(i)-y(i);
w(i+1,:)=w(i,:)+u*e(i)*x;
end;
e(e(k,:))=e;
end;

[a b]=size(e);
for i=1:a
for j=1:b
    e2(i,j)=ee(i,j)^2;
end;
end;

Esquare=mean(e2);
save timesysid;
clear all;
for k=1:100
    u=0.3;
    n=500;
    S=n+50;
    X=randn(S);
    X=X(1,:);
    h=[0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1];

    % Initial conditions
    w(1,:)=zeros(1,16);

    for i=1:n;
        x=X(i:i+15);
        x=fliplr(x);
        d(i)=x*h';
        [A1,D1]=dwt(x,'haar'); % 1st level
        xdtw=[A1 D1];
        y(i)=xdtw*w(i,:);
        e(i)=d(i)-y(i);
        R=xdtw*xdtw;
        RR=R+(0.0001/i)*eye(length(R));
        Rinv=inv(RR);
        g=-2*e(i)*xdtw*Rinv;
        w(i+1,:)=w(i,:)-u*g;
    end;
    ee(k,:)=e;
end;

[a b]=size(ee);
for i=1:a
    for j=1:b
        e2(i,j)=ee(i,j)^2;
    end;
end;
Esquare=mean(e2);
save dwtsysid1;

% MSE of Ensemble of 100 Runs, Wavelet Domain System Identification
% Identify with a 16-tap FIR Filter, u=0.3

clear all;
for k=1:100
u=0.3;
n=500;
S=n+50;
X=randn(S);
X=X(1:);
h=[0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1];

% Initial conditions
w(1,:)=zeros(1,16);

for i=1:n;
x=X(i:i+15);
x=fliplr(x);
d(i)=x*h';
[A1,D1]=dwt(x,'haar'); % 1st level
xdwt=[A1 D1];
y(i)=xdwt*w(i,:);'
e(i)=d(i)-y(i);
R=xdwt*xdwt;
RR=R+(0.0001/i)*eye(length(R));
Rinv=inv(RR);
g=-2*e(i)*xdwt*Rinv;
w(i+1,:)=w(i,:)-u*g;
end;
ee(k,:)=e;
end;

[a b]=size(ee);
for i=1:a
for j=1:b
    e2(i,j)=ee(i,j)^2;
end;
end;

Esquare=mean(e2);
save dwtsysid2;

% Plot MSE of Ensemble of 100 Runs, Time/Wavelet Domain System Identification

clear all;

load timesysid;
time=Esquare;
load dwtsysid1;
dwt1=Esquare;
n=1:500;
figure;
plot(n,time,'-m',n,dwt1,'--b');
xlabel('Number of Iterations');
ylabel('Time Domain LMS (-) vs. Wavelet Domain LMS(--)');
grid;

load dwtsysid2;
dwt2=Esquare;
n=1:500;
figure;
plot(n,time,'-m',n,dwt2,'--b');
xlabel('Number of Iterations');
ylabel('Time Domain LMS (-) vs. Wavelet Domain LMS(--)');
grid;

75
ADPCM with Time Domain LMS Filtering Algorithm
(Input is a sine wave)

% Adaptive Differential Pulse Code Modulation (ADPCM)
% Time Domain LMS Algorithm applied to a sinusoidal input

clear all;

% Generate a sinusoidal signal
Fs = 8000;
f = 100;
t = 0:1/Fs:0.4;
s = sin(2*pi*f*t);

% Jayant Quatizer (4-bit)
b = 4;
do = (2^-(b-1)) * 1;
M = [0.9 0.9 0.9 0.9 1.2 1.6 2.0 2.4];
delta(1) = do;

% LMS Algorithm FIR Filter
n = 500;
u = 0.8;
tap = 16;
previous = zeros(1, tap);
wf = zeros(size(1:tap));
estimateqs(1) = previous * wf;

