UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

HIRSCHMAN OPTIMAL TRANSFORM LEAST MEAN SQUARE ADAPTIVE FILTERS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

By

OSAMA M. ALKHOULI
Norman, Oklahoma
2007

HIRSCHMAN OPTIMAL TRANSFORM LEAST MEAN SQUARE ADAPTIVE FILTERS

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

Acknowledgements

I would like to sincerely thank my supervisors Dr. Victor DeBrunner and Dr. Joseph Havlicek who introduced me to one of the most important and diverse fields in Electrical Engineering, Digital Signal Processing (DSP). They taught me everything I know about this field, inspired me with its wide applications, and guided me throughout my research. They also put all efforts to provide me continuous financial support during my graduate study at University of Oklahoma.

I would like also to thank Dr. Tomasz Przebinda and Dr. Murad Özaydin for being part of out research and for the SigCam lectures that helped me better understand the mathematical theory behind the Hirschman optimal transform. I would like also to thank Dr. Tamer Ibrahim for his advices and comments that helped me better presents the results of my dissertation.

My family has given me their spiritual support. Their encouragement kept me motivated to proceed to expand my knowledge. Their love and support made all of this worthwhile.

Last but not least, I would like to thank my friends at University of Oklahoma. With their presence and help I felt like having a family in Norman.

Notation

The following notations are used throughout the dissertation. Nonbold lowercase letters are used for scalar quantities, bold lowercase is used for vectors, and bold uppercase is used for matrices. Nonbold uppercase letters are used for integer quantities such as length or dimensions. The lowercase letter k is reserved for the block index. The lowercase letter n is reserved for the time index. The time and block indexes are put in brackets, whereas subscripts are used to refer to elements of vectors and matrices. The uppercase letter N is reserved for the filter length and the uppercase letter L is reserved for the block length. The superscripts T and H denote vector or matrix transposition and Hermitian transposition, respectively. The subscripts F and H are used to highlight the DFT and HOT domain quantities, respectively. The $N \times N$ identity matrix is denoted by $\mathbf{I}_{N\times N}$ or $\mathbf{I}$. The $N \times N$ zero matrix is denoted by $\mathbf{0}_{N\times N}$. The linear and circular convolutions are denoted by $*$ and $\star$, respectively. $\mathrm{Diag}\,[\mathbf{v}]$ denotes the diagonal matrix whose diagonal elements are the elements of the vector $\mathbf{v}$.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Least mean square (LMS) adaptive filters, as investigated by Widrow and Hoff in 1960 [1], find applications in many areas of digital signal processing including channel equalization, system identification, adaptive antennas, spectral line enhancement, echo interference cancelation, active vibration and noise control, spectral estimation, and linear prediction [4]. The computational burden and slow convergence speed of the LMS algorithm can render its real time implementation infeasible. The discrete Fourier transform (DFT) has been used to improve the computational cost [2, 3] and the convergence speed [6] of the LMS algorithm.

The Hirschman optimal transform (HOT) is a recently developed discrete unitary transform that uses the orthonormal minimizers of the entropy-based Hirschman uncertainty measure [13]. This measure is different from the energy-based Heisenberg uncertainty measure that is only suited for continuous time signals. The Hirschman uncertainty measure uses entropy to quantify the spread of discrete-time signals in time and frequency [14]. Since the HOT bases are among the minimizers of the uncertainty measure, they have the novel property of being the most compact in discrete-time and frequency. The fact that the HOT basis sequences have many zero-valued samples, as well as their resemblance to the DFT basis sequences, makes the HOT computationally attractive. Furthermore, it has been shown recently that a thresholding algorithm using the HOT yields superior frequency resolution of a pure tone in additive white noise to a similar algorithm based on the DFT [46].

This dissertation introduces new transform domain LMS algorithms based on the HOT. The analyses of presented in this dissertation not only show the improvements in the computational efficiency and convergence speed of the HOT based LMS algorithms but also add more insight into the properties of the HOT and its effects on random signals.

1.2 Original Contributions

The original contribution of this dissertation is four new transform domain LMS algorithms. The First algorithm is the HOT LMS algorithm. This algorithm is somewhat faster than the LMS algorithm and requires less than half the computations of the DFT LMS algorithm. The second algorithm is the self-orthogonalizing block HOT LMS algorithm which requires slightly more multiplications than the block DFT LMS algorithm but converges at a faster rate. The third algorithm is the HOT block LMS algorithm. This algorithm requires less multiplications than the LMS and DFT block LMS algorithms. The fourth algorithm is the HOT DFT block LMS algorithm. This algorithm is very similar to the DFT block LMS algorithm and reduces it computational complexity by about 30% when the filter length is much smaller than the block length.

These computationally efficient transform domain LMS algorithms reduce the computational burden of the conventional LMS algorithm more than the DFT domain LMS algorithms and hence expand the real time applications of the LMS algorithm beyond the real time applications of the DFT domain LMS algorithms.

Chapter 2

Background

In this chapter, the Least Mean Square (LMS) algorithm is reviewed. The review includes the structure of the algorithm and its statistical convergence analysis.

2.1 FIR Adaptive Filters

The general finite impulse response (FIR) adaptive filter problem can be stated as follows: given two random signals $d(n)$ and $u(n)$, what is the impulse response $w(n)$ of an FIR filter which, when driven by $u(n)$, will produce an output $y(n) = w(n) * u(n)$ that is the best estimate in the mean square sense of $d(n)$. The optimal FIR filter can be found by minimizing the following mean square error (MSE):

$$\xi(n) = E\big|e(n)\big|^2 = E\big|d(n) - w(n) * u(n)\big|^2. \tag{2.1}$$

Let

$$\mathbf{u}(n) = \begin{bmatrix} u(n) & u(n-1) & \cdots & u(n-N+1) \end{bmatrix}^T \tag{2.2}$$

and

$$\mathbf{w}(n) = \begin{bmatrix} w_0(n) & w_1(n) & \cdots & w_{N-1}(n) \end{bmatrix}^T \tag{2.3}$$

be the tap-input and tap-weight vectors, respectively. The optimal tap-weight vector can be found from solving

$$\mathbf{R}(n)\mathbf{w}(n) = \mathbf{r}_{du}(n), \tag{2.4}$$

where $\mathbf{R}(n) = E\mathbf{u}^*(n)\mathbf{u}^T(n)$ is the autocorrelation matrix of the input vector and $\mathbf{r}_{du}(n) = E\,d(n)\mathbf{u}^*(n)$ is the cross-correlation vector between the desired and input signals [41]. If all of the involved signals are jointly wide-sense stationary (WSS), then the optimal filter will be time-invariant. Equation (2.4) is called Wiener-Hoff equation and the optimal filter is called the Wiener filter.

2.2 The LMS Algorithm

An adaptive filter based on equation (2.4) is not practical in most applications, since the matrices $\mathbf{R}(n)$ and $\mathbf{r}_{du}(n)$ are not known in advance and inverting $\mathbf{R}(n)$ requires a great deal of computations. To avoid any matrix inversion, the adaptive filter weight vector can be updated according the steepest decent algorithm according to [41]

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\, E\,\mathbf{u}^*(n)\, e(n), \tag{2.5}$$

where μ is the step size that controls the convergence of the algorithm. If $Ee(n)\mathbf{u}^*(n)$ is replaced by the simple estimate $\mathbf{u}^*(n)e(n)$, then we have the stochastic update equation [41]

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\,\mathbf{u}^*(n)\, e(n), \tag{2.6}$$

which is known as the LMS [41] algorithm.

An LMS adaptive filter having N coefficients requires N multiplications and N additions to update the filter weights. In addition, one addition is necessary to compute the error $e(n)$ and one multiplication is needed to form the product $\mu e(n)$. Finally, N multiplications and $N-1$ additions are needed to calculate the output of the adaptive filter. Thus a total of $2N+1$ multiplications and $2N$ additions per input sample are required. Figure 2.1 shows the FIR LMS adaptive filter [41].

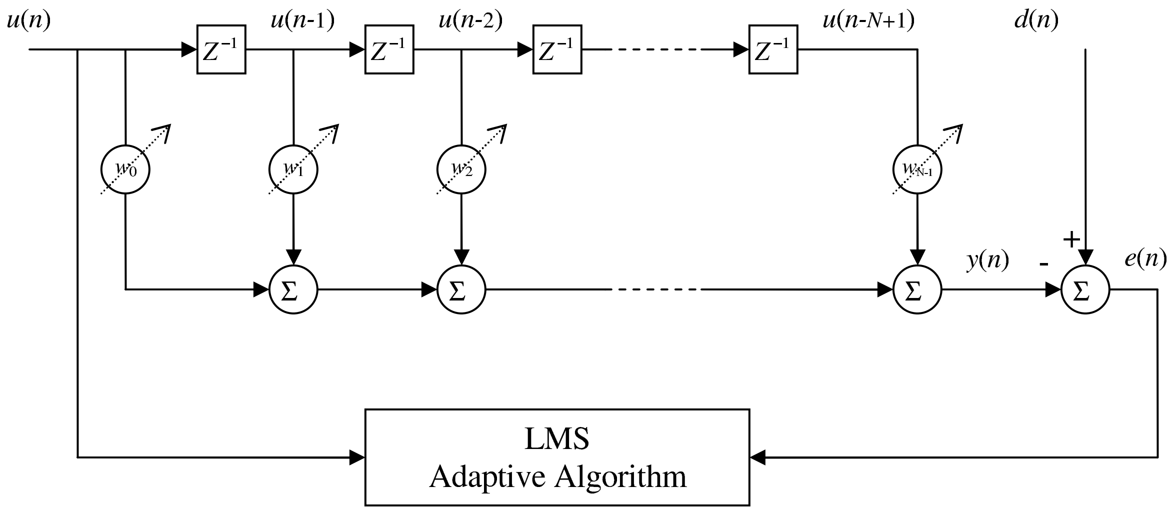


Figure 2.1: FIR adaptive filer block diagram.

The analysis of the LMS Adaptive filter is difficult since it is not linear. With the initial condition $\mathbf{w}(n) = \mathbf{0}$, the solution of equation (2.6) is [4]

$$\mathbf{w}(n) = \mu \sum_{i=-\infty}^{n-1} \mathbf{u}^*(i)e(i). \tag{2.7}$$

The filter output is given by [4]

$$y(n) = \mu \sum_{i=-\infty}^{n-1} \mathbf{u}^H(i)\mathbf{u}(i)e(i). \tag{2.8}$$

From equation (2.8), the LMS adaptive filter is a complicated nonlinear filter. This implies that the analysis of the LMS algorithm is very difficult. However, the LMS algorithm can be analyzed under the condition of small step size for a stationary environment, as will be explained in the next section.

2.3 Statistical LMS Theory

Let $\boldsymbol{\epsilon}(n) = \mathbf{w}^o - \mathbf{w}(n)$ be the error in estimating the filter weight vector, where $\mathbf{w}^o$ is the Wiener optimal solution of equation (2.4). The error $\boldsymbol{\epsilon}(n)$ satisfies the stochastic difference equation [4]

$$\boldsymbol{\epsilon}(n+1) = \Big(\mathbf{I} - \mu\,\mathbf{u}^*(n)\,\mathbf{u}^T(n)\Big)\boldsymbol{\epsilon}(n) - \mu\,\mathbf{u}^*(n)\,e^o(n), \tag{2.9}$$

where $e^o(n)$ is the error produced by the Wiener filter. The above equation can be analyzed based on the following two assumptions:

I. The step size is small, such that the LMS filter acts as a lowpass filter with a low cutoff frequency.

II. The desired response $d(n)$ is generated from the linear regression model $d(n) = w^o(n) * u(n) + e^o(n)$, where $e^o(n)$ is a white-noise process with variance $J_{\min}$ such that $e^o(n)$ is statistically independent of the input.

Equation (2.9) can be iteratively solved by expressing $\boldsymbol{\epsilon}(n)$ as [4]

$$\boldsymbol{\epsilon}(n) = \boldsymbol{\epsilon}_0(n) + \boldsymbol{\epsilon}_1(n) + \boldsymbol{\epsilon}_2(n) + \cdots . \tag{2.10}$$

Substituting equation (2.10) into equation (2.9) yields [4]

$$\boldsymbol{\epsilon}(n+1) = \Big(\mathbf{I} - \mu\,\mathbf{R}\Big)\,\boldsymbol{\epsilon}(n) - \mu\,\mathbf{u}^*(n)\,e^o(n) + \mu\,\mathbf{P}(n)\,\boldsymbol{\epsilon}(n), \tag{2.11}$$

where $\mathbf{P}(n) = \mathbf{u}^*(n)\mathbf{u}^T(n) - \mathbf{R}$, which leads to the following set of coupled difference equations [4]:

$$\boldsymbol{\epsilon}_i(n) = \Big(\mathbf{I} - \mu\,\mathbf{R}\Big)\,\boldsymbol{\epsilon}_i(n) + \begin{cases} -\mu\,\mathbf{u}^*(n)\,e^o(n) & \text{if } i = 0, \\ -\mu\,\mathbf{P}(n)\,\boldsymbol{\epsilon}_{i-1}(n) & \text{if } i \neq 0, \end{cases} \tag{2.12}$$

When μ is small, only the first term in equation (2.10) is significant and the small step size LMS theory can be described by the difference equation [4]

$$\boldsymbol{\epsilon}_0(n+1) = \Big(\mathbf{I} - \mu\,\mathbf{R}\Big)\,\boldsymbol{\epsilon}_0(n) - \mu\,\mathbf{u}^*(n)\,e^o(n). \tag{2.13}$$

If the autocorrelation matrix is decomposed using its eigenvectors according to [41]

$$\mathbf{R} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^H, \tag{2.14}$$

then equation (2.13) can be written as [4]

$$\boldsymbol{\epsilon}_T(n+1) = \Big(\mathbf{I} - \mu\,\boldsymbol{\Lambda}\Big)\,\boldsymbol{\epsilon}_T(n) - \mu\,\mathbf{V}^H\,\mathbf{u}^*(n)\,e^o(n), \tag{2.15}$$

where $\boldsymbol{\epsilon}_T(n) = \boldsymbol{\Lambda}^H \boldsymbol{\epsilon}_0(n)$. The second order statistics of the forcing term in equation (2.15) are given by [4]

$$\begin{aligned} E\,\mu\,\mathbf{V}^H \mathbf{u}^*(n)\,e^o(n) &= \mathbf{0}, && (2.16)\\ E\,\mu^2 \Big(\mathbf{V}^H \mathbf{u}^*(n)\,e^o(n)\Big)\Big(\mathbf{V}^H \mathbf{u}^*(n)\,e^o(n)\Big)^H &= \mu^2 J_{\min}\,\boldsymbol{\Lambda}. && (2.17)\end{aligned}$$

The solution of equation (2.15) is given by [4]

$$\boldsymbol{\epsilon}_T(n+1) = \Big(\mathbf{I} - \mu\,\boldsymbol{\Lambda}\Big)^n \boldsymbol{\epsilon}_T(0) - \sum_{i=0}^{n-1} \Big(\mathbf{I} - \mu\,\boldsymbol{\Lambda}\Big)^{n-1-i} \mu\,\mathbf{V}^H\,\mathbf{u}^*(i)\,e^o(i). \tag{2.18}$$

Equations (2.16 and 2.17 give the following second order statistics of the solution in equation (2.18) [4]:

$$\begin{aligned} E\boldsymbol{\epsilon}_T(n) &= \Big(\mathbf{I} - \mu\,\boldsymbol{\Lambda}\Big)^n \boldsymbol{\epsilon}_T(0) && (2.19)\\ E\,|\epsilon_l(n)|^2 &= \mu\frac{J_{\min}}{2-\mu\lambda_l} + (1-\mu\lambda_l)^{2n}\Big(|\epsilon_l(0)|^2 - \mu\frac{J_{\min}}{2-\mu\lambda_l}\Big). && (2.20)\end{aligned}$$

The most common performance measure of the LMS algorithm is the mean square error (MSE) $J(n) = E\big|e(n)\big|^2$. A plot of the MSE versus time is called the learning curve. The MSE can be written as [4]

$$J(n) = J_{\min} + \mathrm{Tr}\left[\mathbf{R}\,E\,\boldsymbol{\epsilon}_0(n)\,\boldsymbol{\epsilon}_0^H(n)\right]. \tag{2.21}$$

In terms of the eigenvalue decomposition of $\mathbf{R}$, equation (2.21) can be written as

$$J(n) = J_{\min} + \mathrm{Tr}\left[E\,\boldsymbol{\epsilon}_T^H(n)\,\boldsymbol{\Lambda}\,\boldsymbol{\epsilon}_T(n)\right], \tag{2.22}$$

or equivalently,

$$J(n) = J_{\min} + \sum_{l=1}^{N} \lambda_l \, E \, |\epsilon_l(n)|^2 \,. \tag{2.23}$$

Substituting the result of equation (2.20) into equation (2.23) gives [4]

$$J(n) = J_{\min} + \mu \, J_{\min} \sum_{l=1}^{N} \frac{\lambda_l}{2 - \mu\lambda_l} + \sum_{l=1}^{N} \lambda_l \left(|\epsilon_l(n)|^2 - \mu \frac{J_{\min}}{2 - \mu\lambda_l} \right) (1 - \mu\lambda_l)^{2n}. \tag{2.24}$$

The steady state MSE can be found by evaluating the MSE in equation (2.24) at infinity [4]:

$$J(\infty) = J_{\min} + \mu \, J_{\min} \sum_{l=1}^{N} \frac{\lambda_l}{2 - \mu\lambda_l}. \tag{2.25}$$

These results are subject to the assumption that the LMS algorithm is convergent. To guarantee convergence of the LMS algorithm in the MS sense, it is generally required that [4]

$$\mu < \frac{2}{\lambda_{\min}}. \tag{2.26}$$

The second term in equation (2.25) is positive, which implies that the steady state MSE is higher than that of the Wiener filter. Therefore, in spite of the fact that the LMS algorithm converges to the Wiener filter, the MSE of the LMS filter is higher than that of the Wiener filter. This result is expected, since at steady state the LMS weight estimate fluctuates about the Wiener solution. The difference between the MSE of the LMS and Wiener filters is called the excess mean square error J_{ex}. The percent deviation of the steady state error of the LMS filter is called the misadjustment [4] and is given by

$$M = \frac{J_{\text{ex}}}{J_{\min}} = \mu \sum_{l=1}^{N} \frac{\lambda_l}{2 - \mu\lambda_l}. \tag{2.27}$$

Equation (2.24) can be used to determine the average time constant (the time needed for the transient behavior to decay) of the LMS algorithm [4]:

$$\tau_{\text{ave}} = \frac{N}{2\mu \sum_{l=1}^{N} \lambda_l}. \tag{2.28}$$

Another measure of the time constant of the LMS algorithm is the time constant of the slowest mode of the LMS filter [4], which is given by

$$\tau_{\max} = \frac{1}{2\mu\, \lambda_{\min}}. \tag{2.29}$$

But we have $\mu \sim 2/\lambda_{\max}$, so [4]

$$\tau_{\max} \sim \frac{\lambda_{\max}}{\lambda_{\min}}. \tag{2.30}$$

This result shows that the speed of convergence of the LMS algorithm depends on the eigenvalue spread of the autocorrelation matrix of the input.

2.4 The Transform Domain LMS Algorithm

The speed of convergence of the LMS algorithm can be increased if the step size is replaced by the matrix $\alpha\mathbf{R}^{-1}$ to obtain [4]

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \alpha\, \mathbf{R}^{-1}\mathbf{u}^*(n)\, e(n). \tag{2.31}$$

With this change in the LMS algorithm, (2.13) becomes [4]

$$\boldsymbol{\epsilon}_0(n+1) = \Big(\mathbf{I} - \alpha\, \mathbf{R}^{-1}\mathbf{R}\Big)\boldsymbol{\epsilon}_0(n) - \alpha\, \mathbf{R}^{-1}\mathbf{u}^*(n)\, e^o(n). \tag{2.32}$$

Equation 2.32 implies that the convergence of the LMS algorithm is independent of the input statistics and that the convergence does not depend on the eigenvalue spread that slows down the convergence of the conventional LMS algorithm. This algorithm is known as the self-orthogonalizing LMS algorithm [40]. Upon substituting $\mathbf{R}^{-1} = \mathbf{V}\mathbf{\Lambda}^{-1}\mathbf{V}^H$, equation (2.31) can be written as [4]

$$\mathbf{w}_T(n+1) = \mathbf{w}_T(n) + \alpha\,\mathbf{\Lambda}^{-1}\mathbf{u}_T^*(n)\,e(n), \tag{2.33}$$

where $\mathbf{w}_T(n) = \mathbf{V}^H\mathbf{w}(n)$ and $\mathbf{u}_T(n) = \mathbf{V}^T\mathbf{u}(n)$.

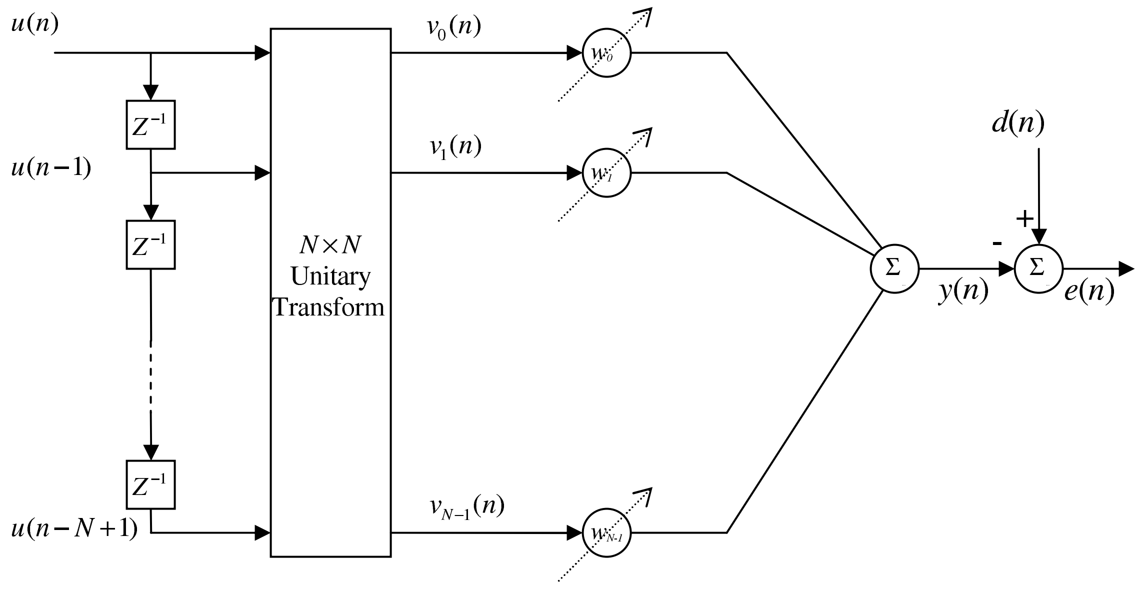


Figure 2.2: Schematic diagram of the transform domain LMS algorithm.

The result in equation (2.33) means that the self-orthogonalizing LMS algorithm can be equivalently implemented by transforming the input vector by $\mathbf{V}^T$ and using an individual step size for each transformed input component that is proportional to the inverse of the corresponding eigenvalue of the autocorrelation matrix. This implementation is referred to as the transform domain LMS algorithm. A schematic of the transform domain LMS algorithm is shown in Figure 2.2.

Chapter 3

Review of the Transform Domain LMS Algorithms

Although the LMS algorithm is computationally simpler than the least squares (LS) algorithm, real time implementation may not be feasible, especially for large filter lengths. To reduce the computational cost of the LMS filter, Ferrara proposed a frequency domain implementation of the LMS algorithm [2]. In this algorithm, the data is partitioned into fixed-length blocks and the weights are allowed to change after each block is processed. This algorithm is called the block LMS algorithm. The computational reduction in the block LMS algorithm comes from using fast DFT convolution to calculate the convolution between the filer input and weights and the gradient estimate. For an adaptive filter of length N, this algorithm, known as the DFT block LMS algorithm, requires $10N\log_2(2N)+16N$ real multiplications whereas the conventional LMS algorithm requires $2N^2+N$ real multiplications.

A time domain convergence analysis of the DFT block LMS algorithm was presented in [3]. It was proved that the conditions under which the block LMS and conventional LMS algorithms converge in the mean are the same. It was also shown that if the ratio between the step sizes of the block LMS algorithm and conventional LMS algorithm equals the block length, then both algorithms will converge at the same rate and have the same misadjustment provided that they are driven by the same input.

The DFT block LMS algorithm is required to perform five DFTs, two of them due to the fact that the DFT can perform circular convolution and the estimation of the gradient requires linear convolution instead. The DFT block algorithm is also referred to as the constrained DFT block LMS filter, since the two additional DFTs are needed to constrain the gradient. Mansour, et. al., removed this constraint to save two DFT computations [24]. This algorithm is called the unconstrained DFT block LMS algorithm. This block LMS algorithm was proposed for applications that require adaptive filters of order up to a few thousands, such as sonar signal processing and echo cancelation. It was shown that with the constraint removed, the algorithm can still converge to the Wiener solution under certain conditions.

The LMS algorithm is known to have low convergence speed when driven by colored input. In [5], it was shown that for stationary data with small step size, the speed of convergence of the LMS algorithm is dependent on the ratio of the maximum to the minimum eigenvalues (the condition number) of the input autocorrelation matrix. To increase the convergence speed of the LMS algorithm, Narayan and Peterson proposed the transform domain LMS (TRLMS) algorithm [6, 7]. This algorithm uses a fixed transform such as the DFT or discrete cosine transform (DCT) to whiten the input and reduce the condition number of the autocorrelation matrix. The tap-input vector is transformed into another vector which is then used to feed the LMS algorithm.

Each filter weight is assigned an individual step size that is inversely proportional to the power of the corresponding component of the transformed vector.

The work in [6, 7] did not include any convergence analysis or an algorithm to estimate the powers needed for the step sizes. Nevertheless, it was shown that with a properly chosen transform, a reduction in the condition number can be expected. The performance of the TRLMS algorithm was presented for the first time in [8]. It was shown that the MSE of the Wiener filter in the transform domain is the same as the time domain Wiener filter MSE. Also, it was shown that the convergence speed and steady state MSE of the TRLMS and LMS algorithms are the same if the TRLMS algorithm is implemented with a constant convergence factor. It was explained in [8] that if the transform domain correlation matrix is approximately diagonal, then the convergence speed of the TRLMS algorithm is improved significantly compared to the LMS algorithm for the same steady state MSE. In [8], two methods were proposed to estimate the diagonal elements of the transform domain autocorrelation matrix that are needed to calculate the step sizes. In the first method, the first row of the input autocorrelation matrix is estimated recursively using a single pole lowpass filer. The diagonal elements are then given by the DFT of the estimated row. The second method is based on the observation that, since the diagonal elements of the transform domain correlation matrix are the powers of the components of the transformed input vector, they can be estimated recursively from the transformed input vector with a single pole lowpass filer. It was mentioned that the DFT performs well, provided that the input autocorrelation sequence decays much faster than the filer order. Assuming a real input, the computational complexity of the DFT LMS algorithm is $N(\log_2 N + 3) + 4$ real multiplications. The multiplication count for the

DCT LMS algorithm is $N(\log_2 N - 3/2) + 4$, which is slightly less than that for the DFT LMS algorithm, since the DCT is a real value transform.

Real Transforms such as the DCT and discrete sine transform (DST) are preferred since they are real and thereby avoid the need for complex multiplications. This results in fewer computations than the DFT. Another real transform, called the discrete Hartley transform (DHT) was also used in the TRLMS algorithm [10]. The running DHT of an input $u(n)$ is given by [10]

$$u_{\mathrm{DHT}}(n,m) = \sum_{k=0}^{N-1} u(n-k)\big(\cos(\frac{2\pi}{N}km) + \sin(\frac{2\pi}{N}km)\big). \tag{3.1}$$

The DHT adaptive filter was in introduced in [9]. It was used in recovering narrowband signals from noise contamination.

The aforementioned research results verified by theory and simulations that the speed of convergence of any TRLMS algorithm depends on the condition number of the correlation matrix of the transformed input. The optimal transform for the TRLMS algorithm is the Karhunen-Loeve transform (KLT). Since the KLT is defined in terms of the statistics of the input, it is certain that no fixed-parameter transform will deliver optimal learning characteristic for all signals. Therefore, the choice of a suitable transform is application dependent. After many transforms were suggested in the literature, a comparative study was presented in [11]. The performances of the DFT, DCT, DHT, Walsh-Hadamard transform (WHT), and power-of-2 (PO2) transform LMS algorithms were compared and tested with different colored inputs. The results of the simulations verified that there is no single transform that can perform better than the others. For example, the DFT performed best with narrowband input whereas the DCT performed best with wideband input. Also, [11] presented a novel explanation for the convergence rate improvement property of the TRLMS algorithm. It was shown that the effect of an ideal transform on the shape of the MSE surface is to convert the equal error contours, which are initially hyperellipses, into hyperspheres. In [11], it was shown through simulations that the error surface theory can only predict the overall convergence rate and it does not take into considerations the distributions of the eigenvalues or the noise level in the desired signal.

Although most research agrees on the advantages of the transform domain LMS algorithms, there were no analytic calculations of the asymptotic eigenvalue spread of the transformed input vector autocorrelation matrix prior to [12], where the asymptotic eigenvalue spread for the DFT and DCT LMS algorithms were derived for the case of first-order Markov input signals. The following summarizes the main results in [12]:

I. The eigenvalue spread of the autocorrelation matrix of a first-order Markov signal of parameter $\rho \in [0\,,1]$ tends to $(1+\rho)^2/(1-\rho)^2$ as the length of filter increases.

II. The eigenvalue spread of the autocorrelation matrix of the same signal after DFT and power normalization tends to $(1+\rho)/(1-\rho)$ as the length of filter increases. For a finite filter length, the eigenvalue spread is always less than $(1+\rho)/(1-\rho)$.

III. The eigenvalue spread of the autocorrelation matrix of a first-order Markov signal transformed by a DCT with power normalization tends to $1+\rho$ as the length of filter increases. For finite filter length, the eigenvalue spread is slightly greater than $1+\rho$ as the length of filter increases.

The TRLMS algorithm was proposed to increase the convergence speed whereas the DFT block LMS algorithm was proposed to reduce the computational complexity of the conventional LMS algorithm from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log_2 N)$. These two algorithms can be combined to improve the overall performance of the LMS filter. In [19], it was explained how to improve the convergence speed of the DFT block LMS algorithm in the context of echo cancelation such as that widely used to reduce the echo signal in telephony systems by normalizing the power in each frequency bin. The power in each frequency bin is estimated recursively from the transformed input vector with single a pole lowpass filer as suggested in [8]. In [19], the gradient constraint was represented by a window function; this window function was used to force the second half of the time domain augmented impulse response to zero in the constrained block LMS algorithm. It was shown that the difference between the unconstrained and constrained block LMS algorithms is dependent on the type of window that is used. The analysis and simulations in [19] showed that, depending on the global envelope of the echo path impulse response, an efficient window can be used to eliminate two of the five DFTs without degrading the convergence performance of the adaptive filter.

Theoretical performance analysis of the DFT block LMS algorithm with power normalization and the unconstrained DFT block LMS algorithm with and without power normalization was given in [25]. The autocorrelation matrices that govern the weight updates in the frequency domain were analyzed in the time domain to conclude the following results, all of which were expected:

I. The unconstrained DFT LMS algorithm converges to the Wiener optimal filter in the mean for sufficiently large filter length with slightly higher misadjustment than that of the constrained DFT LMS algorithm.

II. Both the unconstrained and constrained DFT block LMS algorithms with power normalization converge faster than the conventional LMS algorithm.

III. The constrained LMS algorithm with power normalization converges slightly faster than the unconstrained DFT approach.

