Hirschman Optimal Transform (HOT) DFT Block LMS Algorithm

Osama Alkhouli[1], Victor DeBrunner[2] and Joseph Havlicek[3]

[1]*Caterpillar Inc.,*
[2]*Florida State University,*
[3]*The University of Oklahoma*
USA

1. Introduction

Least mean square (LMS) adaptive filters, as investigated by Widrow and Hoff in 1960 (Widrow & Hoff, 1980), 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 (Farhang-Boroujeny, 1999; Haykin, 2002). The computational burden and slow convergence speed of the LMS algorithm can render its real time implementation infeasible. To reduce the computational cost of the LMS filter, Ferrara proposed a frequency domain implementation of the LMS algorithm (Ferrara, 1980). 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 DFT block LMS algorithm. The computational reduction in the DFT block LMS algorithm comes from using the fast DFT convolution to calculate the convolution between the filer input and weights and the gradient estimate.
The Hirschman optimal transform (HOT) is a recently developed discrete unitary transform (DeBrunner et al., 1999; Przebinda et.al, 2001) that uses the orthonormal minimizers of the entropy-based Hirschman uncertainty measure (Przebinda et.al, 2001). 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 (DeBrunner et al., 1999). 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 (DeBrunner et al., 2005). The HOT is similar to the DFT. For example, the 3^2-point HOT matrix is explicitly given below.

$$\begin{bmatrix} 1&0&0&1&0&0&1&0&0\\ 0&1&0&0&1&0&0&1&0\\ 0&0&1&0&0&1&0&0&1\\ 1&0&0&e^{-j2\pi/3}&0&0&e^{-j4\pi/3}&0&0\\ 0&1&0&0&e^{-j2\pi/3}&0&0&e^{-j4\pi/3}&0\\ 0&0&1&0&0&e^{-j2\pi/3}&0&0&e^{-j4\pi/3}\\ 1&0&0&e^{-j4\pi/3}&0&0&e^{-j8\pi/3}&0&0\\ 0&1&0&0&e^{-j4\pi/3}&0&0&e^{-j8\pi/3}&0\\ 0&0&1&0&0&e^{-j4\pi/3}&0&0&e^{-j8\pi/3} \end{bmatrix} \tag{1}$$

In general, the NK-point HOT basis are generated from the N-point DFT basis as follows. Each of the DFT basis functions are interpolated by K and then circularly shifted to produce the complete set of orthogonal basis signals that define the HOT. The computational saving of any fast block LMS algorithm depends on how efficiently each of the two convolutions involved in the LMS algorithm are calculated (Clark et al., 1980; Ferrara, 1980). The DFT block LMS algorithm is most efficient when the block and filter sizes are equal. Recently, we developed a fast convolution based on the HOT (DeBrunner & Matusiak, 2003). The HOT convolution is more efficient than the DFT convolution when the disparity in the lengths of the sequences being convolved is large. In this chapter we introduce a new fast block LMS algorithm based on the HOT. This algorithm is called the HOT DFT block LMS algorithm. It is very similar to the DFT block LMS algorithm and reduces its computational complexity by about 30% when the filter length is much smaller than the block length. In the HOT DFT block LMS algorithm, the fast HOT convolution is used to calculate the filter output and update the weights.

Recently, the HOT transform was used to develop the HOT LMS algorithm (Alkhouli et al., 2005; Alkhouli & DeBrunner, 2007), which is a transform domain LMS algorithm, and the HOT block LMS algorithm (Alkhouli & DeBrunner, 2007), which is a fast block LMS algorithm. The HOT DFT block LMS algorithm presented here is different from the HOT block LMS algorithm presented in (Alkhouli & DeBrunner, 2007). The HOT DFT block LMS algorithm developed in this chapter uses the fast HOT convolution (DeBrunner & Matusiak, 2003). The main idea behind the HOT convolution is to partition the longer sequence into sections of the same length as the shorter sequence and then convolve each section with the shorter sequence efficiently using the fast DFT convolution. The relevance of the HOT will become apparent when the all of the (sub)convolutions are put together concisely in a matrix form as will be shown later in this chapter.

The following notations are used throughout this chapter. 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 N-point DFT matrix is denoted by $\mathbf{F}_N$ or simply by $\mathbf{F}$. 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. Diag $[\mathbf{u}]$ or $\mathcal{U}$ denotes the diagonal matrix whose diagonal elements are the elements of the vector $\mathbf{u}$.
In section 2, The explicit relation between the DFT and HOT is developed. The HOT convolution is presented in Section 3. In Section 4, the HOT DFT block LMS algorithm is developed. Its computational cost is analyzed in Section 5. Section 6 contains the convergence analysis and Section 7 contains its misadjustment. Simulations are provided in Section 8 before the conclusions in Section 9

2. The relation between the DFT and HOT

In this section, an explicit relation between the DFT and HOT is derived. Let $\mathbf{u}$ be a vector of length NK. The K-band polyphase decomposition of $\mathbf{u}$ decomposes $\mathbf{u}$ into a set of K polyphase components. The k^{th} polyphase componenet of $\mathbf{u}$ is denoted by $\tilde{\mathbf{u}}_k$ and is given by

$$\tilde{\mathbf{u}}_k = \begin{bmatrix} u_k \\ u_{k+K} \\ u_{k+2K} \\ \vdots \\ u_{k+(N-1)K} \end{bmatrix}. \tag{2}$$

The vector that combines the polyphase components of $\mathbf{u}$ is denoted by $\tilde{\mathbf{u}}$, i.e.,

$$\tilde{\mathbf{u}} = \begin{bmatrix} \tilde{\mathbf{u}}_0 \\ \tilde{\mathbf{u}}_1 \\ \tilde{\mathbf{u}}_2 \\ \vdots \\ \tilde{\mathbf{u}}_{K-1} \end{bmatrix}. \tag{3}$$