% 4-bit Adaptive standard Jayant quantizer is applied to s(n)
% LMS Algorithm is applied
for i = 1:n
    e(i) = s(i+tap) - estimateqs(i);
    qe(i) = floor(e(i)/delta(i))*delta(i);
    for j = 2:tap
        qslms(j) = previous(j-1);
    end;
    qslms(1) = estimateqs(i) + qe(i);
    previous = qslms;
end;
reconstruct(i)=estimateqs(i)+qe(i);
qsllms=qsllms-mean(qsllms);
m(i)=mean(qsllms);
w(i+1,:)=w(i,:) + u*qe(i)*qsllms;
estimateqs(i+1)=qsllms*w(i+1,:);
k=floor(abs(qe(i))/delta(i));
if k>8
    k=8;
elseif k==0
    k=1;
end;
delta(i+1)=delta(i)*M(k);
end;
reconstruct=reconstruct+m;

figure;
snplt(411);
plot(s(tap+1:tap+n));
ylabel('s(n)');
title('ADPCM (Time Domain LMS Algorithm, Tap=16, u=0.8, SNR=47.51 dB, SNRQ=66.48 dB)');
grid;
snplt(412);
plot(estimateqs(1:n));
ylabel('Estimated s-(n)');
grid;
snplt(413);
plot(reconstruct(1:n));
ylabel('s-(n)');
grid;
snplt(414);
plot(qe(1:n));
xlabel('Number of Samples');
ylabel('e-(n)');
grid;

SNRQ=10*log10(var(s(tap+400:n+tap))/var(reconstruct(tap+400:n+tap))-s(tap+400:n+tap))
SNR=10*log10(var(s(tap+400:n+tap))/var(estimateqs(tap+400:n+tap))-s(tap+400:n+tap))
ADPCM with Time Domain LMS Filtering Algorithm
(Input is a sine wave with a sudden change)

% Adaptive Differential Pulse Code Modulation (ADPCM)
% Time Domain LMS Algorithm applied to a sinusoidal input with a sudden change
clear all;
% Generate a sinusoidal signal
Fs=8000;
f=100;
t=0:1/Fs:0.4;
s=sin(2*pi*f*t);
s(516)=-0.2;
s(517)=-0.3;
s(518)=-0.5;

% Jayant Quatizer (4-bit)
b=4;
do=(2^-(b-1))*1;
M=[0.9 0.9 0.9 0.9 1.2 1.6 2.0 2.4];
delta(1)=do;

% LMS Algorithm FIR Filter
n=1000;
u=0.5;
tap=16;
previous=zeros(1,tap);
wf=zeros(size(1:tap));
estimates(1)=previous*wf;

% 4-bit Adaptive standard Jayant quantizer is applied to s(n)
% LMS Algorithm is applied
for i=1:n
    e(i)=s(i+tap)-estimates(i);
    qe(i)=floor(e(i)/delta(i))*delta(i);
    for j=2:tap
        qslms(j)=previous(j-1);
    end;
    qslms(1)=estimates(i)+qe(i);
end;
previous=qslms;
reconstruct(i)=estimateqs(i)+qe(i);
qslms=qslms-mean(qslms);
m(i)=mean(qslms);
wfi+1,:)\rightarrow wfi,:)\rightarrow u*qe(i)\rightarrow qslms;
estimateqs(i+1)=qslms*wf(i+1,:);
k=\text{floor}(abs(qe(i))/delta(i));
if k>8
k=8;
elseif k==0
k=1;
end;
delta(i+1)=delta(i)*M(k);
end;
reconstruct=reconstruct+m;

figure;
subplot(411);
plot(s(tap+1:tap+n));
ylabel('s(n)');
title('ADPCM (Time Domain LMS Algorithm, Tap=16, u=0.5, SNR=40.03 dB, SNRQ=59.46 dB)');
grid;
subplot(412);
plot(estimateqs(1:n));
ylabel('Estimated s~(n)');
grid;
subplot(413);
plot(reconstruct(1:n));
ylabel('s~(n)');
grid;
subplot(414);
plot(qe(1:n));
xlabel('Number of Samples');
ylabel('e~(n)');
grid;
SNRQ=10*log10(var(s(1+tap+900:n+tap))/var(reconstruct(1+900:n)-s(1+tap+900:n+tap)))
SNR=10*log10(var(s(1+tap+900:n+tap))/var(estimateqs(1+900:n)-s(1+tap+900:n+tap)))

79
ADPCM with Wavelet Domain LMS Filtering Algorithm
(Input is a sine wave)

% Adaptive Differential Pulse Code Modulation (ADPCM)
% DWT Domain LMS Algorithm applied to a sinusoidal signal
clear all;

% Generate a sinusoidal signal
Fs=8000;
f=100;
t=0:1/Fs:0.4;
s=sin(2*pi*f*t);

% Jayant Quatizer (4-bit)
b=4;
do=(2^-(b-1))*1;
M=[0.9 0.9 0.9 0.9 1.2 1.6 2.0 2.4];
delta(1)=do;

