An analysis of real-Fourier domain based adaptive algorithms implemented with the Hartley transform using cosine-sine symmetries

An Analysis of Real-Fourier Domain-Based Adaptive Algorithms Implemented with the Hartley Transform Using Cosine-Sine Symmetries

Vasanthan Raghavan, *Student Member, IEEE*, K. M. M. Prabhu, *Senior Member, IEEE*, and Piet C. W. Sommen, *Senior Member, IEEE*

***Abstract*—The least mean squared (LMS) algorithm and its variants have been the most often used algorithms in adaptive signal processing. However the LMS algorithm suffers from a high computational complexity, especially with large filter lengths. The Fourier transform-based block normalized LMS (FBNLMS) reduces the computation count by using the discrete Fourier transform (DFT) and exploiting the fast algorithms for implementing the DFT. Even though the savings achieved with the FBNLMS over the direct-LMS implementation are significant, the computational requirements of FBNLMS are still very high, rendering many real-time applications, like audio and video estimation, infeasible. The Hartley transform-based BNLMS (HBNLMS) is found to have a computational complexity much less than, and a memory requirement almost of the same order as, that of the FBNLMS. This paper is based on the cosine and sine symmetric implementation of the discrete Hartley transform (DHT), which is the key in reducing the computational complexity of the FBNLMS by 33% asymptotically (with respect to multiplications). The parallel implementation of the discrete cosine transform (DCT) in turn can lead to more efficient implementations of the HBNLMS.**

***Index Terms*—Adaptive algorithms, DCT, DST, FBNLMS, FFT, FHT, frequency domain algorithms, LMS algorithms.**

I. Introduction

THE ability of an adaptive filter to operate effectively in an unknown environment and track time variations of the input statistics makes it a powerful device. It finds wide applications in diverse fields such as interference and echo cancellation, channel equalization, linear prediction, and spectral estimation.

The adaptive filter is implemented in the time domain by algorithms like the least mean squared (LMS) algorithm [1], the gradient adaptive lattice [2], or the least squares algorithms [3], [4]. However, as the number of computations per sample needed to implement the LMS algorithm grows linearly with the filter order, it is less useful in real-time applications like audio and video estimation. Many techniques have been proposed to reduce this burgeoning computation count. Transforming the LMS algorithm to the frequency domain [5], [6] has been the classical approach, as we can exploit the various forms of fast Fourier transform (FFT) algorithms to reduce the number of computations. With the generalization of the transform domain LMS algorithm [7], we find that we can use orthogonal transforms to yield better computation counts.

Merched *et al.* [8][1] have introduced the implementation of the FBNLMS in the Hartley domain (the so-called "HBNLMS"). This method involves extending the data matrices to circulant symmetric matrices and then using the Hartley transform to diagonalize them. The main motive of [8] is to find the commonality between adaptive filters using multidelay concepts and those using filterbanks, whereas in this paper, our main motive is the efficient implementation of the HBNLMS algorithm. We exploit the symmetry of the Hartley transform in being decomposed into a cosine and a sine transform of reduced order and the symmetrical structure of the HBNLMS to significantly reduce the computational complexity of the FBNLMS. This is possible entirely because the sine transform of a sequence can be implemented using the cosine transform of its alternate sequence [9]. The asymptotic computational savings with the HBNLMS is 20% with respect to multiplications and additions and 33% with respect to multiplications alone. Since fast cosine transforms are commonly used in MPEG coding and are efficiently implemented in special VLSI chips, decomposition of the Hartley transform to a cosine and a sine transform also has other inherent advantages that are associated with VLSI designs. We also note that the HBNLMS requires almost the same order of memory as that of the FBNLMS implementation.

The Hartley transform has been used in the literature for filtering in [10] and [11], but the authors are not aware of an efficient implementation of the Hartley transform routines based on sine and cosine symmetric decompositions in the context of LMS filters. Merched *et al.* [8] use the regular Hartley transform in their work without exploiting the symmetries of the sequences involved. Thus, this work is an efficient implementation of the proposed Hartley transform-based frequency domain LMS. The paper is organized as follows. The Fourier domain-based block normalized LMS algorithm is described in

Manuscript received June 16, 2003; revised January 9, 2004. The associate editor coordinating the review of this paper and approving it for publication was Prof. Trac D. Tran.

V. Raghavan is with the Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, WI 53706 USA (e-mail: raghavan@cae.wisc.edu; vasanth_295@yahoo.com).

K. M. M. Prabhu was with Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands. He is now with the Department of Electrical Engineering, Indian Institute of Technology Madras, Chennai 600036, India (e-mail: prabhu_kmm@hotmail.com).

P. C. W. Sommen is with the Department of Electrical Engineering, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands (e-mail: p.c.w.sommen@tue.nl).

Digital Object Identifier 10.1109/TSP.2004.838983

[1]We would like to thank one of the reviewers for introducing us to the results of [8].

Section II. The Hartley transform-based BNLMS (HBNLMS) is introduced in Section III. Sections IV and V deal with the implementation issues of the HBNLMS and the computation counts of the discrete Fourier transform (DFT) and discrete Hartley transform (DHT)-based algorithms, respectively. The memory requirements of both the algorithms are compared in Section VI and concluding remarks are provided in Section VII.

A. Notation

We assume real inputs, real "desired" outputs and we start with a real $\mathbf{w}(0)$. This will ensure that the weight vectors over all iterations are real. The operations $\mathbf{u}(\mathrm{n})^{\mathrm{T}}$, $\mathbf{u}(\mathrm{n})^{\mathrm{H}}$ and $\mathbf{u}(\mathrm{n})^{*}$ refer, respectively, to the transpose, Hermitian, and the conjugate of $\mathbf{u}(\mathrm{n})$, respectively. The $\otimes$ operation refers to elementwise multiplication of two sequences. The superscripts in $\mathbf{U^c}(\mathrm{n})$ and $\mathbf{U^{cs}}(\mathrm{n})$ refer, respectively, to "circulant" and "circulant symmetric." $\mathbf{F}$, $\mathbf{H}$, $\mathbf{F^{-1}}$ and $\mathbf{H^{-1}}$ refer to the Fourier, the Hartley, the inverse Fourier and the inverse Hartley matrices, respectively. X(k) and X_H(k) refer, respectively, to the DFT and the DHT of the sequence x(n). The computation count of an algorithm is defined as the number of multiplications and additions, since in most modern-day digital signal processors (DSPs), the time needed for implementing an addition is almost the same as that for a multiplication.