The square matrix that relates $\tilde{\mathbf{u}}$ and $\mathbf{u}$ is denoted by $\mathbf{P}$, i.e.,

$$\tilde{\mathbf{u}} = \mathbf{P}\mathbf{u}. \tag{4}$$

For example, $\mathbf{P}$ for the case of $N = 4$ and $K = 3$ is given by

$$\mathbf{P} = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&1&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&1&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&1&0&0 \\
0&1&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&1&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&1&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&1&0 \\
0&0&1&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&1&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&1&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&1
\end{bmatrix}. \tag{5}$$

Without loss of generality, we consider the special case of $N = 4$ and $K = 3$ to find an explicit relation between the DFT and HOT. The 4×3-point HOT is given by

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & e^{-j2\pi/4} & 0 & 0 & e^{-j4\pi/4} & 0 & 0 & e^{-j6\pi/4} & 0 & 0 \\ 0 & 1 & 0 & 0 & e^{-j2\pi/4} & 0 & 0 & e^{-j4\pi/4} & 0 & 0 & e^{-j6\pi/4} & 0 \\ 0 & 0 & 1 & 0 & 0 & e^{-j2\pi/4} & 0 & 0 & e^{-j4\pi/4} & 0 & 0 & e^{-j6\pi/4} \\ 1 & 0 & 0 & e^{-j4\pi/4} & 0 & 0 & e^{-j8\pi/4} & 0 & 0 & e^{-j12\pi/4} & 0 & 0 \\ 0 & 1 & 0 & 0 & e^{-j4\pi/4} & 0 & 0 & e^{-j8\pi/4} & 0 & 0 & e^{-12\pi/4} & 0 \\ 0 & 0 & 1 & 0 & 0 & e^{-j4\pi/4} & 0 & 0 & e^{-j8\pi/4} & 0 & 0 & e^{-j12\pi/4} \\ 1 & 0 & 0 & e^{-j6\pi/4} & 0 & 0 & e^{-j12\pi/4} & 0 & 0 & e^{-j18\pi/4} & 0 & 0 \\ 0 & 1 & 0 & 0 & e^{-j6\pi/4} & 0 & 0 & e^{-j12\pi/4} & 0 & 0 & e^{-j18\pi/4} & 0 \\ 0 & 0 & 1 & 0 & 0 & e^{-j6\pi/4} & 0 & 0 & e^{-j12\pi/4} & 0 & 0 & e^{-j18\pi/4} \end{bmatrix}. \tag{6}$$

Equation (6) shows that the HOT takes the 4-point DFTs of the 3 polyphase components and then reverses the polyphase decomposition. Therefore, the relation between the DFT and HOT can be written as

$$\mathbf{H} = \mathbf{P} \begin{bmatrix} \mathbf{F}_4 & \mathbf{0}_{4\times4} & \mathbf{0}_{4\times4} \\ \mathbf{0}_{4\times4} & \mathbf{F}_4 & \mathbf{0}_{4\times4} \\ \mathbf{0}_{4\times4} & \mathbf{0}_{4\times4} & \mathbf{F}_4 \end{bmatrix} \mathbf{P}. \tag{7}$$

Also, it can be easily shown that

$$\mathbf{H}^{-1} = \mathbf{P} \begin{bmatrix} \mathbf{F}_4^{-1} & \mathbf{0}_{4\times4} & \mathbf{0}_{4\times4} \\ \mathbf{0}_{4\times4} & \mathbf{F}_4^{-1} & \mathbf{0}_{4\times4} \\ \mathbf{0}_{4\times4} & \mathbf{0}_{4\times4} & \mathbf{F}_4^{-1} \end{bmatrix} \mathbf{P}. \tag{8}$$

3. The HOT convolution

In this section we present a computationally efficient convolution algorithm based on the HOT. Let $h(n)$ be a signal of length N and $u(n)$ be a signal of length KN. The linear convolution between $h(n)$ and $u(n)$ is given by

$$y(n) = \sum_{l=0}^{N-1} h(l)u(n-l). \tag{9}$$

According to the overlap-save method (Mitra, 2000), $y(n)$ for $0 \leq n \leq KN$, where K is an integer, can be calculated by dividing $u(n)$ into K overlapping sections of length $2N$ and $h(n)$ is post appended with N zeros as shown in Figure 1 for $K = 3$. The linear convolution in (9) can be calculated from the circular convolutions between and $h(n)$ and the sections of $u(n)$. Let $u_k(n)$ be the k^{th} section of $u(n)$. Denote the $2N$-point circular convolution between $u_k(n)$ and $h(n)$ by $c_k(n) = u_k(n) \star h(n)$.

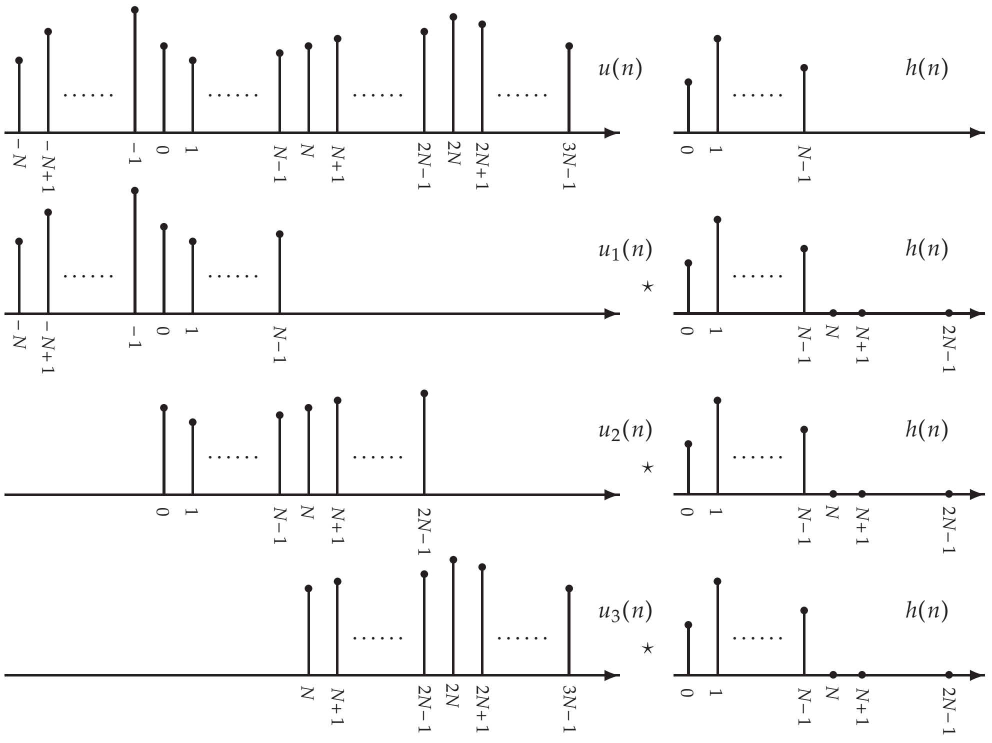

Fig. 1. Illustration of how $u(n)$ is divided into 3 overlapping sections. Each section is convolved with the the appended $h(n)$.