% LMS Algorithm FIR Filter
n=500;
u=0.6;
tap=16;
previous=zeros(1,tap);
w=zeros(size(1:tap));
estimateqs(1)=previous*w';

% 4-bit Adaptive standard Jayant quantizer is applied to s(n)
% LMS Algorithm is applied
for i=1:n
  e(i)=s(i+tap)-estimateqs(i);
  qe(i)=floor(e(i)/delta(i))*delta(i);
  for j=2:tap
    qslms(j)=previous(j-1);
  end;
  qslms(1)=estimateqs(i)+qe(i);
  reconstruct(i)=estimateqs(i)+qe(i);
previous=qslms;
qslms=qslms-mean(qslms);
m(i)=mean(qslms);

% wdt input of the FIR filter input qslms

[A1,D1]=dwt(qslms,'dB6'); % 1st level
A11=A1(6:length(A1)); % Low frequency component of 1st level
D11=D1(1:length(D1)-5); % high frequency component of 1st level

[A2,D2]=dwt(A11,'dB3'); % 2nd level
A22=A2(3:length(A2));
D22=D2(1:length(D2)-2);
qsdwt=[A22 D22 D11]; % wdt LMS FIR filter input

R=qsdwt'*qsdwt;
R=R+(0.0001/i)*eye(tap);
Rinv=inv(R);

g=-2*e(i)*qsdwt*Rinv;
w(i+1,:)w(i,:)-u*g;

estimateqs(i+1)=qsdwt*w(i+1,:);

k=floor(abs(qe(i))/delta(i));
if k>8
k=8;
elseif k==0
k=1;
end;
delta(i+1)=delta(i)*M(k);
end;
reconcrut=reconstruct+m;

figure;
subplot(411);
plot(s(1+tap:n+tap));
ylabel('s(n);
title('ADPCM (DWT Domain LMS Algorithm, Tap=16, u=0.6, SNRQ=110.68 dB, SNR=126.09 dB));
grid;
subplot(412);
plot(estimateqs(1:n));
ylabel('Estimated s-(n)');
grid;
subplot(413);
plot(reconstruct(1:n));
ylabel('s-(n)');
grid;
subplot(414);
plot(qe(1:n));
xlabel('Number of Samples');
ylabel('e-(n)');
grid;

SNRQ=10*log10(var(s(1+tap+400:n+tap))/var(reconstruct(1+400:n)-s(1+tap+400:n+tap)))
SNR=10*log10(var(s(1+tap+400:n+tap))/var(estimateqs(1+400:n)-s(1+tap+400:n+tap)))
ADPCM with Wavelet Domain LMS Filtering Algorithm
(Input is a sine wave with a sudden change)

% Adaptive Differential Pulse Code Modulation (ADPCM)
% DWT Domain LMS Algorithm, input is a sine wave with a sudden change
clear all;

% Generate a sinusoidal signal
Fs=8000;
f=100;
t=0:1/Fs:0.4;
s=sin(2*pi*f*t);
s(516)=-0.2;
s(517)=-0.3;
s(518)=-0.5;

% Jayant Quatizer (4-bit)
b=4;
do=(2^-(b-1))*1;
M=[0.9 0.9 0.9 0.9 1.2 1.6 2.0 2.4];
delta(1)=do;

% LMS Algorithm FIR Filter
n=1000;
u=0.5;
tap=16;
previous=zeros(1,tap);
w=zeros(size(1:tap));
estimateqs(1)=previous*w';

% 4-bit Adaptive standard Jayant quantizer is applied to s(n)
% LMS Algorithm is applied
for i=1:n
    e(i)=s(i+tap)-estimateqs(i);
    qe(i)=floor(e(i)/delta(i))*delta(i);
    for j=2:tap
        qslms(j)=previous(j-1);
    end;
end;
qslms(1)=estimateqs(i)+qe(i);
reconstruct(i)=estimateqs(i)+qe(i);
previous=qslms;
qslms=qslms-mean(qslms);
m(i)=mean(qslms);

%wtd input of the FIR filter input qslms
[A1,D1]=dwt(qslms,'dB6');  % 1st level
A11=A1(6:length(A1));  % Low frequency component of 1st level
D11=D1(1:length(D1)-5);  % high frequency component of 1st level
[A2,D2]=dwt(A11,'dB3');  % 2nd level
A22=A2(3:length(A2));
D22=D2(1:length(D2)-2);
qsdwt=[A22 D22 D11];  % wdt LMS FIR filter input