The recursive least square (RLS) adaptive filters are known to have superior convergence speed compared to the LMS filters at the expense of increased computational complexity, which is of order $\mathcal{O}(N^2)$, compared to $\mathcal{O}(N)$ for the LMS filter [43]. The convergence of the RLS algorithm is consistent and does not depend on the statistics of the input. Panda, et. al., worked on an algorithm that lay between the RLS and the LMS algorithms in both computational complexity and performance, providing a rate of convergence that is independent of the input signal conditioning [26]. To make the algorithm independent of the input signal conditioning, they looked at estimating the autocorrelation matrix of the input and to improve the computational complexity they used the DFT block LMS algorithm. Their algorithm is called the self-orthogonalizing block adaptive filter (SOBAF). They did not use the well-known unbiased autocorrelation matrix estimate [42]

$$\frac{1}{N}\sum_{i=0}^{N-1}\mathbf{u}(i)\mathbf{u}^T(i), \tag{3.2}$$

since it is not Toeplitz and its inverse can not be calculated efficiently. Instead, they estimated the first row of the autocorrelation matrix and generated the rest of the elements assuming that the autocorrelation matrix is symmetric and Toeplitz. Therefore, the inverse of the estimate can be found efficiently using the Levinson recursion, which requires $\mathcal{O}(N)$ operations per sample, or using the fast algorithm given in [27] that requires $\mathcal{O}(\log_2 N)$ operations. Since the DFT block LMS algorithm also requires $\mathcal{O}(\log_2 N)$ operations, the overall SOBAF requires $\mathcal{O}(\log_2 N)$ operations, a dramatic reduction in computational load compared to the basic LMS algorithm. Their simulations showed that the convergence of SOBAF is close to that of the ideal KLT LMS algorithm.

DFT convolution may not necessarily be the most efficient convolution algorithm. An efficient block LMS algorithm given in [28] employed the rectangular transform (RT) to implement circular convolution in the block LMS filter [29]. RT convolution is more efficient than DFT convolution up to a signal length of 420 points. However, for applications with filter lengths of several hundreds, where efficient block LMS algorithm is crucial, the DFT block LMS algorithm is more efficient than the RT LMS algorithm. Another efficient block LMS adaptive filter based on the DHT was proposed in [30], see equation (3.1). The circular convolution between $u(n)$ and $h(n)$ in the DHT domain is given by [30]

$$\big(h^{\mathrm{e}}_{\mathrm{DHT}}(k) + h^{\mathrm{o}}_{\mathrm{DHT}}(k)\big)u^{\mathrm{e}}_{\mathrm{DHT}}(k) + \big(h^{\mathrm{e}}_{\mathrm{DHT}}(k) - h^{\mathrm{o}}_{\mathrm{DHT}}(k)\big)u^{\mathrm{o}}_{\mathrm{DHT}}(k). \tag{3.3}$$

This algorithm is similar to the unconstrained DFT LMS filter [24] and hence has higher steady state mean square error than the DFT block LMS algorithm. The convergence speed of the DHT block LMS algorithm is similar to that of the unconstrained DFT LMS filter and enjoys about a 30% reduction in the computational complexity.

The computational reduction of the aforementioned block LMS algorithms comes from fast implementation of convolution using specific transforms while keeping the weights fixed in each block. As mentioned before [3], for the block LMS algorithm to converge at the same rate as the conventional LMS algorithm, the step size of the block LMS filter has to be scaled by the block length; this has the effect of reducing the stability domain of the block LMS filter. In [18], Benesty and Duhamel presented a block LMS algorithm that is mathematically equivalent to the LMS algorithm. The algorithm is called the fast exact LMS (FELMS) algorithm. The computational saving in the FELMS algorithm comes from writing the LMS update equations in each block and eliminating all weight vectors except the last one. This way, the computations required to calculates the weight vectors inside each block can be avoided to improve the overall computational count. The FELMS algorithm is computationally most efficient for small block lengths. For example, the number of multiplications needed for a 512-point filter and a block length of 32 is 1024, 289, and 243 with the LMS, FELMS, and DFT block LMS algorithms, respectively. The performance of the FELMS algorithm is worse than that of the LMS algorithm when the filter and block lengths are equal. This is opposite from the case of the DFT block LMS algorithm, which is most efficient when the filter and block lengths are the same. In any case, the DFT block LMS algorithm is more efficient than the FELMS algorithm.

Although the DFT block LMS algorithm with a block length smaller than the filter order is not the most efficient, it is preferred in order to avoid undesirable excessive delay in the output samples that would limit the application of the DFT block LMS filter in some cases. To increase the computational efficiency of the block LMS algorithm with smaller block length, Boroujeny, et. al., developed the generalized sliding FFT (GSFFT) that computes the DFT of the current block input vector using the already computed DFT of the previous block input vector [32]. It is worth mentioning that the sliding DFT is a special case of GSFFT when the input sequence slides one sample at a time as is the case in the TRLMS algorithm [31]. It was shown that the computational complexity of the DFT block LMS algorithm for smaller block lengths can be improved by 30% to 70% depending on the ratio between the filter length and block length. This improvement comes at the expense of increasing the misadjustment, which is negligible for small block lengths.

Lee and Un provided a convergence analysis of the DFT block LMS algorithm following a mapping of the frequency domain information to the time domain before proceeding with the analysis of the algorithm [25]. Therefore, their analysis was difficult to follow. Boroujeny and Chan [22] presented an equivalent analysis of the DFT block LMS algorithm in the frequency domain. Their analysis gives better insight into the effect of various processing components in the algorithm structure on its convergence behavior. In addition to the conclusions of [25], they added the following observations:

I. For both the constrained and unconstrained DFT block LMS algorithms, the eigenvalues of all modes of convergence are asymptotically the same. This is due to the window function that is used to extract the circular convolution samples that correspond to linear convolution. Without this effect, the eigenvalue spread would have been as predicted by F. Beaufays [12].

II. The constrained DFT block LMS algorithm converges slightly faster than the unconstrained block LMS algorithm because of the window, which constrains the weights and reduces the eigenvalue spread compared to the spread of the unconstrained block LMS filter.

The DFT block LMS algorithm is implemented using five DFTs. Narasimha [33] reduced the number of DFTs from five to three by combining the required convolution and correlation operations into a single complex filtering process. Although the new structure did not reduce the number of multiplications, it decreased the number of additions. The corresponding architecture is more elegant and simpler to implement in hardware or software compared to previous LMS implementations.

Modifications to the original TRLMS algorithm were also suggested. Ogunfunmi and Peterson proposed a TRLMS algorithm that contained two adaptive algorithms running at the same time [34]. The first LMS algorithm estimates the DFT using the input signal as the desired signal and the phasor vector

$$\mathbf{u}(i) = \begin{bmatrix} 1 & e^{j\frac{2\pi}{N}i} & e^{j\frac{2\pi}{N}2i} & \ldots & e^{j\frac{2\pi}{N}(N-1)i} \end{bmatrix}^T \tag{3.4}$$

as input. This algorithm requires $\mathcal{O}(N)$ computations per sample to estimate the DFT of the input while the conventional DFT algorithms require $\mathcal{O}(N\log_2 N)$ computations per sample. Such computational improvement is not new, since the sliding DFT can also be used to compute the DFT with $\mathcal{O}(N)$ computations per sample.

All of the previously mentioned versions of the LMS algorithm used fixed step sizes. Chao, et. al., proposed another fast adaptive filter algorithm that uses variable step sizes [35]. They showed that there exist finite optimum update positions in the gradient direction of the LMS algorithm and the optimum step sizes to reach these positions are the reciprocal eigenvalues of the input autocorrelation matrix. This algorithm is different from the previous TRLMS algorithms that used the reciprocal of the eigenvalues as step sizes for the convergence modes of the filter. This new algorithm uses the reciprocal of one eigenvalue as a step size before it uses another eigenvalue in the next update. This algorithm requires good estimation of the eigenvalues and the gradient. They used the DCT to estimate the eigenvalues and block averaging for better gradient estimation. The number of multiplications required is about $11 \log_2 N + 12$ per sample. Another transform domain variable step size LMS algorithm was proposed in [37]. The step size for the i^{th} frequency bin is given by $\mu_i(n) = \mu(n)/\sigma_i^2$, where $\mu(n)$ is a time-dependant step size, constant for all modes and is called the global step size. It is chosen to be dependant on the output error of the filter. Simulations showed that this variable step size can speed up the convergence compared to other TRLMS algorithms.

A wavelet TRLMS algorithm was presented by Hosur and Tewfik in [38]. The algorithm exploits the special sparse structure of the wavelet transform of wide classes of correlation matrices and their Cholesky factorizations in order to compute a whitening transformation of the data in the wavelet domain and minimize the computational complexity. They described two approaches. The first approach explicitly computes a sparse estimate of the wavelet domain correlation matrix of the input process. It then computes the Cholesky factorization of that matrix and uses the inverse to whiten the input. The complexity of this approach is $\mathcal{O}(N \log_2^2 N)$. In contrast, the second approach computes a sparse estimate of the Cholesky factorization of the wavelet domain correlation matrix directly. This second approach has a computational complexity of $\mathcal{O}(N \log_2 N)$ operations. However, it requires a more complex book keeping procedure. Both algorithms have a convergence rate that is faster than that of the time domain, DFT, and DCT LMS algorithms.

Chapter 4

Hirschman Optimal Transform

The Hirschman optimal transform (HOT) is a recently developed discrete unitary transform that uses the orthonormal minimizers of the entropy-based Hirschman uncertainty measure [13]. This measure is different from the energy-based Heisenberg uncertainty measure that is only suited for continuous time signals. The fact that the HOT basis sequences have many zero-valued samples, as well as their resemblance to the DFT basis sequences, makes the HOT computationally attractive. This chapter describes the mathematical theory behind the HOT.

4.1 The Phase Plane for Continuous Time Signals

Let $u(t)$ be a signal in the space of square integrable functions $L^2(R)$. The norm of $u(t)$ is given by [14]

$$\|u\|^2 = \int_R |u(t)|^2 dt. \tag{4.1}$$

The Fourier transform of $u(t)$ is given by [14]

$$u_F(j\,\omega) = \int_R u(t)\, e^{-j\omega t} dt. \tag{4.2}$$

The set of all points $(t, \omega) \in R^2$ define the phase plane. The uncertainty in position is a measure of the duration where the signal has non-negligible energy and is defined by [14]

$$\sigma_t^2 = \frac{1}{\|u\|^2} \int_R (t - \bar{t}\,)^2 \, |u(t)|^2 \, dt, \tag{4.3}$$

where

$$\bar{t} = \frac{1}{\|u\|^2} \int_R t\, |u(t)|^2 \, dt. \tag{4.4}$$

The uncertainty in frequency is a measure of the bandwidth where the signal has non-negligible energy and is defined by [14]

$$\sigma_\omega^2 = \frac{1}{\|u_F\|^2} \int_R (\omega - \bar{\omega}\,)^2 \, |u_F(j\omega)|^2 \, d\omega, \tag{4.5}$$

where

$$\bar{\omega} = \frac{1}{\|u_F\|^2} \int_R \omega\, |u_F(j\omega)|^2 \, d\omega. \tag{4.6}$$

A time-frequency uncertainty measure may be defined as the product of the uncertainties in both position and frequency according to [14]

$$U = \sigma_\omega^2 \, \sigma_t^2. \tag{4.7}$$

The uncertainty measure in equation (4.7) is invariant under the following operations [14]:

(i) Translation $u(t) \to u(t - t_o)$,

(ii) Dilation $u(t) \to \frac{1}{\sqrt{a}} u(\frac{t}{a})$,

(iii) Modulation $u(t) \to e^{j\omega t} u(t)$.

Due to the reciprocal spreading nature of the Fourier transform, no signal can have arbitrarily small uncertainty simultaneously in time and frequency. The uncertainty measure equation (4.7) has a lower limit of $1/4\pi$, i.e.,

$$\sigma_\omega^2 \, \sigma_t^2 \geq \frac{1}{4\pi}. \tag{4.8}$$

The result in equation (4.8) is known as the Heisenberg-Weyl uncertainty principle [14]. The lower limit in the Heisenberg-Weyl uncertainty principle is achieved by the Gaussian

$$\sqrt{2\pi} e^{-\frac{t^2}{4}} \tag{4.9}$$

or by any composition of translation, dilation, or multiplication of the Gaussian by a complex number of magnitude one [14].

4.2 The Phase Plane for Discrete-Time Finite Duration Signals

Let $u(n)$ be a signal in the space of square summable sequences defined on the finite abelian group $\Big\{0, 1, \ldots, N-1\Big\}$. The norm of $u(n)$ is given by [14]

$$\|u\|^2 = \sum_{n=0}^{N-1} |u(n)|^2. \tag{4.10}$$

The Fourier transform of $u(n)$ is given by [14]

$$u_F(k) = \sum_{n=0}^{N-1} u(n)\, e^{-j\frac{2\pi}{N}kn}. \tag{4.11}$$

The set of all points $(n,k) \in \Big\{0,1,\ldots,N-1\Big\}^2$ defines the phase plane. The uncertainty in position and frequency of discrete-time signals can be defined similarly to continuous time signals. However, such direct extension might generate a measure that is not invariant under translation and modulations [44]. Therefore, an alternative measure is needed for discrete-time signals. The translation and modulation of discrete-time signals are defined follows [14]:

(i) Translation $u(n) \rightarrow u(\langle n - n_o \rangle_N)$,

(ii) Modulation $u(n) \rightarrow e^{j\frac{2\pi}{N}kn} u(n)$,

where $\langle n \rangle_N$ means n modulo N.

Entropy is a measure of how much a probability distribution is spread and is defined as follows for a discrete-time signal with $\|u\|^2 = 1$ [14]:

$$H(u) = -\sum_{n=0}^{N-1} |u(n)|^2 \log |u(n)|^2. \tag{4.12}$$

If $|u(n)|^2$ is considered a probability distribution, then the entropy of u is the mean of $\log |u(n)|^2$. The entropy based uncertainty of $u(n)$ in frequency is given by [14]

$$H(u_F) = -\sum_{k=0}^{N-1} |u_F(k)|^2 \log |u_F(k)|^2. \tag{4.13}$$

A joint time-frequency uncertainty measure may be defined as a weighted sum of the uncertainties in time and frequency [14]:

$$H_p(n) = pH(u) + (1-p)H(u_F). \tag{4.14}$$

For $p = 1/2$, equation (4.14) is known as the digital Hirschman measure [21]. The parameter p allows for a trade-off between concentration in time and in frequency. When $p = 1$, the frequency is ignored. When $p = 0$, the time is ignored. Before stating the lower limit of $H_{1/2}$ and the signals that achieve that limit (the minimizers), periodization is defined [39].

Definition 1. (periodization) For $N = KL$, the periodization of $v \in C^K$ is $x \in C^N$ defined as $x(sK+n) = \frac{1}{\sqrt{L}}v(n)$ for $0 \leq s \leq L-1$ and $0 \leq n \leq K-1$.

The lower limit of $H_{1/2}(u)$ may be stated as follows [14, 39].

Conjecture 1. The minimal value of $H_{1/2}(u)$ is $\frac{1}{2}\log N$. The only sequences $u \in C^N$ for which $H_{1/2}(u)$ is minimal are obtained from the Kronecker delta by applying any composition of periodization, translation, modulation, the DFT, or multiplication by a complex number of magnitude 1.

The first part of **Conjecture 1** is called the Hirschman uncertainty principle. The minimizers in **Conjecture 1** are different from Gaussians, which are the minimizers of the energy based uncertainty measure [14]. The minimizers of the Hirschman uncertainty measure $H_{1/2}(u)$ are of special interest since they form a basis that is useful for representing a wide variety of signals. Based on group theory, Przebinda, et. al., presented an original a proof of the Hirschman uncertainty principle [13]. Before the theorem is stated, some notations are quoted from [13].

Let A be an abelian group with respect to addition. The Heisenberg group of degree one with coefficients in A is the group $G_1(A)$ of all matrices of the form [13]

$$\begin{bmatrix} 1 & x & z \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix}, \qquad x, y, z \in A. \tag{4.15}$$

Let

$$\chi(a) = e^{j\frac{2\pi}{N}a}, \qquad a \in A. \tag{4.16}$$

This is a unitary character of the additive group. Let $L^2(A)$ be the set of all $u : A \to C$ that are square summable. Let

$$\rho(x, y, z)u(a) = \chi(ay + z)u(a + x). \tag{4.17}$$

Then ρ is a group homomorphism from $G_1(A)$ to the group of unitary operators in $L^2(A)$ [13]. In simple terms, x represents translation, y represents modulation, and z represents multiplication by a unit-magnitude constant. Consider $u \in L^2(A)$ with $\|u\|^2 = 1$ equivalent to $v = \lambda u$ where $|\lambda| = 1$. As $H(u) = H(v)$ and $H_p(u) = H_p(v)$ for equivalent u and v, H and H_p are defined on the equivalence classes. This set of equivalence classes, which is denoted $P(A)$, forms a complex projective space. Now, after listing the necessary notation, the main theorem in [13] is stated.

Theorem 1.

(a) If $u \in P(A)$ then $H_{1/2} = \frac{1}{2}\log|A|$.

(b) The set of all vectors $u \in P(A)$ and $H_{1/2} = \frac{1}{2}\log|A|$ coincide with the union of the orbits

$$\rho(x,y,z)\frac{1}{\sqrt{|B|}}1_B(B \text{ — a subgroup of } A)^1. \tag{4.18}$$

(c) Each orbit is an orthogonal basis of $L^2(A)$.

(d) The set of vectors $u \in P(A)$ and $H_{1/2} = \frac{1}{2}\log|A|$ for all $0 \leq p \leq 1$ is not empty if and only if $|A|$ is a square. In this case, this set coincides with the orbit in equation (4.18) for the unique subgroup $B \subseteq A$ of cardinality $|B| = \sqrt{|A|}$.

Part (d) indicates that Hirschman uncertainty minimizers with $H_{1/2} = \frac{1}{2}\log|A|$ for all $0 \leq p \leq 1$ exist only when N is a perfect square. These minimizers form bases that define a unitary transform called the Hirschman optimal transform (HOT). The 3^2-point HOT matrix is explicitly given below [13]:

$$\begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{I}_{3\times3} \\ \mathbf{I}_{3\times3} & e^{-j\frac{2\pi}{3}}\mathbf{I}_{3\times3} & e^{-j\frac{2\pi}{3}2}\mathbf{I}_{3\times3} \\ \mathbf{I}_{3\times3} & e^{-j\frac{2\pi}{3}2}\mathbf{I}_{3\times3} & e^{-j\frac{2\pi}{3}4}\mathbf{I}_{3\times3} \end{bmatrix}. \tag{4.19}$$

Equation (4.19) shows that the K^2-point HOT is equivalent to taking the K-point DFT of the K polyphase components of $u(n)$ separately.

The algorithms that we proposing are best analyzed if the relation between the HOT and DFT is presented in matrix form. This matrix form is shown in Figure 4.1, where $\mathbf{I}_0$, $\mathbf{I}_1$,..., $\mathbf{I}_{K-1}$ are $K \times K^2$ matrices such that multiplication of a vector with $\mathbf{I}_i$ produces the i^{th} polyphase component of the vector. The matrix $\mathbf{I}_K$ is formed from

$^1 1_B$ is the indicator function of B.

$\mathbf{I}_0$, $\mathbf{I}_1$,..., $\mathbf{I}_{K-1}$, i.e.,

$$\mathbf{I}_K = \begin{bmatrix} \mathbf{I}_0 \\ \mathbf{I}_1 \\ \vdots \\ \mathbf{I}_{K-2} \\ \mathbf{I}_{K-1} \end{bmatrix}. \tag{4.20}$$

Since the rows of $\left\{\mathbf{I}_i\right\}$ are taken from the rows of the $K^2 \times K^2$ identity matrix, multiplications with such matrices does not impose any computational burden. The K^2-point HOT requires fewer computations than does the K^2-point DFT. The computational efficiency of the HOT was used to implement fast convolution algorithms in [15]. When K is an integer power of 2, $K^2 \log_2 K$ (complex) multiplications are needed to compute the HOT, which is half the number required when computing the DFT. For the special case $K = 3$, we have

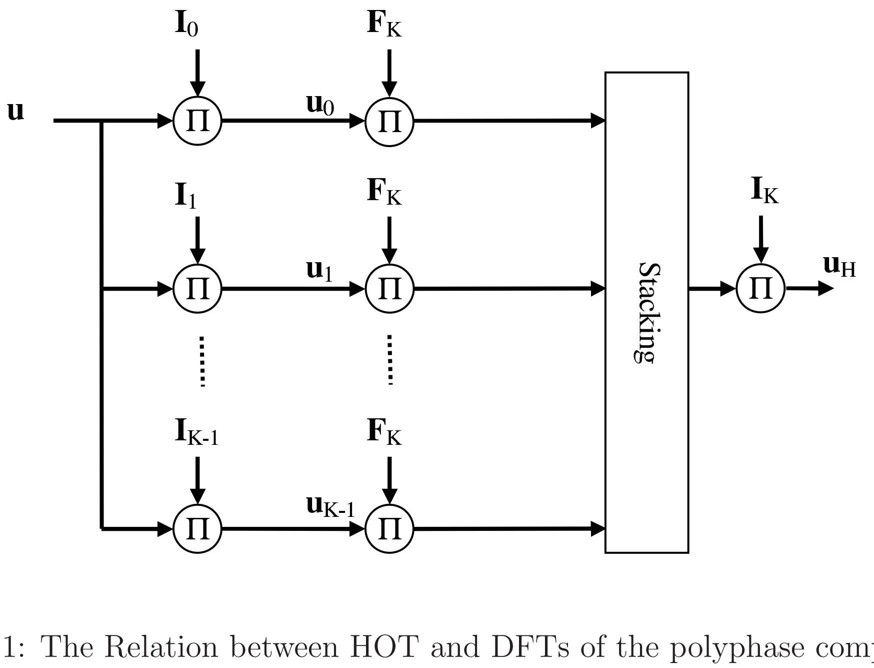


Figure 4.1: The Relation between HOT and DFTs of the polyphase components.

$$\mathbf{I}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \tag{4.21}$$

$$\mathbf{I}_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \tag{4.22}$$

$$\mathbf{I}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.23}$$

The K^2-point HOT matrix is denoted by $\mathbf{H}$. It satisfies the following:

$$\mathbf{H}\mathbf{H}^H = K\mathbf{I}_{K^2 \times K^2}, \tag{4.24}$$

$$\mathbf{H} = \mathbf{H}^T. \tag{4.25}$$

Chapter 5

Hirschman Optimal Transform LMS Algorithm

HOT bases are optimally compact in time and frequency, i.e., their spread in time is optimal without compromising their spread in frequency. The compactness of the HOT basis in time is manifested by the many zeros in the transform. Therefore, the HOT transform is computationally efficient [15]. The compactness in frequency is also desirable to filter the input into disjoint frequency bands [4]. The frequency bands of the HOT basis are wider than the DFT and DCT bands, which constitutes a trade-off of performance against computational efficiency. In this chapter, we present new transform domain LMS algorithms based on the HOT. Since the frequency bands of the HOT basis are wider than those of the DFT basis, the performance of the HOT LMS algorithms are theoretically and numerically investigated to study the trade-off of performance against computational efficiency.

5.1 Development of the Basic Algorithm

Let $u(n)$ be the filter input and $\mathbf{u}(n)$ be the tap-input vector. The HOT transform of $\mathbf{u}(n)$ is denoted by $\mathbf{u}_H(n)$ and the output of the filter is given by $y(n) = \mathbf{w}_H^T(n)\mathbf{u}_H(n)$. In the HOT LMS algorithm, the filter tap-weight vector $\mathbf{w}_H(n)$ is updated using

$$\mathbf{w}_H(n+1) = \mathbf{w}_H(n) + \alpha\, \mathbf{\Lambda}^{-1}(n)\, \mathbf{u}_H^*(n)\, e(n), \tag{5.1}$$

where the filter error is given by $e(n) = d(n) - y(n)$. The diagonal matrix $\mathbf{\Lambda}(n)$ contains the estimated power of the HOT coefficients and is updated using the recursion

$$\mathbf{\Lambda}(n) = \mathbf{\Lambda}(n-1) + \frac{1}{n}\Big(\mathbf{U}_H^*(n-1)\mathbf{U}_H(n-1) - \mathbf{\Lambda}(n-1)\Big), \tag{5.2}$$

where α is a constant given by $1/2K^2$, K^2 is the filter length, and $\mathbf{U}_H(n)$ is a diagonal matrix that contains the elements of $\mathbf{u}_H(n)$. We found that the HOT of the tap-input vector can be calculated efficiently using the following sliding HOT recursion:

$$\mathbf{u}_H(n) = \mathbf{I}_K \begin{bmatrix} \mathbf{D}\mathbf{I}_{K-1}\mathbf{u}_H(n-1) + \big(u(n) - u(n-K^2)\big)\mathbf{I}_{K\times K} \\ \mathbf{I}_0\mathbf{u}_H(n-1) \\ \mathbf{I}_1\mathbf{u}_H(n-1) \\ \vdots \\ \mathbf{I}_{K-2}\mathbf{u}_H(n-1) \end{bmatrix}, \tag{5.3}$$

where

$$\mathbf{D} = \mathrm{Diag}\left\{1, e^{-j\frac{2\pi}{K}}, \ldots, e^{-j\frac{2\pi}{K}(K-1)}\right\}. \tag{5.4}$$

Since the rows of $\left\{\mathbf{I}_i\right\}$ are taken from the $K^2 \times K^2$ identity matrix, multiplications with such matrices do not impose any computational burden. The sliding HOT requires only K multiplications; it is K times more efficient than the sliding DFT, which requires $N = K^2$ multiplications [31].

5.2 Asymptotic Autocorrelation Matrix in the HOT Domain

The DFT perfectly diagonalizes circulant autocorrelation matrices. In general, only a periodic wide-sense stationary (WSS) process has such an autocorrelation matrix. Asymptotically however, the input autocorrelation matrix in most application can be approximated by a circulant matrix in the sense that both matrices have the same eigenvalue distribution [8]. This result was proven by many researchers [17, 8, 22, 12], each with a different approach. Pearl [17], for example, formed a diagonal matrix from the diagonal elements of the DFT autocorrelation matrix and then inverse transformed the resulting matrix. The resulting matrix is called the DFT autocorrelation matrix approximation. The Hilbert-Schmidt norm[1] of the difference between the autocorrelation matrix and the DFT autocorrelation matrix approximation was used as measure for the diagonalizing power of the DFT. Pearl showed that as $N \to \infty$ this norm goes to zero as $\mathcal{O}(1/N)$ for square summable autocorrelation sequences.

To investigate the decorrelation power of the HOT basis, we used the same measure. Let $u(n)$ be a WSS random process with autocorrelation sequence $r(n)$ and autocorrelation matrix $\mathbf{R}$. The HOT autocorrelation matrix approximation is de-

[1]The Hilbert-Schmidt norm of a square matrix is defined as $|\mathbf{T}|^2 = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|T_{ij}|^2$.

noted by $\tilde{\mathbf{R}}$ and defined as [17]

$$\tilde{\mathbf{R}} = \mathbf{H}\,\mathbf{S}\,\mathbf{H}^H, \tag{5.5}$$

where $\mathbf{S}$ is the diagonal matrix that contains the diagonal elements of the HOT autocorrelation matrix $\mathbf{H}^H\mathbf{R}\mathbf{H}$, i.e.,

$$S_{ij} = \begin{cases} \left[\mathbf{H}^H\mathbf{R}\mathbf{H}\right]_{ij} & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases} \tag{5.6}$$

where $\mathbf{H}$ is the HOT matrix. The $(i,j)^{\text{th}}$ element of $\tilde{\mathbf{R}}$ is given by

$$\tilde{R}_{ij} = \left[\mathbf{H}^H\mathbf{R}\mathbf{H}\right]_{ij} = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1} H_{ik} S_{kl} H^H_{lj}. \tag{5.7}$$

Since the matrix $\mathbf{S}$ is diagonal,

$$\tilde{R}_{ij} = \sum_{k=0}^{N-1} H_{ik} S_{kk} H^H_{kj}. \tag{5.8}$$

Substituting equation (5.6) into equation (5.8), we have that

$$\tilde{R}_{ij} = \sum_{k=0}^{N-1} H_{ik} \left[\mathbf{H}^H\mathbf{R}\mathbf{H}\right]_{kk} H^H_{kj}. \tag{5.9}$$

The diagonal elements of the HOT autocorrelation matrix are given by

$$\left[\mathbf{H}^H\mathbf{R}\mathbf{H}\right]_{kk} = \sum_{l=0}^{N-1}\sum_{m=0}^{N-1} H^H_{kl}\, r(l-m)\, H_{mk}. \tag{5.10}$$

Substituting this result in equation (5.9) gives

$$\tilde{R}_{ij} = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\sum_{m=0}^{N-1} H_{ik}H_{kl}^{H}\, r(l-m)\, H_{mk}H_{kj}^{H}. \tag{5.11}$$

Since the HOT matrix is symmetric, we have then that

$$\tilde{R}_{ij} = \sum_{l=0}^{N-1}\sum_{m=0}^{N-1} r(l-m) \sum_{k=0}^{N-1} H_{ik}H_{kl}^{*}H_{mk}H_{kj}^{*}. \tag{5.12}$$

To find a functional form for the HOT basis, we used the periodic Kronecker delta and periodic ramp sequences, which are defined by ($0 \le n < \sqrt{N}$)

$$\delta(n) = \begin{cases} 1 & \text{if } n = 0, \\ 0 & \text{otherwise,} \end{cases} \tag{5.13}$$

$$s(n) = n. \tag{5.14}$$

The k^{th} HOT basis signal is given by

$$h_k(n) = \frac{1}{\sqrt[4]{N}}\, \delta(n-k)\, e^{j\frac{2\pi}{N}(n-s(n))\left[\frac{k}{\sqrt{N}}\right]}, \tag{5.15}$$

where $[x]$ is the largest integer function. Using equation (5.15) in equation (5.12), we obtain

$$\begin{aligned} \tilde{R}_{ij} \;=\; & \frac{1}{N}\sum_{l=0}^{N-1}\sum_{m=0}^{N-1} r(l-m) \sum_{k=0}^{N-1} \delta(i-k)\, e^{j\frac{2\pi}{N}(i-s(i))\left[\frac{k}{\sqrt{N}}\right]}\, \delta(l-k)\, e^{-j\frac{2\pi}{N}(l-s(l))\left[\frac{k}{\sqrt{N}}\right]} \\ & \times \delta(m-k)\, e^{j\frac{2\pi}{N}(m-s(m))\left[\frac{k}{\sqrt{N}}\right]}\, \delta(j-k)\, e^{-j\frac{2\pi}{N}(j-s(j))\left[\frac{k}{\sqrt{N}}\right]}, \end{aligned} \tag{5.16}$$

which, after routine algebra may be simplified as

$$\tilde{R}_{ij} = \frac{1}{N} \sum_{l=0}^{N-1} \sum_{m=0}^{N-1} \sum_{k=0}^{N-1} r(l-m)\delta(i-k)\delta(l-k)\delta(m-k)$$

$$\times \delta(j-k)\, e^{j\frac{2\pi}{N}(i-s(i)-l+s(l)+m-s(m)-j+s(j))\left[\frac{k}{\sqrt{N}}\right]}. \tag{5.17}$$

Since $\tilde{\mathbf{R}}$ is Toeplitz and symmetric, only the first row is needed, which is given by

$$\tilde{R}_{0j} = \frac{1}{N} \sum_{l=0}^{N-1} \sum_{m=0}^{N-1} \sum_{k=0}^{N-1} r(l-m)\delta(k)\delta(l-k)\delta(m-k)$$

$$\times \delta(j-k) e^{j\frac{2\pi}{N}(-l+s(l)+m-s(m)-j+s(j))\left[\frac{k}{\sqrt{N}}\right]}. \tag{5.18}$$