The circular convolution $c_k(n)$ can be calculated using the $2N$-point DFT as follows. First, form the vectors

$$\mathbf{h} = \begin{bmatrix} h(0) \\ h(1) \\ \vdots \\ h(N-1) \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{10}$$

$$\mathbf{c}_k = \begin{bmatrix} c_k(0) \\ c_k(1) \\ \vdots \\ c_k(2N-1) \end{bmatrix}, \tag{11}$$

Then the $2N$-point DFT of $\mathbf{c}_k$ is given by

$$\mathbf{F}_{2N}\mathbf{c}_k = \mathbf{F}_{2N}\mathbf{u}_k \cdot \mathbf{F}_{2N}\mathbf{h}, \tag{12}$$

where $\mathbf{u}_k$ is the vector that contains the elements of $u_k(n)$. "$\cdot$" indicates pointwise matrix multiplication and, throughout this chapter, pointwise matrix multiplication takes a lower precedence than conventional matrix multiplication. Combining all of the circular convolutions into one matrix equation, we should have

$$\begin{bmatrix} \mathbf{F}_{2N}\mathbf{c}_0 \\ \mathbf{F}_{2N}\mathbf{c}_1 \\ \vdots \\ \mathbf{F}_{2N}\mathbf{c}_{K-1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{2N}\mathbf{h} \\ \mathbf{F}_{2N}\mathbf{h} \\ \vdots \\ \mathbf{F}_{2N}\mathbf{h} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{F}_{2N}\mathbf{u}_0 \\ \mathbf{F}_{2N}\mathbf{u}_1 \\ \vdots \\ \mathbf{F}_{2N}\mathbf{u}_{K-1} \end{bmatrix}. \tag{13}$$

Using equation (7), equation (13) can be written as

$$\mathbf{H}\tilde{\mathbf{c}} = \mathbf{H}\tilde{\mathbf{u}} \cdot \mathbf{H}\tilde{\mathbf{h}}_r, \tag{14}$$

where

$$\mathbf{h}_r = \begin{bmatrix} \mathbf{h} \\ \mathbf{h} \\ \vdots \\ \mathbf{h} \end{bmatrix}, \tag{15}$$

and

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{k-1} \end{bmatrix}. \tag{16}$$

Therefore, The vector of the circular convolutions is given by

$$\mathbf{c} = \mathbf{P}\mathbf{H}^{-1}\left(\mathbf{H}\tilde{\mathbf{u}} \cdot \mathbf{H}\tilde{\mathbf{h}}_r\right). \tag{17}$$

According to the overlap-save method, only the second half of $\mathbf{c}_k$ corresponds to the k^{th} section of the linear convolution. Denote the k^{th} section of the linear convolution by $\mathbf{y}_k$ and the vector that contains the elements of $y(n)$ by $\mathbf{y}$. Then $\mathbf{y}_k$ can be written as

$$\mathbf{y}_k = \begin{bmatrix} \mathbf{0}_{N\times N} & \mathbf{I}_{N\times N} \end{bmatrix} \mathbf{c}_k, \tag{18}$$

and $\mathbf{y}$ as

$$\mathbf{y} = \mathbf{G}\mathbf{c}, \tag{19}$$

where

$$\mathbf{G} = \begin{bmatrix} \mathbf{0}_{N\times N}\ \mathbf{I}_{N\times N} & \mathbf{0}_{2N\times 2N} & \cdots & \mathbf{0}_{2N\times 2N} \\ \mathbf{0}_{2N\times 2N} & \mathbf{0}_{N\times N}\ \mathbf{I}_{N\times N} & \cdots & \mathbf{0}_{2N\times 2N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{2N\times 2N} & \mathbf{0}_{2N\times 2N} & \cdots & \mathbf{0}_{N\times N}\ \mathbf{I}_{N\times N} \end{bmatrix}. \tag{20}$$

Finally, the linear convolution using the HOT is given by

$$\mathbf{y} = \mathbf{G}\mathbf{P}\mathbf{H}^{-1}\left(\mathbf{H}\tilde{\mathbf{u}} \cdot \mathbf{H}\tilde{\mathbf{h}}_r\right). \tag{21}$$

In summery, the convolution between $(K+1)N$-point input $u(n)$ and N-point impulse response $h(n)$ can be calculated efficiently using the HOT as follows:

1. Divide $u(n)$ into K overlapping sections and combine them into one vector to from $\mathbf{u}$.
2. Perform K-band polyphase decomposition of $\mathbf{u}$ to form $\tilde{\mathbf{u}}$.
3. Take the HOT of $\tilde{\mathbf{u}}$.
4. Post append $h(n)$ with N zeros and then stack the appended $h(n)$ K times into one vector to form $\mathbf{h}_r$.
5. Perform K-band polyphase decomposition of $\mathbf{h}_r$ to form $\tilde{\mathbf{h}}_r$.
6. Take the HOT of $\tilde{\mathbf{h}}_r$.
7. Point-wise multiply the vectors from steps 3 and 6.
8. Take the inverse HOT of the vector from step 7.
9. Perform K-band polyphase decomposition of the result from step 8.
10. Multiply the result of step 9 with $\mathbf{G}$.