R=qsdwt'*qsdwt;
R=R+(0.0001/i)*eye(tap);
Rinv=inv(R);

g=-2*e(i)*qsdwt*Rinv;
w(i+1,:)=w(i,:)-u*g;

estimateqs(i+1)=qsdwt*w(i+1,:);

k=floor(abs(qe(i))/delta(i));
if k>8
  k=8;
elseif k==0
  k=1;
end;
delta(i+1)=delta(i)*M(k);
end;
reconstruct=reconstruct+m;

figure;
subplot(411);
plot(s(1+tap:n+tap));
ylabel('s(n)');
title('ADPCM (DWT Domain LMS Algorithm, Tap=16, u=0.5, SNRQ=165.54, SNR=152.87)');
grid;
subplot(412);
plot(estimateqs(1:n));
ylabel('Estimated s-(n)');
grid;
subplot(413);
plot(reconstruct(1:n));
ylabel('s-(n)');
grid;
subplot(414);
plot(qe(1:n));
xlabel('Number of Samples');
ylabel('e-(n)');
grid;

$$SNRQ = 10 \times \log_{10} \left( \frac{\text{var}(s(1+tap+900:n+tap))}{\text{var}(\text{reconstruct}(1+900:n)-s(1+tap+900:n+tap))} \right)$$

$$SNR = 10 \times \log_{10} \left( \frac{\text{var}(s(1+tap+900:n+tap))}{\text{var}(\text{estimateqs}(1+900:n)-s(1+tap+900:n+tap))} \right)$$
MSE of Ensemble of 50 Runs, Time/Wavelet Domain ADPCM
(Input is a sine wave)

% MSE of an Ensemble of 50 runs
% Time Domain Adaptive Differential Pulse Code Modulation (ADPCM)
% LMS Algorithm applied to a sinusoidal input

clear all;
for k=1:50
  % Generate a sinusoidal signal
  Fs=8000;
f=100;
t=0:1/Fs:0.4;
s=sin(2*pi*f*t);
  
  % Jayant Quantizer (4-bit)
b=4;
do=(2^-(b-1))*1;
M=[0.9 0.9 0.9 0.9 1.2 1.6 2.0 2.4];
delta(1)=do;
  
  % LMS Algorithm FIR Filter
  n=500;
u=0.8;
tap=16;
previous=zeros(1,tap);
wf=zeros(size(1:tap));
estimateqs(1)=previous*wf;
  
  % 4-bit Adaptive standard Jayant quantizer is applied to s(n)
% LMS Algorithm is applied
for i=1:n
  e(i)=s(i+tap)-estimateqs(i);
  qe(i)=floor(e(i)/delta(i))*delta(i);
  for j=2:tap
    qslms(j)=previous(j-1);
  end;
  qslms(1)=estimateqs(i)+qe(i);
end;
previous=qslms;
reconstruct(i)=estimateqs(i)+qe(i);
qslms=qslms-mean(qslms);
m(i)=mean(qslms);
wf(i+1,:)=wf(i,:)+u*qe(i)*qslms;
estimateqs(i+1)=qslms*wf(i+1,:);’;
k=floor(abs(qe(i))/delta(i));
if k>8
  k=8;
elseif k==0
  k=1;
end;
  delta(i+1)=delta(i)*M(k);
end;
reconstruct=reconstruct+m;
eq(k,:)=qe;
end;
[a b]=size(ee);
for i=1:a
  for j=1:b
    e2(i,j)=ee(i,j)^2;
  end;
end;

Esquare=mean(e2);
save timeMSE;

% Adaptive Differential Pulse Code Modulation (ADPCM)
% DWT Domain LMS Algorithm
clear all;
for kk=1:50

% Generate a sinusoidal signal
Fs=8000;
f=100;
t=0:1/Fs:0.4;
s=sin(2*pi*f*t);
% Jayant Quatizer (4-bit)
b=4;
do=(2^-(b-1))*1;
M=[0.9 0.9 0.9 0.9 1.2 1.6 2.0 2.4];
delta(1)=do;

% LMS Algorithm FIR Filter
n=500;
u=0.6;
tap=16;
previous=zeros(1,tap);
w=zeros(size(1:tap));
estimateqs(1)=previous*w';