The tap inputs of the LMS filter are denoted by $\mathrm{u}(\mathrm{n}), \mathrm{u}(\mathrm{n}-1), \ldots, \mathrm{u}(\mathrm{n}-\mathrm{M}+1)$ and the tap weights by $\mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_{\mathrm{M}}$. The output, the desired output, and the error in the output are denoted by y(n), d(n), and e(n), respectively. Here, M is the order of the filter used. The filter output y(n) is given by the convolution sum

$$\mathrm{y}(\mathrm{n}) = \sum_{\mathrm{k}=1}^{\mathrm{k}=\mathrm{M}} \mathrm{w}(\mathrm{k})\mathrm{u}(\mathrm{n}-\mathrm{k}+1). \tag{1}$$

This output y(n) is used to estimate the desired response d(n). Let $\mathbf{w}^{\mathrm{T}} = [\mathrm{w}_1\ \mathrm{w}_2\ \ldots\ \mathrm{w}_{\mathrm{M}}]$ and $\mathbf{u}(\mathrm{n})^{\mathrm{T}} = [\mathrm{u}(\mathrm{n})\ \mathrm{u}(\mathrm{n}-1)\ \ldots\ \mathrm{u}(\mathrm{n}-\mathrm{M}+1)]$. We do not use an explicit dependence of the weight vector and the error vector on the discrete-time variable "n" as we are dealing with a particular iteration, and for that iteration, these vectors can be assumed to be constant. In the block algorithm (where B is the filter block length), we define the block input $\mathbf{U}(\mathrm{n})$ (an M × B matrix) as

$$\mathbf{U}(\mathrm{n}) = [\mathbf{u}(\mathrm{n}-\mathrm{B}+1)\cdots\mathbf{u}(\mathrm{n}-1)\mathbf{u}(\mathrm{n})]. \tag{2}$$

Then, the block output $\mathbf{y}_{\mathrm{B}}(\mathrm{n})$ is defined as $\mathbf{y}_{\mathrm{B}}(\mathrm{n}) = \mathbf{U}(\mathrm{n})^{\mathrm{T}}\mathbf{w}$, where

$$\mathbf{y}_{\mathrm{B}}(\mathrm{n})^{\mathrm{T}} = [\mathrm{y}(\mathrm{n}-\mathrm{B}+1)\cdots\mathrm{y}(\mathrm{n}-1)\mathrm{y}(\mathrm{n})]. \tag{3}$$

The estimation error $\mathbf{e}_{\mathrm{B}}(\mathrm{n})$ is defined as

$$\mathbf{e}_{\mathrm{B}}(\mathrm{n})^{\mathrm{T}} = [\mathrm{e}(\mathrm{n}-\mathrm{B}+1)\cdots\mathrm{e}(\mathrm{n}-1)\mathrm{e}(\mathrm{n})]. \tag{4}$$

μ refers to the adaptation constant in the adaptive algorithm.

II. FBNLMS Algorithm

The Fourier transform-based block normalized least mean square (FBNLMS) algorithm [12], [13] is an efficient way of implementing the block normalized least mean squares (BNLMS) algorithm [12]. The weight update equation of the BNLMS algorithm is

$$\mathbf{w}[(\mathrm{n}+1)\mathrm{B}] = \mathbf{w}(\mathrm{nB}) + \frac{1}{\sigma_{\mathrm{U}}^2}[\mu\mathbf{U}(\mathrm{n})\mathbf{e}_{\mathrm{B}}(\mathrm{n})]$$
$$\sigma_{\mathrm{U}}^2 = \frac{1}{\mathrm{BN}}\sum_{\mathrm{i}=0}^{\mathrm{i}=\mathrm{B}-1}\mathbf{E}\left[\mathbf{u}(\mathrm{n}-\mathrm{i})^{\mathrm{T}}\mathbf{u}(\mathrm{n}-\mathrm{i})\right]. \tag{5}$$

Here, σ_{U}^2 is an estimate of block variance of $\mathbf{U}(\mathrm{n})$. The structure of $\mathbf{U}(\mathrm{n})$ allows us to extend it circulantly as follows:

$$\mathbf{U^c}(\mathrm{n})^{\mathrm{T}} = \begin{bmatrix}\mathbf{X} & \mathbf{U}(\mathrm{n})\\ \mathbf{Y} & \mathbf{Z}\end{bmatrix} \tag{6}$$

where $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ are chosen appropriately. The first row of $\mathbf{U^c}(\mathrm{n})^{\mathrm{T}}$ is given as

$$\mathbf{U}_1^{\mathrm{c}}(\mathrm{n})^{\mathrm{T}} = [\mathrm{u}(\mathrm{n}-\mathrm{N}+1)\cdots\mathrm{u}(\mathrm{n}-1)\mathrm{u}(\mathrm{n})]. \tag{7}$$