4. 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 is 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 HOT convolution developed in the previous section can be used to calculate the first convolution. However, the second convolution is a convolution between two signals of the same length and the fast HOT convolution can not be used directly without modification.
Let N be the filer length and $L = NK$ be the block length, where N, L, and K are all integers. Let

$$\hat{\mathbf{w}}(k) = \begin{bmatrix} w_0(k) \\ w_1(k) \\ \vdots \\ w_{N-2}(k) \\ w_{N-1}(k) \end{bmatrix} \tag{22}$$

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

$$\hat{\mathbf{u}}(k) = \begin{bmatrix} u\,(kL-N) \\ \vdots \\ u\,(kL) \\ u\,(kL+1) \\ \vdots \\ u\,(kL+L-1) \end{bmatrix} \tag{23}$$

be the vector of input samples needed in the k^{th} block. To use the fast HOT convolution described in the previous section, $\hat{\mathbf{u}}(k)$ is divided is into K overlapping sections. Such sections can be formed by multiplying $\hat{\mathbf{u}}(k)$ with the following matrix:

$$\mathbf{J} = \begin{bmatrix} \mathbf{I}_{N\times N} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N\times N} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N\times N} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_{N\times N} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_{N\times N} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I}_{N\times N} \end{bmatrix}. \tag{24}$$

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

$$\mathbf{w}(k) = \begin{bmatrix} \hat{\mathbf{w}}(k) \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{25}$$

According the fast HOT convolution, see equation (21), the output of the adaptive filter in the k^{th} block

$$\mathbf{y}(k) = \begin{bmatrix} y(kL) \\ y(kL+1) \\ \vdots \\ y(kL+L-2) \\ y(kL+L-1) \end{bmatrix} \tag{26}$$

is given by

$$\mathbf{y}(k) = \mathbf{GPH}^{-1}\Big(\mathbf{HPw}_r(k) \cdot \mathbf{HPJ}\hat{\mathbf{u}}(k)\Big). \tag{27}$$

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

$$\hat{\mathbf{d}}(k) = \begin{bmatrix} d(kL) \\ d(kL+1) \\ \vdots \\ d(kL+L-2) \\ d(kL+L-1) \end{bmatrix} \tag{28}$$

and

$$\hat{\mathbf{e}}(k) = \begin{bmatrix} e(kL) \\ e(kL+1) \\ \vdots \\ e(kL+L-2) \\ e(kL+L-1) \end{bmatrix}, \tag{29}$$

respectively, where

$$e(n) = d(n) - y(n). \tag{30}$$

The filter update equation is given by

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) + \frac{\mu}{L}\sum_{i=0}^{L-1}\begin{bmatrix} u\left(kL+i\right) \\ u\left(kL+i-1\right) \\ \vdots \\ u\left(kL+i-N+2\right) \\ u\left(kL+i-N+1\right) \end{bmatrix} e(kL+i). \tag{31}$$

The sum in equation (31) can be efficiently calculated using the $(L+N)$-point DFTs of the error vector $e(n)$ and input vector $u(n)$. However, the $(L+N)$-point DFT of $u(n)$ is not available and only the $2N$-point DFTs of the K sections of $\hat{\mathbf{u}}(k)$ are available. Therefore, the sum in equation (31) should be divided into K sections as follows:

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

$$\sum_{l=0}^{K-1}\sum_{j=0}^{N-1}\begin{bmatrix} u\left(kL+lN+j\right) \\ u\left(kL+lN+j-1\right) \\ \vdots \\ u\left(kL+lN+j-N+2\right) \\ u\left(kL+lN+j-N+1\right) \end{bmatrix} e(kL+lK+j). \tag{32}$$

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

$$\mathbf{u}_l(k) = \begin{bmatrix} u(kL+lN-N) \\ \vdots \\ u(kL+lN+N-2) \\ u(kL+lN+N-1) \end{bmatrix}, \tag{33}$$

$$\mathbf{e}_l(k) = \begin{bmatrix} \mathbf{0}_{N\times 1} \\ e(kL+lN) \\ \vdots \\ e(kL+lN+N-2) \\ e(kL+lN+N-1) \end{bmatrix}. \tag{34}$$

Then the sum over j is just the first N elements of the circular convolution of $\mathbf{e}_l(k)$ and circularly shifted $\mathbf{u}_l(k)$ and it can be computed using the DFT as shown below:

$$\sum_{j=0}^{N-1}\mathbf{u}_l(k)e(kL+lK+j) = \mathbf{U}_N\left(\mathbf{u}_{lF}^{*}(k)\cdot\mathbf{e}_{lF}(k)\right), \tag{35}$$

where

$$\mathbf{U}_N = \begin{bmatrix} \mathbf{I}_{N\times N} & \mathbf{0}_{N\times N} \\ \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} \end{bmatrix}, \tag{36}$$

$$\mathbf{u}_{lF}(k) = \mathbf{F}_{2N}\mathbf{u}_l(k), \tag{37}$$

and

$$\mathbf{e}_{lF}(k) = \mathbf{F}_{2N}\mathbf{e}_l(k). \tag{38}$$

Therefore, the filter update equation for the HOT DFT block LMS algorithm can be written as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu}{L}\sum_{l=0}^{K-1}\mathbf{U}_N\mathbf{F}^{-1}\Big(\mathbf{u}_{lF}^{*}(k)\cdot\mathbf{e}_{lF}(k)\Big). \tag{39}$$

Next, we express the sum in equation (39) in terms of the HOT. Form the vectors

$$\mathbf{u}(k) = \begin{bmatrix} \mathbf{u}_0(k) \\ \mathbf{u}_1(k) \\ \vdots \\ \mathbf{u}_{K-1}(k) \end{bmatrix}, \tag{40}$$

$$\mathbf{e}(k) = \begin{bmatrix} \mathbf{e}_0(k) \\ \mathbf{e}_1(k) \\ \vdots \\ \mathbf{e}_{K-1}(k) \end{bmatrix}. \tag{41}$$