% 4-bit Adaptive standard Jayant quantizer is applied to s(n)
% LMS Algorithm is applied
for i=1:n
    e(i)=s(i+tap)-estimateqs(i);
    qe(i)=floor(e(i)/delta(i))*delta(i);
    for j=2:tap
        qslms(j)=previous(j-1);
    end;
    qslms(1)=estimateqs(i)+qe(i);
    reconstruct(i)=estimateqs(i)+qe(i);
    previous=qslms;
    qslms=qslms-mean(qslms);
    m(i)=mean(qslms);
end;

% wdt input of the FIR filter input qslms
[A1,D1]=dwt(qslms,'dB6'); % 1st level
A11=A1(6:length(A1)); % Low frequency component of 1st level
D11=D1(1:length(D1)-5); % high frequency component of 1st level
[A2,D2]=dwt(A11,'dB3'); % 2nd level
A22=A2(3:length(A2));
D22=D2(1:length(D2)-2);
qsdwt=[A22 D22 D11]; % wdt LMS FIR filter input
R=qsdwt*qsdwt;
R=R+(0.0001/i)*eye(tap);
Rinv=inv(R);
g=-2*e(i)*qsdwt*Rinv;
w(i+1,:)=w(i,:)-u*g;
estimates(i+1)=qsdwt*w(i+1,:);
k=floor(abs(qe(i))/delta(i));
if k>8
    k=8;
elseif k==0
    k=1;
end;
delta(i+1)=delta(i)*M(k);
end;
reconstruct=reconstruct+m;

[a b]=size(ee);
for i=1:a
    for j=1:b
        e2(i,j)=ee(i,j)^2;
    end;
end;
Esquare=mean(e2);
save dwtMSE;
clear all;
load timeMSE;
timeE=Esquare;
load dwtMSE;
dwtE=Esquare;
N=1:length(timeE);
figure;
plot(N,timeE,'-m',N,dwtE,'--b');
xlabel('Number of Iterations');
ylabel('Time Domain ADPCM MSE (-) vs. Wavelet Domain ADPCM MSE (--);');
title('MSE of an Ensemble of 50 Runs (ADPCM, Time/Wavelet Domain LMS)');
grid;
MSE of Ensemble of 50 Runs, Time/Wavelet Domain ADPCM
(Input is a sine wave with a sudden change)

% Adaptive Differential Pulse Code Modulation (ADPCM)
% LMS Algorithm applied to a sinusoidal input with a sudden change

clear all;
for kk=1:50
% Generate a sinusoidal signal
Fs=8000;
f=100;
t=0:1/Fs:0.4;
s=sin(2*pi*f*t);
s(516)=-0.2;
s(517)=-0.3;
s(518)=-0.5;

% Jayant Quatizer (4-bit)
b=4;
do=(2^-(b-1))'*1;
M=[0.9 0.9 0.9 0.9 1.2 1.6 2.0 2.4];
delta(1)=do;

% LMS Algorithm FIR Filter
n=1000;
u=0.5;
tap=16;
previous=zeros(1,tap);
wf=zeros(size(1:tap));
estimateqs(1)=previous*wf;

% 4-bit Adaptive standard Jayant quantizer is applied to s(n)
% LMS Algorithm is applied
for i=1:n
    e(i)=s(i+tap)-estimateqs(i);
    qe(i)=floor(e(i)/delta(i))*delta(i);
    for j=2:tap
        qslms(j)=previous(j-1);
        end;
end;
qslms(1)=estimateqs(i)+qe(i);
previous=qslms;
reconstruct(i)=estimateqs(i)+qe(i);
qslms=qslms-mean(qslms);
m(i)=mean(qslms);
wf(i+1,:)=wf(i,:)+u*qe(i)*qslms;
estimateqs(i+1)=qslms*wf(i+1,:);
k=floor(abs(qe(i))/delta(i));
if k>8
  k=8;
elseif k==0
  k=1;
end;
delta(i+1)=delta(i)*M(k);
end;
reconstruct=reconstruct+m;
e(e,kk,:)=qe;
end;

[a b]=size(e);
for i=1:a
  for j=1:b
    e2(i,j)=ee(i,j)^2;
  end;
end;
Esquare=mean(e2);
save timeMSE;

% Adaptive Differential Pulse Code Modulation (ADPCM)
% DWT Domain LMS Algorithm
clear all;
for kk=1:50
  % Generate a sinusoidal signal
  Fs=8000;
  f=100;
  t=0:1/Fs:0.4;
  s=sin(2*pi*f*t);
  s(516)=-0.2;
\[ s(517) = -0.3; \]
\[ s(518) = -0.5; \]