Here, the size of $\mathbf{U^c}(\mathrm{n})$ (a square matrix of N × N), which is denoted by N, is constrained by $\mathrm{N} > (\mathrm{M}+\mathrm{B}-1)$. Usually, N is chosen to be a power of two so that the FFTs can be used to reduce computation (described later in this section). Further manipulation is made easy if we notice that $\mathbf{U}(\mathrm{n})\mathbf{e}_{\mathrm{B}}(\mathrm{n})$ is a submatrix of $\mathbf{U^c}(\mathrm{n})^{\mathrm{T}}\mathbf{e^c}$, where $\mathbf{e}^{\mathrm{cT}} = [\mathbf{0}_{1\times\mathrm{N}-\mathrm{B}}\mathbf{e}_{\mathrm{B}}(\mathrm{n})_{1\times\mathrm{B}}^{\mathrm{T}}]$. This submatrix can then be extracted from the product $\mathbf{U^c}(\mathrm{n})^{\mathrm{T}}\mathbf{e^c}$ by premultiplication with a M × N windowing matrix $\mathbf{B}$ defined as $\mathbf{B} = [\mathbf{I}_{\mathrm{M}\times\mathrm{M}}\ \mathbf{0}_{\mathrm{M}\times(\mathrm{N}-\mathrm{M})}]$. The following algebraic manipulations then follow trivially:

$$\mathbf{U}(\mathrm{n})\mathbf{e}_{\mathrm{B}}(\mathrm{n}) = \mathbf{B}\mathbf{U^c}(\mathrm{n})^{\mathrm{T}}\mathbf{e^c}$$
$$\mathbf{U}(\mathrm{n})\mathbf{e}_{\mathrm{B}}(\mathrm{n}) = \mathbf{B}\mathbf{F^{-1}}\mathbf{F}\mathbf{U^c}(\mathrm{n})^{\mathrm{T}}\mathbf{F^{-1}}\mathbf{F}\mathbf{e^c}. \tag{8}$$

It can be seen that $\mathbf{F}\ \mathbf{U^c}(\mathrm{n})^{\mathrm{T}}\ \mathbf{F^{-1}} = [\mathbf{F}\ \mathbf{U^c}(\mathrm{n})\ \mathbf{F^{-1}}]^{\mathrm{H}} = \mathrm{diag}[\mathbf{U}_1^{\mathrm{c}}(\mathrm{k})]^{\mathrm{H}}$, where we exploit the realness of the input vector and the fact that Fourier matrices diagonalize circulant matrices [14]. This, therefore, leads to

$$\mathbf{U}(\mathrm{n})\mathbf{e}_{\mathrm{B}}(\mathrm{n}) = \mathbf{B}\mathbf{F^{-1}}[\mathbf{U}_1^{\mathrm{c}}(\mathrm{k})^{*}\otimes\mathbf{E}^{\mathrm{c}}(\mathrm{k})] \tag{9}$$

where the symbol $\otimes$ denotes the elementwise multiplication of the two N-point sequences. The weight update equation of the BNLMS algorithm then becomes

$$\mathbf{w}[(\mathrm{n}+1)\mathrm{B}] = \mathbf{w}(\mathrm{nB}) + \frac{1}{\sigma_{\mathrm{U}}^2}\left[\mu\mathbf{B}\mathbf{F^{-1}}\left(\mathbf{U}_1^{\mathrm{c}}(\mathrm{k})^{*}\otimes\mathbf{E}^{\mathrm{c}}(\mathrm{k})\right)\right]. \tag{10}$$

$\mathbf{E^c}(\mathrm{k})$ can be written as follows:

$$\mathbf{E}^{\mathrm{c}}(\mathrm{k}) = \mathbf{D}^{\mathrm{c}}(\mathrm{k}) - \mathbf{Y}^{\mathrm{c}}(\mathrm{k}),\ \mathbf{Y}^{\mathrm{c}}(\mathrm{k}) = \mathbf{F}\mathbf{y}^{\mathrm{c}},\ \mathbf{D}^{\mathrm{c}}(\mathrm{k}) = \mathbf{F}\mathbf{d}^{\mathrm{c}}$$
$$\mathbf{d}^{\mathrm{cT}} = \left[\mathbf{0}_{1\times\mathrm{N}-\mathrm{B}}\mathbf{d}_{\mathrm{B}}(\mathrm{n})_{1\times\mathrm{B}}^{\mathrm{T}}\right],\ \mathbf{y}^{\mathrm{cT}} = \left[\mathbf{0}_{1\times\mathrm{N}-\mathrm{B}}\mathbf{y}_{\mathrm{B}}(\mathrm{n})_{1\times\mathrm{B}}^{\mathrm{T}}\right]$$
$$\mathbf{y}_{\mathrm{B}}(\mathrm{n}) = \mathbf{A}\mathbf{U^c}(\mathrm{n})\mathbf{w}^{\mathrm{c}} \tag{11}$$

where $\mathbf{A} = [\mathbf{0}_{\mathrm{B}\times\mathrm{N}-\mathrm{B}}\mathbf{I}_{\mathrm{B}\times\mathrm{B}}]$, and $\mathbf{w}^{\mathrm{cT}} = [\mathbf{w}_{1\times\mathrm{M}}^{\mathrm{T}}\mathbf{0}_{1\times\mathrm{N}-\mathrm{M}}]$. Equation (11) can then be rewritten similarly as

$$\mathbf{y}_{\mathrm{B}}(\mathrm{n}) = \mathbf{A}\mathbf{F^{-1}}\mathbf{F}\mathbf{U^c}(\mathrm{n})\mathbf{F^{-1}}\mathbf{F}\mathbf{w}^{\mathrm{c}}$$
$$\mathbf{y}_{\mathrm{B}}(\mathrm{n}) = \mathbf{A}\mathbf{F^{-1}}[\mathbf{U}_1^{\mathrm{c}}(\mathrm{k})\otimes\mathbf{W}^{\mathrm{c}}(\mathrm{k})]. \tag{12}$$

Since $\mathbf{d^c}$ is a sequence that is well-known even before the start of iterations, we can assume that it is a constant sequence and therefore, $\mathbf{D^c}(\mathrm{k})$ is a known sequence (or can be computed offline). We therefore need three forward Fourier transforms, two inverse Fourier transforms, two elementwise multiplications, a variance estimator and two windows for each iteration. We do not concern ourselves with the dynamic behavior (convergence or tracking properties) of the FBNLMS or the Hartley transform-based algorithm that will be introduced in Section III, because both the algorithms are in principle the same when real inputs and weights are used. The main motive of this paper is to exploit the structure that lies hidden in the vectors used in HBNLMS, in order to reduce the computation count. However, we would like to comment that, since the FBNLMS algorithm is an exact transformation of the BNLMS algorithm, the dynamic behavior of both of them will be essentially the same [12].