After the change of variable $k \to k\sqrt{N}$, equation (5.18) simplifies to

$$\tilde{R}_{0j} = \frac{1}{N} \sum_{l=0}^{N-1} \sum_{m=0}^{N-1} r(l-m) \sum_{k=0}^{\sqrt{N}-1} \delta(l-k\sqrt{N})\delta(m-k\sqrt{N})$$

$$\times \delta(j-k\sqrt{N})\, e^{j\frac{2\pi}{N}(-l+s(l)+m-s(m)-j+s(j))k}. \tag{5.19}$$

But we have $\delta(j-k\sqrt{N}) = \delta(j)$, so

$$\tilde{R}_{0j} = \frac{1}{N}\delta(j) \sum_{l=0}^{N-1} \sum_{m=0}^{N-1} \delta(l)\delta(m) r(l-m) \sum_{k=0}^{\sqrt{N}-1} e^{j\frac{2\pi}{N}(-l+s(l)+m-s(m)-j+s(j)k}. \tag{5.20}$$

Using the property which is only valid for periodic Kronecker delta,

$$\delta(l)\delta(m) = \delta(l-m), \tag{5.21}$$

we have that

$$\tilde{R}_{0j} = \frac{1}{N}\delta(j) \sum_{l=0}^{N-1} \sum_{m=0}^{N-1} \delta(l-m) r(l-m) \sum_{k=0}^{\sqrt{N}-1} e^{j \frac{2\pi}{\sqrt{N}} \frac{(m-l-j)}{\sqrt{N}} k}. \tag{5.22}$$

Using the orthogonality of the DFT basis, e.g.,

$$\sum_{k=0}^{\sqrt{N}-1} e^{j \frac{2\pi}{\sqrt{N}} \frac{(m-l-j)}{\sqrt{N}} k} = \sqrt{N}\, \delta\left(\frac{m-l-j}{\sqrt{N}}\right), \tag{5.23}$$

the first row of $\tilde{\mathbf{R}}$ is given by

$$\tilde{R}_{0j} = \frac{1}{\sqrt{N}}\delta(j) \sum_{l=0}^{N-1} \sum_{m=0}^{N-1} \delta(l-m) r(l-m) \delta\left(\frac{m-l-j}{\sqrt{N}}\right). \tag{5.24}$$

Then equation (5.24) can be simplified as

$$\tilde{R}_{0j} = \frac{1}{\sqrt{N}}\delta(j) \left\{ r(-j) \left(\frac{N-j}{\sqrt{N}}\right) + r(N-j) \left(\frac{j}{\sqrt{N}}\right) \right\}. \tag{5.25}$$

Since $\tilde{\mathbf{R}}$ is Toeplitz, it follows that

$$\begin{aligned} \tilde{R}_{ii+j} &= \tilde{R}_{0j}, \\ &= \frac{1}{\sqrt{N}}\delta(j) \left\{ r(-j) \left(\frac{N-j}{\sqrt{N}}\right) + r(N-j) \left(\frac{j}{\sqrt{N}}\right) \right\}, \end{aligned} \tag{5.26}$$

and

$$\tilde{R}_{ik} = \frac{1}{\sqrt{N}}\delta(k-i) \left\{ r(-(k-i)) \left(\frac{N-(k-i)}{\sqrt{N}}\right) + r(N-(k-i)) \left(\frac{k-i}{\sqrt{N}}\right) \right\}, \tag{5.27}$$

provided that $k - i \geq 0$. Since $\tilde{\mathbf{R}}$ is also symmetric for real signals, we have

$$\tilde{R}_{ik} = \delta(k-i)\left\{ r(k-i) + \frac{|k-i|}{N}\Big(r(N - |k-i|) - r(k-i) \Big) \right\} \tag{5.28}$$

for any $0 \leq k \leq N-1$ and $0 \leq i \leq N-1$. Equation (5.28) shows that the autocorrelation matrix $\tilde{\mathbf{R}}$ has many zero entries. In fact, each row has only $\sqrt{N}$ non-zero entries. This result is inherited from the HOT basis (where each basis signal has the same number of zero valued samples). For $N = 9$, $\tilde{\mathbf{R}}$ is shown explicitly in equation (5.29).

$$\tilde{\mathbf{R}} = \begin{bmatrix} r_0 & 0 & 0 & r_{-1} & 0 & 0 & r_{-2} & 0 & 0 \\ 0 & r_0 & 0 & 0 & r_{-1} & 0 & 0 & r_{-2} & 0 \\ 0 & 0 & r_0 & 0 & 0 & r_{-1} & 0 & 0 & r_{-2} \\ r_1 & 0 & 0 & r_0 & 0 & 0 & r_{-1} & 0 & 0 \\ 0 & r_1 & 0 & 0 & r_0 & 0 & 0 & r_{-1} & 0 \\ 0 & 0 & r_1 & 0 & 0 & r_0 & 0 & 0 & r_{-1} \\ r_2 & 0 & 0 & r_1 & 0 & 0 & r_0 & 0 & 0 \\ 0 & r_2 & 0 & 0 & r_1 & 0 & 0 & r_0 & 0 \\ 0 & 0 & r_2 & 0 & 0 & r_1 & 0 & 0 & r_0 \end{bmatrix}, \tag{5.29}$$

where

$$r_0 = r(0), \tag{5.30}$$

$$r_{\pm 1} = \frac{2r(\pm 3) + r(\mp 6)}{3}, \tag{5.31}$$

$$r_{\pm 2} = \frac{2r(\mp 3) + r(\pm 6)}{3}. \tag{5.32}$$

We are interested in calculating the Hilbert-Schmidt norm of $\mathbf{R}-\tilde{\mathbf{R}}$. Since $R_{i,j} = r(i-j)$, it follows that

$$\begin{aligned} R_{i,j} - \tilde{R}_{i,j} &= r(i-j) - \delta(i-j)\Big\{ r(i-j) + \frac{|i-j|}{N}\Big(r(N-|i-j|) - r(i-j)\Big)\Big\} \\ &= \Big(1-\delta(i-j)\Big) r(i-j) - \delta(i-j)\frac{|i-j|}{N}\Big(r(N-|i-j|) - r(i-j)\Big). \end{aligned} \tag{5.33}$$

Equation (5.33) is used to calculate the Hilbert-Schmidt norm of $\mathbf{R}-\tilde{\mathbf{R}}$ according to

$$\begin{aligned} \Big|\mathbf{R}-\tilde{\mathbf{R}}\Big|^2 = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\Big| &\Big(1-\delta(i-j)\Big) r(i-j) - \\ &\delta(i-j)\frac{|i-j|}{N}\Big(r(N-|i-j|) - r(i-j)\Big)\Big|^2. \end{aligned} \tag{5.34}$$

It can be shown that

$$\begin{aligned} &\Big|\Big(1-\delta(i-j)\Big) r(i-j) - \delta(i-j)\frac{|i-j|}{N}\Big(r(N-|i-j|) - r(i-j)\Big)\Big|^2 \\ &= \Big(1-\delta(i-j)\Big)\Big|r(i-j)\Big|^2 + \delta(i-j)\Big|\frac{|i-j|}{N}\Big(r(N-|i-j|) - r(i-j)\Big)\Big|^2. \end{aligned} \tag{5.35}$$

Using equation (5.35) in equation (5.34), we obtain

$$\begin{aligned} \Big|\mathbf{R}-\tilde{\mathbf{R}}\Big|^2 = &\frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\Big(1-\delta(i-j)\Big)\Big|r(i-j)\Big|^2 \\ &+\frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\delta(i-j)\Big|\frac{|i-j|}{N}\Big(r(N-|i-j|) - r(i-j)\Big)\Big|^2. \end{aligned} \tag{5.36}$$

The second term in equation (5.36) can be split into two terms according to

$$\begin{aligned}\left|\mathbf{R}-\tilde{\mathbf{R}}\right|^2 &= \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\Big(1-\delta(i-j)\Big)\Big|r(i-j)\Big|^2 \\ &+\frac{1}{N^3}\sum_{i=0}^{N-1}\sum_{\substack{j=0\\ j<i}}^{N-1}\delta(i-j)\left|\frac{|i-j|}{N}\Big(r(N-|i-j|)-r(i-j)\Big)\right|^2 \\ &+\frac{1}{N^3}\sum_{i=0}^{N-1}\sum_{j>i}^{N-1}\delta(i-j)\left|\frac{|i-j|}{N}\Big(r(N-|i-j|)-r(i-j)\Big)\right|^2. \qquad (5.37)\end{aligned}$$

By a change of variable, equation (5.37) may be further simplified as

$$\begin{aligned}\left|\mathbf{R}-\tilde{\mathbf{R}}\right|^2 &= \frac{1}{N}\sum_{k=-(N-1)}^{N-1}(N-k)\Big(1-\delta(k)\Big)\Big|r(k)\Big|^2 \\ &+\frac{1}{N^3}\sum_{k=0}^{N-1}\delta(k)k^2(N-k)\Big(r(N-k)-r(k)\Big)^2 \\ &+\frac{1}{N^3}\sum_{k=0}^{N-1}\delta(k)k^2(N-k)\Big(r(N-k)-r(k)\Big)^2. \qquad (5.38)\end{aligned}$$

The third term in equation (5.38) can be shown to be equal to

$$\begin{aligned}&\frac{1}{N^3}\sum_{k=0}^{N-1}\delta(k)k^2(N-k)\Big(r(N-k)-r(k)\Big)^2 \\ &\qquad = \frac{1}{N^3}\sum_{k=0}^{N-1}\delta(k)k(N-k)^2\Big(r(N-k)-r(k)\Big)^2. \qquad (5.39)\end{aligned}$$

Therefore, equation (5.38) may be rewritten as

$$\begin{aligned}\left|\mathbf{R}-\tilde{\mathbf{R}}\right|^2 &= \frac{1}{N}\sum_{k=-(N-1)}^{N-1}(N-k)\Big(1-\delta(k)\Big)\Big|r(k)\Big|^2 \\ &+\frac{1}{N^2}\sum_{k=0}^{N-1}\delta(k)k(N-k)\Big(r(N-k)-r(k)\Big)^2. \qquad (5.40)\end{aligned}$$

The corresponding Hilbert-Schmidt norm for the DFT case is given by [17]

$$\left|\mathbf{R} - \tilde{\mathbf{R}}\right|^2 = \frac{1}{N^2} \sum_{k=0}^{N-1} k(N-k)\Big(r(k) - r(N-k)\Big)^2. \tag{5.41}$$

The second term in equation (5.40) is almost the same as the result in equation (5.41), except for the appearance of $\delta(k)$ in the former. Therefore, the second term in equation (5.40) converges to zero faster than the result in equation 5.41 for square summable autocorrelation sequences. However, the first term in equation (5.40) does not converge to zero as $N \to \infty$. This term is plotted in Figure 5.1 for $r(k) = 1/k^2$. Also, we found that for the same autocorrelation sequence,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=-(N-1)}^{N-1} (N-k)\Big(1 - \delta(k)\Big)\Big|r(k)\Big|^2 = \frac{\pi^4}{45}. \tag{5.42}$$

The reason for this nonvanishing Hilbert-Schmidt norm is the zeros in the HOT basis which leads to many zeros in $\tilde{\mathbf{R}}$ (see equation (5.29) for the case of $N = 9$).

The HOT does not diagonalize the autocorrelation matrix even for large filter lengths. This result shows that the effect of the HOT on the autocorrelation matrix is more subtle and more analysis is needed to fully understand the convergence modes of the HOT LMS algorithm.

5.3 Convergence Analysis of the HOT LMS Adaptive Filter

More information can be obtained about the performance of the HOT LMS filter by looking directly at the input autocorrelation in the HOT domain. In this section, we

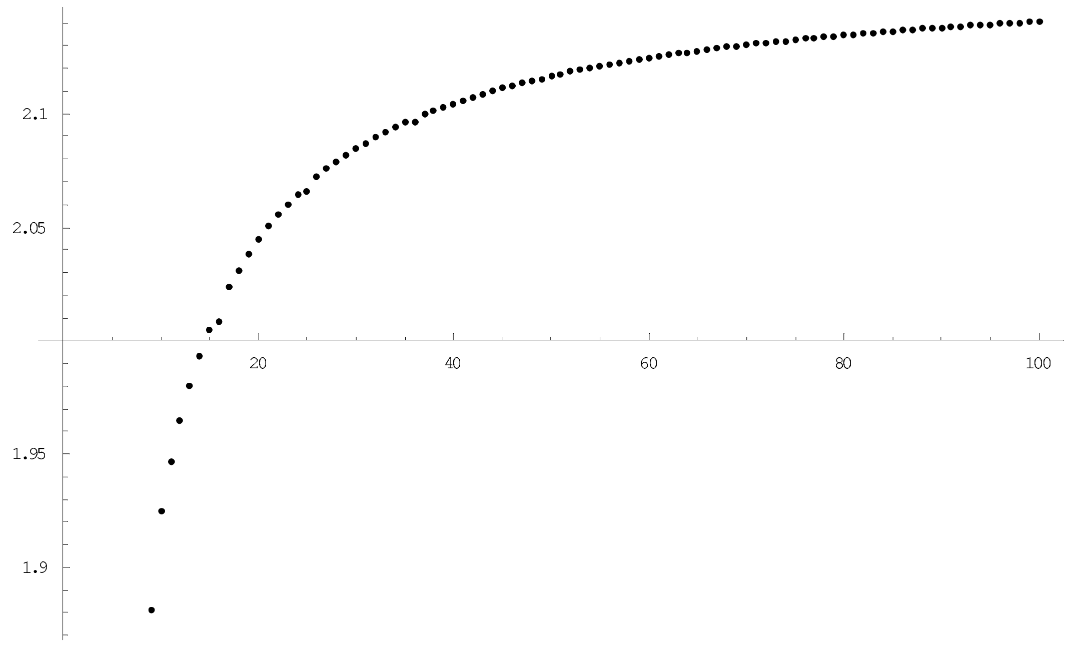


Figure 5.1: Plot of the second term of the Hilbert-Schmidt norm of $\mathbf{R} - \tilde{\mathbf{R}}$ versus N for $r(k) = 1/k^2$.

present a detailed convergence analysis of the HOT LMS algorithm. we show that the autocorrelation matrix in the HOT domain is asymptotically block diagonal and that the HOT LMS algorithm adjusts the learning rate of each block to improve the convergence speed of the adaptive filter as compared to the standard LMS algorithm.

The filter weight recursion can be written as

$$\mathbf{w}_H(n+1) = \mathbf{w}_H(n) + \alpha\,\mathbf{\Lambda}^{-1}\left(d(n) - \mathbf{u}_H^T(n)\mathbf{w}_H(n)\right)\mathbf{u}_H^*(n), \tag{5.43}$$

or equivalently

$$\boldsymbol{\epsilon}_H(n+1) = \left(\mathbf{I}_{K^2\times K^2} - \alpha\,\mathbf{\Lambda}^{-1}\mathbf{u}_H^*(n)\mathbf{u}_H^T(n)\right)\boldsymbol{\epsilon}_H(n) - \alpha\,\mathbf{\Lambda}^{-1}\mathbf{u}_H^*(n)e^o(n), \tag{5.44}$$

where $\boldsymbol{\epsilon}_H(n) = \mathbf{w}_H^o - \mathbf{w}_H(n)$ and $\mathbf{w}_H^o$ is the Wiener optimal filter in the HOT domain.

Taking the expectation of equation (5.44) and assuming that the filter weights and the input are statistically independent (this assumption directly follow from the small step size statistical LMS theory presented in chapter 2), the convergence of the weight vector in the mean is governed by the recursion

$$E\boldsymbol{\epsilon}_H(n+1) = \Big(\mathbf{I}_{K^2\times K^2} - \alpha\,\boldsymbol{\Lambda}^{-1}E\mathbf{u}_H^*(n)\mathbf{u}_H^T(n)\Big)E\boldsymbol{\epsilon}_H(n). \tag{5.45}$$

Let $\mathbf{H}$ be the HOT matrix. It can be easily shown that

$$\mathbf{R}_H = E\mathbf{u}_H^*(n)\mathbf{u}_H^T(n) = \mathbf{H}^H\mathbf{R}\mathbf{H}, \tag{5.46}$$

where $\mathbf{R} = E\mathbf{u}^*(n)\mathbf{u}^T(n)$ is the autocorrelation matrix of the tap-input vector. Now we look at the structure of $\mathbf{R}_H$. Let $\mathbf{F}_K$ be the K-point DFT matrix. Then

$$\mathbf{H} = \mathbf{I}_K \begin{bmatrix} \mathbf{F}_K & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_K & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{F}_K \end{bmatrix} \mathbf{I}_K. \tag{5.47}$$

The HOT autocorrelation matrix $\mathbf{R}_H$ can be written as

$$\mathbf{R}_H = \mathbf{I}_K \begin{bmatrix} \mathbf{F}_K^H & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_K^H & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{F}_K^H \end{bmatrix} \mathbf{I}_K\mathbf{R}\mathbf{I}_K \begin{bmatrix} \mathbf{F}_K & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_K & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{F}_K \end{bmatrix} \mathbf{I}_K. \tag{5.48}$$

Let $\mathbf{u}_i(n)$ be the i^{th} polyphase component of the tap-input vector. Then

$$\mathbf{u} = \mathbf{I}_K \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{K-1} \end{bmatrix} \tag{5.49}$$

and

$$\mathbf{R} = \mathbf{I}_K \begin{bmatrix} E\mathbf{u}_0^*\mathbf{u}_0 & E\mathbf{u}_0^*\mathbf{u}_1^T & \cdots & E\mathbf{u}_0^*\mathbf{u}_{K-1}^T \\ E\mathbf{u}_1^*\mathbf{u}_0^T & E\mathbf{u}_1^*\mathbf{u}_1^T & \cdots & E\mathbf{u}_1^*\mathbf{u}_{K-1}^T \\ \vdots & \vdots & \ddots & \vdots \\ E\mathbf{u}_{K-1}^*\mathbf{u}_0^T & E\mathbf{u}_{K-1}^*\mathbf{u}_1^T & \cdots & E\mathbf{u}_{K-1}^*\mathbf{u}_{K-1}^T \end{bmatrix} \mathbf{I}_K. \tag{5.50}$$

Substituting equation (5.50) into equation (5.48) gives

$$\begin{aligned} \mathbf{R}_H &= \mathbf{I}_K \begin{bmatrix} \mathbf{F}_K^H & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_K^H & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{F}_K^H \end{bmatrix} \begin{bmatrix} E\mathbf{u}_0^*\mathbf{u}_0^T & E\mathbf{u}_0^*\mathbf{u}_1^T & \cdots & E\mathbf{u}_0^*\mathbf{u}_{K-1}^T \\ E\mathbf{u}_1^*\mathbf{u}_0^T & E\mathbf{u}_1^*\mathbf{u}_1^T & \cdots & E\mathbf{u}_1^*\mathbf{u}_{K-1}^T \\ \vdots & \vdots & \ddots & \vdots \\ E\mathbf{u}_{K-1}^*\mathbf{u}_0^T & E\mathbf{u}_{K-1}^*\mathbf{u}_1^T & \cdots & E\mathbf{u}_{K-1}^*\mathbf{u}_{K-1}^T \end{bmatrix} \\ &\times \begin{bmatrix} \mathbf{F}_K & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_K & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{F}_K \end{bmatrix} \mathbf{I}_K, \end{aligned} \tag{5.51}$$

or equivalently

$$\mathbf{R}_H = \mathbf{I}_K \begin{bmatrix} \mathbf{F}_K^H E\mathbf{u}_0^*\mathbf{u}_0^T\mathbf{F}_K & \mathbf{F}_K^H E\mathbf{u}_0^*\mathbf{u}_1^T\mathbf{F}_K & \cdots & \mathbf{F}_K^H E\mathbf{u}_0^*\mathbf{u}_{K-1}^T\mathbf{F}_K \\ \mathbf{F}_K^H E\mathbf{u}_1^*\mathbf{u}_0^T\mathbf{F}_K & \mathbf{F}_K^H E\mathbf{u}_1^*\mathbf{u}_1^T\mathbf{F}_K & \cdots & \mathbf{F}_K^H E\mathbf{u}_1^*\mathbf{u}_{K-1}^T\mathbf{F}_K \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{F}_K^H E\mathbf{u}_{K-1}^*\mathbf{u}_0^T\mathbf{F}_K & \mathbf{F}_K^H E\mathbf{u}_{K-1}^*\mathbf{u}_1^T\mathbf{F}_K & \cdots & \mathbf{F}_K^H E\mathbf{u}_{K-1}^*\mathbf{u}_{K-1}^T\mathbf{F}_K \end{bmatrix} \mathbf{I}_K. \tag{5.52}$$

Looking at the (i,j)$^{\text{th}}$ block in equation (5.52), it follows that

$$\mathbf{F}_K^H E\mathbf{u}_i^*\mathbf{u}_j^T\mathbf{F}_K = \mathbf{F}_K^H \\ \times \begin{bmatrix} r(i-j) & r(i-j-K) & \cdots & r(i-j-(K-1)K) \\ r(i-j+K) & r(i-j) & \cdots & r(i-j-(K-2)K) \\ \vdots & \vdots & \ddots & \vdots \\ r(i-j+(K-1)K) & r(i-j+(K-1)K) & \cdots & r(i-j) \end{bmatrix} \mathbf{F}_K,$$

which is a Toeplitz matrix asymptotically diagonalized by the DFT matrix [17]. It can be easily verified that if each of these blocks is perfectly diagonal, then the autocorrelation matrix $\mathbf{R}_H$ is block diagonal. Asymptotically, the HOT LMS adaptive filter transforms the K^2 modes into K decoupled sets of modes. This result is summarized in equation (5.53) — $\left\{\mathbf{\Lambda}_i\right\}$ contains the diagonal elements of $\mathbf{R}_H$:

$$E\boldsymbol{\epsilon}_H(n+1) = E\boldsymbol{\epsilon}_H(n) - \alpha \begin{bmatrix} \mathbf{\Lambda}_0^{-1}\mathbf{R}_H^0 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_1^{-1}\mathbf{R}_H^1 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{\Lambda}_{K-1}^{-1}\mathbf{R}_H^{K-1} \end{bmatrix} E\boldsymbol{\epsilon}_H(n). \tag{5.53}$$

The HOT adaptive filter is expected to have an increased convergence rate, since it equalizes the learning rates of each block. This expectation was confirmed by simulations. Furthermore, the diagonal elements of Λ_i are identical; a formal proof of this fact is presented in the next section. This induces further reduction in the computational complexity of the HOT adaptive filter compared to the DFT adaptive filter since one need only to estimate the power of K HOT coefficients, whereas the DFT adaptive filter requires estimation of the power of all the K^2 DFT coefficients. The total number of multiplications required for the HOT LMS adaptive filter is $2K^2+3K+1$. For comparison, the computational complexities of the LMS and DFT LMS adaptive filters are $2N+1$ and $5N+1$, respectively. The HOT LMS algorithm requires less than half the number of multiplications required for the HOT LMS algorithm and "almost" the same number of multiplications as the LMS algorithm. The computational improvement of the HOT LMS algorithm over the DFT algorithm comes at the cost of a reduction in performance.

The previous theoretical predictions were verified through the following simulations. The matrix $\mathbf{R}_H$ was computed for first-order Markov signals with autocorrelation functions given by $r(k) = 0.9^{|k|}$. Figures 5.2 and 5.3 show the image representations for $\mathbf{R}_H$ with $K = 8$ and $K = 16$, respectively. It is evident that $\mathbf{R}_H$ is asymptotically block diagonal. Figures 5.2 and 5.3 indicate that the blocks of $\mathbf{R}_H$ are both symmetric and Toeplitz. The first property is easy to verify. The second property is proved in the next section.

The performance of the HOT LMS adaptive filer was simulated and compared with the performance of the standard LMS adaptive filter. The desired input was generated using the linear model $d(n) = w^o(n) * u(n) + e^o(n)$, where $e^o(n)$ is the

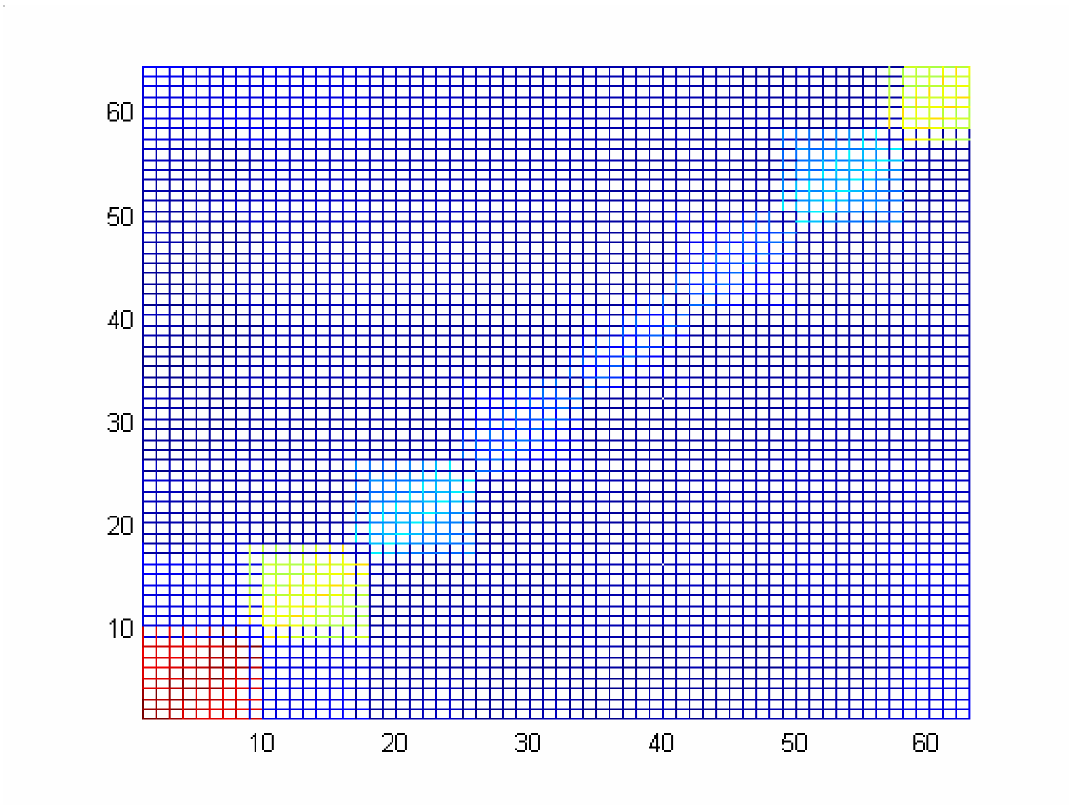


Figure 5.2: Image representation of the HOT autocorrelation matrix $\mathbf{R}_H$ with $K = 8$.

measurement white gaussian noise with variance 10^{-8}. The input was a first-order Markov signal with autocorrelation functions given by $r(k) = 0.9^{|k|}$. The filter was a 64-point lowpass filter with a cutoff frequency of $\pi/2$ rad. The learning curves of both the LMS and HOT LMS adaptive filters are shown in Figure 5.4. It is evident that the speed of convergence of the HOT LMS algorithm is higher than that of the LMS algorithm.

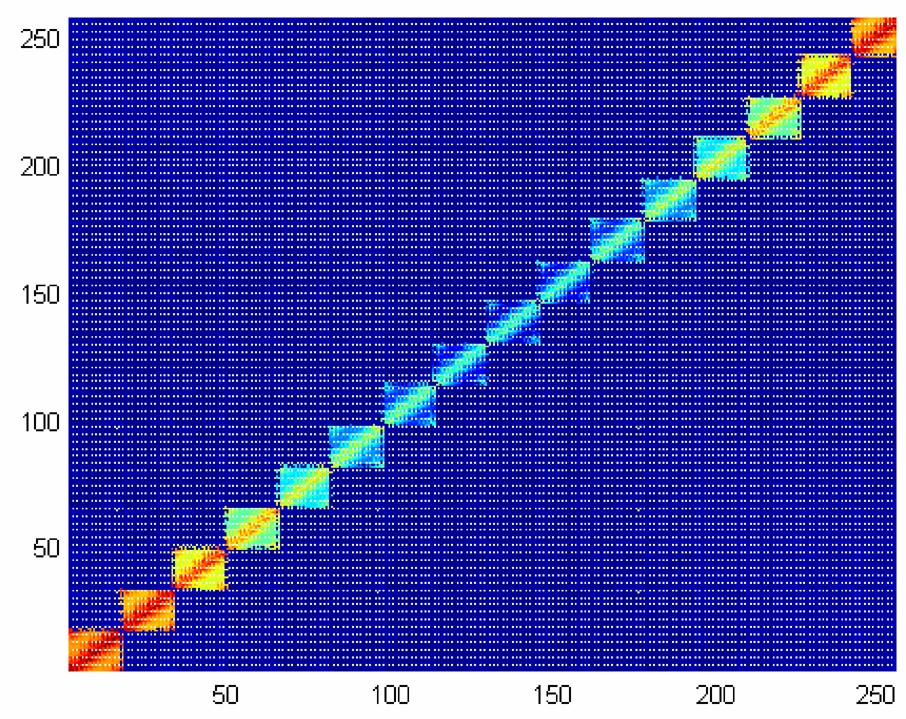


Figure 5.3: Image representation of the HOT autocorrelation matrix $\mathbf{R}_H$ with $K = 16$.

5.4 Self-Orthogonalizing HOT Adaptive Filter

The HOT LMS algorithm can be improved if the blocks of the HOT autocorrelation matrix can be efficiently diagonalized. Each block in the autocorrelation matrix is symmetric and Toeplitz. A formal proof of this fact is presented in **Theorem 1** on page 54. Therefore, each block of the HOT autocorrelation matrix can be efficiently diagonalized using a Levinson recursion that requires $\mathcal{O}(K^2)$ operations or using the technique given in [27] that requires $\mathcal{O}(K \log K)$ operations. With this modification, the HOT LMS algorithm becomes the self-orthogonalizing HOT LMS algorithm. Therefore, the self-orthogonalizing HOT LMS algorithm implements the

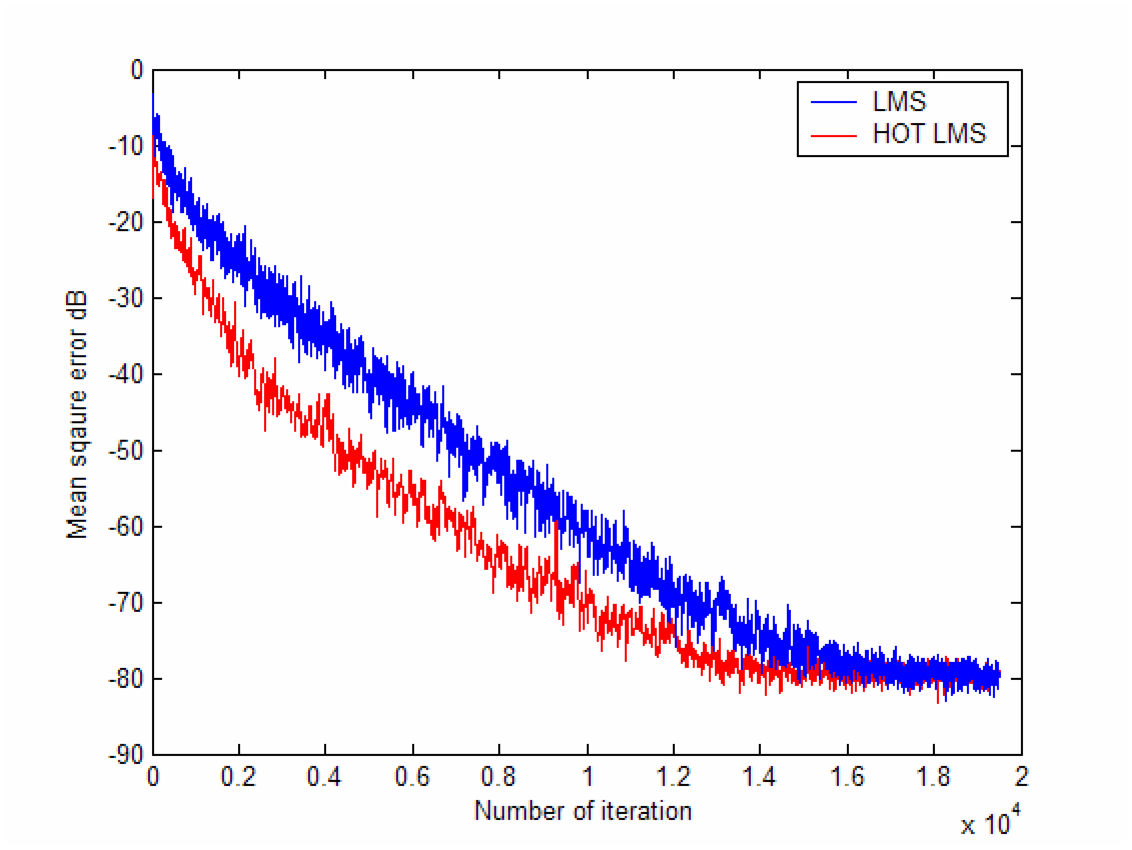


Figure 5.4: Learning curves for the LMS and HOT LMS algorithms.