Then using equation (7), the filter update equation can be written as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu}{L}\mathbf{SPH}^{-1}\Big(\mathbf{H}^{*}\tilde{\mathbf{u}}(k)\cdot\mathbf{H}\tilde{\mathbf{e}}(k)\Big), \tag{42}$$

where the matrix $\mathbf{S}$ is given by

$$\mathbf{S} = \begin{bmatrix} \mathbf{1}_{K\times 1} & \mathbf{0}_{K\times 1} & \cdots & \mathbf{0}_{K\times 1} & \\ \mathbf{0}_{K\times 1} & \mathbf{1}_{K\times 1} & \cdots & \mathbf{0}_{K\times 1} & \\ \vdots & \vdots & \ddots & \vdots & \mathbf{0}_{N\times KN} \\ \mathbf{0}_{K\times 0} & \mathbf{0}_{K\times 1} & \cdots & \mathbf{1}_{K\times 1} & \\ & \mathbf{0}_{N\times KN} & & & \mathbf{0}_{N\times KN} \end{bmatrix}. \tag{43}$$

Figure 2 shows the flow block diagram of the HOT DFT block LMS adaptive filter.

5. 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.

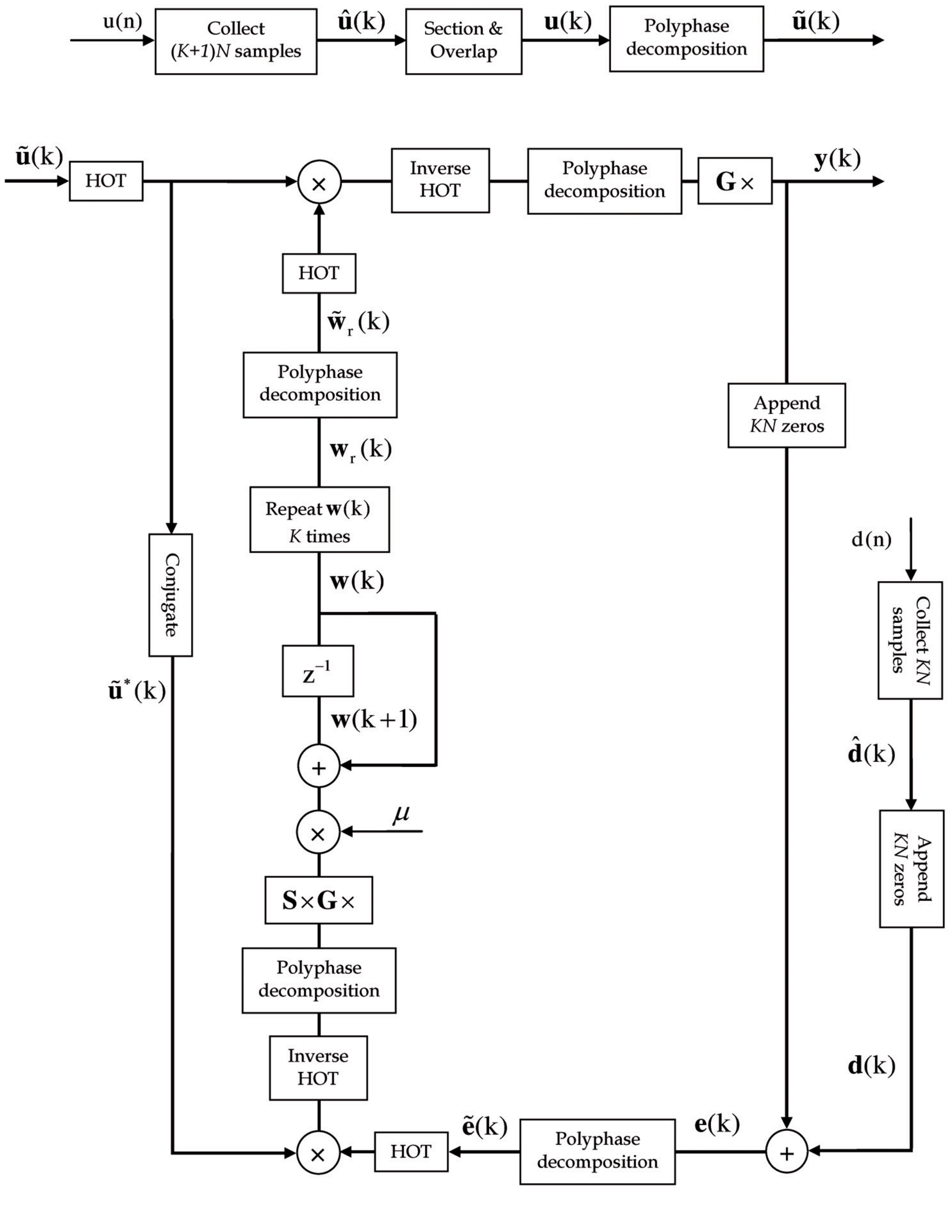

Fig. 2. The flow block diagram of the HOT DFT block LMS adaptive filter.

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 3 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.

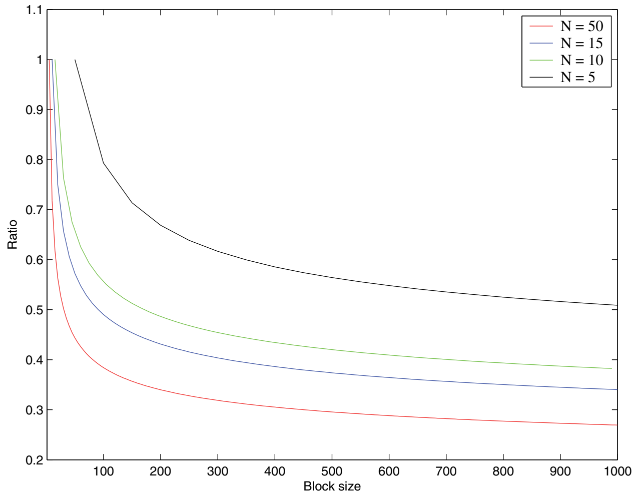

Fig. 3. Ratio between the number of multiplications required for the HOT DFT and the DFT block LMS algorithms.