III. Hartley Transform-Based BNLMS (HBNLMS)

The DHT [15], [16] of a real valued N-point sequence x(n) is defined as follows:

$$\mathbf{X}_{\mathrm{H}}(\mathrm{k}) = \sum_{\mathrm{n=0}}^{\mathrm{n=N-1}} \mathrm{x(n)cas}\left(\frac{2\pi \mathrm{kn}}{\mathrm{N}}\right), \quad \mathrm{k} = 0, 1, \cdots, \mathrm{N}-1 \tag{13}$$

where, $\mathrm{cas(x)} = \cos(\mathrm{x}) + \sin(\mathrm{x})$. A circulant matrix $\mathbf{C}$ can be diagonalized by a Fourier matrix, viz. $\mathbf{F\,C\,F^{-1}} = \mathrm{diag}[\mathrm{C}_1(\mathrm{k})]$, where $\mathrm{C}_1(\mathrm{k})$ is the DFT of the first column of $\mathbf{C}$ [14]. A circulant symmetric matrix $\mathbf{C}$ can be diagonalized by a Hartley matrix.

$$\mathbf{HCH^{-1}} = \mathrm{diag}[\mathbf{HC}_1] \tag{14}$$

where C_1 is the first column of C. The FBNLMS algorithm can then be implemented by extending all data matrices to be circulant and symmetric and using $\mathbf{H}$ to diagonalize them [8]. The order of all circulant symmetric matrices should satisfy the constraint $\mathrm{N^{cs}} > 2(\mathrm{M} + \mathrm{B} - 1)$. We will abuse notation and use $\mathrm{N^{cs}} = 2\mathrm{N}$ (where N stands for the size of the extended circulant matrix in the FBNLMS) henceforth. The following equations can then be seen to be true for HBNLMS [8].

$$\begin{aligned}
\mathbf{w}\left[(\mathrm{n}+1)\mathrm{B}\right] &= \mathbf{w}(\mathrm{nB}) + \frac{1}{\sigma_{\mathrm{U}}^2}\left[\mu \mathbf{U}(\mathrm{n})\mathbf{e}_{\mathrm{B}}(\mathrm{n})\right] \\
\mathbf{U}(\mathrm{n})\mathbf{e}_{\mathrm{B}}(\mathrm{n}) &= \mathbf{B^{cs}U^{cs}}(\mathrm{n})\mathbf{e^{cs}}
\end{aligned}$$

where $\mathbf{e}^{\mathrm{csT}} = [\mathbf{0}_{1\times 2\mathrm{N}-\mathrm{B}}\mathbf{e}_{\mathrm{B}}(\mathrm{n})^{\mathrm{T}}_{1\times \mathrm{B}}]$ and $\mathbf{B^{cs}} = [\mathbf{I}_{\mathrm{M}\times\mathrm{M}}\mathbf{0}_{\mathrm{M}\times 2\mathrm{N}-\mathrm{M}}]$.

$$\begin{aligned}
\mathbf{U}(\mathrm{n})\mathbf{e}_{\mathrm{B}}(\mathrm{n}) &= \mathbf{B^{cs}H^{-1}HU^{cs}}(\mathrm{n})\mathbf{H^{-1}He^{cs}} \\
\mathbf{U}(\mathrm{n})\mathbf{e}_{\mathrm{B}}(\mathrm{n}) &= \mathbf{B^{cs}H^{-1}}\left[\mathbf{U}^{\mathrm{cs}}_{1\mathrm{H}}(\mathrm{k}) \otimes \mathbf{E}^{\mathrm{cs}}_{\mathrm{H}}(\mathrm{k})\right] \\
\mathbf{w}\left[(\mathrm{n}+1)\mathrm{B}\right] &= \mathbf{w}(\mathrm{nB}) + \frac{1}{\sigma_{\mathrm{U}}^2}\left[\mu \mathbf{B^{cs}H^{-1}}[\mathbf{U}^{\mathrm{cs}}_{1\mathrm{H}}(\mathrm{k}) \otimes \mathbf{E}^{\mathrm{cs}}_{\mathrm{H}}(\mathrm{k})]\right] \\
\mathbf{E}^{\mathrm{cs}}_{\mathrm{H}}(\mathrm{k}) &= \mathbf{D}^{\mathrm{cs}}_{\mathrm{H}}(\mathrm{k}) - \mathbf{Y}^{\mathrm{cs}}_{\mathrm{H}}(\mathrm{k}) \\
\mathbf{Y}^{\mathrm{cs}}_{\mathrm{H}}(\mathrm{k}) &= \mathbf{Hy^{cs}} = \mathbf{Fy^{cs}} \\
\mathbf{D}^{\mathrm{cs}}_{\mathrm{H}}(\mathrm{k}) &= \mathbf{Hd^{cs}} = \mathbf{Fd^{cs}} \\
\mathbf{d}^{\mathrm{csT}} &= \left[\mathbf{0}_{1\times 2\mathrm{N}-\mathrm{B}}\mathbf{d}_{\mathrm{B}}(\mathrm{n})^{\mathrm{T}}_{1\times \mathrm{B}}\right] \\
\mathbf{y}^{\mathrm{csT}} &= \left[\mathbf{0}_{1\times 2\mathrm{N}-\mathrm{B}}\mathbf{y}_{\mathrm{B}}(\mathrm{n})^{\mathrm{T}}_{1\times \mathrm{B}}\right] \\
\mathbf{y}_{\mathrm{B}}(\mathrm{n}) &= \mathbf{A^{cs}U^{cs}}(\mathrm{n})\mathbf{w^{cs}}
\end{aligned} \tag{15}$$

where $\mathbf{A^{cs}} = [\mathbf{0}_{\mathrm{B}\times 2\mathrm{N}-\mathrm{B}}\mathbf{I}_{\mathrm{B}\times\mathrm{B}}]$, $\mathbf{w}^{\mathrm{csT}} = [\mathbf{w}^{\mathrm{T}}_{1\times\mathrm{M}}\mathbf{0}_{1\times 2\mathrm{N}-\mathrm{M}}]$, and $\mathbf{U}^{\mathrm{cs}}_{1\mathrm{H}}(\mathrm{k})$ is the DHT of the first column of $\mathbf{U^{cs}}(\mathrm{n})$. We then have

$$\begin{aligned}
\mathbf{y}_{\mathrm{B}}(\mathrm{n}) &= \mathbf{A^{cs}H^{-1}HU^{cs}}(\mathrm{n})\mathbf{H^{-1}Hw^{cs}} \\
\mathbf{y}_{\mathrm{B}}(\mathrm{n}) &= \mathbf{A^{cs}H^{-1}}\left[\mathbf{U}^{\mathrm{cs}}_{1\mathrm{H}}(\mathrm{k}) \otimes \mathbf{W}^{\mathrm{cs}}_{\mathrm{H}}(\mathrm{k})\right].
\end{aligned} \tag{16}$$