HOT to block-wise diagonalize the autocorrelation matrix and then uses K self-orthogonalizing LMS sections [40] to diagonalize the blocks of the HOT autocorrelation matrix. Since there are K blocks in $\mathbf{R}_H$, it can be efficiently diagonalized with $\mathcal{O}(K^3)$ or $\mathcal{O}(K^2 \log_2 K)$ operations. The details of this new algorithm are described next.

Since the HOT autocorrelation matrix $\mathbf{R}_H$ of the tap-input vector is asymptotically block diagonal, the HOT adaptive filter convergence speed can be increased if the step size in equation (5.1) is replaced by the inverse of $\mathbf{R}_H$:

$$\mathbf{w}_H(n+1) = \mathbf{w}_H(n) + \alpha\, \mathbf{R}_H^{-1}\, \mathbf{u}_H^*(n)\, e(n). \tag{5.54}$$

Computing $\mathbf{R}_H^{-1}$ is computationally expensive and is replaced by the inverse of

$$\tilde{\mathbf{R}}_H = \begin{bmatrix} \mathbf{R}_{H0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{H1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{R}_{HK-1} \end{bmatrix}, \tag{5.55}$$

where $\mathbf{R}_{Hi}$ is the i^{th} diagonal block of $\mathbf{R}_H$.

Theorem 1. The set $\left\{\mathbf{R}_{H0}, \mathbf{R}_{H1}, \ldots, \mathbf{R}_{HK-1}\right\}$ is a set of Hermitian Toeplitz matrices.

Proof: That $\mathbf{R}_{Hi}$ is Hermitian follows directly from the fact that $\mathbf{R}_H$ is Hermitian. The second part can be proved by explicitly calculating the elements of $\mathbf{R}_{Hi}$. We have

$$[\mathbf{R}_{Hi}]_{lm} = E \sum_{k=0}^{N-1} h^*_{iK+l}(k) u(n-k) \sum_{k=0}^{N-1} h_{iK+m}(k) u(n-k), \tag{5.56}$$

where $0 \leq l < K$, $0 \leq m < K$, and $h_{iK+l}(k)$ is the $(iK+m)^{\text{th}}$ HOT basis signal given in equation (5.15). Equation (5.56) can be simplified to

$$[\mathbf{R}_{Hi}]_{lm} = \sum_{k=0}^{N-1} \sum_{k'=0}^{N-1} h^*_{iK+l}(k) h_{iK+m}(k') r(k-k'). \tag{5.57}$$

Substituting the HOT basis signal in equation (5.57),

$$\begin{aligned} [\mathbf{R}_{Hi}]_{lm} &= \sum_{k=0}^{N-1} \sum_{k'=0}^{N-1} \delta(k - iK - l) e^{-j\frac{2\pi}{N}(k - s(k))\left[\frac{iK+l}{\sqrt{N}}\right]} \delta(k' - iK - m) \\ &\quad \times e^{j\frac{2\pi}{N}(k' - s(k'))\left[\frac{iK+m}{\sqrt{N}}\right]} r(k-k') \\ &= \sum_{k=0}^{N-1} \sum_{k'=0}^{N-1} \delta(k-l)\delta(k'-m) e^{j\frac{2\pi}{N}(k' - k - s(k') + s(k))i} r(k-k'). \end{aligned} \tag{5.58}$$

The sums over k and k' can be broken into four sums using $k = k_1K + k_2$ and $k' = k_1'K + k_2'$, where $0 \le k_1 < K$, $0 \le k_2 < K$, $0 \le k_1' < K$, and $0 \le k_2' < K$:

$$\begin{aligned}[\mathbf{R}_{Hi}]_{lm} = \sum_{k_1=0}^{K-1}\sum_{k_2=0}^{K-1}\sum_{k_1'=0}^{K-1}\sum_{k_2'=0}^{K-1} &\delta(k_1K + k_2 - l)\delta(k_1'K + k_2' - m) \\ &\times e^{j\frac{2\pi}{N}((k_1'-k_1)K+k_2'-k_2-s(k_1'K+k_2')+s(k_1K+k_2))i} r(k - k'). \qquad (5.59)\end{aligned}$$

It can easily be verified that

$$(k_1' - k_1)K + k_2' - k_2 - s(k_1'K + k_2') + s(k_1K + k_2) = (k_1' - k_1)K. \qquad (5.60)$$

Therefore, equation (5.59) simplifies to

$$\begin{aligned}[\mathbf{R}_{Hi}]_{lm} &= \sum_{k_1=0}^{K-1}\sum_{k_2=0}^{K-1}\sum_{k_1'=0}^{K-1}\sum_{k_2'=0}^{K-1} \delta(k_2 - l)\delta(k_2' - m)e^{j\frac{2\pi}{K}(k_1'-k_1)i} \\ &\quad \times r((k_1 - k_1')K + k_2 - k_2') \\ &= \sum_{k_1=0}^{K-1}\sum_{k_1'=0}^{K-1} e^{j\frac{2\pi}{K}(k_1'-k_1)i} r((k_1 - k_1')K + l - m). \qquad (5.61)\end{aligned}$$

Equation (5.61) shows that $[\mathbf{R}_{Hi}]_{lm}$ depends only on $l - m$, or equivalently $\mathbf{R}_{Hi}$ is Toeplitz. Q.E.D.

The matrices $\left\{\mathbf{R}_{H0}, \mathbf{R}_{H1}, \ldots, \mathbf{R}_{HK-1}\right\}$ are not directly available and should be estimated from the data. Each autocorrelation matrix is Toeplitz and estimating its first column is not only sufficient but also computationally less expensive than estimating all of the other columns. Let us divide $\mathbf{u}_H(n)$ into K sections, where each section corresponds to one of the K modes of the HOT adaptive filter. Thus,

$$\mathbf{u}_H(n) = \begin{bmatrix} \mathbf{u}_{H0}^T(n) & \mathbf{u}_{H1}^T(n) & \cdots & \mathbf{u}_{HK-1}^T(n) \end{bmatrix}^T. \tag{5.62}$$

Then the first column of $\mathbf{R}_{Hi}$ can be estimated recursively using

$$\mathbf{p}_{Hi}(n) = \frac{n-1}{n}\mathbf{p}_{Hi} + \frac{1}{n}\mathbf{u}_{Hi}^*(n)[\mathbf{u}_{Hi}(n)]_0, \tag{5.63}$$

where $[\mathbf{u}_{Hi}(n)]_0$ is the first element of $\mathbf{u}_{Hi}(n)$. Each recursion requires $2K+1$ multiplications. Since there are K recursions, $2K^2 + K$ multiplications are required to estimates all of the K autocorrelation matrices. Each matrix can be efficiently inverted using a Levinson recursion with K^2 multiplications. The total number of multiplications required for the self-orthogonalizing HOT LMS adaptive filter is therefore $2K^3 + 4K^2 + 2K + 1$. For comparison, the computational complexities of the LMS, DFT LMS, and HOT LMS algorithms are $2N+1$, $5N+1$, and $2K^2+3K+1$, respectively.

To verify the predictions in this section, the self-orthogonalizing HOT LMS algorithm was simulated. The simulation was identical to the simulation of the LMS and HOT LMS algorithms in the previous section, except that the HOT LMS algorithm was replaced by the self-orthogonalizing HOT LMS algorithm. Figure 5.5, which shows the learning curves of both the LMS and self-orthogonalizing HOT LMS adaptive filters, depicts the substantial improvement of the self-orthogonalizing HOT LMS algorithm as compared to the basic HOT LMS algorithm. Although the performance of the self-orthogonalizing HOT LMS algorithm is greatly improved compared to the HOT LMS algorithm, the self-orthogonalizing HOT LMS algorithm is not computationally efficient. This is because it requires $\mathcal{O}(K^3)$ operations and any other transform domain LMS algorithm requires at most $\mathcal{O}(K^2)$ operations. To improve the computational efficiency of the self-orthogonalizing HOT LMS algorithm, the block LMS algorithm [2] can be used as shown in the next section.

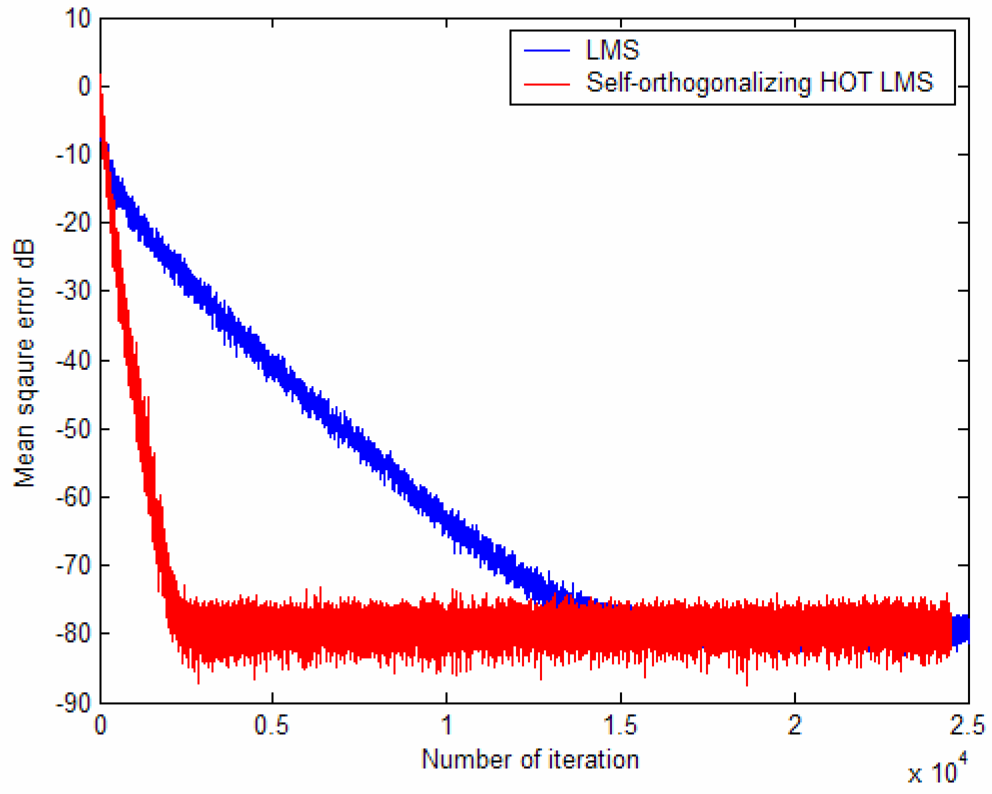


Figure 5.5: learning curves for the LMS and self-orthogonalizing HOT LMS algorithms.

5.5 Self-Orthogonalizing HOT Block Adaptive Filter

To reduce the computational complexity of the self-orthogonalizing HOT LMS algorithm, the computations are carried out block-by-block. This algorithm is called the self-orthogonalizing HOT block adaptive filter (SOHBAF). Let $\mathbf{w}_H(k)$ be the tap-weight vector in the k^{th} block and define the matrix

$$\mathbf{U}_H(k) = \begin{bmatrix} \mathbf{u}_H(kL) & \mathbf{u}_H(kL+1) & \cdots & \mathbf{u}_H(kL+L-1) \end{bmatrix}, \tag{5.64}$$

which contains the tap-input vectors in the k^{th} block, where L is the block length. The output of the filter in the k^{th} block is given by

$$\mathbf{y}(k) = \begin{bmatrix} y(kL) & y(kL+1) & \cdots & y(kL+L-1) \end{bmatrix}^T, \tag{5.65}$$

while the filter weight vector in the k^{th} block is given by

$$\mathbf{w}_H(k) = \begin{bmatrix} w_H(kL) & w_H(kL+1) & \cdots & w_H(kL+L-1) \end{bmatrix}^T. \tag{5.66}$$

We have then that

$$\mathbf{y}(k) = \mathbf{U}_H^T(k)\mathbf{w}_H(k). \tag{5.67}$$

Since $\mathbf{U}_H(k)$ contains the transformed input vectors in the k^{th} block rather than the the input vectors in the time domain, equation (5.67) is not a convolution and hence can not be efficiently calculated using the DFT. Let us divide $\mathbf{U}_H(k)$ into K sections that correspond to the K blocks of the HOT LMS algorithm as follows:

$$\mathbf{U}_H(k) = \begin{bmatrix} \mathbf{u}_{H0}(kL) & \mathbf{u}_{H0}(kL+1) & \cdots & \mathbf{u}_{H0}(kL+L-1) \\ \mathbf{u}_{H1}(kL) & \mathbf{u}_{H1}(kL+1) & \cdots & \mathbf{u}_{H1}(kL+L-1) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{u}_{HK-1}(kL) & \mathbf{u}_{HK-1}(kL+1) & \cdots & \mathbf{u}_{HK-1}(kL+L-1) \end{bmatrix}. \tag{5.68}$$

Then the output is given by

$$\mathbf{y}(k) =$$

$$\begin{bmatrix} \mathbf{u}^T_{H0}(kL) & \mathbf{u}^T_{H1}(kL) & \cdots & \mathbf{u}^T_{HK-1}(kL) \\ \mathbf{u}^T_{H0}(kL+1) & \mathbf{u}^T_{H1}(kL+1) & \cdots & \mathbf{u}^T_{HK-1}(kL+1) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{u}^T_{H0}(kL+L-1) & \mathbf{u}^T_{H1}(kL+L-1) & \cdots & \mathbf{u}^T_{HK-1}(kL+L-1) \end{bmatrix} \begin{bmatrix} \mathbf{w}_{H0}(k) \\ \mathbf{w}_{H1}(k) \\ \vdots \\ \mathbf{w}_{HK-1}(k) \end{bmatrix},$$

or

$$\mathbf{y}(k) = \sum_{i=0}^{K-1} \begin{bmatrix} \mathbf{u}_{Hi}(kL) & \mathbf{u}_{Hi}(kL+1) & \cdots & \mathbf{u}_{Hi}(kL+L-1) \end{bmatrix}^T \mathbf{w}_{Hi}(k). \tag{5.69}$$

Fortunately, each term in equation (5.69) is a convolution and can be efficiently calculated using the DFT with $6L\log_2 2L + 2L$ multiplications and the output of the filter can be calculated with $6KL\log_2 2L + 2LK$ multiplications. Let the i^{th} autocorrelation matrix in the k^{th} block be given by $\mathbf{R}_{Hi}(k)$. Then the first column of $\mathbf{R}_{Hi}(k)$ can be estimated by the recursion

$$\mathbf{p}_{Hi}(k) = \frac{k-1}{k}\mathbf{p}_{Hi}(k-1) + \frac{1}{kL}\sum_{l=0}^{L-1} \mathbf{u}^*_{Hi}(kL+l)[\mathbf{u}_{Hi}(kL+l)]_0, \tag{5.70}$$

or

$$\mathbf{p}_{Hi}(k) = \frac{k-1}{k}\mathbf{p}_{Hi}(k-1) + \frac{1}{kL}$$

$$\times \begin{bmatrix} \mathbf{u}_{Hi}^*(kL) & \mathbf{u}_{Hi}^*(kL+1) & \cdots & \mathbf{u}_{Hi}^*(kL+L-1) \end{bmatrix} \begin{bmatrix} [\mathbf{u}_{Hi}(kL)]_0 \\ [\mathbf{u}_{Hi}(kL+1)]_0 \\ \vdots \\ [\mathbf{u}_{Hi}(kL+L-1)]_0 \end{bmatrix}. \quad (5.71)$$

The second term in equation (5.71) is a convolution and can be efficiently calculated using the DFT with $6L\log_2 2L + 2L$ multiplications. All of the K recursions can be calculated with $6KL\log_2 2L + 2KL + 2K^2$ multiplications.

The filter weight vectors in the k^{th} block are updated by the recursion

$$\mathbf{w}_H(k+1) = \mathbf{w}_H(k) + \alpha\tilde{\mathbf{R}}_H^{-1}(k)\mathbf{U}_H^*(k)\mathbf{e}(k), \quad (5.72)$$

where $\mathbf{e}(k)$ is the error vector in the k^{th} block and

$$\tilde{\mathbf{R}}_H(k) = \begin{bmatrix} \mathbf{R}_{H0}(k) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{H1}(k) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{R}_{HK-1}(k) \end{bmatrix}. \quad (5.73)$$

The recursion in equation (5.72) requires $2K^3 + 6KL\log_2 2L + 2KL + 2K^2$ multiplications. Therefore, the computational complexity of the SOHBAF depends on the block length. If the block length is chosen to be K^2, then the computational complexity per sample is $\mathcal{O}(K\log_2 K)$. Therefore, the SOHBAF is computationally more efficient than the SOBAF, which requires $\mathcal{O}(K^2)$ operations per sample. On the other hand, if the block length is chosen to be K, then the computational complexity per sample is $\mathcal{O}(K^2)$ and both algorithms have similar computational complexities.

The total number of multiplications required for the SOHBAF is found by adding the number of multiplications required for each step. Keeping in mind that we need to calculate the DFT of $\mathbf{U}_H(k)$ only once and the convolutions in equation (5.69) are convolutions between long and short sequences [2], the total number of multiplications required for the SOHBAF is $2K^3+12.2KL\log_2 2L+5.4KL+4K^2$. The total number of multiplications required for the SOBAF is $2K^4 + 4K^2 + 14L\log_2 2L + 6L$. The multiplication counts per sample for the SOBAF and SOHBAF are plotted in Figure 5.10. The figure shows that the filter length at which the SOHBAF becomes more efficient than the SOBAF is about 7000.

To investigate the convergence speed of the SOHBAF algorithm, the learning curves of the LMS, SOBAF, SOHBAF, and DFT block LMS algorithms were simulated. The desired input was generated using the linear model $d(n) = w^o(n) * u(n) + e^o(n)$, where $e^o(n)$ is the measurement white gaussian noise with variance 10^{-8}. The input was a first-order Markov signal with autocorrelation function given by $r(k) = 0.9^{|k|}$. Two filter lengths were used in the simulation, 64-point and 256-point, both lowpass with cutoff frequency $\pi/2$ rad.

Figure 5.7 shows the learning curves for the LMS, SOBAF, SOHBAF, and DFT block LMS algorithms for the 64-point filter. It is clear that the LMS algorithm converges much slower than the other algorithms. The learning curves that correspond to the 256-point filter are shown in Figure 5.8. In both Figures, the block length used for

[2]This convolution can be calculated using the algorithm developed in [15] which is about 30% more efficient than the convolution using the DFT.

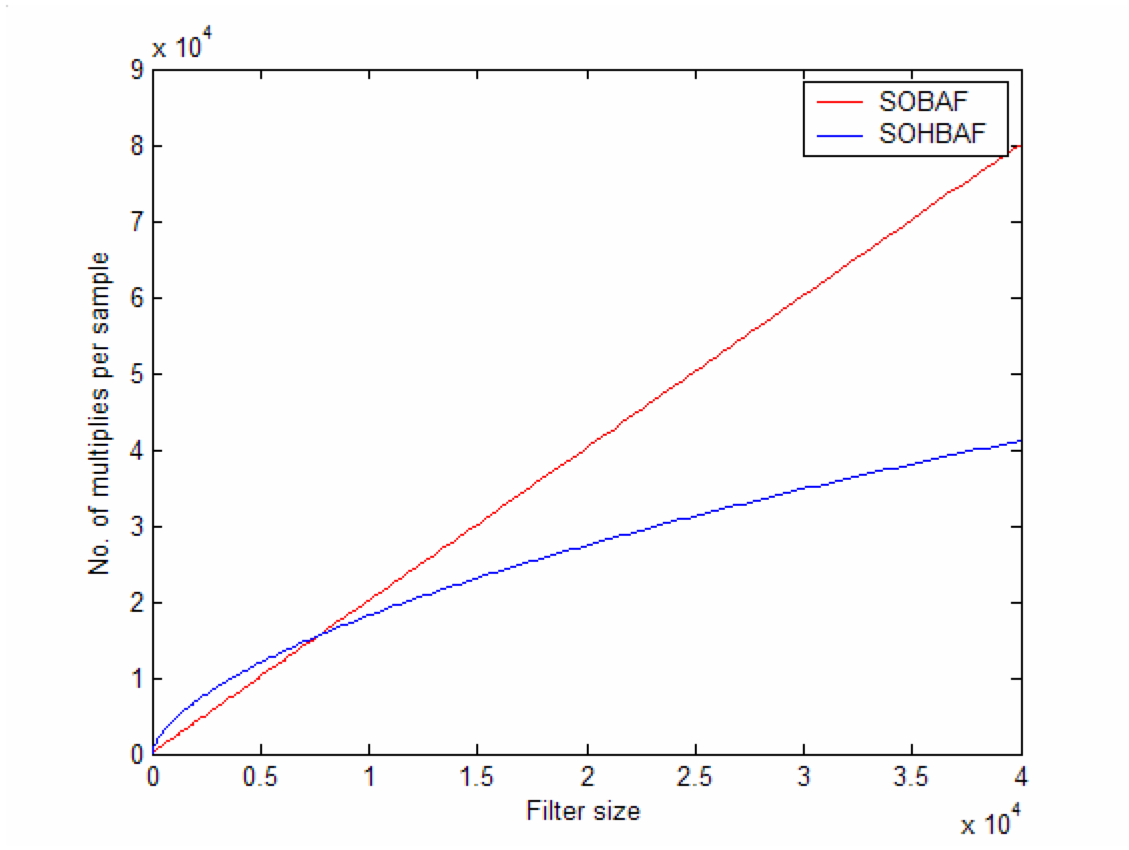


Figure 5.6: Multiplication counts per sample for the SOBAF and SOHBAF. The filter length at which the SOHBAF becomes more efficient than the SOBAF is about 7000.

the SOHBAF is K^2. The learning curves in both figures show that the convergence speed of the SOHBAF filter lies between that of the DFT block and SOBAF filters. Noting that the computational complexities per sample of the SOHBAF, SOBAF, and block DFT algorithms are $\mathcal{O}(K\log_2 K)$, $\mathcal{O}(K^2)$, and $\mathcal{O}(\log_2 K)$, respectively, the SOHBAF filter represents an LMS algorithm with computational complexity and convergence speed performance lying between the SOBAF and DFT block LMS algorithms.

The above simulations were repeated with $r(k) = 0.8^{|k|}$. The corresponding learning curves are shown in Figure 5.9. As expected, the convergence of the LMS al-

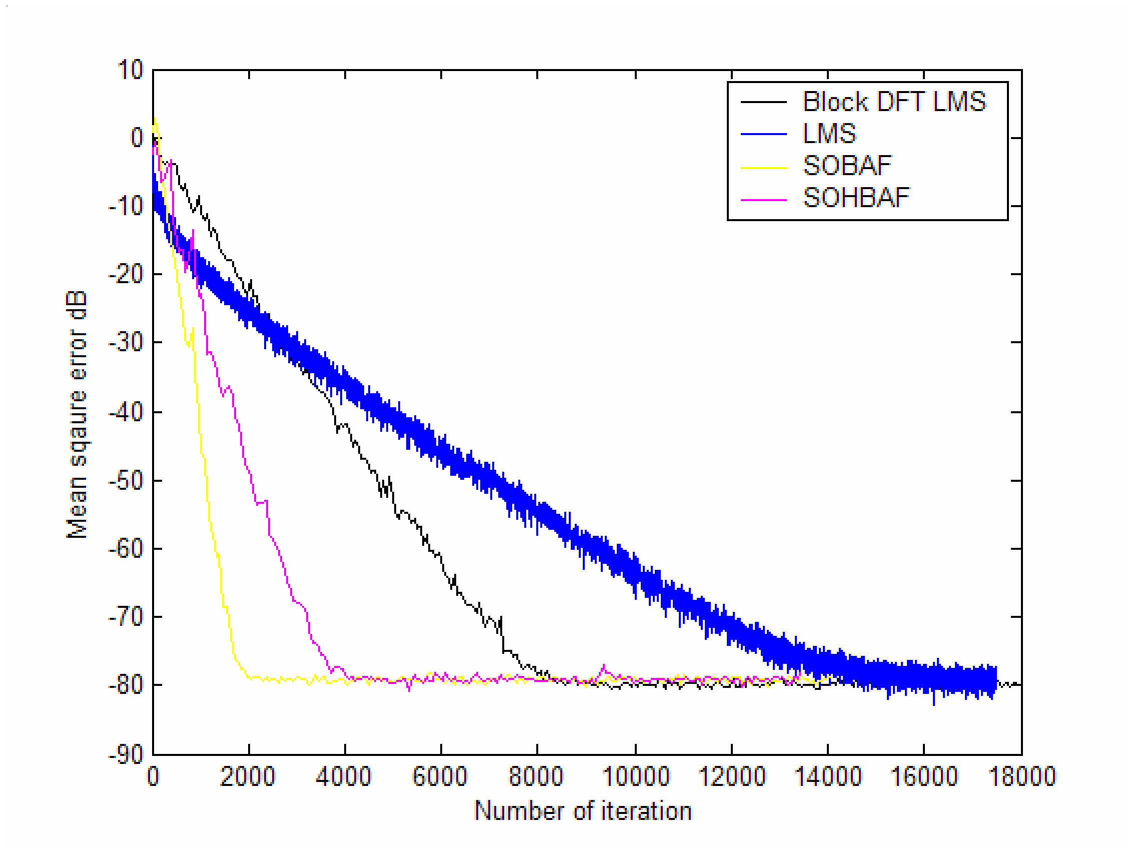


Figure 5.7: Learning curves for the LMS, SOBAF, SOHBAF, and DFT block LMS algorithms. The filter length was 64 and the block length for the SOHBAF filter was 64.

gorithm improved since the input is now less correlated. The convergence speed of the SOBAF did not change because the SOBAF does not depend on the filter input statistics. The convergence speed of the DFT block LMS algorithm improved a little more than how much the convergence speed of the SOHBAF did. Therefore, the SOHBAF also represents an LMS algorithm with sensitivity to the input statistics lying between the SOBAF and DFT block LMS algorithms.

Another set of simulations were run with input colored by the coloring filter whose impulse response listed in Table 5.1. The frequency response of the coloring filter

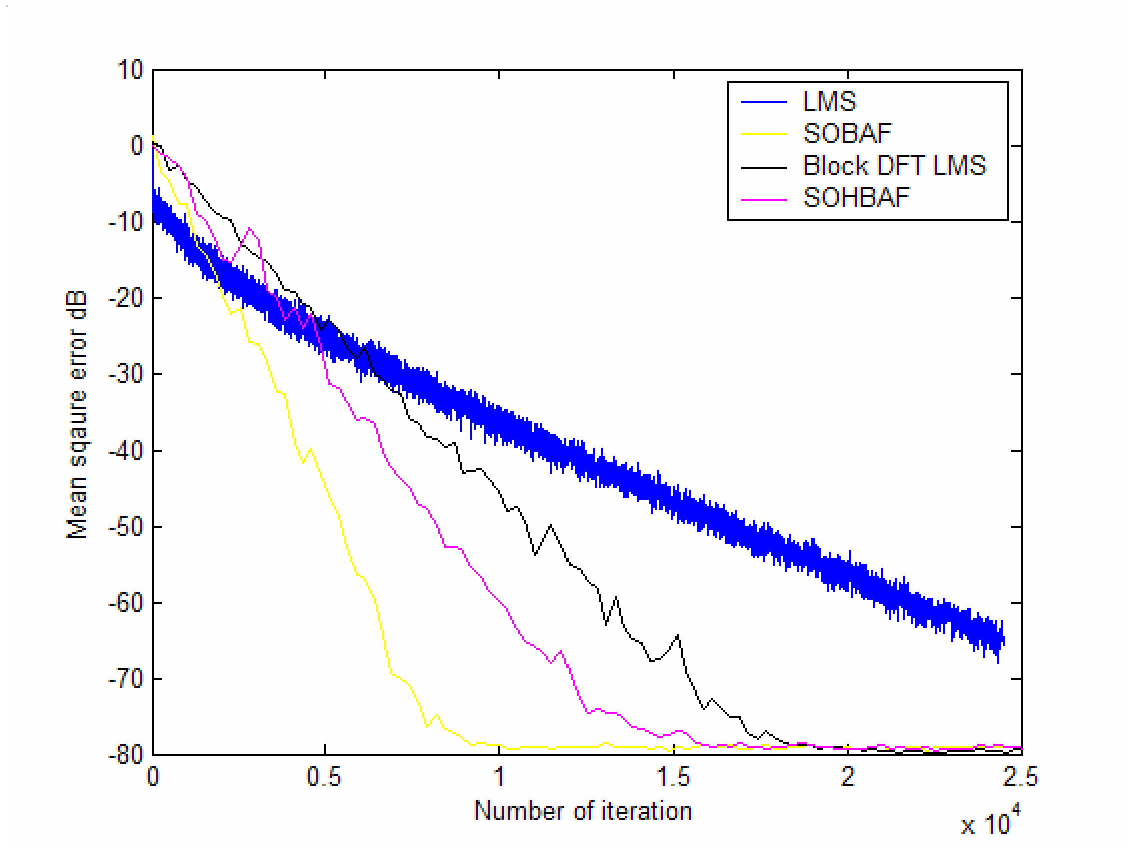


Figure 5.8: Learning curves for the LMS, SOBAF, SOHBAF, and DFT block LMS algorithms. The filter length was 256 and the block length for the SOHBAF filter was 256.

is shown in Figure 5.10.The filter was a 256-point lowpass filter with a cutoff frequency of 0.8π rad. The corresponding learning curves are shown in Figure 5.11. The SOBAF converged way much faster than the other algorithms. However, the SOHBAF converged much better than the DFT block LMS adaptive filter with such highly correlated input.