6. 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

$$\mathbf{w}_F(k+1) = \mathbf{w}_F(k) + \frac{\mu}{L}\sum_{l=0}^{K-1} \mathbf{F}\mathbf{U}_N\mathbf{F}^{-1}\Big(\mathbf{u}_{lF}^*(k)\cdot\mathbf{e}_{lF}(k)\Big). \tag{44}$$

Let the desired signal be generated using the linear regression model

$$d(n) = w^o(n) * u(n) + e^o(n), \tag{45}$$

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

$$\hat{\mathbf{d}}_l(k) = \left[\,\mathbf{0}_{N\times N}\ \mathbf{I}_{N\times N}\,\right]\mathbf{F}^{-1}\Big(\mathbf{w}_F^o(k)\cdot\mathbf{u}_{lF}(k)\Big) + \hat{\mathbf{e}}_l^o(k), \tag{46}$$

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

$$\mathbf{e}_l(k) = \mathbf{L}_N\mathbf{F}^{-1}\Big(\epsilon_F(k)\cdot\mathbf{u}_{lF}(k)\Big) + \mathbf{e}_l^o(k), \tag{47}$$

where

$$\mathbf{L}_N = \begin{bmatrix} \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} \\ \mathbf{0}_{N\times N} & \mathbf{I}_{N\times N} \end{bmatrix}, \tag{48}$$

and $\boldsymbol{\epsilon}_F(k) = \mathbf{w}_F^o - \mathbf{w}_F(k)$. Using equation (44), the error in the estimation of the adaptive filter weight vector $\boldsymbol{\epsilon}_F(k)$ is updated according to

$$\boldsymbol{\epsilon}_F(k+1) = \boldsymbol{\epsilon}_F(k) - \frac{\mu}{L}\sum_{l=0}^{K-1} \mathbf{U}_{N,F}\Big(\mathbf{u}_{lF}^*(k)\cdot \mathbf{e}_{lF}(k)\Big), \tag{49}$$

where

$$\mathbf{U}_{N,F} = \mathbf{F}\begin{bmatrix} \mathbf{I}_{N\times N} & \mathbf{0}_{N\times N} \\ \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} \end{bmatrix}\mathbf{F}^{-1}. \tag{50}$$

Taking the DFT of equation (47), we have that

$$\mathbf{e}_{lF}(k) = \mathbf{L}_{N,F}\Big(\boldsymbol{\epsilon}_F(k)\cdot \mathbf{u}_{lF}(k)\Big) + \mathbf{e}_{lF}^o(k), \tag{51}$$

where

$$\mathbf{L}_{N,F} = \mathbf{F}\begin{bmatrix} \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} \\ \mathbf{0}_{N\times N} & \mathbf{I}_{N\times N} \end{bmatrix}\mathbf{F}^{-1}. \tag{52}$$

Using equation (51), we can write

$$\mathbf{u}_{lF}^*(k)\cdot \mathbf{e}_{lF}(k) = \mathcal{U}_{lF}^*(k)\Big(\mathbf{L}_{N,F}\mathcal{U}_{lF}(k)\boldsymbol{\epsilon}_F(k) + \mathbf{e}_{lF}^o(k)\Big). \tag{53}$$

Using

$$\mathcal{U}_{lF}^*(k)\mathbf{L}_{N,F}\mathcal{U}_{lF}(k) = \mathbf{u}_{lF}^*(k)\mathbf{u}_{lF}^T(k)\cdot \mathbf{L}_{N,F}, \tag{54}$$

equation (53) can be simplified to

$$\mathbf{u}_{lF}^*(k)\cdot \mathbf{e}_{lF}(k) = \Big(\mathbf{u}_{lF}^*(k)\mathbf{u}_{lF}^T(k)\cdot \mathbf{L}_{N,F}\Big)\boldsymbol{\epsilon}_F(k) + \mathbf{u}_{lF}^*(k)\cdot \mathbf{e}_{lF}^o(k). \tag{55}$$

Substituting equation (55) into equation (49), we have that

$$\boldsymbol{\epsilon}_F(k+1) = \Big(\mathbf{I} - \frac{\mu}{L}\mathbf{U}_{N,F}\sum_{l=0}^{K-1}\mathbf{u}_{lF}^*(k)\mathbf{u}_{lF}^T(k)\cdot \mathbf{L}_{N,F}\Big)\boldsymbol{\epsilon}_F(k) - \frac{\mu}{L}\mathbf{U}_{N,F}\sum_{l=0}^{K-1}\mathbf{u}_{lF}^*(k)\cdot \mathbf{e}_{lF}^o(k). \tag{56}$$

Taking the expectation of the above equation yields

$$\mathrm{E}\boldsymbol{\epsilon}_F(k+1) = \Big(\mathbf{I} - \frac{\mu}{N}\mathbf{U}_{N,F}\Big(\mathbf{R}_{u,F}\cdot \mathbf{L}_{N,F}\Big)\Big)\mathrm{E}\boldsymbol{\epsilon}_F(k), \tag{57}$$

where $\mathbf{R}_{u,F} = \mathbf{F}^H\mathbf{R}_u\mathbf{F}$ and $\mathbf{R}_u$ is the $2N \times 2N$ autocorrelation matrix of $u(n)$. Equation (57) is similar to the result that corresponds to the DFT block LMS algorithm (Farhang-Boroujeny & Chan, 2000). Therefore, the convergence characteristics of the HOT DFT block LMS algorithm are similar to that of the DFT block LMS algorithm.