Since $\mathbf{d^{cs}}$ is a sequence that is well known even before the start of iterations, we can assume that its transform is a known sequence. Thus, three Hartley transforms and two inverse Hartley transforms are needed to compute $\mathbf{U}(\mathrm{n})\mathbf{e}_{\mathrm{B}}(\mathrm{n})$ in the frequency domain. These Hartley transforms can be computed using the more efficient DCT and discrete sine transform (DST) systems. It might initially appear that the HBNLMS has an inherent disadvantage because of the order of the transforms needed here. For every Fourier transform (and inverse) of order N, we need a Hartley transform (and inverse) of order 2N. The HBNLMS would be computationally disadvantageous when compared to the FBNLMS (if the symmetries in the sequences are not used), as is done in [8], where a Hartley transform of order 2N is implicitly used. The main motive of this work is to exploit the symmetries inherent in the sequences to use the computationally more efficient DCT and DST systems to recursively compute the needed transforms. In this process, we reduce the needed computations even below that needed by the FBNLMS (with *real*-FFT's used for a fair comparison). The symmetry in the sequences helps us in replacing a 2N-point Hartley transform with cosine transforms of order of N/2 down until 1 [17], [18].

IV. Implementation of the HBNLMS

When real data vectors are used, the HBNLMS is just another mathematically equivalent way of implementing the FBNLMS. The basic idea of our implementation is that the recursive implementation of the DCT-I, DST-I, DCT-II and DST-II involve the same building blocks with signs changed appropriately to give the needed transforms. The first column of $\mathbf{U^{cs}}(\mathrm{n})$, $\mathbf{U}^{\mathbf{cs}}_1(\mathrm{n})$ satisfies the following properties:

$$\begin{aligned}
\mathbf{U}^{\mathrm{cs}}_1(\mathrm{n}) &= \mathbf{U}^{\mathrm{cs}}_1(2\mathrm{N}-\mathrm{n}),\ \mathrm{n} = 1, \cdots, \mathrm{N}-1 \\
\mathbf{U}^{\mathrm{cs}}_1(0) &= \mathbf{U}^{\mathrm{cs}}_1(\mathrm{N}).
\end{aligned} \tag{17}$$

This then implies the following about $\mathbf{U}^{\mathrm{cs}}_{1\mathrm{H}}(\mathrm{k})$:

$$\begin{aligned}
\mathbf{U}^{\mathrm{cs}}_{1\mathrm{H}}(\mathrm{k}) &= \sum_{\mathrm{n=0}}^{\mathrm{n=2N-1}} \mathbf{U}^{\mathrm{cs}}_1(\mathrm{n})\mathrm{cas}\left(\frac{2\pi \mathrm{kn}}{2\mathrm{N}}\right) \\
&\quad \mathrm{k} = 0, 1, \cdots, 2\mathrm{N}-1 \\
\mathbf{U}^{\mathrm{cs}}_{1\mathrm{H}}(\mathrm{k}) &= \sum_{\mathrm{n=0}}^{\mathrm{n=N-1}} \mathbf{U}^{\mathrm{csm}}_1(\mathrm{n})\cos\left(\frac{2\pi \mathrm{kn}}{2\mathrm{N}}\right) \\
&\quad + \mathbf{U}^{\mathrm{cs}}(0)\cos(\pi \mathrm{k}) \\
&\quad \mathrm{k} = 0, 1, \cdots, \mathrm{N}-1 \\
\mathbf{U}^{\mathrm{cs}}_{1\mathrm{H}}(\mathrm{k}+\mathrm{N}) &= \sum_{\mathrm{n=0}}^{\mathrm{n=N-1}} (-1)^{\mathrm{n}}\mathbf{U}^{\mathrm{csm}}_1(\mathrm{n})\cos\left(\frac{2\pi \mathrm{kn}}{2\mathrm{N}}\right) \\
&\quad + \mathbf{U}^{\mathrm{cs}}(0)\cos(\pi \mathrm{k}) \\
&\quad \mathrm{k} = 0, 1, \cdots, \mathrm{N}-1.
\end{aligned} \tag{18}$$

Here, $\mathbf{U}_1^{\mathrm{csm}}(\mathrm{n})$ is defined as

$$\mathbf{U}_1^{\mathrm{csm}}(\mathrm{n})^{\mathrm{T}} = [\mathbf{U}_1^{\mathrm{cs}}(0), 2\mathbf{U}_1^{\mathrm{cs}}(1), 2\mathbf{U}_1^{\mathrm{cs}}(2), \cdots, 2\mathbf{U}_1^{\mathrm{cs}}(\mathrm{N}-1)]. \tag{19}$$

This implies that all we need to do to find the 2N-point Hartley transform is to find the DCT-I [19] of the sequences $\mathbf{U}_1^{\mathrm{csm}}(\mathrm{n})$ and $\mathbf{U}_1^{\mathrm{csm}}(\mathrm{n})(-1)^{\mathrm{n}}$.