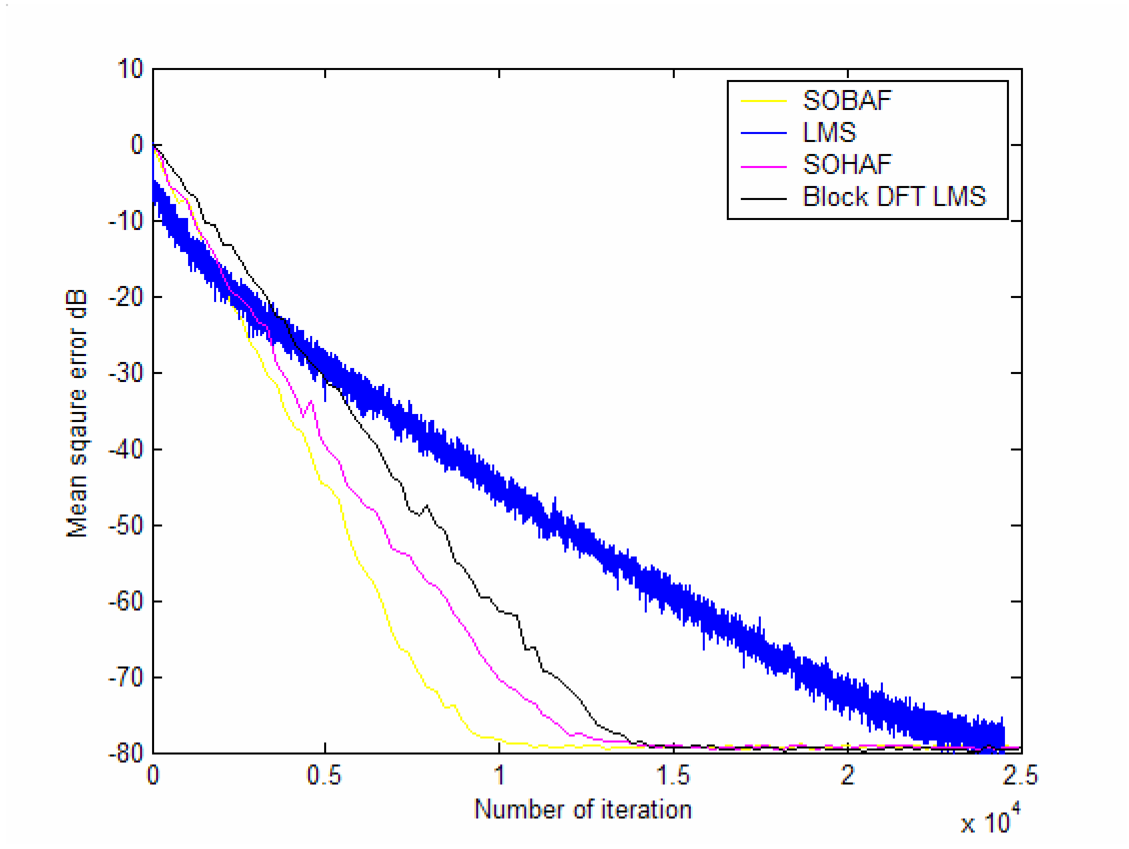


Figure 5.9: Learning curves for the LMS, SOBAF, SOHBAF, and DFT block LMS algorithms after the correlation parameter ρ was changed to 0.8. The filter length was 256 and the block length for the SOHBAF filter was 256.

Table 5.1: Coloring filter impulse response

h(1) =	-0.07390498507667	= h(32)
h(2) =	-0.08979904880683	= h(31)
h(3) =	0.1697858570223	= h(30)
h(4) =	0.1322851634558	= h(29)
h(5) =	-0.2619159234493	= h(28)
h(6) =	0.1505798251251	= h(27)
h(7) =	0.4151743979132	= h(26)
h(8) =	-0.5751693227394	= h(25)
h(9) =	-0.2779895539074	= h(24)
h(10) =	0.8359508110275	= h(23)
h(11) =	-0.4144632502419	= h(22)
h(12) =	-0.7776864201642	= h(21)
h(13) =	1.126049898253	= h(20)
h(14) =	0.1606359395014	= h(19)
h(15) =	-1.286906985691	= h(18)
h(16) =	0.7597038107278	= h(17)

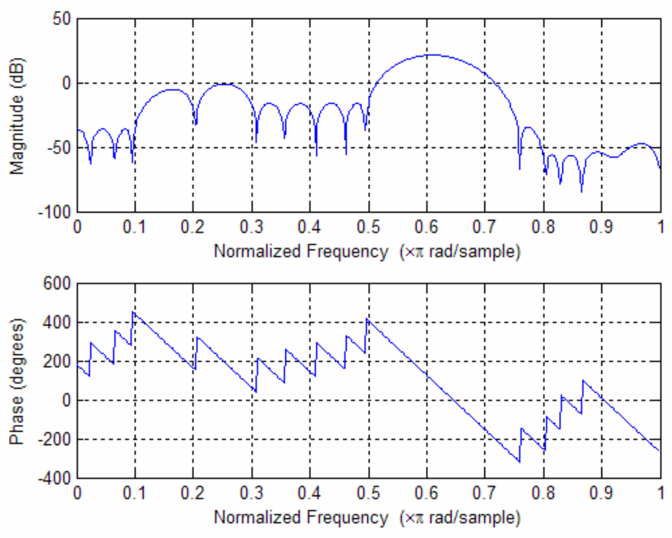


Figure 5.10: Frequency response of the coloring filter.

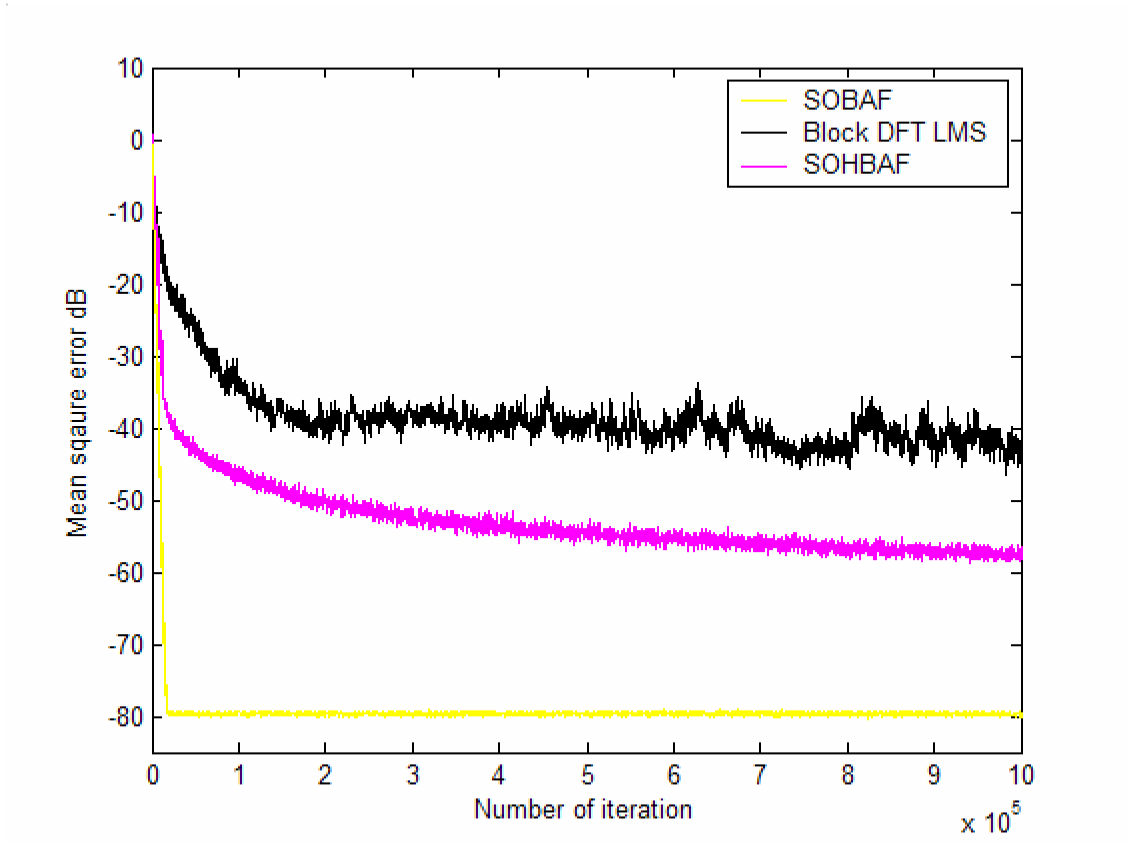


Figure 5.11: Learning curves for the SOBAF, SOHBAF, and DFT block LMS algorithms. The filter length was 256 and the block length for the SOHBAF filter was 256.

Chapter 6

Hirschman Optimal Transform Block LMS Algorithm

In this chapter, a “HOT convolution” is derived. The result is used to develop a fast block LMS adaptive filter, which is called the HOT block LMS adaptive filter. This filter requires slightly less than half of the computations that are required for the DFT block LMS adaptive filter. The convergence of the HOT block LMS adaptive filter is investigated in both the the time and HOT domains.

6.1 Convolution Using the HOT

In this section, the “HOT convolution,” a relation between the HOT of two signals and their circular convolution, is derived. Let u and w be two signals of length K^2. The circular convolution of the signals is $y = w \star u$. In the DFT domain, the convolution is given by the pointwise multiplication of the respective DFTs of the signals, i.e., $y_F(k) = w_F(k)u_F(k)$. A similar relation in the HOT domain can be readily found through the relation between the DFT and HOT. The DFT of u can be written as

$$\begin{aligned} u_F(k) &= \sum_{n=0}^{K^2-1} u(n)\, e^{-j\frac{2\pi}{K^2}kn} \\ &= \sum_{i=0}^{K-1} e^{-j\frac{2\pi}{K^2}ki} \sum_{l=0}^{K-1} u(lK+i)\, e^{-j\frac{2\pi}{K}kl}. \end{aligned} \tag{6.1}$$

The signal $u(lK+i)$, denoted by $u_i(l)$, is the i^{th} polyphase component of $u(n)$ with DFT given by

$$u_{iF}(k) = \sum_{l=0}^{K-1} u_i(l)\, e^{-j\frac{2\pi}{K}kl}. \tag{6.2}$$

Therefore, the DFT of the signal u can be written in terms of the DFTs of the polyphase components, or the HOT of u. The relation between the HOT and the DFTs of the polyphase components is descried in Figure 4.1. Equation (6.1) may be written as

$$u_F(k) = \sum_{i=0}^{K-1} e^{-j\frac{2\pi}{K^2}ki} u_{iF}(k). \tag{6.3}$$

Define the diagonal matrix

$$\mathbf{D}_{i,j}(k) = \text{Diag}\left\{ e^{-j\frac{2\pi}{K^2}ki}, e^{-j\frac{2\pi}{K^2}k(i+1)}, \ldots, e^{-j\frac{2\pi}{K^2}kj} \right\}. \tag{6.4}$$

Then the DFT of the signal can be written in a matrix form

$$\mathbf{u}_F = \sum_{i=0}^{K-1} \mathbf{D}_{0,K^2-1}(i) \begin{bmatrix} \mathbf{F}_K \\ \mathbf{F}_K \\ \vdots \\ \mathbf{F}_K \end{bmatrix} \mathbf{u}_i. \tag{6.5}$$

The above is the desired relation between the DFT and HOT. It should be noted that equation (6.5) represents a radix-K FFT algorithm which is less efficient than the radix-2 FFT algorithm. Therefore, HOT convolution is expected to be less efficient than DFT convolution. Now, we can use equation (6.5) to transform $\mathbf{y}_F = \mathbf{w}_F \otimes \mathbf{u}_F$ into the HOT domain. The symbol $\otimes$ indicates pointwise matrix multiplication and, throughout this discussion, pointwise matrix multiplication takes a higher precedence than conventional matrix multiplication. We have that

$$\sum_{i=0}^{K-1} \mathbf{D}_{0,K^2-1}(i) \begin{bmatrix} \mathbf{F}_K \\ \mathbf{F}_K \\ \vdots \\ \mathbf{F}_K \end{bmatrix} \mathbf{y}_i$$

$$= \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} \mathbf{D}_{0,K^2-1}(i+j) \begin{bmatrix} \mathbf{F}_K\mathbf{w}_i \\ \mathbf{F}_K\mathbf{w}_i \\ \vdots \\ \mathbf{F}_K\mathbf{w}_i \end{bmatrix} \otimes \begin{bmatrix} \mathbf{F}_K\mathbf{u}_j \\ \mathbf{F}_K\mathbf{u}_j \\ \vdots \\ \mathbf{F}_K\mathbf{u}_j \end{bmatrix}. \tag{6.6}$$

The above matrix equation can be separated into a system of K equations

$$\sum_{i=0}^{K-1} \mathbf{D}_{rK,(r+1)K-1}(i)\mathbf{F}_K\mathbf{y}_i = \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} \mathbf{D}_{rK,(r+1)K-1}(i+j) \left(\mathbf{F}_K\mathbf{w}_i\right) \otimes \left(\mathbf{F}_K\mathbf{w}_j\right), \tag{6.7}$$

where $r = 0, 1, \ldots, K-1$. Since

$$\mathbf{D}_{rK,(r+1)K-1}(i) = e^{-j\frac{2\pi}{K}ri}\mathbf{D}_{0,K-1}(i), \tag{6.8}$$

the HOT of the output can be obtained by solving the following set of K matrix equations:

$$\sum_{i=0}^{K-1} e^{-j\frac{2\pi}{K}ri}\mathbf{D}_{0,K-1}(i)\mathbf{F}_K\mathbf{y}_i = \sum_{i=0}^{K-1}\sum_{j=0}^{K-1} e^{-j\frac{2\pi}{K}r(i+j)}\mathbf{D}_{0,K-1}(i+j)\left(\mathbf{F}_K\mathbf{w}_i\right)\otimes\left(\mathbf{F}_K\mathbf{u}_j\right). \quad (6.9)$$

Since the DFT matrix is unitary, the solution of equation (6.9) can be expressed as

$$\mathbf{D}_{0,K-1}(s)\mathbf{F}_K\mathbf{y}_s = \frac{1}{K}\sum_{r=0}^{K-1}\sum_{i=0}^{K-1}\sum_{j=0}^{K-1} e^{j\frac{2\pi}{K}r(s-(i+j))}\mathbf{D}_{0,K-1}(i+j)\left(\mathbf{F}_K\mathbf{w}_i\right)\otimes\left(\mathbf{F}_K\mathbf{u}_j\right), \quad (6.10)$$

where

$$\mathbf{F}_K\mathbf{y}_s = \frac{1}{K}\sum_{r=0}^{K-1}\sum_{i=0}^{K-1}\sum_{j=0}^{K-1} e^{j\frac{2\pi}{K}r(i+j-s)}\mathbf{D}_{0,K-1}(i+j-s)\left(\mathbf{F}_K\mathbf{w}_i\right)\otimes\left(\mathbf{F}_K\mathbf{u}_j\right). \quad (6.11)$$

Moreover, as

$$\sum_{r=0}^{K-1} e^{j\frac{2\pi}{K}r(i+j-s)} = K\delta(i+j-s), \quad (6.12)$$

where $\delta(n)$ denotes the periodic Kronecker delta of periodicity K, equation (6.11) can be simplified to

$$\mathbf{F}_K\mathbf{y}_s = \sum_{i=0}^{K-1}\sum_{j=0}^{K-1}\delta(i+j-s)\mathbf{D}_{0,K-1}(i+j-s)\left(\mathbf{F}_K\mathbf{w}_i\right)\otimes\left(\mathbf{F}_K\mathbf{u}_j\right), \quad (6.13)$$

where $s = 0, 1, 2, \ldots, K-1$. The pointwise matrix multiplication in equation equation (6.13) can be converted into conventional matrix multiplication if we define $\mathbf{W}_i$ as the diagonal matrix for $\mathbf{F}_K\mathbf{w}_i$. We have then that

$$\mathbf{F}_K\mathbf{y}_s = \sum_{i=0}^{K-1}\sum_{j=0}^{K-1}\delta(i+j-s)\mathbf{D}_{0,K-1}(i+j-s)\mathbf{W}_i\mathbf{F}_K\mathbf{u}_j. \quad (6.14)$$

Combining the above K equations into one matrix equation, the HOT convolution can be written as

$$\begin{bmatrix} \mathbf{F}_K\mathbf{y}_0 \\ \mathbf{F}_K\mathbf{y}_1 \\ \mathbf{F}_K\mathbf{y}_2 \\ \vdots \\ \mathbf{F}_K\mathbf{y}_{K-2} \\ \mathbf{F}_K\mathbf{y}_{K-1} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_0 & \mathbf{D}\mathbf{W}_{K-1} & \mathbf{D}\mathbf{W}_{K-2} & \cdots & \mathbf{D}\mathbf{W}_2 & \mathbf{D}\mathbf{W}_1 \\ \mathbf{W}_1 & \mathbf{W}_0 & \mathbf{W}_{K-1} & \cdots & \mathbf{D}\mathbf{W}_3 & \mathbf{D}\mathbf{W}_2 \\ \mathbf{W}_2 & \mathbf{W}_1 & \mathbf{W}_0 & \cdots & \mathbf{D}\mathbf{W}_4 & \mathbf{D}\mathbf{W}_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{W}_{K-2} & \mathbf{W}_{K-3} & \mathbf{W}_{K-4} & \cdots & \mathbf{W}_0 & \mathbf{D}\mathbf{W}_{K-1} \\ \mathbf{W}_{K-1} & \mathbf{W}_{K-2} & \mathbf{W}_{K-3} & \cdots & \mathbf{W}_1 & \mathbf{W}_0 \end{bmatrix} \begin{bmatrix} \mathbf{F}_K\mathbf{u}_0 \\ \mathbf{F}_K\mathbf{u}_1 \\ \mathbf{F}_K\mathbf{u}_2 \\ \vdots \\ \mathbf{F}_K\mathbf{u}_{K-2} \\ \mathbf{F}_K\mathbf{u}_{K-1} \end{bmatrix} \tag{6.15}$$

where

$$\mathbf{D} = \mathrm{Diag}\left\{1, e^{-j\frac{2\pi}{K}}, \ldots, e^{-j\frac{2\pi}{K}(K-1)}\right\}. \tag{6.16}$$

Notice that the square matrix in equation (6.15) is arranged in a block Toeplitz structure.

A better understanding of this result may be obtained by comparing equation (6.15) with the K-point circular convolution

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{K-2} \\ y_{K-1} \end{bmatrix} = \begin{bmatrix} w_0 & w_{K-1} & w_{K-2} & \cdots & w_2 & w_1 \\ w_1 & w_0 & w_{K-1} & \cdots & w_3 & w_2 \\ w_2 & w_1 & w_0 & \cdots & w_4 & w_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{K-2} & w_{K-3} & w_{K-4} & \cdots & w_0 & w_{K-1} \\ w_{K-1} & w_{K-2} & w_{K-3} & \cdots & w_1 & w_0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{K-2} \\ u_{K-1} \end{bmatrix}. \tag{6.17}$$

The square matrix in equation (6.17) is also Toeplitz. However, equation (6.17) is a pure time domain result, whereas equation (6.15) is a pure HOT domain relation, which may be interpreted in terms of both the time domain and the DFT domain features. This fact can be explained in terms of fact that the HOT basis is optimal in the sense of the entropic joint time-frequency uncertainty measure $H_p(u) = pH(u) + (1-p)H(u_F)$ for all $0 \leq p \leq 1$. Before moving on to the computational complexity analysis of HOT convolution, we make the same observations about the term $\mathbf{DF}_K\mathbf{w}_i$ appearing in equation (6.15). This term is the complex conjugate of the DFT of the upside down flipped i^{th} polyphase component of w.

It should be noted that equation (6.15) does not show explicitly the HOT of $u(n)$ and $w(n)$. However, the DFT of the polyphase components that are shown explicitly in equation (6.15) are related to the HOT of the corresponding signal as shown in Figure. 4.1. For example, the 0^{th} polyphase component of the output is given by

$$\mathbf{y}_0(k) = \mathbf{F}_K^{-1}\mathbf{I}_0\mathbf{w}_H(k) \otimes \mathbf{I}_0\mathbf{u}_H(k) + \mathbf{F}_K^{-1}\mathbf{D}\sum_{i=1}^{K-1}\mathbf{I}_{K-i}\mathbf{w}_H(k) \otimes \mathbf{I}_i\mathbf{u}_H(k). \tag{6.18}$$

Next, we examine the computational complexity of HOT convolution. To find the HOT of the two signals w and u, $2K^2\log_2 K$ multiplications are required. Multiplication with the diagonal matrix D requires $K(K-1)$ multiplications. Finally, the matrix multiplication requires K^3 scalar multiplications. Therefore, the total number of multiplications required is $2K^2\log_2 K + K^3 + K^2 - K$. Thus, computation of the output y using the HOT requires $K^3 + 3K^2\log_2 K + K^3 + K^2 - K$ multiplications, which is more than $6K^2\log_2 K + K^2$ as required by the DFT. When it is required to calculate only one polyphase component of the output, only $K^2 + 2K^2\log_2 K + K\log_2 K$ multiplications are necessary. Asymptotically in K, we see that the HOT could be three times more efficient than the DFT.

6.2 Development of the Basic Algorithm

In the block adaptive filter, the adaptation proceeds block-by-block with the weight update equation

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu}{L}\sum_{i=0}^{L-1}\mathbf{u}(kL+i)e(kL+i), \tag{6.19}$$

where $d(n)$ and $y(n)$ are the desired and output signals, respectively, $\mathbf{u}(n)$ is the tap-input vector, L is the block length or the filter length, and $e(n) = d(n) - y(n)$ is the filter error. The DFT is commonly used to efficiently calculate the output of the filter and the sum in the update equation. Since the HOT is more efficient than the DFT when it is only required to calculate one polyphase component of the output, the block LMS algorithm equation (6.19) is modified such that only one polyphase component of the error in the k^{th} block is used to update the filter weights. For reasons that will become clear later, the filter length L is chosen such that $L = K^2/2$. With this modification, equation (6.19) becomes

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{2\mu}{K}\sum_{i=0}^{K/2-1}\mathbf{u}(kL+iK+j)e(kL+iK+j). \tag{6.20}$$

Since the DFT is most efficient when the length of the filter is equal to the block length [2], this will be assumed in equation (6.20). The parameter j determines which polyphase component of the error signal is being used in the adaptation. This parameter can be changed from block to block. If $j = 0$, the output can be computed using the HOT as in equation (6.18). A second convolution is needed to compute the sum in equation (6.20). This sum contains only one polyphase component of the error. If this vector is up-sampled by K, the sum is just a convolution between the input vector and the up-sampled error vector. Although all the polyphase components are needed in the sum, the convolution can be computed by the HOT with the same computational complexity as the first convolution since only one polyphase component of the error vector is non-zero.

The block adaptive filter that implements the above algorithm is called the HOT block LMS adaptive filter and is shown in Figure 6.1. The complete steps of this new, efficient, adaptive algorithm are summarized below:

(a) Append the weight vector with $K^2/2$ zeros (the resulting vector is now K^2 points long as required in the HOT definition) and find its HOT.

(b) Compute the HOT of the input vector

$$\left[\begin{array}{cccccc} u\left((k-1)\frac{K^2}{2}\right) & \cdots & u\left(k\frac{K^2}{2}\right) & u\left(k\frac{K^2}{2}+1\right) & \cdots & u\left((k+1)\frac{K^2}{2}-1\right) \end{array}\right]^T. \tag{6.21}$$

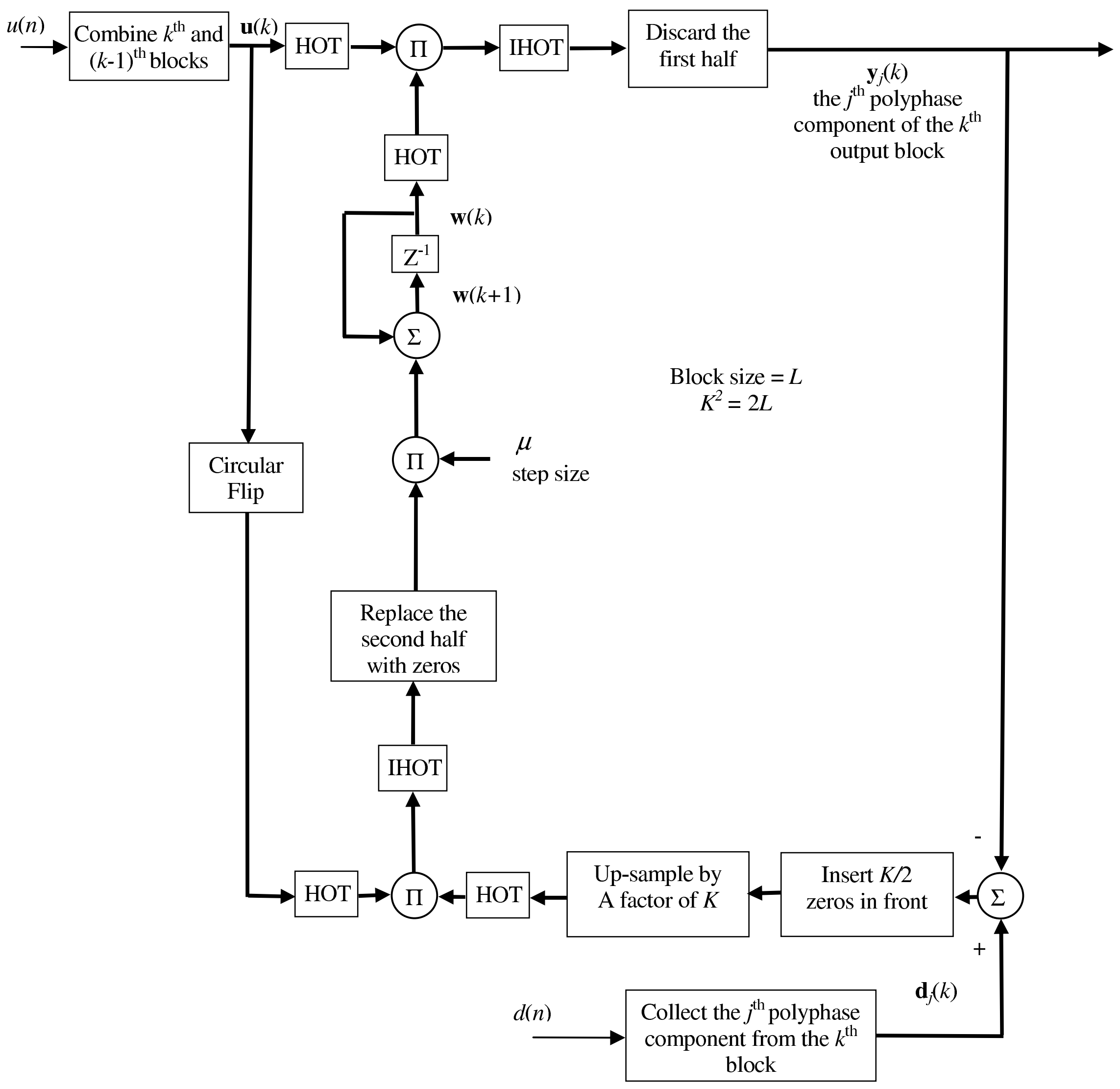
Note that this vector contains the input samples for the current and previous blocks.

(c) Use the inverse HOT and equation (6.15) to calculate the j^{th} polyphase component of the circular convolution. The j^{th} polyphase component of the output can be found by discarding the first half of the j^{th} polyphase component of the circular convolution.

(d) Calculate the j^{th} polyphase component of the error, insert a block of $K/2$ zeros, up-sample by K, then calculate its HOT.

(e) Circularly flip the vector in (b) and then compute its HOT.

(f) Compute the sum in the update equation using equation (6.15). This sum is the first half of the elements of the circular convolution between the vectors in parts (e) and (d).



$u(n)$
Combine kth and $(k$-1)th blocks
$\mathbf{u}(k)$
HOT
Π
IHOT
Discard the first half
$\mathbf{y}_j(k)$
the jth polyphase component of the kth output block
HOT
$\mathbf{w}(k)$
Z^{-1}
$\mathbf{w}(k+1)$
Σ
Block size = L
$K^2 = 2L$
Circular Flip
Π
μ
step size
Replace the second half with zeros
IHOT
HOT
Π
HOT
Up-sample by A factor of K
Insert $K/2$ zeros in front
-
Σ
+
$\mathbf{d}_j(k)$
$d(n)$
Collect the jth polyphase component from the kth block

Figure 6.1: HOT block LMS adaptive filter.

6.3 Computational Complexity Analysis

In this section, we analyze the computational cost of the algorithm and compare it to that of the DFT block adaptive algorithm. Parts (a), (b), and (e) require $3K^2 \log_2 K$ multiplications. Part (c) requires $K \log_2 K + K^2$. Part (d) requires $K \log_2 K$ multiplications, and part (f) requires $K^2 + K^2 \log_2 K$ multiplications. The total number of multiplications is thus $4K^2 \log_2 K + 2K \log_2 K + 2K^2$. The corresponding DFT block adaptive algorithm requires $10K^2 \log_2 K + 2K^2$ multiplications — asymptotically more than twice as many. Therefore, by using only one polyphase component for the adaptation in a block, the computational cost can be reduced by a factor of 2.5. While this complexity reduction comes at the cost of not using all available information, the proposed algorithm provides better estimates than the LMS filter. The reduction of the computational complexity in this algorithm comes from using the polyphase components of the input signal to calculate one polyphase component of the output via the HOT.

It is worth mentioning that the fast exact LMS (FELMS) adaptive algorithm [18] also reduces the computational complexity by finding the output by processing the polyphase components of the input. However, the computational complexity reduction of the FELMS algorithm is less than that found in the DFT and HOT block adaptive algorithms because the FELMS algorithm is designed to have exact mathematical equivalence to, and hence the same convergence properties as, the conventional LMS algorithm. Comparing the HOT block LMS algorithm with the block LMS algorithms described in Chapter 3, the HOT filter performs computationally better.

The multiplication counts for both the DFT block and HOT block LMS algorithms are plotted in Figure 6.2. The HOT block LMS adaptive filter is always more efficient than the DFT block LMS adaptive filter and the asymptotic ratio between their computational cost is almost reached at small filter lengths. The computational complexity of the HOT filter can be further improved by relating the HOT of the circularly flipped vector in step (e) to the HOT of the vector in step (b). Another possibility to reduce the computational cost of the HOT block algorithm is by removing the gradient constraint in the filter weight update equation as has been done in the unconstrained DFT block LMS algorithm [24].

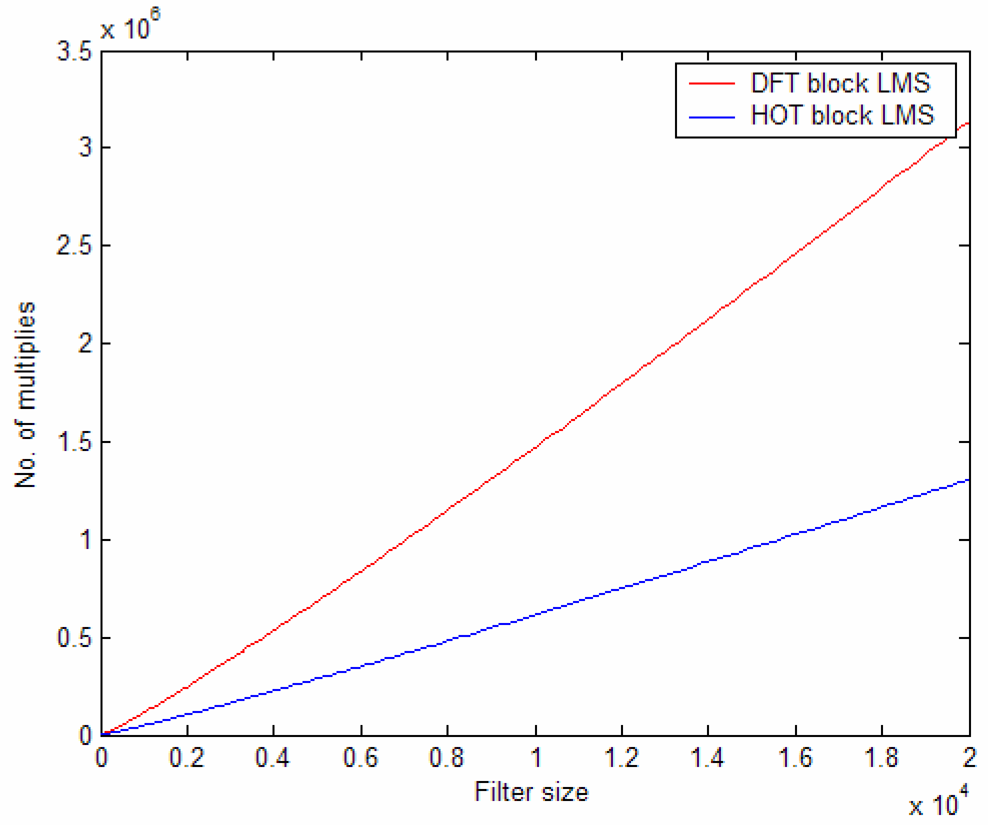


Figure 6.2: Multiplication counts for both the DFT block and HOT block LMS algorithms.