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

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu}{L}\sum_{l=0}^{K-1}\mathbf{U}_N\mathbf{F}^{-1}\Lambda_l^{-1}(k)\Big(\mathbf{u}_{lF}^*(k)\cdot \mathbf{e}_{lF}(k)\Big) \tag{58}$$

and

$$\Lambda_l(k+1) = \frac{k-1}{k}\Lambda_l(k) + \frac{1}{kL}\mathrm{Diag}\Big[\mathbf{u}_{lF}^*(k)\cdot \mathbf{u}_{lF}(k)\Big]. \tag{59}$$

7. Misadjustment of the HOT DFT LMS algorithm

In this section, the misadjusment of the HOT DFT block LMS algorithm is derived. The mean square error of the conventional LMS algorithm is given by

$$J(n) = \mathrm{E}\Big|e(n)\Big|^2. \tag{60}$$

For the block LMS algorithm, the mean square error is given by

$$J(k) = \frac{1}{L}\mathrm{E}\sum_{i=0}^{L-1}\Big|e(kL+i)\Big|^2, \tag{61}$$

which is also equivalent to

$$J(k) = \frac{1}{2NL}\mathrm{E}\sum_{l=0}^{K-1}\mathbf{e}_{lF}^H(k)\mathbf{e}_{lF}(k). \tag{62}$$

Using equation (51), the mean square error of the HOT DFT block LMS algorithm is given by

$$J(k) = J^o + \frac{1}{2NL}\mathrm{E}\sum_{l=0}^{K-1}\Big|\Big|\mathbf{L}_{N,F}\mathcal{U}_{lF}(k)\boldsymbol{\epsilon}(k)\Big|\Big|^2, \tag{63}$$

where J^o is the mean square of $e^o(n)$. Assuming that $\boldsymbol{\epsilon}(k)$ and $\mathrm{Diag}[\mathbf{u}_{lF}(k)]$ are independent, the excess mean square error is given by

$$J_{\mathrm{ex}}(k) = \frac{1}{2NL}\mathrm{E}\sum_{l=0}^{K-1}\boldsymbol{\epsilon}_F^H(k)\mathrm{E}\mathcal{U}_{lF}^H(k)\mathbf{L}_{N,F}\mathcal{U}_{lF}^H(k)\boldsymbol{\epsilon}_F(k). \tag{64}$$

Using equation (54), the excess mean square error can be written as

$$J_{\mathrm{ex}} = \frac{K}{2NL}\mathrm{E}\boldsymbol{\epsilon}_F^H(k)\Big(\mathbf{R}_{u,F}\cdot\mathbf{L}_{N,F}\Big)\boldsymbol{\epsilon}_F(k), \tag{65}$$

or equivalently

$$J_{\mathrm{ex}} = \frac{K}{2NL}\mathrm{tr}\Big[\Big(\mathbf{R}_{u,F}\cdot\mathbf{L}_{N,F}\Big)\mathrm{E}\boldsymbol{\epsilon}_F(k)\boldsymbol{\epsilon}_F^H(k)\Big]. \tag{66}$$

8. 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 4 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 5 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 6 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.

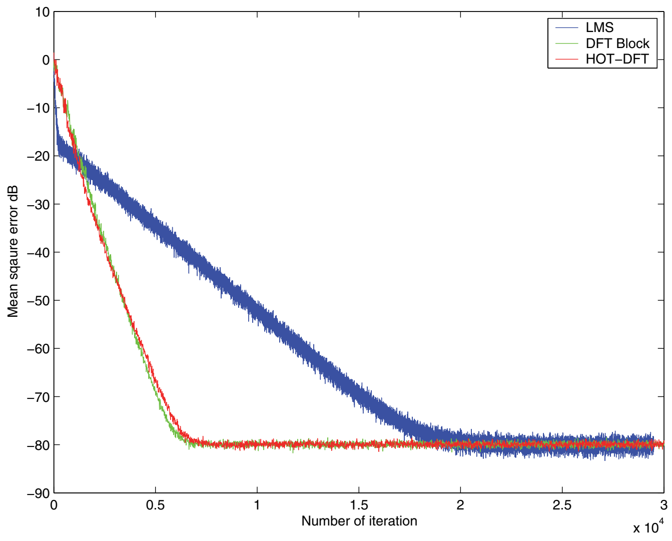

Fig. 4. Learning curves for the LMS, HOT DFT block LMS, and DFT block LMS algorithms. $N = 4$ and $K = 3$. $\rho = 0.9$.

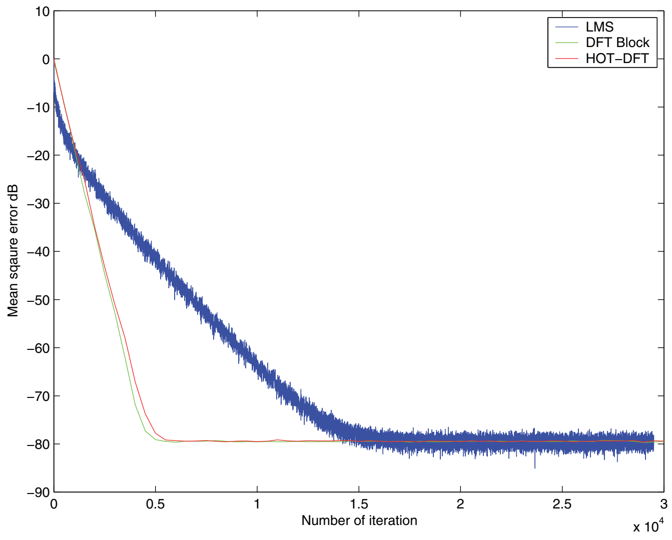

Fig. 5. Learning curves for the LMS, HOT DFT block LMS, and DFT block LMS algorithms. $N = 50$ and $K = 10$. $\rho = 0.9$.

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. The learning curves are shown in Figure 8. The simulations are again consistent with the theoretical predictions presented in this chapter.