% Jayant Quatizer (4-bit)
\[ b = 4; \]
\[ do = (2^{-(b-1)})^*1; \]
\[ M = [0.9 \ 0.9 \ 0.9 \ 0.9 \ 1.2 \ 1.6 \ 2.0 \ 2.4]; \]
\[ \text{delta(1)} = \text{do}; \]

% LMS Algorithm FIR Filter
\[ n = 1000; \]
\[ u = 0.3; \]
\[ \text{tap} = 16; \]
\[ \text{previous} = \text{zeros(1, tap)}; \]
\[ w = \text{zeros(size(1:tap))}; \]
\[ \text{estimateqs(1)} = \text{previous}^*w^\prime; \]

% 4-bit Adaptive standard Jayant quantizer is applied to s(n)
% LMS Algorithm is applied
\[
\text{for } i = 1:n
\text{e(i)} = s(i+\text{tap}) - \text{estimateqs(i)};
\text{qe(i)} = \text{floor}(\text{e(i)}/\text{delta(i)})^*\text{delta(i)};
\text{for } j = 2:\text{tap}
\text{qslms(j)} = \text{previous(j-1)};
\text{end};
\text{qslms(1)} = \text{estimateqs(i)} + \text{qe(i)};
\text{reconstruct(i)} = \text{estimateqs(i)} + \text{qe(i)};
\text{previous} = \text{qslms};
\text{qslms} = \text{qslms} - \text{mean(qslms)};
\text{m(i)} = \text{mean(qslms)};
\text{% wdt input of the FIR filter input qslms}
\text{[A1, D1]} = \text{dwt(qslms, 'dB6')}; % 1st level
\text{A11} = \text{A1}(6:\text{length(A1)}); % Low frequency component of 1st level
\text{D11} = \text{D1}(1:\text{length(D1)}-5); % high frequency component of 1st level
\text{[A2, D2]} = \text{dwt(A11, 'dB3')}; % 2nd level
\text{A22} = \text{A2}(3:\text{length(A2)});
\text{D22} = \text{D2}(1:\text{length(D2)}-2);
\text{qsdwt} = [\text{A22} \ \text{D22} \ \text{D11}]; % wdt LMS FIR filter input
\[ R = qsdwt^*qsdwt; \]
\[ R = R + (0.0001/i) \cdot \text{eye}(\text{tap}); \]
\[ Rinv = \text{inv}(R); \]
\[ g = -2 \cdot e(i) \cdot qsdwt \cdot Rinv; \]
\[ w(i+1,:) = w(i,:) - u \cdot g; \]
\[ \text{estimateqs}(i+1) = qsdwt \cdot w(i+1,:); \]
\[ k = \text{floor}(\text{abs}(qe(i))/\text{delta}(i)); \]
\[ \text{if } k > 8 \]
\[ k = 8; \]
\[ \text{elseif } k = 0 \]
\[ k = 1; \]
\[ \text{end}; \]
\[ \text{delta}(i+1) = \text{delta}(i) \cdot M(k); \]
\[ \text{end}; \]
\[ \text{reconstruct} = \text{reconstrcut} + m; \]
\[ \text{ee}(kk,:) = qe; \]
\[ \text{end}; \]
\[
[a \ b] = \text{size}(\text{ee});
\]
\[ \text{for } i = 1:a \]
\[ \quad \text{for } j = 1:b \]
\[ \quad \quad e2(i,j) = \text{ee}(i,j)^2; \]
\[ \quad \text{end}; \]
\[ \text{end}; \]
\[ \text{Esquare} = \text{mean}(e2); \]
\[ \text{save dwtMSE}; \]
\[ \text{clear all}; \]
\[ \text{load timeMSE}; \]
\[ \text{timeE} = \text{Esquare}; \]
\[ \text{load dwtMSE}; \]
\[ \text{dwtE} = \text{Esquare}; \]
\[ N = 1: \text{length}(\text{timeE}); \]
\[ \text{figure}; \]
\[ \text{plot}(N,\text{timeE},'-m',N,\text{dwtE},'--b'); \]
\[ \text{xlabel('Number of Iterations')}; \]
\[ \text{ylabel('Time Domain ADPCM MSE (-) vs. Wavelet Domain ADPCM MSE (--))';} \]
\[ \text{title('MSE of an Ensemble of 50 Runs (ADPCM, Time/Wavelet Domain LMS)');} \]
\[ \text{grid}; \]
REFERENCES