6.4 Convergence Analysis in the Time Domain

In this section, we analyze the convergence of the HOT block LMS algorithm in the time domain. we assume throughout that the step size is small. The HOT block LMS filter minimizes the cost

$$\hat{\xi} = \frac{2}{K} \sum_{i=0}^{\frac{K}{2}-1} \Big| e(kL + iK + j) \Big|^2, \tag{6.22}$$

which is the average of the squared errors in the j^{th} polyphase error component. From statistical LMS theory [4], the block LMS algorithm can be analyzed using the stochastic difference equation [4]

$$\boldsymbol{\epsilon}_T(k+1) = \Big(\mathbf{I} - \mu\boldsymbol{\Lambda}\Big)\boldsymbol{\epsilon}_T(k) + \boldsymbol{\phi}(k), \tag{6.23}$$

where

$$\boldsymbol{\phi}(k) = -\frac{\mu}{L}\,\mathbf{V}^H \sum_{i=0}^{L-1} \mathbf{u}(kL+i)\, e^o(kL+i) \tag{6.24}$$

is the driving force of for the block LMS algorithm [4]. we found that the HOT block LMS algorithm has the following driving force

$$\boldsymbol{\phi}_{\text{HOT}}(k) = -\frac{2\mu}{K}\,\mathbf{V}^H \sum_{i=0}^{\frac{K}{2}-1} \mathbf{u}(kL+iK+j)\, e^o(kL+iK+j). \tag{6.25}$$

It is easily shown that

$$E\boldsymbol{\phi}_{\text{HOT}}(k) = 0, \tag{6.26}$$

$$E\boldsymbol{\phi}_{\text{HOT}}(k)\boldsymbol{\phi}_{\text{HOT}}^H(k) = \frac{2\mu^2 J_{\min}\boldsymbol{\Lambda}}{K}. \tag{6.27}$$

The mean square of the l^{th} component of equation (6.27) is given by

$$E\left|\epsilon_l(k)\right|^2 = \frac{2\mu\frac{J_{\min}}{K}}{2-\mu\lambda_l} + (1-\mu\lambda_l)^{2k}\left(\left|\epsilon_l(0)\right|^2 - \frac{2\mu\frac{J_{min}}{K}}{2-\mu\lambda_l}\right), \tag{6.28}$$

where λ_l is the l^{th} eigenvalue of the input autocorrelation matrix. Therefore, the average time constant of the HOT block LMS algorithm is given by

$$\tau = \frac{L^2}{2\mu\sum_{l=1}^{L}\lambda_l}. \tag{6.29}$$

The misadjustment can be calculated directly and is given by

$$M = \frac{\sum_{l=1}^{L}\lambda_l E\left|\epsilon_l(\infty)\right|^2}{J_{\min}}. \tag{6.30}$$

Using equation (6.23), one may find $E|\epsilon_l(\infty)|^2$ and substitute the result into equation (6.30). The misadjustment of the HOT block LMS filter is then given by

$$M = \frac{\mu}{K}\sum_{l=1}^{L}\lambda_l. \tag{6.31}$$

Thus, the average time constant of the HOT block LMS filter is the same as that of the DFT block LMS filter [1]. However, the HOT block LMS filter has K times higher misadjustment than the DFT block LMS algorithm [2].

The HOT and DFT block LMS algorithms were simulated using white noise inputs. The desired signal was generated using the linear model $d(n) = w^o(n) * u(n) + e^o(n)$, where $e^o(n)$ is the measurement white gaussian noise with variance 10^{-4} and

[1]The average time constant of the DFT block LMS filter is [4] $\tau = L^2/2\mu\sum_{l=1}^{L}\lambda_l$.

[2]The misadjustment of the DFT block LMS algorithm is [4] $M = \frac{\mu}{K^2}\sum_{l=1}^{L}\lambda_l$.

$W^o(z) = 1 + 0.5z^{-1} - 0.25z^{-2} + 0.03z^{-3} + 0.1z^{-4} + 0.002z^{-5} - 0.01z^{-6} + 0.007z^{-7}$. The learning curves are shown in Figure 6.3 with the learning curve of the conventional LMS algorithm. The step sizes of all algorithms were chosen to be the same. The higher mean square error of the HOT algorithm, compared to the DFT algorithm, shows the trade-off for complexity reduction by more than half. As expected the HOT and DFT block LMS algorithms converge at the same rate.

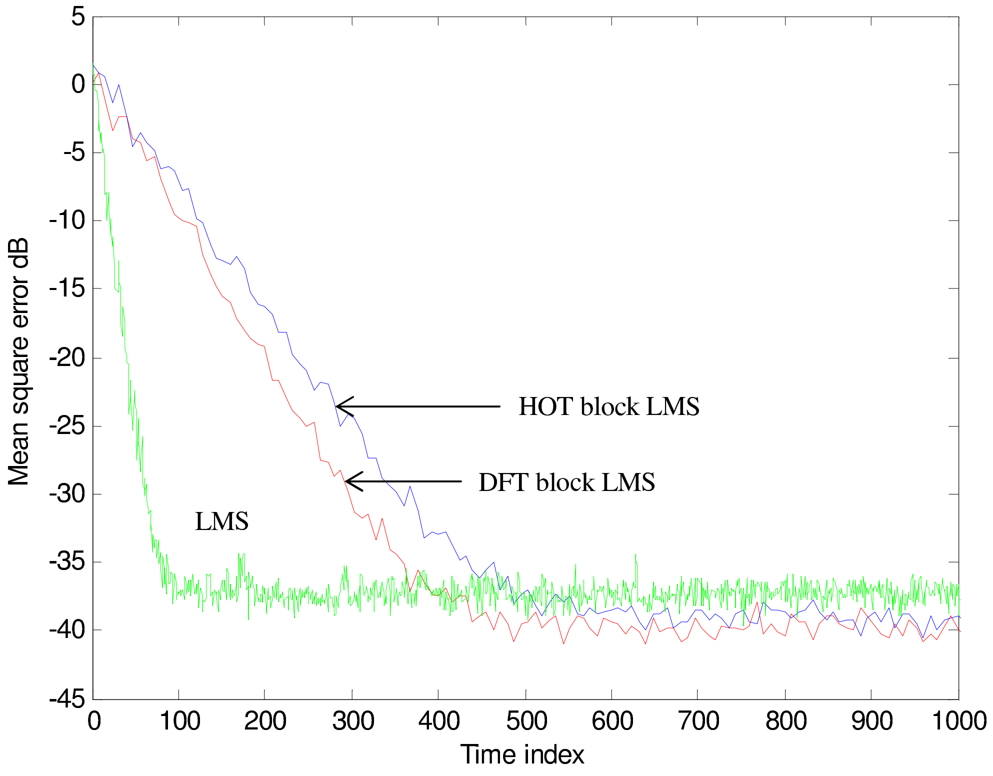


Figure 6.3: Learning curves of the DFT and HOT block LMS algorithms with the conventional LMS filter.

6.5 Convergence Analysis in the HOT Domain

Let $u(n)$ be the input to the adaptive filter and

$$\hat{\mathbf{w}}(k) = \left[\begin{array}{cccc} w_0(k) & w_1(k) & \cdots & w_{\frac{K2}{2}-1}(k) \end{array} \right]^T \tag{6.32}$$

be the tap-weight vector of the adaptive filter, where k is the block index. Define the extended tap-weight vector

$$\mathbf{w}(k) = \left[\begin{array}{ccccc} \hat{\mathbf{w}}^T(k) & 0 & 0 & \cdots & 0 \end{array} \right]^T \tag{6.33}$$

and the tap-input vector

$$\mathbf{u}(k) = \left[\begin{array}{cccccc} u\left((k-1)\frac{K^2}{2}\right) & \cdots & u\left(k\frac{K^2}{2}\right) & u\left(k\frac{K^2}{2}+1\right) & \cdots & u\left((k+1)\frac{K^2}{2}-1\right) \end{array} \right]^T . \tag{6.34}$$

Denote the HOT transforms of $\mathbf{u}(k)$ and $\mathbf{w}(k)$ by $\mathbf{u}_H(k) = \mathbf{H}\mathbf{u}(k)$ and $\mathbf{w}_H(k) = \mathbf{H}\mathbf{w}(k)$, respectively, where $\mathbf{H}$ is the HOT matrix. The 0^{th} polyphase component of the circular convolution of $\mathbf{u}(k)$ and $\mathbf{w}(k)$ is given by

$$\mathbf{F}_K\mathbf{y}_0(k) = \mathbf{F}_K\mathbf{w}_0(k) \otimes \mathbf{F}_K\mathbf{u}_0(k) + \mathbf{D}\sum_{i=1}^{K-1} \mathbf{F}_K\mathbf{w}_{K-i}(k) \otimes \mathbf{F}_K\mathbf{u}_i(k). \tag{6.35}$$

Using $\mathbf{F}_K\mathbf{u}_i(k) = \mathbf{I}_i\mathbf{H}\mathbf{u}(k) = \mathbf{I}_i\mathbf{u}_H(k)$, equation (6.35) can be written in terms of the HOT of $\mathbf{u}$(k) and $\mathbf{w}$(k). The result is given by

$$\mathbf{F}_K\mathbf{y}_0(k) = \mathbf{I}_0\mathbf{w}_H(k) \otimes \mathbf{I}_0\mathbf{u}_H(k) + \mathbf{D}\sum_{i=1}^{K-1} \mathbf{I}_{K-i}\mathbf{w}_H(k) \otimes \mathbf{I}_i\mathbf{u}_H(k). \tag{6.36}$$

The 0th polyphase component of the linear convolution of $\hat{\mathbf{w}}(k)$ and $\mathbf{u}(\mathrm{n})$, the output of the adaptive filter in the kth block, is given by the last $K/2$ elements of $\mathbf{y}_0(k)$. Let the desired signal be $d(n)$ and define the extended 0th polyphase component of the desired signal in the kth block as

$$\mathbf{d}_0(k) = \begin{bmatrix} \mathbf{0}_{\frac{K}{2}} \\ \hat{\mathbf{d}}_0(k) \end{bmatrix}. \tag{6.37}$$

The extended 0th polyphase component of error signal in the kth block is given by

$$\begin{aligned} \mathbf{e}_0(k) &= \begin{bmatrix} \mathbf{0}_{\frac{K}{2}} \\ \hat{\mathbf{e}}_0(k) \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{\frac{K}{2}} \\ \hat{\mathbf{d}}_0(k) \end{bmatrix} - \begin{bmatrix} \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} & \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} \\ \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} & \mathbf{I}_{\frac{K}{2}\times\frac{K}{2}} \end{bmatrix} \mathbf{F}_K^{-1} \\ &\quad \times \left[\mathbf{I}_0\mathbf{w}_H(k) \otimes \mathbf{I}_0\mathbf{u}_H(k) + \mathbf{D}\sum_{i=1}^{K-1} \mathbf{I}_{K-i}\mathbf{w}_H(k) \otimes \mathbf{I}_i\mathbf{u}_H(k) \right]. \end{aligned} \tag{6.38}$$

Multiplying equation (6.38) by the DFT matrix yields

$$\begin{aligned} \mathbf{F}_K\mathbf{e}_0(k) &= \mathbf{F}_K \begin{bmatrix} \mathbf{0}_{\frac{K}{2}} \\ \hat{\mathbf{d}}_0(k) \end{bmatrix} - \mathbf{F}_K \begin{bmatrix} \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} & \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} \\ \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} & \mathbf{I}_{\frac{K}{2}\times\frac{K}{2}} \end{bmatrix} \mathbf{F}_K^{-1} \\ &\quad \times \left[\mathbf{I}_0\mathbf{w}_H(k) \otimes \mathbf{I}_0\mathbf{u}_H(k) + \mathbf{D}\sum_{i=1}^{K-1} \mathbf{I}_{K-i}\mathbf{w}_H(k) \otimes \mathbf{I}_i\mathbf{u}_H(k) \right]. \end{aligned} \tag{6.39}$$

Define $\mathbf{u}_H^c(k) = \mathbf{H}\mathbf{u}^c(k)$, where $\mathbf{u}^c(k)$ is the circularly shifted version of $\mathbf{u}(k)$. The adaptive filter update equation in the kth block is given by

$$\mathbf{w}_H(k+1) = \mathbf{w}_H(k) + \mu\,\mathbf{H} \begin{bmatrix} \mathbf{I}_{\frac{K^2}{2}\times\frac{K^2}{2}} & \mathbf{0}_{\frac{K^2}{2}\times\frac{K^2}{2}} \\ \mathbf{0}_{\frac{K^2}{2}\times\frac{K^2}{2}} & \mathbf{0}_{\frac{K^2}{2}\times\frac{K^2}{2}} \end{bmatrix} \mathbf{H}^{-1}\boldsymbol{\phi}_H(k), \tag{6.40}$$

where $\boldsymbol{\phi}_H(k)$ is found from

$$\begin{bmatrix} \mathbf{I}_0\boldsymbol{\phi}_H(k) \\ \mathbf{I}_1\boldsymbol{\phi}_H(k) \\ \mathbf{I}_2\boldsymbol{\phi}_H(k) \\ \vdots \\ \mathbf{I}_{K-2}\boldsymbol{\phi}_H(k) \\ \mathbf{I}_{K-1}\boldsymbol{\phi}_H(k) \end{bmatrix} = \begin{bmatrix} \mathbf{F}_K\mathbf{e}_0(k) \\ \mathbf{F}_K\mathbf{e}_0(k) \\ \mathbf{F}_K\mathbf{e}_0(k) \\ \vdots \\ \mathbf{F}_K\mathbf{e}_0(k) \\ \mathbf{F}_K\mathbf{e}_0(k) \end{bmatrix} \otimes \begin{bmatrix} \mathbf{I}_0\mathbf{u}_H^c(k) \\ \mathbf{I}_1\mathbf{u}_H^c(k) \\ \mathbf{I}_2\mathbf{u}_H^c(k) \\ \vdots \\ \mathbf{I}_{K-2}\mathbf{u}_H^c(k) \\ \mathbf{I}_{K-1}\mathbf{u}_H^c(k) \end{bmatrix}, \tag{6.41}$$

as

$$\boldsymbol{\phi}_H(k) = \mathbf{I}_K^{-1} \begin{bmatrix} \mathbf{F}_K\mathbf{e}_0(k) \\ \mathbf{F}_K\mathbf{e}_0(k) \\ \mathbf{F}_K\mathbf{e}_0(k) \\ \vdots \\ \mathbf{F}_K\mathbf{e}_0(k) \\ \mathbf{F}_K\mathbf{e}_0(k) \end{bmatrix} \otimes \begin{bmatrix} \mathbf{I}_0\mathbf{u}_H^c(k) \\ \mathbf{I}_1\mathbf{u}_H^c(k) \\ \mathbf{I}_2\mathbf{u}_H^c(k) \\ \vdots \\ \mathbf{I}_{K-2}\mathbf{u}_H^c(k) \\ \mathbf{I}_{K-1}\mathbf{u}_H^c(k) \end{bmatrix}. \tag{6.42}$$

Finally, the HOT block LMS filter in the HOT domain can be written as

$$\begin{aligned} \mathbf{w}_H(k+1) &= \mathbf{w}_H(k) \\ &+ \mu\,\mathbf{H} \begin{bmatrix} \mathbf{I}_{\frac{K^2}{2}\times\frac{K^2}{2}} & \mathbf{0}_{\frac{K^2}{2}\times\frac{K^2}{2}} \\ \mathbf{0}_{\frac{K^2}{2}\times\frac{K^2}{2}} & \mathbf{0}_{\frac{K^2}{2}\times\frac{K^2}{2}} \end{bmatrix} \mathbf{H}^{-1}\mathbf{I}_K^{-1} \begin{bmatrix} \mathbf{F}_K\mathbf{e}_0(k) \\ \mathbf{F}_K\mathbf{e}_0(k) \\ \mathbf{F}_K\mathbf{e}_0(k) \\ \vdots \\ \mathbf{I}_{K-1}\mathbf{e}_0(k) \\ \mathbf{I}_{K-1}\mathbf{e}_0(k) \end{bmatrix} \otimes \begin{bmatrix} \mathbf{I}_0\mathbf{u}_H^c(k) \\ \mathbf{I}_1\mathbf{u}_H^c(k) \\ \mathbf{I}_2\mathbf{u}_H^c(k) \\ \vdots \\ \mathbf{I}_{K-2}\mathbf{u}_H^c(k) \\ \mathbf{I}_{K-1}\mathbf{u}_H^c(k) \end{bmatrix}. \end{aligned} \tag{6.43}$$

Next, we investigate the convergence properties of equation (6.43). we assume the following linear statistical model for the desired signal:

$$d(n) = w^o(n) * u(n) + e^o(n), \tag{6.44}$$

where w^o is the impulse response of the Wiener optimal filter and $e^o(n)$ is the irreducible estimation error, which is white noise and statistically independent of the adaptive filter input. The above equation can be written in the HOT domain form

$$\begin{bmatrix} \mathbf{0}_{\frac{K}{2}} \\ \hat{\mathbf{d}}_0(k) \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} & \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} \\ \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} & \mathbf{I}_{\frac{K}{2}\times\frac{K}{2}} \end{bmatrix} \mathbf{F}_K^{-1}$$

$$\times \left[\mathbf{I}_0 \mathbf{w}_H^o(k) \otimes \mathbf{I}_0 \mathbf{u}_H(k) + \mathbf{D} \sum_{i=1}^{K-1} \mathbf{I}_{K-i} \mathbf{w}_H^o(k) \otimes \mathbf{I}_i \mathbf{u}_H(k) + \mathbf{F}_K \mathbf{e}_0^o(k) \right]. \tag{6.45}$$

This form will be useful to obtain the stochastic difference equation that describes the convergence of the adaptive algorithm. Using the above equation to replace the desired signal in equation (6.39), we have

$$\mathbf{F}_K \mathbf{e}_0(k) = \mathbf{F}_K \begin{bmatrix} \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} & \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} \\ \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} & \mathbf{I}_{\frac{K}{2}\times\frac{K}{2}} \end{bmatrix} \mathbf{F}_K^{-1}$$

$$\times \left[\mathbf{I}_0 \boldsymbol{\epsilon}_H(k) \otimes \mathbf{I}_0 \mathbf{u}_H(k) + \mathbf{D} \sum_{i=1}^{K-1} \mathbf{I}_{K-i} \boldsymbol{\epsilon}_H(k) \otimes \mathbf{I}_i \mathbf{u}_H(k) + \mathbf{F}_K \mathbf{e}_0^o(k) \right], \tag{6.46}$$

where $\boldsymbol{\epsilon}_H(k)$ is the error in the estimation of the adaptive filter weight vector, i.e.,

$\boldsymbol{\epsilon}_H(k) = \mathbf{w}_H^o - \mathbf{w}_H(k)$. The i^{th} block in equation (6.43) is given by

$$\mathbf{F}_K\mathbf{e}_0(k) \otimes \mathbf{I}_i\mathbf{u}_H^c(k) = \text{Diag}\left[\mathbf{I}_i\mathbf{u}_H^c(k)\right]\mathbf{F}_K\mathbf{e}_0(k). \tag{6.47}$$

Substituting equation (6.46) into equation (6.47) yields

$$\mathbf{F}_K\mathbf{e}_0(k) \otimes \mathbf{I}_i\mathbf{u}_H^c(k) = \text{Diag}\left[\mathbf{I}_i\mathbf{u}_H^c(k)\right]\mathbf{F}_K\begin{bmatrix} \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} & \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} \\ \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} & \mathbf{I}_{\frac{K}{2}\times\frac{K}{2}} \end{bmatrix}\mathbf{F}_K^{-1} \times$$

$$\left[\text{Diag}\left[\mathbf{I}_0\mathbf{u}_H(k)\right]\mathbf{I}_0\boldsymbol{\epsilon}_H(k) + \mathbf{D}\sum_{i=1}^{K-1}\text{Diag}\left[\mathbf{I}_{K-i}\mathbf{u}_H(k)\right]\mathbf{I}_i\boldsymbol{\epsilon}_H(k) + \mathbf{F}_K\mathbf{e}^o(k)\right]. \tag{6.48}$$

Upon defining

$$\mathbf{T}_{i,j} = \text{Diag}\left[\mathbf{I}_i\mathbf{u}_H^c(k)\right]\mathbf{L}_K\text{Diag}\left[\mathbf{I}_j\mathbf{u}_H(k)\right], \tag{6.49}$$

where

$$\mathbf{L}_K = \mathbf{F}_K\begin{bmatrix} \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} & \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} \\ \mathbf{0}_{\frac{K}{2}\times\frac{K}{2}} & \mathbf{I}_{\frac{K}{2}\times\frac{K}{2}} \end{bmatrix}\mathbf{F}_K^{-1}, \tag{6.50}$$

the i^{th} block of equation (6.43) can be written as

$$\begin{aligned}\mathbf{F}_K\mathbf{e}^o(k) \otimes \mathbf{I}_i\mathbf{u}_H^c(k) &= \begin{bmatrix} \mathbf{T}_{i,0} & \mathbf{T}_{i,K-1} & \mathbf{T}_{i,K-2} & \cdots & \mathbf{T}_{i,1} \end{bmatrix}\begin{bmatrix} \mathbf{I}_0\boldsymbol{\epsilon}_H(k) \\ \mathbf{D}\mathbf{I}_1\boldsymbol{\epsilon}_H(k) \\ \mathbf{D}\mathbf{I}_2\boldsymbol{\epsilon}_H(k) \\ \vdots \\ \mathbf{D}\mathbf{I}_{K-1}\boldsymbol{\epsilon}_H(k) \end{bmatrix} \\ &\quad + \text{Diag}\left[\mathbf{I}_i\mathbf{u}_H^c(k)\right]\mathbf{L}_K\mathbf{e}^o(k). \end{aligned} \tag{6.51}$$

Using the fact that

$$\mathrm{Diag}\,[\mathbf{v}]\,\mathbf{R}\,\mathrm{Diag}\,[\mathbf{u}] = \left(\mathbf{v}\mathbf{u}^T\right) \otimes \mathbf{R}, \tag{6.52}$$

equation (6.49) can be written as

$$\mathbf{T}_{i,j} = \left(\mathbf{I}_i \mathbf{u}_H^c(k) \left(\mathbf{I}_j \mathbf{u}_H(k)\right)^T\right) \otimes \mathbf{L}_K. \tag{6.53}$$

Define

$$\mathbf{U}_{K^2} = \mathbf{H} \begin{bmatrix} \mathbf{I}_{\frac{K^2}{2}\times\frac{K^2}{2}} & \mathbf{0}_{\frac{K^2}{2}\times\frac{K^2}{2}} \\ \mathbf{0}_{\frac{K^2}{2}\times\frac{K^2}{2}} & \mathbf{0}_{\frac{K^2}{2}\times\frac{K^2}{2}} \end{bmatrix} \mathbf{H}^{-1}. \tag{6.54}$$

Then

$$\begin{aligned} \mathbf{w}_H(k+1) &= \mathbf{w}_H(k) \\ &+ \mu\, \mathbf{U}_{K^2} \mathbf{I}_K^{-1} \mathbf{T} \begin{bmatrix} \mathbf{I}_0 \boldsymbol{\epsilon}_H(k) \\ \mathbf{D}\mathbf{I}_1 \boldsymbol{\epsilon}_H(k) \\ \mathbf{D}\mathbf{I}_2 \boldsymbol{\epsilon}_H(k) \\ \vdots \\ \mathbf{D}\mathbf{I}_{K-1} \boldsymbol{\epsilon}_H(k) \end{bmatrix} + \mu\, \mathbf{U}_{K^2} \mathbf{I}_K^{-1} \begin{bmatrix} \mathrm{Diag}\,[\mathbf{I}_0 \mathbf{u}_H^c(k)] \\ \mathrm{Diag}\,[\mathbf{I}_1 \mathbf{u}_H^c(k)] \\ \mathrm{Diag}\,[\mathbf{I}_2 \mathbf{u}_H^c(k)] \\ \vdots \\ \mathrm{Diag}\,[\mathbf{I}_{K-1} \mathbf{u}_H^c(k)] \end{bmatrix} \mathbf{L}_K \mathbf{e}^o(k). \end{aligned} \tag{6.55}$$

The matrix $\mathbf{T}$ can be written as

$$\mathbf{T} = \left(\mathbf{1}_{K\times K} \times \mathbf{L}_K\right) \otimes$$

$$\begin{bmatrix} \mathbf{I}_0 \mathbf{u}_H^c(k) \left[\mathbf{I}_0 \mathbf{u}_H(k)\right]^T & \mathbf{I}_0 \mathbf{u}_H^c(k) \left[\mathbf{I}_{K-1} \mathbf{u}_H(k)\right]^T & \cdots & \mathbf{I}_0 \mathbf{u}_H^c(k) \left[\mathbf{I}_1 \mathbf{u}_H(k)\right]^T \\ \mathbf{I}_1 \mathbf{u}_H^c(k) \left[\mathbf{I}_0 \mathbf{u}_H(k)\right]^T & \mathbf{I}_1 \mathbf{u}_H^c(k) \left[\mathbf{I}_{K-1} \mathbf{u}_H(k)\right]^T & \cdots & \mathbf{I}_1 \mathbf{u}_H^c(k) \left[\mathbf{I}_1 \mathbf{u}_H(k)\right]^T \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{I}_{K-1} \mathbf{u}_H^c(k) \left[\mathbf{I}_0 \mathbf{u}_H(k)\right]^T & \mathbf{I}_{K-1} \mathbf{u}_H^c(k) \left[\mathbf{I}_{K-1} \mathbf{u}_H(k)\right]^T & \cdots & \mathbf{I}_{K-1} \mathbf{u}_H^c(k) \left[\mathbf{I}_1 \mathbf{u}_H(k)\right]^T \end{bmatrix},$$

where $\times$ denotes the Kronecker product and $\mathbf{1}_{K\times K}$ is the $K\times K$ matrix with all element being equal to one. The matrix $\mathbf{T}$ can be written as

$$\begin{aligned}\mathbf{T} &= \begin{bmatrix}\mathbf{I}_0\mathbf{u}_H^c(k)\\ \mathbf{I}_1\mathbf{u}_H^c(k)\\ \vdots\\ \mathbf{I}_{K-2}\mathbf{u}_H^c(k)\\ \mathbf{I}_{K-1}\mathbf{u}_H^c(k)\end{bmatrix}\begin{bmatrix}\mathbf{I}_0\mathbf{u}_H(k)\\ \mathbf{I}_{K-1}\mathbf{u}_H(k)\\ \vdots\\ \mathbf{I}_2\mathbf{u}_H(k)\\ \mathbf{I}_1\mathbf{u}_H(k)\end{bmatrix}^T \otimes \Big(\mathbf{1}_{K\times K}\times\mathbf{L}_K\Big)\\ &= \Big(\mathbf{I}_K\mathbf{u}_H^c(k)\mathbf{u}_H^T(k){\mathbf{I}_K^c}^T\Big)\otimes\Big(\mathbf{1}_{K\times K}\times\mathbf{L}_K\Big),\end{aligned} \tag{6.56}$$

where

$$\mathbf{I}_K^c = \begin{bmatrix}\mathbf{I}_0\\ \mathbf{I}_{K-1}\\ \vdots\\ \mathbf{I}_1\end{bmatrix}. \tag{6.57}$$

Finally, the error in the estimation of the adaptive filter is given by

$$\begin{aligned}\boldsymbol{\epsilon}_H(k+1) &= \bigg(\mathbf{I}-\mu\mathbf{U}_{K^2}\mathbf{I}_K^{-1}\Big(\mathbf{I}_K\mathbf{u}_H^c(k)\mathbf{u}_H^T(k){\mathbf{I}_K^c}^T\Big)\otimes\Big(\mathbf{1}_{K\times K}\times\mathbf{L}_K\Big)\mathbf{I}_K^D\bigg)\boldsymbol{\epsilon}_H(k)\\ &\quad -\mu\mathbf{U}_{K^2}\mathbf{I}_K^{-1}\begin{bmatrix}\mathrm{Diag}[\mathbf{I}_0\mathbf{u}_H^c(k)]\\ \mathrm{Diag}[\mathbf{I}_1\mathbf{u}_H^c(k)]\\ \vdots\\ \mathrm{Diag}[\mathbf{I}_{K-2}\mathbf{u}_H^c(k)]\\ \mathrm{Diag}[\mathbf{I}_{K-1}\mathbf{u}_H^c(k)]\end{bmatrix}\mathbf{L}_K\mathbf{e}^o(k),\end{aligned} \tag{6.58}$$

where

$$\mathbf{I}_K^D = \begin{bmatrix} \mathbf{I}_0 \\ \mathbf{D}\mathbf{I}_1 \\ \mathbf{D}\mathbf{I}_2 \\ \vdots \\ \mathbf{D}\mathbf{I}_{K-2} \\ \mathbf{D}\mathbf{I}_{K-1} \end{bmatrix}. \tag{6.59}$$

Therefore, the adaptive block HOT filter convergence is governed by the matrix

$$\mathbf{\Psi} = \mathbf{H} \begin{bmatrix} \mathbf{I}_{\frac{K^2}{2}\times\frac{K^2}{2}} & \mathbf{0}_{\frac{K^2}{2}\times\frac{K^2}{2}} \\ \mathbf{0}_{\frac{K^2}{2}\times\frac{K^2}{2}} & \mathbf{0}_{\frac{K^2}{2}\times\frac{K^2}{2}} \end{bmatrix} \mathbf{H}^{-1}\mathbf{I}_K^{-1}\Big(\mathbf{I}_K E\mathbf{u}_H^c(k)\mathbf{u}_H^T(k)\mathbf{I}_K^{cT}\Big) \otimes \Big(\mathbf{1}_{K\times K} \times \mathbf{L}_K\Big)\mathbf{I}_K^D. \tag{6.60}$$

The structure of $\mathbf{\Psi}$ is now analyzed. Using the relation between the HOT and the DFT transforms, we can write

$$\mathbf{I}_K\mathbf{u}_H^c = \begin{bmatrix} \mathbf{F}_K\mathbf{u}_0^c \\ \mathbf{F}_K\mathbf{u}_1^c \\ \vdots \\ \mathbf{F}_K\mathbf{u}_{K-2}^c \\ \mathbf{F}_K\mathbf{u}_{K-1}^c \end{bmatrix}. \tag{6.61}$$

It can be easily shown that

$$\mathbf{F}_K\mathbf{u}_i^c = \begin{cases} \mathbf{F}_K^H\mathbf{u}_i & \text{if } i = 0, \\ \mathbf{D}^*\mathbf{F}_K^H\mathbf{u}_{K-i} & \text{if } i \neq 0. \end{cases} \tag{6.62}$$

Then we have

$$\mathbf{I}_K \mathbf{u}_K^c = \begin{bmatrix} \mathbf{F}_K^H \mathbf{u}_0 \\ \mathbf{D}^* \mathbf{F}_K^H \mathbf{u}_{K-1} \\ \vdots \\ \mathbf{D}^* \mathbf{F}_K^H \mathbf{u}_2 \\ \mathbf{D}^* \mathbf{F}_K^H \mathbf{u}_1 \end{bmatrix} \tag{6.63}$$

and

$$\mathbf{I}_K \mathbf{u}_H^c(k) \mathbf{u}_H^T(k) \mathbf{I}_K^{cT} = \begin{bmatrix} \mathbf{F}_K^H \mathbf{u}_0 \\ \mathbf{D}^* \mathbf{F}_K^H \mathbf{u}_{K-1} \\ \vdots \\ \mathbf{D}^* \mathbf{F}_K^H \mathbf{u}_2 \\ \mathbf{D}^* \mathbf{F}_K^H \mathbf{u}_1 \end{bmatrix} \begin{bmatrix} \mathbf{F}_K \mathbf{u}_0 \\ \mathbf{F}_K \mathbf{u}_{K-1} \\ \vdots \\ \mathbf{F}_K \mathbf{u}_2 \\ \mathbf{F}_K \mathbf{u}_1 \end{bmatrix}^T . \tag{6.64}$$

Taking the expectation of equation (6.64) yields

$$\mathbf{I}_K E \mathbf{u}_H^c(k) \mathbf{u}_H^T(k) \mathbf{I}_K^{cT} =$$

$$\begin{bmatrix} \mathbf{F}_K^H E \mathbf{u}_0 \mathbf{u}_0^T \mathbf{F}_K & \mathbf{F}_K^H E \mathbf{u}_0 \mathbf{u}_{K-1}^T \mathbf{F}_K & \dots & \mathbf{F}_K^H E \mathbf{u}_0 \mathbf{u}_1^T \mathbf{F}_K \\ \mathbf{D}^* \mathbf{F}_K^H E \mathbf{u}_{K-1} \mathbf{u}_0^T \mathbf{F}_K & \mathbf{D}^* \mathbf{F}_K^H E \mathbf{u}_{K-1} \mathbf{u}_{K-1}^T \mathbf{F}_K & \dots & \mathbf{D}^* \mathbf{F}_K^H E \mathbf{u}_{K-1} \mathbf{u}_1^T \mathbf{F}_K \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{D}^* \mathbf{F}_K^H E \mathbf{u}_1 \mathbf{u}_0^T \mathbf{F}_K & \mathbf{D}^* \mathbf{F}_K^H E \mathbf{u}_1 \mathbf{u}_{K-1}^T \mathbf{F}_K & \dots & \mathbf{D}^* \mathbf{F}_K^H E \mathbf{u}_1 \mathbf{u}_1^T \mathbf{F}_K \end{bmatrix} .$$

Each block in the above equation is an autocorrelation matrix that is asymptotically diagonalized by the DFT matrix. Each block will be also pointwise multiplied by $\mathbf{L}_K$. Three-dimensional representations of $\mathbf{L}_K$ for $K = 16$ and $K = 32$ are shown in Figures 6.4 and 6.5, respectively. The diagonal elements of $\mathbf{L}_K$ are much higher than the off diagonal elements. Therefore, pointwise multiplying each block in the previous equation with $\mathbf{L}_K$ makes it more diagonal. If each block is perfectly diagonal, then $\mathbf{I}_K\left(\mathbf{I}_K E\mathbf{u}_H^c(k)\mathbf{u}_H^T(k)\mathbf{I}_K^{cT}\right)\otimes(\mathbf{1}_{K\times K}\times\mathbf{L}_K)\mathbf{I}_K^D$ will be block diagonal. Asymptotically the HOT block LMS adaptive filter transforms the K^2 modes into K decoupled sets of modes. The convergence rate of the HOT block LMS adaptive filter can be increased if each block is given an individual step size that is inversely proportional the diagonal elements of $\mathbf{I}_K\left(\mathbf{I}_K E\mathbf{u}_H^c(k)\mathbf{u}_H^T(k)\mathbf{I}_K^{cT}\right)\mathbf{I}_K^D$.