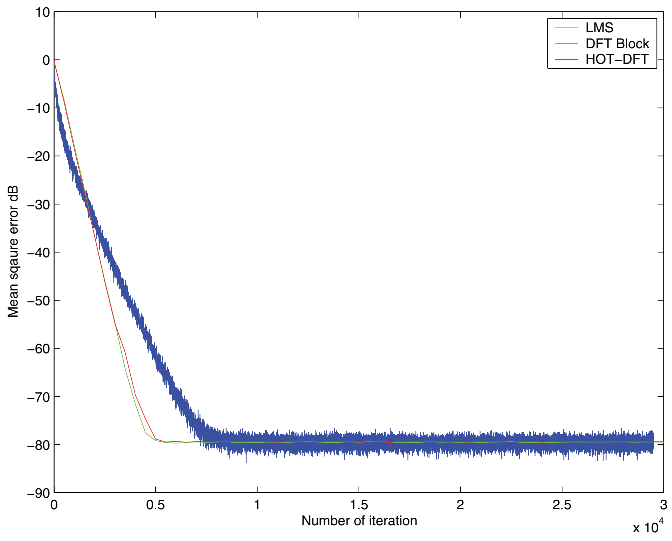

Fig. 6. Learning curves for the LMS, HOT DFT block LMS, and DFT block LMS algorithms. $N = 50$ and $K = 10$. $\rho = 0.8$.

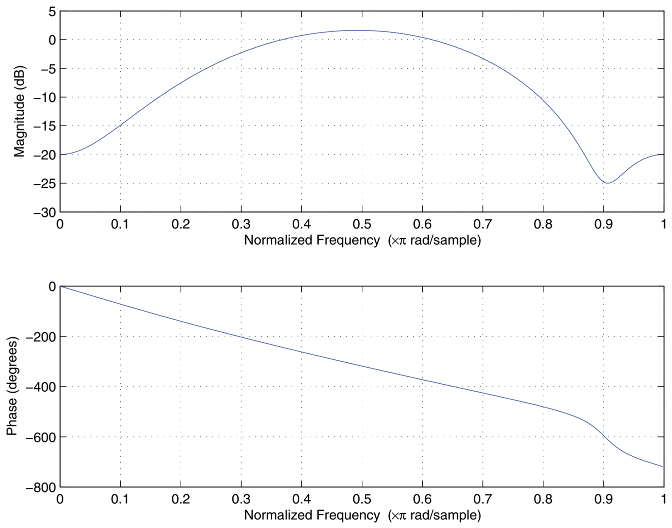

Fig. 7. Frequency response of the coloring filter.

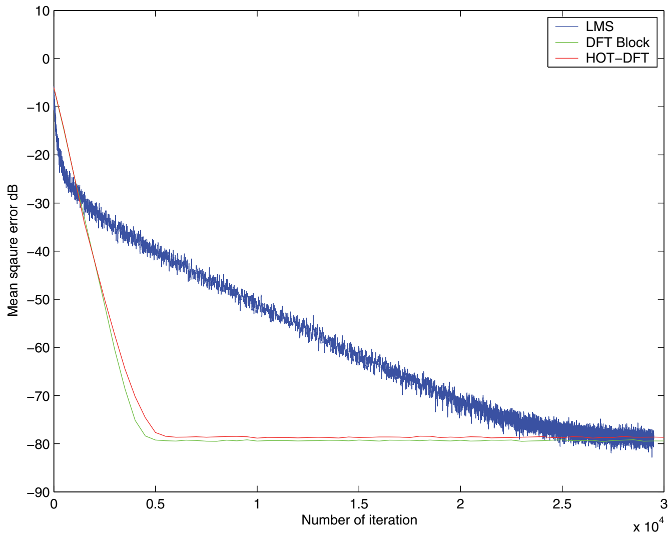

Fig. 8. Learning curves for the LMS, HOT DFT block LMS, and DFT block LMS algorithms. $N = 50$ and $K = 10$.

9. Conclusions

In this chapter a new computationally efficient block LMS algorithm was presented. This algorithm is called the HOT DFT block LMS algorithm. It is based on a newly developed transform called the HOT. The basis of the HOT has many zero-valued samples and resembles the DFT basis, which makes the HOT computationally attractive. The HOT DFT block LMS 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. The analytical predictions and simulations showed that the convergence characteristics of the HOT DFT block LMS algorithm are similar to that of the DFT block LMS algorithm.

10. References

Alkhouli, O.; DeBrunner, V.; Zhai, Y. & Yeary, M. (2005). “FIR Adaptive filters based on Hirschman optimal transform,” IEEE/SP 13th Workshop on Statistical Signal Processing, 2005.

Alkhouli, O. & DeBrunner, V. (2007). “Convergence Analysis of Hirschman Optimal Transform (HOT) LMS adaptive filter,” IEEE/SP 14th Workshop on Statistical Signal Processing, 2007.

Alkhouli, O. & DeBrunner, V. (2007). “Hirschman optimal transform block adaptive filter,” International conference on Acoustics, Speech, and Signal Processing (ICASSP), 2007.

Clark, G.; Mitra S. & Parker, S. (1981). “Block implementation of adaptive digital filters,” *IEEE Trans. ASSP*, pp. 744-752,ăJun 1981.

DeBrunner, V.; Özaydin, M. & Przebinda T. (1999). “Resolution in time-frequency,” *IEEE Trans. ASSP*, pp. 783-788, Mar 1999.

DeBrunner, V. & Matusiak, E. (2003). “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.

DeBrunner, V.; Havlicek, J.; Przebinda, T. & Özaydin M. (2005). “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.

Farhang-Boroujeny, B. (2000). *Adaptive Filters Theory and Applications*. Wiley, 1999.

Farhang-Boroujeny, B. & Chan, K. (2000). “Analysis of the frequency-domain block LMS algorithm,” *IEEE Trans. ASSP*, pp. 2332, Aug. 2000.

Ferrara, E. (1980). “Fast implementation of LMS adaptive filters,” *IEEE Trans. ASSP*, vol. ASSP-28, NO. 4, Aug 1980.

Mitra, S. (2000). *Digital Signal Processing*. Mc Graw Hill, Second edition, 2000.

Haykin S. (2002). *Adaptive Filter Theory*. Prentice Hall information and system sciences series, Fourth edition, 2002.

Przebinda, H.; DeBrunner, V. & Özaydin M. (2001). “The optimal transform for the discrete Hirschman uncertainty principle,” *IEEE Trans. Infor. Theory*, pp. 2086-2090, Jul 2001.

Widrow, B. & Hoff, Jr., M. (1980). “Adaptive switching circuit,” *IRE WESCON Conv. Rec.*, pt. 4 pp. 96-104, 1980.