The DCT-I of an N-point sequence is computed recursively using the DCT-I of the even sampled sequence and the DCT-II of the odd sampled sequence [19], as follows:

$$\begin{aligned}\sum_{\mathrm{n}=0}^{\mathrm{n}=\mathrm{N}-1} \mathrm{x}(\mathrm{n}) \cos\left(\frac{2\pi \mathrm{kn}}{2\mathrm{N}}\right) &= \underbrace{\sum_{\mathrm{n}=0}^{\mathrm{n}=\frac{\mathrm{N}}{2}-1} \mathrm{x}(2\mathrm{n}) \cos\left(\frac{2\pi \mathrm{k}.2\mathrm{n}}{2\mathrm{N}}\right)}_{\substack{\text{Even samples}\\ \text{DCT-I}}} \\ &\quad + \underbrace{\sum_{\mathrm{n}=0}^{\mathrm{n}=\frac{\mathrm{N}}{2}-1} \mathrm{x}(2\mathrm{n}+1) \cos\left(\frac{2\pi \mathrm{k}(2\mathrm{n}+1)}{2\mathrm{N}}\right)}_{\substack{\text{Odd samples}\\ \text{DCT-II}}} \\ &\quad \mathrm{k} = 0, 1, \cdots, \mathrm{N}-1 \end{aligned} \tag{20}$$

If A and B are the DCT-I of the even samples and DCT-II of the odd samples of $\mathbf{U}_1^{\mathrm{csm}}(\mathrm{n})$ and A_1 and B_1 are the last $(\mathrm{N}/2-1)$ points of A and B *in the same order*, respectively, $\mathbf{U}_{1\mathrm{H}}^{\mathbf{cs}}(\mathrm{k})$ for $\mathrm{k}=0$ to $(\mathrm{N}-1)$ can be written as follows:

$$\begin{aligned} \mathbf{U}_{1\mathrm{H}}^{\mathrm{cs}}(\mathrm{n}) &= \mathrm{A}+\mathrm{B}, \quad \mathrm{k}=0,1,\cdots,\frac{\mathrm{N}}{2}-1 \\ \mathbf{U}_{1\mathrm{H}}^{\mathrm{cs}}(\mathrm{N}-1-\mathrm{k}) &= \mathrm{A}_1-\mathrm{B}_1, \quad \mathrm{k}=0,\cdots,\frac{\mathrm{N}}{2}-2 \\ \mathrm{U}_{1\mathrm{H}}^{\mathrm{cs}}\left(\frac{\mathrm{N}}{2}\right) &= \sum_{\mathrm{n}=0}^{\mathrm{n}=\frac{\mathrm{N}}{4}-1} \mathbf{U}_1^{\mathrm{csm}}(4\mathrm{n}) \\ &\quad - \sum_{\mathrm{n}=0}^{\mathrm{n}=\frac{\mathrm{N}}{4}-1} \mathbf{U}_1^{\mathrm{csm}}(4\mathrm{n}+2). \end{aligned} \tag{21}$$

The DCT-I of $\mathbf{U}_1^{\mathrm{csm}}(\mathrm{n})(-1)^{\mathrm{n}}$ can be calculated using A, B, A_1 and B_1 as follows:

$$\begin{aligned} \mathbf{U}_{1\mathrm{H}}^{\mathrm{cs}}(\mathrm{n}) &= \mathrm{A}-\mathrm{B}, \quad \mathrm{k}=\mathrm{N},\cdots,\frac{3\mathrm{N}}{2}-1 \\ \mathbf{U}_{1\mathrm{H}}^{\mathrm{cs}}(2\mathrm{N}-1-\mathrm{k}) &= \mathrm{A}_1+\mathrm{B}_1, \quad \mathrm{k}=0,\cdots,\frac{\mathrm{N}}{2}-2 \\ \mathrm{U}_{1\mathrm{H}}^{\mathrm{cs}}\left(\frac{3\mathrm{N}}{2}\right) &= \sum_{\mathrm{n}=0}^{\mathrm{n}=\frac{\mathrm{N}}{4}-1} \mathbf{U}_1^{\mathrm{csm}}(4\mathrm{n}) \\ &\quad - \sum_{\mathrm{n}=0}^{\mathrm{n}=\frac{\mathrm{N}}{4}-1} \mathbf{U}_1^{\mathrm{csm}}(4\mathrm{n}+2). \end{aligned} \tag{22}$$

This method is employed recursively to obtain A, B, and thus $\mathbf{U}_{1\mathrm{H}}^{\mathbf{cs}}(\mathrm{k})$. Thus, we will need DCT-II of order N/2 down until 1 to compute $\mathbf{U}_{1\mathrm{H}}^{\mathbf{cs}}(\mathrm{k})$. The block diagram of an N-point DCT-I transformer is shown in Fig. 1. The restructured block diagram for computing the DCT-I of an alternate sequence is also shown in Fig. 1.

$\mathbf{W}_{\mathrm{H}}^{\mathbf{cs}}(\mathrm{k})$ can be computed using a similar approach. Let $\mathbf{w}'^{\mathrm{T}} = [\mathrm{w}_2 \ \cdots \ \mathrm{w}_{\mathrm{M}}]$ and $\mathbf{w}''^{\mathrm{T}} = [\mathrm{w}_{\mathrm{M}} \ \cdots \ \mathrm{w}_2]$.

$$\mathbf{w}^{\mathrm{cs}} = \frac{1}{2}\begin{bmatrix} \mathrm{w}_1 \\ \mathbf{w}'_{\mathrm{M}-1\times 1} \\ \mathbf{0}_{\mathrm{N}-\mathrm{M}\times 1} \\ \mathrm{w}_1 \\ \mathbf{0}_{\mathrm{N}-\mathrm{M}\times 1} \\ \mathbf{w}''_{\mathrm{M}-1\times 1} \end{bmatrix}_{\downarrow \mathbf{w}_1^{\mathrm{cs}}} + \frac{1}{2}\begin{bmatrix} \mathrm{w}_1 \\ \mathbf{w}'_{\mathrm{M}-1\times 1} \\ \mathbf{0}_{\mathrm{N}-\mathrm{M}\times 1} \\ -\mathrm{w}_1 \\ \mathbf{0}_{\mathrm{N}-\mathrm{M}\times 1} \\ -\mathbf{w}''_{\mathrm{M}-1\times 1} \end{bmatrix}_{\downarrow \mathbf{w}_2^{\mathrm{cs}}}. \tag{23}$$