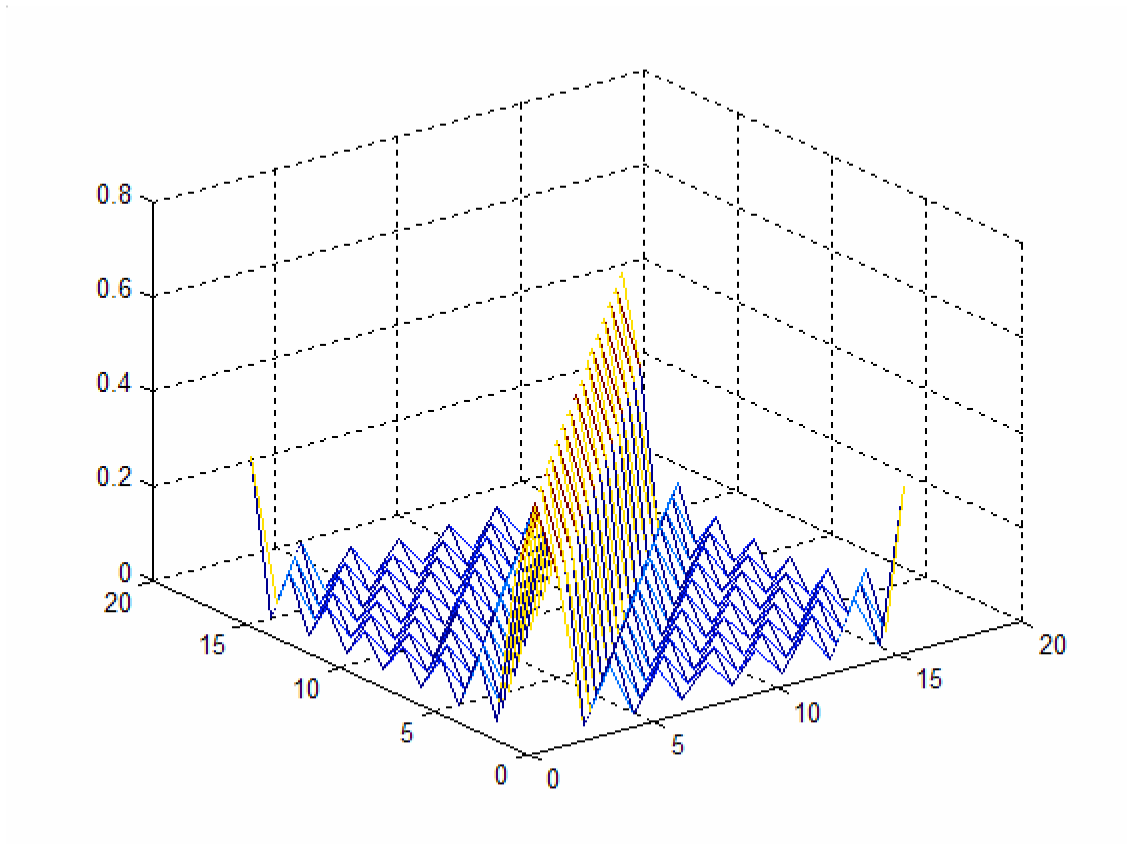


Figure 6.4: Three-dimensional representation of $\mathbf{L}_{16}$.

To numerically verify the above theoretical predictions, the performance of the HOT block LMS adaptive filter was simulated with colored input. The input of the adaptive filter was generated by coloring unit variance white noise using the FIR filter

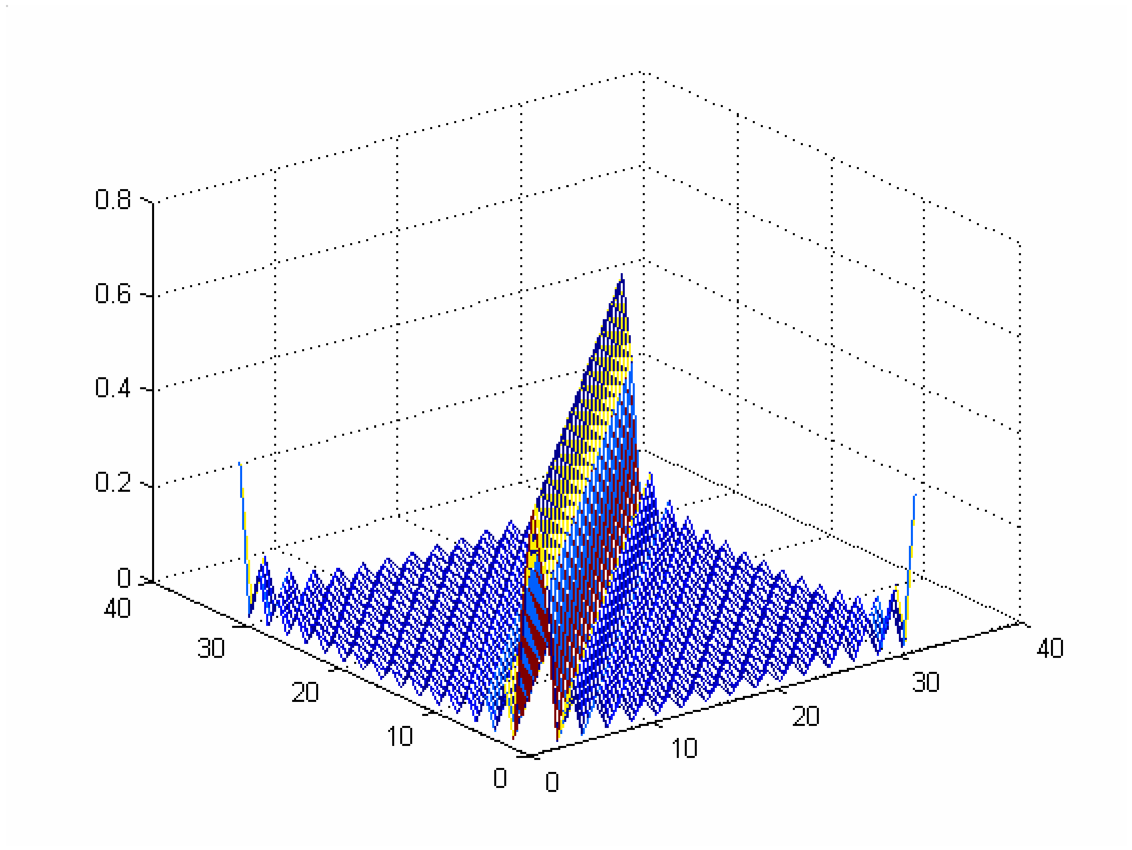


Figure 6.5: Three-dimensional representation of $\mathbf{L}_{32}$.

$H(z) = 0.1 + 0.2z^{-1} + 0.3z^{-2} + 0.4z^{-3} + 0.4z^{-4} + 0.2z^{-5} + 0.1z^{-6}$. The desired input was generated using the linear model $d(n) = w^o(n) * u(n) + e^o(n)$, where $e^o(n)$ is the measurement white noise with variance 10^{-4}. The filter length was 256. The learning curves of the LMS, DFT block LMS, and HOT block LMS algorithms are shown in Figure 6.6. The learning curves show that the convergence rate of the HOT block LMS filter is close to that of the DFT block LMS filter.

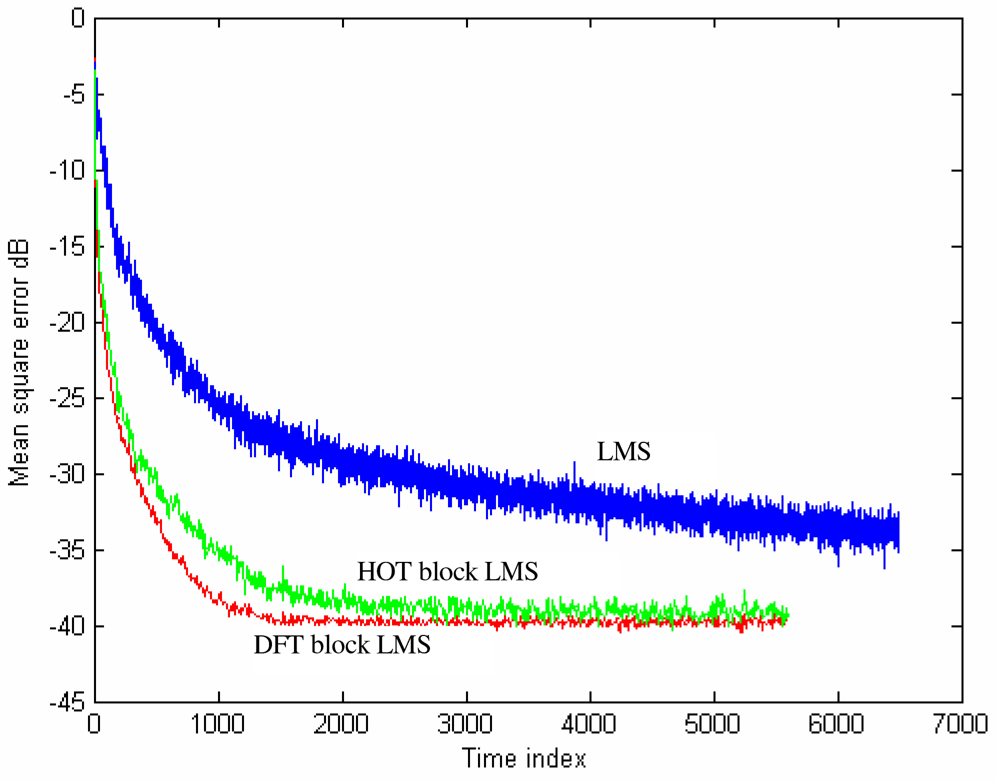


Figure 6.6: Learning curves of the LMS, DFT block LMS, and HOT block LMS algorithms. K = 16.

Chapter 7

HOT DFT Block LMS Algorithm

Based on a fast HOT convolution developed by Matusiak and DeBrunner [15], another block LMS algorithm is developed. This new block LMS algorithm is called the HOT DFT block LMS algorithm. This new LMS algorithm assumes that the filter length is much smaller than the block length. The computational efficiency of the HOT DFT block LMS algorithm is verified and its convergence is analyzed.

7.1 Development of the HOT DFT Block LMS Algorithm

Recall that in the block LMS algorithm there are two convolutions needed. The first convolution is a convolution between the filter impulse response and the filter input and needed to calculate the output of the filter in each block. The second convolution is a convolution between the filter input and error and is needed to estimate the gradient in the filter weight update equation. If the block length is much larger than the filter length, then the fast convolution in [15] can be used to calculate the first convolution. However, the second convolution is a convolution between two signals of the same length and the method in [15] can not be used directly without modification. The fast convolution in [15] is based on the overlap-add method [45]. Since the overlap-save method is more convenient in the block LMS algorithm, the fast convolution in [15] is developed with the overlap-save method here and then applied to the block LMS algorithm.

Let N be the filer length and $L = NK$ be the block length, where N, L, and K are all integers. Let

$$\hat{\mathbf{w}}(k) = \begin{bmatrix} w_0(k) \\ w_1(k) \\ \vdots \\ w_{N-2}(k) \\ w_{N-1}(k) \end{bmatrix} \tag{7.1}$$

be the filter tap-weight vector in the k^{th} block and

$$\mathbf{u}(k) = \begin{bmatrix} u\left(kL - N + 1\right) \\ \vdots \\ u\left(kL\right) \\ u\left(kL + 1\right) \\ \vdots \\ u\left(kL + L - 1\right) \end{bmatrix} \tag{7.2}$$

be the vector of input samples needed in the k^{th} block. The overlap-save method divides this vector is into K N-overlapping sections. Such sections can be formed by multiplying $\mathbf{u}(k)$ with the following matrix:

$$\mathbf{J} = \begin{bmatrix} \mathbf{I}_{N\times N} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N\times N} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N\times N} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{N\times N} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_{N\times N} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_{N\times N} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I}_{N\times N} \end{bmatrix}. \tag{7.3}$$

The vector $\mathbf{z}(k) = \mathbf{J}\mathbf{u}(k)$ will be then of length $2NK$. Let $\mathbf{z}_i(k)$ be the i^{th} polyphase component of $\mathbf{z}(k)$. Then

$$\begin{bmatrix} \mathbf{z}_0(k) \\ \mathbf{z}_1(k) \\ \vdots \\ \mathbf{z}_{K-2}(k) \\ \mathbf{z}_{K-1}(k) \end{bmatrix} = \mathbf{I}_{2NK}\mathbf{J}\mathbf{u}(k). \tag{7.4}$$

Define the extended tap-weight vector (post appended with N zeros)

$$\mathbf{w}(k) = \begin{bmatrix} \hat{\mathbf{w}}(k) \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{7.5}$$

The DFT of $\mathbf{w}(k)$ is given by

$$\mathbf{w}_F(k) = \mathbf{F}_{2N}\mathbf{w}(k). \tag{7.6}$$

Let $\mathbf{e}_{K\times 1}$ be a column vector of all ones, i.e.,

$$\mathbf{e}_{K\times 1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{7.7}$$

Then the $2NK \times 2N$ matrix $\mathbf{E}$ is given by

$$\mathbf{E} = \begin{bmatrix} \mathbf{e}_{K\times 1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{e}_{K\times 1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{e}_{K\times 1} \end{bmatrix}. \tag{7.8}$$

Define the matrices

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{N\times N} & \mathbf{I}_{N\times N} \end{bmatrix}, \tag{7.9}$$

and

$$\mathbf{G} = \begin{bmatrix} \mathbf{A} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A} \end{bmatrix}. \tag{7.10}$$

According to the overlap-save method, the output of the adaptive filter in the k^{th}

block

$$\mathbf{y}(k) = \left[\begin{array}{ccccc} y(kL) & y(kL+1) & \cdots & y(kL+L-2) & y(kL+L-1) \end{array}\right]^T \tag{7.11}$$

is given by

$$\mathbf{y}(k) = \mathbf{G}\mathbf{I}_{2NK}\mathbf{H}^{-1}\Big(\mathbf{E}\mathbf{w}_F(k)\Big) \otimes \Big(\mathbf{H}\mathbf{I}_{2NK}\mathbf{J}\mathbf{u}(k)\Big). \tag{7.12}$$

The desired signal vector and the filter error in the k^{th} block are given by

$$\mathbf{d}(k) = \left[\begin{array}{ccccc} d(kL) & d(kL+1) & \cdots & d(kL+L-2) & d(kL+L-1) \end{array}\right]^T \tag{7.13}$$

and

$$\mathbf{e}(k) = \left[\begin{array}{ccccc} e(kL) & e(kL+1) & \cdots & e(kL+L-2) & e(kL+L-1) \end{array}\right]^T, \tag{7.14}$$

respectively, where

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{y}(k). \tag{7.15}$$

The filter update equation is given by

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) + \frac{\mu}{L}\sum_{i=0}^{L-1} \left[\begin{array}{c} u\left(kL+i\right) \\ u\left(kL+i-1\right) \\ \vdots \\ u\left(kL+i-N+2\right) \\ u\left(kL+i-N+1\right) \end{array}\right] e(kL+i). \tag{7.16}$$

The sum in equation (7.16) can be efficiently calculated using the L-point DFTs of the error vector $\mathbf{e}(k)$ and input vector $\mathbf{u}(k)$. However, the L-point DFT of $\mathbf{u}(k)$ is not available and only the $2N$-point DFTs of the K sections of $\mathbf{u}(k)$ are available. Therefore, the sum in equation (7.16) should be divided into K sections as follows:

$$\sum_{i=0}^{L-1} \begin{bmatrix} u\left(kL+i\right) \\ u\left(kL+i-1\right) \\ \vdots \\ u\left(kL+i-N+2\right) \\ u\left(kL+i-N+1\right) \end{bmatrix} e(kL+i) =$$

$$\sum_{l=0}^{K-1}\sum_{i=0}^{N-1} \begin{bmatrix} u\left(kL+lN+i\right) \\ u\left(kL+lN+i-1\right) \\ \vdots \\ u\left(kL+lN+i-N+2\right) \\ u\left(kL+lN+i-N+1\right) \end{bmatrix} e(kL+lK+i). \tag{7.17}$$

For each l, the sum over i can be calculated as follows. First, form the vectors

$$\mathbf{u}_l(k) = \begin{bmatrix} u(kL+lN-N) & \cdots & u(kL+lN+N-2) & u(kL+lN+N-1) \end{bmatrix}^T, \tag{7.18}$$

$$\mathbf{e}_l(k) = \begin{bmatrix} \mathbf{0}_{N\times 1} & e(kL+lN) & \cdots & e(kL+lN+N-2) & e(kL+lN+N-1) \end{bmatrix}^T. \tag{7.19}$$

Then the sum over i is just the first N elements of the circular convolution of $\mathbf{e}_l(k)$ and circularly shifted $\mathbf{u}_l(k)$. Therefore, the filter update equation for the HOT DFT block LMS algorithm can be written as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu}{L}\sum_{l=0}^{K-1}\begin{bmatrix} \mathbf{I}_{N\times N} & \mathbf{0}_{N\times N} \\ \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} \end{bmatrix} \mathbf{F}_{2N}^{-1}\Big(\mathbf{F}_{2N}\mathbf{u}_l(k)\Big)^* \otimes \Big(\mathbf{F}_{2N}\mathbf{e}_l(k)\Big). \quad (7.20)$$

7.2 Computational Cost of the HOT DFT Block LMS Algorithm

Before looking at the convergence analysis of the new adaptive filter, we look at its computational cost. To calculate the the output of the k^{th} block, $2K+1$ $2N$-point DFTs are needed. Therefore, $(2K+1)2N\log_2 2N+2NK$ multiplications are needed to calculate the output. To calculate the gradient estimate in the filter update equation, $2K$ $2N$-point DFTs are required. Therefore, $6KN\log_2 2N + 2NK$ multiplications are needed. The total multiplication count of the new algorithm is then $(4K+1)2N\log_2 2N + 4NK$. The multiplication count for the DFT block LMS algorithm is $10KN\log_2 2NK + 4NK$. Therefore, as K gets larger the HOT DFT block LMS algorithm becomes more efficient than the DFT block LMS algorithm. For example, for $N = 100$ and $K = 10$, the HOT DFT LMS algorithm is about 30% more efficient and for for $N = 50$ and $K = 20$ the HOT DFT LMS algorithm is about 40% more efficient.

The ratio between the number of multiplications required for the HOT DFT block LMS algorithm and the number of multiplications required for the DFT block LMS algorithm is plotted in Figure 7.1 for different filter lengths. The HOT DFT block LMS filter is always more efficient than the DFT block LMS filter and the efficiency increases as the block length increases.

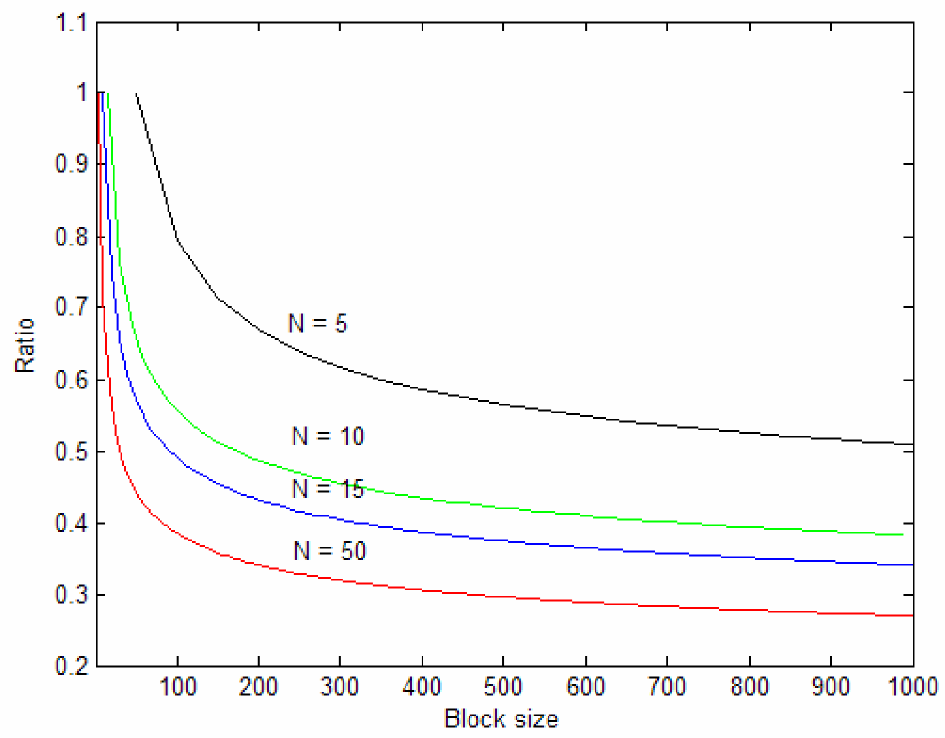


Figure 7.1: Ratio between the number of multiplications required for the HOT DFT and the DFT block LMS algorithms.

7.3 Convergence Analysis of the HOT DFT LMS Algorithm

Now the convergence of the new algorithm is analyzed. The analysis is performed in the DFT domain. The adaptive filter update equation in the DFT domain is given by

$$\mathbf{w}_F(k+1) = \mathbf{w}_F(k) + \frac{\mu}{L}\sum_{l=0}^{K-1} \mathbf{F}_{2N} \begin{bmatrix} \mathbf{I}_{N\times N} & \mathbf{0}_{N\times N} \\ \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} \end{bmatrix} \mathbf{F}_{2N}^{-1}\Big(\mathbf{F}_{2N}\mathbf{u}_l(k)\Big)^* \otimes \Big(\mathbf{F}_{2N}\mathbf{e}_l(k)\Big). \tag{7.21}$$

Let the desired signal be generated using the linear regression model

$$d(n) = w^o(n) * u(n) + e^o(n), \tag{7.22}$$

where $w^o(n)$ is the impulse response of the Wiener optimal filter and $e^o(n)$ is the irreducible estimation error, which is white noise and statistically independent of the adaptive filter input. In the k^{th} block, the l^{th} section of the desired signal in the DFT domain is given by

$$\mathbf{d}_l(k) = \begin{bmatrix} \mathbf{0}_{N\times N} & \mathbf{I}_{N\times N} \end{bmatrix} \mathbf{F}_{2N}^{-1}\mathbf{w}_F^o(k) \otimes \Big(\mathbf{F}_{2N}\mathbf{u}_l(k)\Big) + \hat{\mathbf{e}}_l^o(k). \tag{7.23}$$

Therefore, the l^{th} section of the error is given by

$$\mathbf{e}_l(k) = \begin{bmatrix} \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} \\ \mathbf{0}_{N\times N} & \mathbf{I}_{N\times N} \end{bmatrix} \mathbf{F}_{2N}^{-1}\Big(\mathbf{w}_F^o(k) - \mathbf{w}_F(k)\Big) \otimes \Big(\mathbf{F}_{2N}\mathbf{u}_l(k)\Big) + \mathbf{e}_l^o(k). \tag{7.24}$$

Using equation (7.21), the error in the estimation of the adaptive filter weight vector $\boldsymbol{\epsilon}_F(k) = \mathbf{w}_F^o - \mathbf{w}_F(k)$ is updated according to

$$\boldsymbol{\epsilon}_F(k+1) = \boldsymbol{\epsilon}_F(k) - \frac{\mu}{L}\sum_{l=0}^{K-1} \mathbf{U}\Big(\mathbf{F}_{2N}\mathbf{u}_l(k)\Big)^* \otimes \Big(\mathbf{F}_{2N}\mathbf{e}_l(k)\Big), \tag{7.25}$$

where

$$\mathbf{U} = \mathbf{F}_{2N} \begin{bmatrix} \mathbf{I}_{N\times N} & \mathbf{0}_{N\times N} \\ \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} \end{bmatrix} \mathbf{F}_{2N}^{-1}. \tag{7.26}$$

Taking the DFT of equation (7.24), we have that

$$\mathbf{F}_{2N}\mathbf{e}_l(k) = \mathbf{L}\boldsymbol{\epsilon}_F(k) \otimes \Big(\mathbf{F}_{2N}\mathbf{u}_l(k)\Big) + \mathbf{F}_{2N}\mathbf{e}_l^o(k), \tag{7.27}$$

where

$$\mathbf{L} = \mathbf{F}_{2N}\begin{bmatrix} \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} \\ \mathbf{0}_{N\times N} & \mathbf{I}_{N\times N} \end{bmatrix}\mathbf{F}_{2N}^{-1}. \tag{7.28}$$

Using equation (7.27), we can write

$$\Big(\mathbf{F}\mathbf{u}_l(k)\Big)^* \otimes \Big(\mathbf{F}\mathbf{e}_l(k)\Big) = \mathrm{Diag}\left[\mathbf{F}\mathbf{u}_l(k)\right]^* \Big(\mathbf{L}\mathrm{Diag}\left[\mathbf{F}\mathbf{u}_l(k)\right]\boldsymbol{\epsilon}_F(k) + \mathbf{F}\,\mathbf{e}_l^o(k)\Big). \tag{7.29}$$

With equation (6.52), the above equation can be simplified to

$$\Big(\mathbf{F}\mathbf{u}_l(k)\Big)^* \otimes \Big(\mathbf{F}\mathbf{e}_l(k)\Big) = \Big(\mathbf{F}\mathbf{u}_l(k)\Big)^*\Big(\mathbf{F}\mathbf{u}_l(k)\Big)^T \otimes \mathbf{L}\boldsymbol{\epsilon}_F(k) + \mathrm{Diag}\left[\mathbf{F}\mathbf{u}_l(k)\right]^*\mathbf{F}\,\mathbf{e}_l^o(k). \tag{7.30}$$

Substituting equation (7.30) into equation (7.25), we have that

$$\boldsymbol{\epsilon}_F(k+1) = \Big(\mathbf{I} - \frac{\mu}{L}\sum_{l=0}^{K-1}\mathbf{U}\Big(\mathbf{F}\mathbf{u}_l(k)\Big)^*\Big(\mathbf{F}\mathbf{u}_l(k)\Big)^T \otimes \mathbf{L}\Big)\boldsymbol{\epsilon}_F(k) - \frac{\mu}{L}\sum_{l=0}^{K-1}\mathbf{U}\mathrm{Diag}\left[\mathbf{F}\mathbf{u}_l(k)\right]^*\mathbf{F}\mathbf{e}_l^o(k). \tag{7.31}$$

Taking the expectation of the above equation

$$E\boldsymbol{\epsilon}_F(k+1) = \Big(\mathbf{I} - \frac{\mu}{N}\mathbf{U}\left(\mathbf{F}_{2N}^H\mathbf{R}\mathbf{F}_{2N}\right) \otimes \mathbf{L}\Big)E\boldsymbol{\epsilon}_F(k), \tag{7.32}$$

which is similar to the result that corresponds to the DFT block LMS algorithm [22].

The convergence speed of the HOT DFT LMS algorithm can be increased if the convergence moods are normalized using the estimated power of the tap-input vector in the DFT domain. The complete HOT DFT block LMS weight update equation is given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu}{L}\sum_{l=0}^{K-1}\begin{bmatrix} \mathbf{I}_{N\times N} & \mathbf{0}_{N\times N} \\ \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} \end{bmatrix} \mathbf{F}_{2N}^{-1}\Lambda_l^{-1}(k)\Big(\mathbf{F}_{2N}\mathbf{u}_l(k)\Big)^* \otimes \Big(\mathbf{F}_{2N}\mathbf{e}_l(k)\Big) \tag{7.33}$$

and

$$\Lambda_l(k+1) = \frac{k-1}{k}\Lambda_l(k) + \frac{1}{kL}\mathrm{Diag}\left[\Big(\mathbf{F}_{2N}\mathbf{u}_l(k)\Big)^* \otimes \Big(\mathbf{F}_{2N}\mathbf{u}_l(k)\Big)\right]. \tag{7.34}$$

7.4 Simulation of the HOT DFT Block LMS Algorithm

The learning curves of the HOT DFT block LMS algorithm were simulated. The desired input was generated using the linear model $d(n) = w^o(n) * u(n) + e^o(n)$, where $e^o(n)$ is the measurement white gaussian noise with variance 10^{-8}. The input was a first-order Markov signal with autocorrelation function given by $r(k) = \rho^{|k|}$. The filter was lowpass with a cutoff frequency $\pi/2$ rad.

Figure 7.2 shows the learning curves for the HOT DFT block LMS filter with those for the LMS and DFT block LMS filters for $N = 4$, $K = 3$, and $\rho = 0.9$. Figure 7.3 shows similar curves for $N = 50$, $K = 10$, and $\rho = 0.9$. Both figures show that the HOT DFT block LMS algorithm converges at the same rate as the DFT block LMS algorithm and yet is computationally more efficient. Figure 7.4 shows similar curves for $N = 50$ and $K = 10$ and $\rho = 0.8$. As the correlation coefficient decreases the algorithms converges faster and the HOT DFT block LMS algorithm converges at the same rate as the DFT block LMS algorithm.

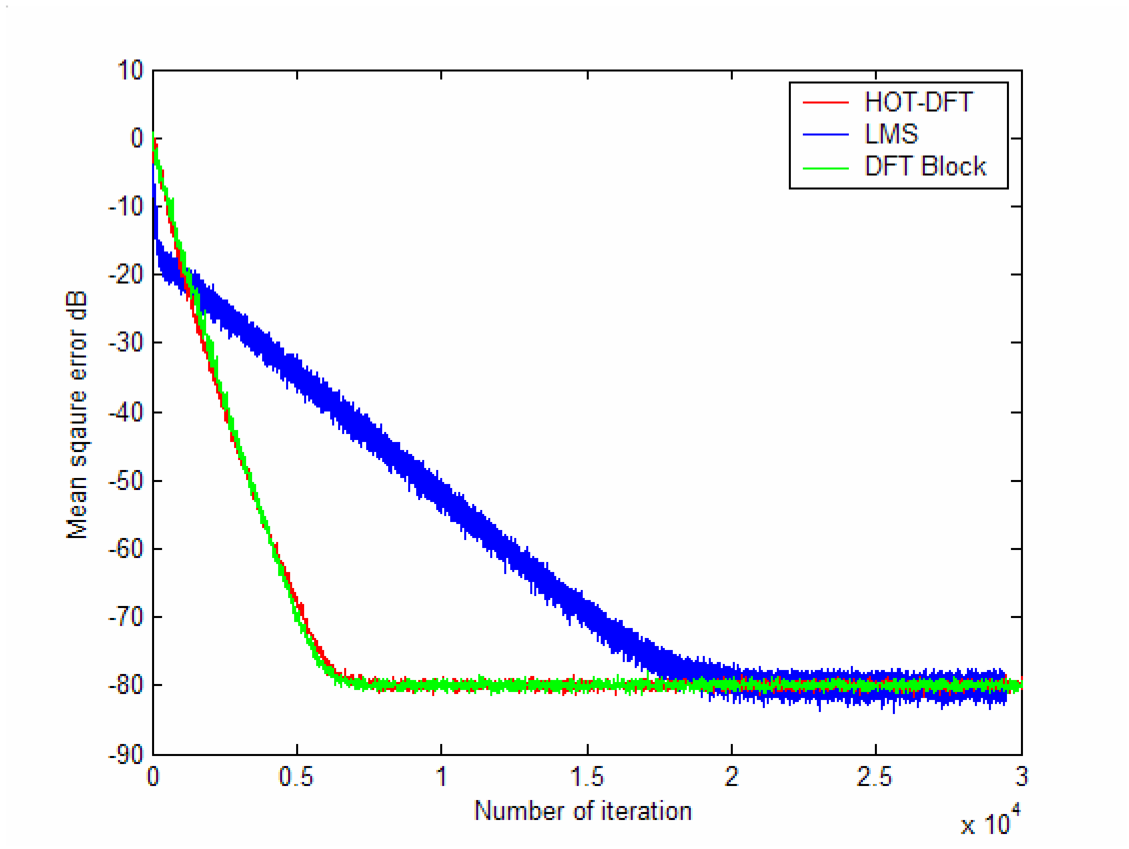


Figure 7.2: Learning curves for the LMS, HOT DFT block LMS, and DFT block LMS algorithms. $N = 4$ and $K = 3$. $\rho = 0.9$.

Another coloring filter was also used to simulate the learning curves of the algorithms. The coloring filter was a bandpass filter with $H(z) = 0.1 - 0.2z^{-1} - 0.3z^{-2} + 0.4z^{-3} + 0.4z^{-4} - 0.2z^{-5} - 0.1z^{-6}$. The frequency response of the coloring filter is shown in Figure 7.5. The learning curves are shown in Figure 7.6. The simulations are again consistent with the theoretical predictions presented in this chapter.

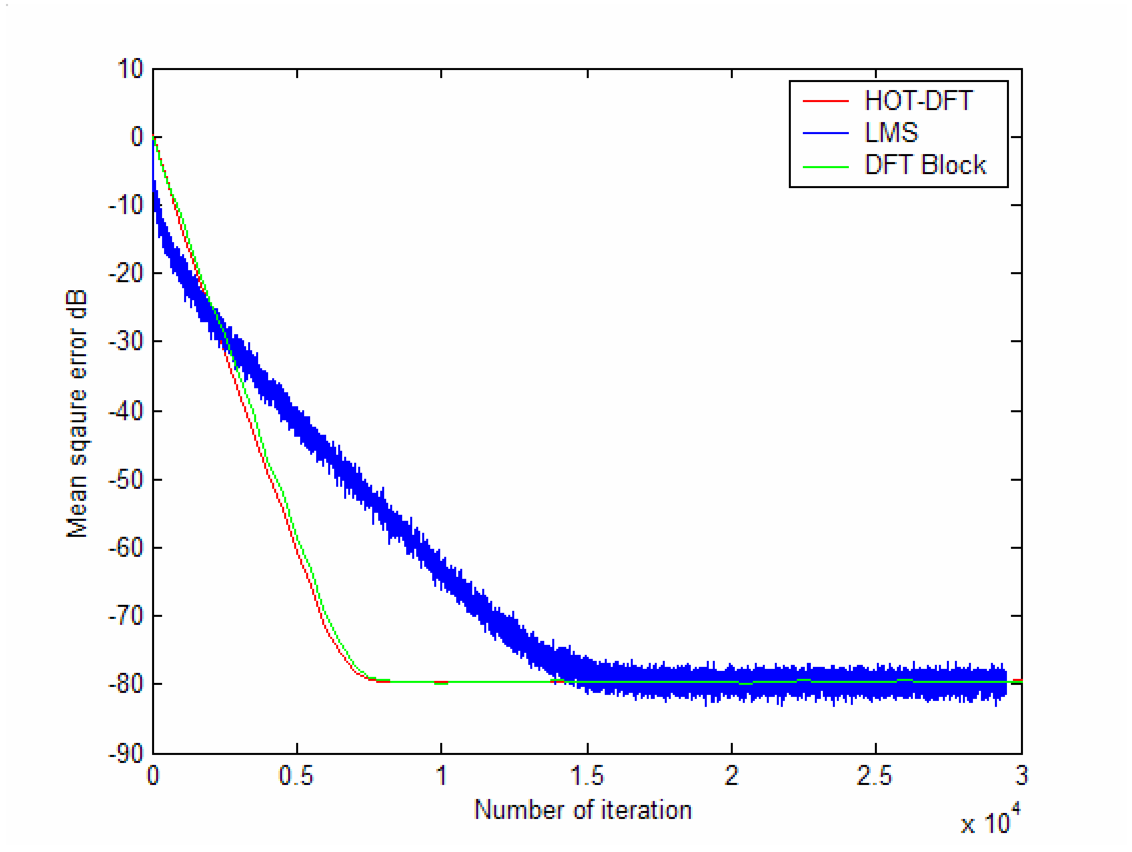


Figure 7.3: Learning curves for the LMS, HOT DFT block LMS, and DFT block LMS algorithms. $N = 50$ and $K = 10$. $\rho = 0.9$.

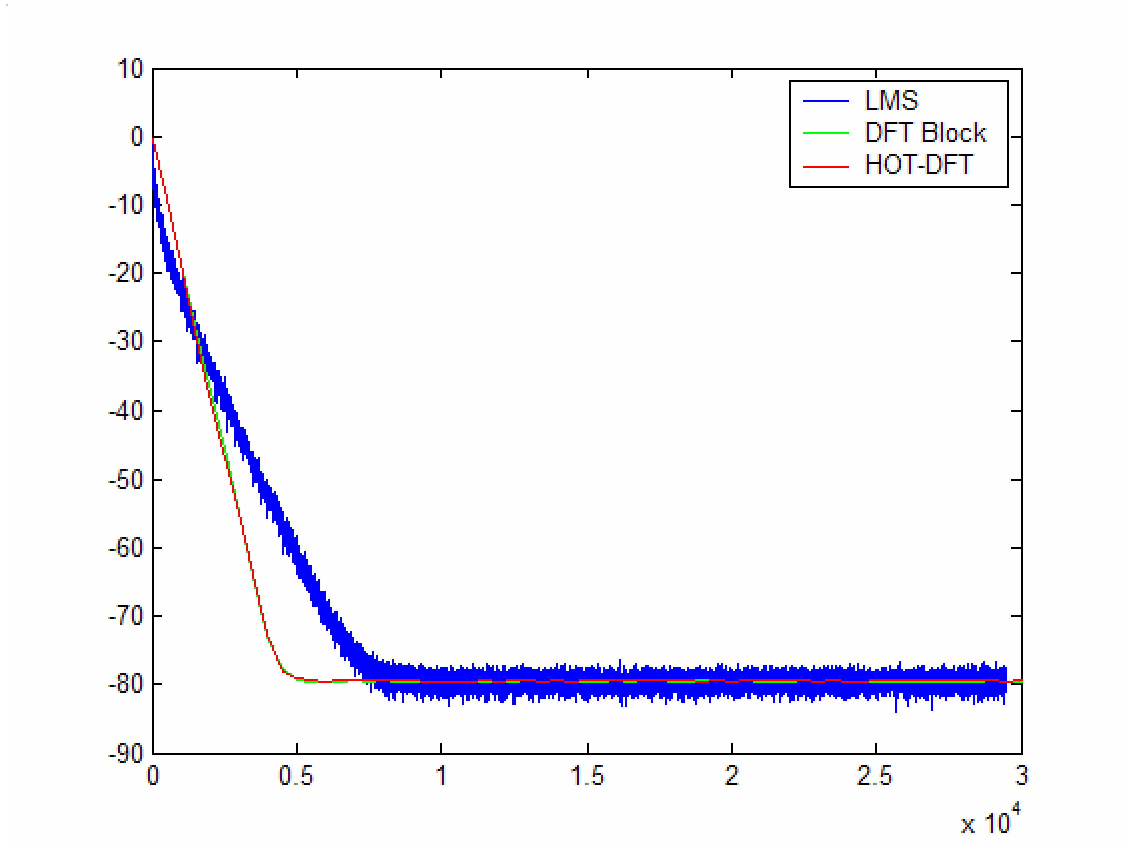


Figure 7.4: Learning curves for the LMS, HOT DFT block LMS, and DFT block LMS algorithms. $N = 50$ and $K = 10$. $\rho = 0.8$.

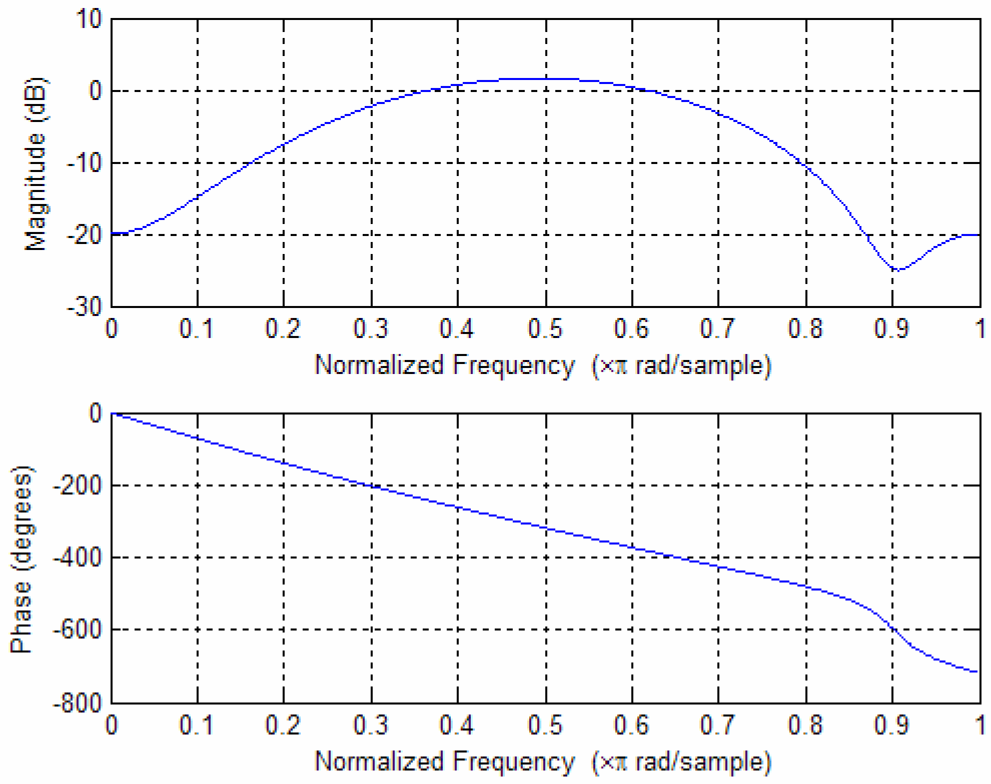


Figure 7.5: Frequency response of the coloring filter.

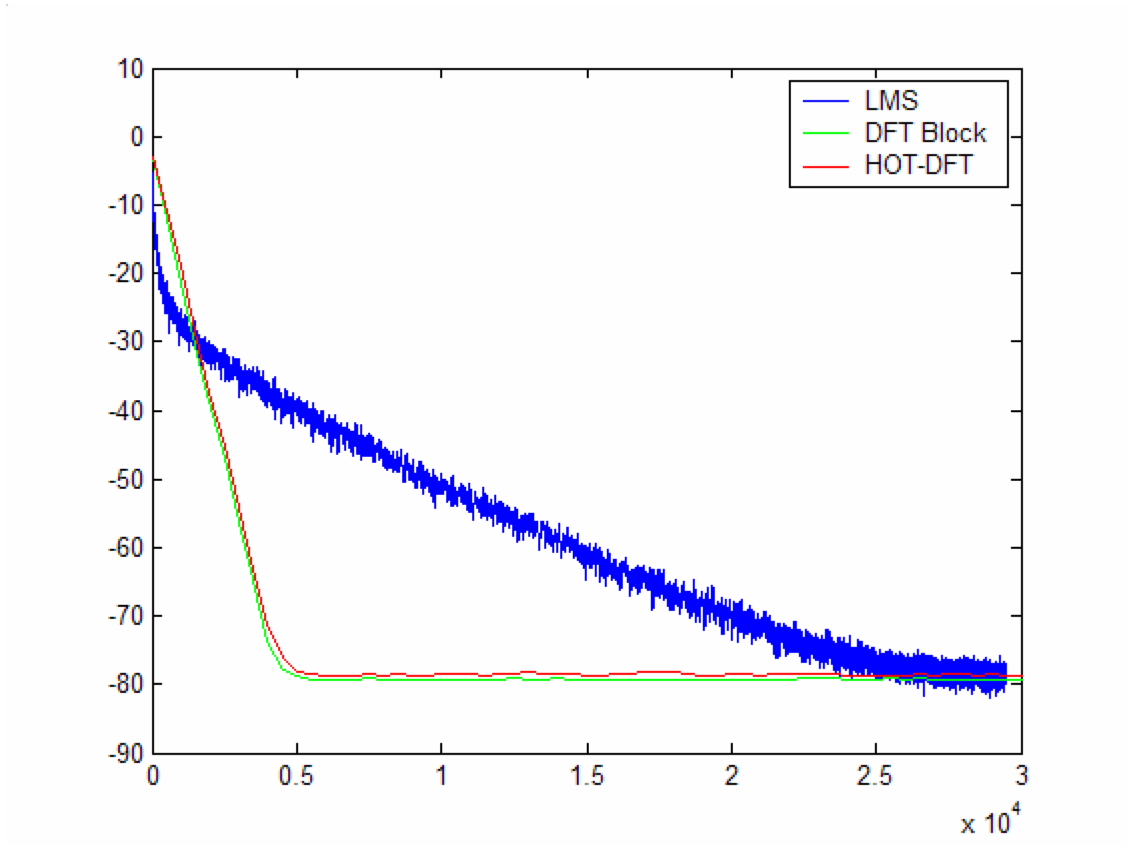


Figure 7.6: Learning curves for the LMS, HOT DFT block LMS, and DFT block LMS algorithms. $N = 50$ and $K = 10$.

Chapter 8

Conclusions and Recommendations for Future Research

8.1 Conclusions

Real time implementation of LMS adaptive filters are restricted due to the high computational cost for large filter lengths and slow convergence speed. The HOT is a recently developed discrete unitary transform that uses the orthonormal minimizers of the entropy-based Hirschman uncertainty measure. The fact that the HOT basis sequences have many zero-valued samples, along with their resemblance to the DFT basis sequences, makes the computationally efficient HOT an attractive alternative for the DFT to improve the performance of LMS adaptive filters. This dissertation introduces new transform domain LMS algorithms based on the HOT, the HOT LMS algorithm, the self-orthogonalizing HOT block adaptive filter, the HOT block LMS algorithm, and the HOT DFT LMS algorithm.

The performance of these algorithms was analyzed in detail. Since the convergence speed of the HOT LMS algorithm depends directly on the HOT autocorrelation matrix of the filter input, it was thoroughly investigated. The Hilbert-Schmidt norm of the difference between the input autocorrelation matrix and the HOT autocorrelation matrices approximation was calculated and used as measure for the diagonalizing power of the HOT. This measure does not vanish as the filter length approaches infinity. Therefore, the HOT does not diagonalize the autocorrelation matrix even for large filter lengths.

By looking directly at the input autocorrelation in the HOT domain, it was found that the autocorrelation matrix in the HOT domain is asymptotically block diagonal and that the HOT LMS algorithm adjust the learning rate of each block to improve the convergence speed of the adaptive filter as compared to the standard LMS algorithm. This expectation was confirmed by simulations. The total number of multiplications required for the HOT LMS adaptive filter is $2K^2 + 3K + 1$, which is less than half the multiplications required for the DFT LMS algorithm (which requires $5N + 1$ multiplications) and "almost" the same as number of multiplications required for the LMS algorithm.

It was found that each block in the HOT autocorrelation matrix is symmetric and Toeplitz. Each block of the HOT autocorrelation matrix can be efficiently diagonalized using a Levinson recursion. With this modification, the HOT LMS algorithm becomes the self-orthogonalizing HOT LMS algorithm. Therefore, the self-orthogonalizing HOT LMS algorithm implements the HOT to block-wise diagonalize the autocorrelation matrix and then uses K self-orthogonalizing LMS sections to diagonalize the blocks of the HOT autocorrelation matrix. Simulations showed a substantial improvement of the self-orthogonalizing HOT LMS filter compared to the basic

HOT LMS filter. Although the performance of the self-orthogonalizing HOT LMS filter is greatly improved compared to the HOT LMS algorithm, the self-orthogonalizing HOT LMS filter is not computationally efficient. To reduce the computational complexity of the self-orthogonalzing HOT LMS algorithm, the computations are done block-by-block. With this modification, the self-orthogonalizing HOT LMS algorithm becomes the self-orthogonalizing HOT block adaptive filter (SOHBAF). The computational complexity of the SOHBAF depends on the block length. If the block length is chosen to be K^2, then the computational complexity per sample is $\mathcal{O}(K \log_2 K)$. Therefore, SOHBAF is computationally more efficient than SOBAF, which requires $\mathcal{O}(K^2)$ operations per sample. The SOHBAF represents an LMS algorithm with computational complexity, convergence speed, and sensitivity to the input statistics lying between the SOBAF and DFT block LMS algorithms.

The "HOT convolution," a relation between the HOT of two signals and their circular convolution was derived. The result was used to develop a fast block LMS adaptive filter called the HOT block LMS adaptive filter. This filter requires slightly less than half of the multiplications that are required for the DFT block LMS adaptive filter. The reduction in the computational complexity of the HOT block LMS comes from using only one polyphase component of the filter error used to update the filter weights. Convergence analysis of the HOT block LMS algorithm showed that the average time constant is the same as that of the DFT block LMS algorithm and that the misadjustment is K times greater than that of the DFT block LMS algorithm.

The HOT block algorithm assumes that the filter and block lengths are the same. Based on a fast HOT convolution developed by Matusiak and DeBrunner [15], another block HOT LMS algorithm was developed. This new block LMS algorithm assumes that the filter length is much smaller than the block length. This algorithm is very similar to the block DFT LMS algorithm and reduces the computational complexity by about 30% when the filter length is much smaller than the block length.

The adaptive LMS algorithms presented in this dissertation and their computational complexities are summarized in Table 8.1. The third column lists the values of K at which the corresponding algorithm becomes more efficient than the one above.

8.2 Recommendations for Future Research

Throughout the analysis and development of the HOT LMS and HOT block LMS algorithms, it was obvious that the HOT transforms the filter input random signal into a midpoint between the time and DFT domains, i.e., the K^2-point HOT divides a K^2-point random vector into K K-point sections, where each section is transformed to the DFT domain. This property of the HOT allowed the HOT adaptive filers to use efficient time domain algorithms such as fast convolution and the Levinson recursion and the DFT. Recall that the HOT basis functions are minimizers of the Hirschman uncertainty principle with $H_{1/2} = \frac{1}{2}\log K^2$ for all $0 \leq p \leq 1$. Hence they are the most compact bases in the phase plane. There should be a connection between this fact and the structure of random vectors in the HOT domain. This connection is worthy of being investigated explicitly and analytically. With this investigation, the performance of the HOT LMS algorithms could be improved or optimized for specific applications.

The proposed transform domain LMS algorithms in this dissertation can be implemented to improve the performance of many communication and control systems where the adaptive filter is required to be large. These systems can be simulated

Table 8.1: Adaptive LMS algorithms and their computational complexities.

HOT LMS Filter and SOHBAF

$N = K^2$ (filter length)

$L = N$ (block length)

Algorithm	Number of Multiplies	K	Comparison to the DFT block LMS
LMS	$2K^2 + 1$		
DFT LMS	$5K^2 + 1$		
HOT LMS	$2K^2 + 3K + 1$	all K	Converges faster than the LMS. Requires less than half the multiplies.
DFT block-LMS	$20K^2 \log_2(2K)$ $+4K^2$		
SOBAF	$2K^4 + 4K^2 +$ $14L \log_2(2L) + 6L$		
SOHBAF	$2K^3 + 4K^2 + 5.4KL$ $+12.2KL \log_2(2L)$	85	Converges faster. Requires more multiplies.

HOT Block LMS Filter

$N = K^2/2$ (filter length)

$L = K^2$ (block length)

Algorithm	Number of Multiplies	K	Comparison to the DFT block LMS
LMS	$K^2 + 1$		
DFT Block-LMS	$10K^2 \log_2 K + 2K^2$	all K	
HOT Block-LMS	$4K^2 \log_2 K$ $+2K \log_2 K + 2K^2$	all K	Requires less than half the multiplies. K times higher misadjustment.

HOT DFT LMS Filter

N (filter length)

$L = KN$ (block length)

Algorithm	Number of Multiplies	K	Comparison to the DFT block LMS
LMS	$2N + 1$		
DFT block-LMS	$10KN \log_2(2NK)$ $+4NK$		
HOT DFT-block LMS	$(4K + 1)2N \log_2(2N)$ $+4NK$	all K	Converges at the same rate.

with the HOT adaptive filter presented in this dissertation to explicitly verify the performance improvement.

Bibliography

[1] B. Widrow and M. E. Hoff, Jr. “Adaptive switching circuit,” *IRE WESCON Conv. Rec.*, pt. 4 pp. 96-104, 1980.

[2] E. R. Ferrara, “Fast implementation of LMS adaptive filters,” *IEEE Trans. ASSP*, vol. ASSP-28, NO. 4, Aug 1980.

[3] G. Clark, S. Mitra, and S Parker, “Block implementation of adaptive digital filters,” *IEEE Trans. ASSP*, pp. 744-752,Jun 1981.

[4] Simon Haykin, *Adaptive Filter Theory.* Prentice Hall information and system sciences series, Fourth edition, 2002.

[5] B. Widrow, “Stationary and nonstationary learning characteristics of the LMS adaptive filter,” *Proc. IEEE*, vol. 64, pp. 1151-1162, Aug. 1976.

[6] S. Narayan, A. Peterson, M. Narasimha, “Frequency domain least-square algorithm,” *Proc. IEEE*, vol. 69, NO. 1, pp. 124-126, Jan 1981.

[7] S. Narayan, A. Peterson, and M. Narasimha, “Transform domain LMS algorithm,” *IEEE Trans. ASSP*, pp. 609-615, Jun 1983.

[8] J. Lee and C. K. Un, “Performance of transform-domain LMS adaptive digital filters,” *IEEE Trans. ASSP*, pp. 499-510, June 1986.

[9] P. K. Bondyopadhyay, “Application of running Hartley transform in adaptive digital Filtering,” *Proc. IEEE*, vol. 76, No. 10, pp. 1370-1372, Oct. 1988.

[10] R. N. Bracwell, “The discrete Hartley transform,” *J. Opt. Soc. Amr.*, vol. 73, pp. 1832-1835, Dec. 1983.

[11] D. F. Marshall, W. K. Jenkins, and J. J. Murphy, “The use of orthogonal transforms for improving performance of adaptive filters,” *IEEE Trans. Cir. and Sys.*, pp. 474-484, Apr 1989.

[12] F. Beaufays, “Transform-domain adaptive filters: An Analytical Approach,” *IEEE Trans. ASSP*, vol. 43, NO. 2, pp. 422-431, Feb. 1995.

[13] H T. Przebinda, V. DeBrunner, and M. Özaydin, “The optimal transform for the discrete Hirschman uncertainty principle,” *IEEE Trans. Infor. Theory*, pp. 2086-2090, Jul 2001.

[14] V. DeBrunner, M. Özaydin, and T. Przebinda, “Resolution in time-frequency,” *IEEE Trans. ASSP*, pp. 783-788, Mar 1999.

[15] V. DeBrunner and E. Matusiak, “An algorithm to reduce the complexity required to convolve finite length sequences using the Hirschman optimal transform (HOT),” *ICASSP 2003*, Hong Kong, China, pp. II-577-580, Apr 2003.

[16] E. Jacobsen and R. Lyons, “The sliding DFT,” *IEEE Signal Processing Mag.*, pp. 74-80, Mar 2003.

[17] J. Pearl, “On coding and filtering stationary signals by discrete Fourier transforms,” *IEEE Trans. Infor. Theory*, pp. 229-232, Mar 1973.

[18] J. Benesty and P. Duhamel, “A fast exact least mean square adaptive algorithm,” *IEEE Trans. ASSP*, pp. 2904-2920, Dec 1992.

[19] Sommen, P.; van Gerwen, P.; Kotmans, H.; Janssen, A.; “Convergence analysis of a frequency-domain adaptive filter with exponential power averaging and generalized window function,” Circuits and Systems, *IEEE Transactions on*, Volume 34, Issue 7, Page(s):788-798, Jul 1987.

[20] J. J. Shynk, “Frequency-domain and multirate adaptive filtering,” *IEEE Signal Processing Mag.*, vol. 9, no. 1, pp. 14-37, Jan. 1992.

[21] I. I. Hirschman, “A note on entropy,” *Amer. J. Math.*, vol. 79, pp. 152-156, 1957.

[22] B. Farhang-Boroujeny and Kheong Sann Chan, “Analysis of the frequency-domain block LMS algorithm,” *IEEE Trans. ASSP*, pp. 2332, Aug. 2000.

[23] E. Jacobsen and R. Lyons, “The sliding DFT,” *IEEE SP Mag.*, pp. 74-80, Mar 2003.

[24] D. Mansour and A. H. Gray, “Unconstrained frequency-domain adaptive filter,” *IEEE Trans. ASSP*, pp. 726-734, Oct 1982.

[25] Jae Chon Lee; Chong Kwan Un; “Performance analysis of frequency-domain block LMS adaptive digital filters,” *Circuits and Systems, IEEE Transactions on*, Volume 36, Issue 2, Page(s):173-189, Feb. 1989.

[26] G. Panda, B. Mulgrew, C. F. N. Cowan, P. M. Grant, “A self-orthogonalizing efficient block adaptive filter,” *IEEE Trans. ASSP*, VOL. ASSP-34, NO. 6, pp. 1573-1582, Dec. 1986.

[27] R. Kumar, "A fast algorithm for solving Toeplitz system of equations," *IEEE Trans. ASSP*, VOL. ASSP-33,pp. 254-267, Feb 1985.

[28] G. Panda, A. M. Alvarez, P. M. Grant, C. F. N. Cowan, "A transform domain circular convolution algorithm for adaptive filtering," *IEEE Trans. ASSP*, VOL. ASSP-35, NO. 8,pp. 1217-1220, Aug. 1987.

[29] R. C. Agarwal and J. W. Cooley, "New algorithms for digital convolution," *IEEE Trans. ASSP*, VOL. ASSP-25, pp. 392-410, Oct. 1977.

[30] T. W. Wong and C. B. Kwong, "Adaptive filtering using Hartley transform and overlap-save method," *IEEE Trans. ASSP*, VOL. 39. NO. 7, July 1991.

[31] L. R. Rabiner and B. Gold, Theory and applications of Digital Signal Processing. Englewood Cliffs, NJ: Prentice Hall, 1975.

[32] B. Farhang-Boroujeny and S. Gazor, "Generalized sliding FFT and its applications to implementation of block LMS adaptive filters," *IEEE Trans. ASSP*,VOL. 42, NO. 3, pp. 532-538 March 1994.

[33] M. Narasimha,"Block adaptive filter with time-domain update using three transforms," *IEEE Signal Processing Letters*, VOL. 14, NO. 1, pp. 51-53, Jan 2007.

[34] A. O. Ogunfunmi and A. M. Peterson, "On the implementation of the frequency-domain LMS adaptive filter," *IEEE Trans. ASSP*, VOL. 39, NO. 5, pp. 532-538, May 1992.

[35] J. Chao, and H. Perez, S. Tsujii, "A fast adaptive filter algorithm using eigenvalue reciprocals," *IEEE Trans. ASSP*, VOL. 38, NO. 8, pp. 1343-1352, Aug. 1990.

[36] S. Hosur and A. Tewfik "Wavelet transform domain adaptive FIR filtering," *IEEE Trans. ASSP*,VOL. 45, NO. 3, pp. 617-630, March 1997.

[37] R. C. Bilcu, P. Kuosmanen, K. Egiazarian. "A transformd domain LMS adaptive filter with variable step-size," *IEEE Signal Processing Letters*, vol. 9, NO. 2, pp. 51-53, Feb. 2002.

[38] S. Hosur and A. Tewfik "Wavelet transform domain adaptive FIR filtering," *IEEE Trans. ASSP*, vol. 45, NO. 3, pp. 617-630, March 1997.

[39] V. DeBrunner, M. Özaydin, and T. Przebinda, "Analysis in a finite time-frequency plane," *IEEE Trans. ASSP*, pp. 1831-1832, June 2000.

[40] Chang R. W. "A new equalizer structure for fast start-up digital communications," *Bell Syst. Tech. J.*, vol.50, pp. 169-2014, 1971.

[41] Monson H. Hayes, *Statistical Digital Signal Processing and Modeling.* Wiley Fourth edition, 1996.

[42] K. Sam Shanmugan and A. M. Breipohl, *Random Signals: Detection, Estimation and Data Analysis.* John Wiley and Sons, Fourth edition, 1988.

[43] G. C. Goodwin and R. L. Payne, *Dynamic System Identification: Experiment Design and Data Analysis.* New York: Academic, 1977.

[44] P. Tay, J.P. Havlicek, and V. DeBrunner, "A novel translation and modulation invariant discrete-discrete uncertainty measure," in Proc. IEEE Int'l. Conf. Acoust., Speech, Signal Proc., Orlando, FL, May 13-17, 2002, vol. 2, pp. 1461-1464.

[45] S. Mitra, *Digital Signal Processing.* Mc Graw Hill, Second edition, 2000.

[46] V. DeBrunner, J. Havlicek, T. Przebinda, and M. Özaydin, "Entropy-based uncertainty measures for $L^2(R)^n$, $\ell^2(Z)$, and $\ell^2(Z/NZ)$ with a Hirschman optimal transform for $\ell^2(Z/NZ)$," *IEEE Trans. ASSP*, pp. 2690-2696, August 2005.