Since $\mathbf{w}^{\mathbf{cs}}$ can be written as the sum of two symmetric vectors ($\mathbf{w}_1^{\mathbf{cs}}$ and $\mathbf{w}_2^{\mathbf{cs}}$), $\mathbf{W}_{\mathrm{H}}^{\mathbf{cs}}(\mathrm{k})$ can be computed using an N-point transform as follows:

$$\begin{aligned} \mathbf{W}_{\mathrm{H}}^{\mathrm{cs}}(\mathrm{k}) &= \sum_{\mathrm{n}=0}^{\mathrm{n}=\mathrm{N}-1} \mathrm{w}^{\mathrm{csm}}(\mathrm{n}) \\ &\quad \times \left[\cos\left(\frac{2\pi \mathrm{kn}}{2\mathrm{N}}\right) + \sin\left(\frac{2\pi \mathrm{kn}}{2\mathrm{N}}\right)\right] \\ &\quad + \frac{\mathrm{w}^{\mathrm{cs}}(0)}{2}, \quad \mathrm{k}=0,1,\cdots,\mathrm{N}-1 \\ \mathbf{W}_{\mathrm{H}}^{\mathrm{cs}}(\mathrm{k}+\mathrm{N}) &= \sum_{\mathrm{n}=0}^{\mathrm{n}=\mathrm{N}-1} (-1)^{\mathrm{n}} \mathrm{w}^{\mathrm{csm}}(\mathrm{n}) \\ &\quad \times \left[\cos\left(\frac{2\pi \mathrm{kn}}{2\mathrm{N}}\right) + \sin\left(\frac{2\pi \mathrm{kn}}{2\mathrm{N}}\right)\right] \\ &\quad + \frac{\mathrm{w}^{\mathrm{cs}}(0)}{2}, \quad \mathrm{k}=0,1,\cdots,\mathrm{N}-1 \end{aligned} \tag{24}$$

where $\mathbf{w}^{\mathbf{csm}}(\mathrm{n})^{\mathrm{T}} = [1/2\mathbf{w}^{\mathbf{cs}}(0)\mathbf{w}^{\mathbf{cs}}(1)\mathbf{w}^{\mathbf{cs}}(2)\ldots \mathbf{w}^{\mathbf{cs}}(\mathrm{N}-1)]$.

From (24), it can be seen that $\mathbf{W}_{\mathrm{H}}^{\mathbf{cs}}(\mathrm{k})$ can be computed from the DCT-I and DST-I [20] of the two sequences, $\mathbf{w}^{\mathbf{csm}}(\mathrm{n})$ and $\mathbf{w}^{\mathbf{csm}}(\mathrm{n})(-1)^{\mathrm{n}}$. Let $\mathrm{T}_{\mathrm{N}/2}$ and $\mathrm{U}_{\mathrm{N}/2}$ represent the DCT-I and the DCT-II of the even and the odd samples, respectively, of $\mathbf{w}^{\mathbf{csm}}(\mathrm{n})$. The suffix R shall denote reordering. [The reordering $(\mathrm{Y})_{\mathrm{R}}(\mathrm{n})$ (which is an $\mathrm{N}-1$ point sequence) of an N-point sequence Y(n) is defined as $(\mathrm{Y})_{\mathrm{R}}(\mathrm{n}) = \mathrm{Y}(\mathrm{N}-\mathrm{n})$ for $\mathrm{n}=1$ to $\mathrm{N}-1$.] Then, the 2N-point Hartley transform of $\mathbf{w}_1^{\mathbf{cs}}$ is

$$\mathbf{H}\mathbf{w}_1^{\mathrm{cs}} = \begin{bmatrix} \mathrm{T}_{\frac{\mathrm{N}}{2}} + \mathrm{U}_{\frac{\mathrm{N}}{2}} \\ \mathrm{w}_0 - \mathrm{w}_2 + \mathrm{w}_4 - \cdots \\ \left(\mathrm{T}_{\frac{\mathrm{N}}{2}} - \mathrm{U}_{\frac{\mathrm{N}}{2}}\right)_{\mathrm{R}} \\ \mathrm{T}_{\frac{\mathrm{N}}{2}} - \mathrm{U}_{\frac{\mathrm{N}}{2}} \\ \mathrm{w}_0 - \mathrm{w}_2 + \mathrm{w}_4 - \cdots \\ \left(\mathrm{T}_{\frac{\mathrm{N}}{2}} + \mathrm{U}_{\frac{\mathrm{N}}{2}}\right)_{\mathrm{R}} \end{bmatrix}. \tag{25}$$

The DST-I of a sequence can also be obtained from the DCT-II of that sequence [9]. The DST-I is computed using the DST-I and DST-II of the even and the odd samples, respectively. Thus, a DST-I of order N requires DST-II's of order N/2 down until 1. The DST-II of order L can be computed from the DCT-II of the alternate sequence as shown below:

$$\begin{aligned} \sum_{\mathrm{n}=0}^{\mathrm{n}=\mathrm{L}-1} \mathrm{x}(\mathrm{n}) \sin\left(\frac{\pi \mathrm{k}(2\mathrm{n}+1)}{2\mathrm{L}}\right) &= \sum_{\mathrm{n}=0}^{\mathrm{n}=\mathrm{L}-1} (-1)^{\mathrm{n}} \mathrm{x}(\mathrm{n}) \cos\left(\frac{\pi(\mathrm{L}-\mathrm{k})(2\mathrm{n}+1)}{2\mathrm{L}}\right) \\ &\quad \mathrm{k}=1,2,\cdots,\mathrm{L}-1 \\ \sum_{\mathrm{n}=0}^{\mathrm{n}=\mathrm{L}-1} \mathrm{x}(\mathrm{n}) \sin\left(\frac{\pi \mathrm{k}(2\mathrm{n}+1)}{2\mathrm{L}}\right) &= 0, \quad \mathrm{k}=0. \end{aligned} \tag{26}$$

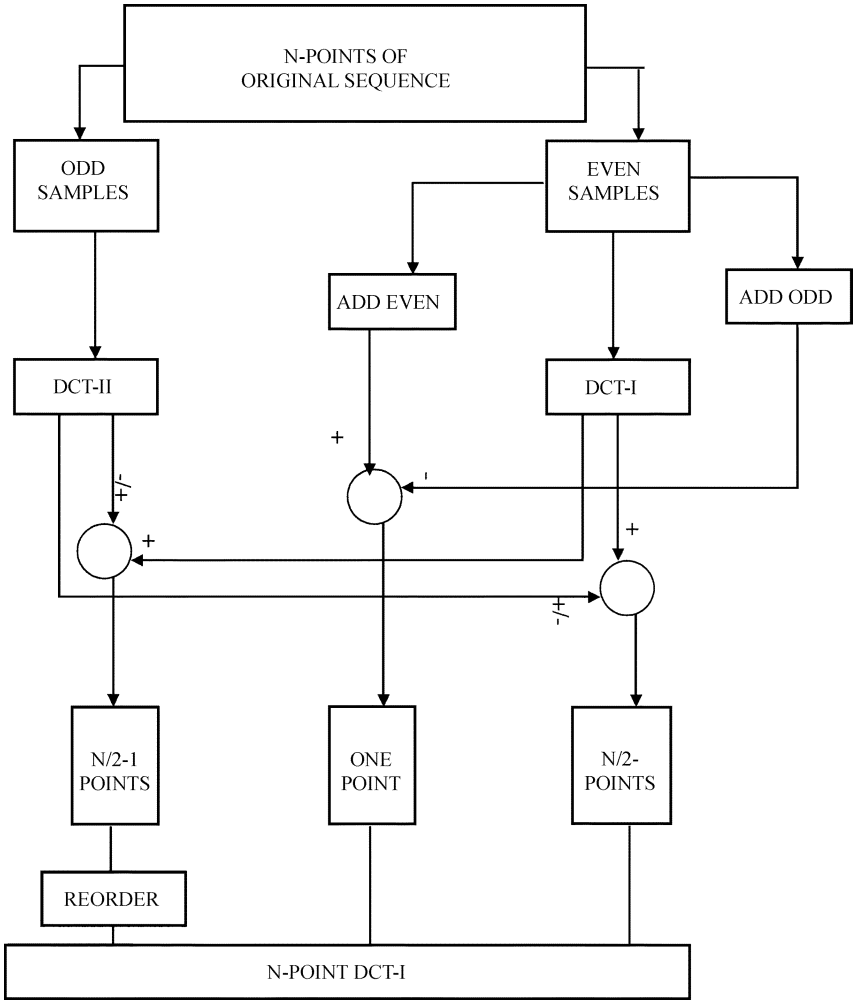


Fig. 1. Block diagram for finding the N-point DCT-I of the original and the alternate sequence from the original data sequence. (The signs a and b in a/b correspond to the respective signs to be used in computing the DCT-I of the original sequence and the alternate sequence.) Circular blocks represent adders.

Let $\mathrm{V}_{\mathrm{N}/2}$ and $\mathrm{W}_{\mathrm{N}/2}$ represent the DST-I and the DST-II, of the even and the odd samples, respectively. Then, the 2N-point Hartley transform of $\mathbf{w}_2^{\mathbf{cs}}$ is

$$\mathbf{H}\mathbf{w}_2^{\mathrm{cs}} = \begin{bmatrix} \mathrm{V}_{\frac{\mathrm{N}}{2}} + \mathrm{W}_{\frac{\mathrm{N}}{2}} \\ \mathrm{w}_1 - \mathrm{w}_3 + \mathrm{w}_5 - \cdots \\ \left(-\mathrm{V}_{\frac{\mathrm{N}}{2}} + \mathrm{W}_{\frac{\mathrm{N}}{2}}\right)_{\mathrm{R}} \\ \mathrm{V}_{\frac{\mathrm{N}}{2}} - \mathrm{W}_{\frac{\mathrm{N}}{2}} \\ -\mathrm{w}_1 + \mathrm{w}_3 - \mathrm{w}_5 + \cdots \\ \left(-\mathrm{V}_{\frac{\mathrm{N}}{2}} - \mathrm{W}_{\frac{\mathrm{N}}{2}}\right)_{\mathrm{R}} \end{bmatrix}. \tag{27}$$

Since the DCT-II of x(n) is implemented recursively using an even and an odd breakup [21], [22], the DCT-II of $\mathbf{x}(\mathrm{n})(-1)^{\mathrm{n}}$ (and thus the DST-II and the DST-I) can be easily computed. The block diagram of a DST-I transformer is shown in Fig. 2. The 2N-point inverse Hartley transform of the product of $\mathbf{U}_{1\mathrm{H}}^{\mathbf{cs}}(\mathrm{k})$ and $\mathbf{W}_{\mathrm{H}}^{\mathbf{cs}}(\mathrm{k})$ can also be computed using an N-point inverse transform. The elementwise product in (15) and (16) (ignoring the (N/2)th and the (3N/2)th points) can be seen to be of the form

$$\begin{bmatrix} \mathrm{A}_{\frac{\mathrm{N}}{2}} + \mathrm{B}_{\frac{\mathrm{N}}{2}} \\ \left(\mathrm{A}_{\frac{\mathrm{N}}{2}} - \mathrm{B}_{\frac{\mathrm{N}}{2}}\right)_{\mathrm{R}} \\ \mathrm{A}_{\frac{\mathrm{N}}{2}} - \mathrm{B}_{\frac{\mathrm{N}}{2}} \\ \left(\mathrm{A}_{\frac{\mathrm{N}}{2}} + \mathrm{B}_{\frac{\mathrm{N}}{2}}\right)_{\mathrm{R}} \end{bmatrix} + \begin{bmatrix} \mathrm{C}_{\frac{\mathrm{N}}{2}} + \mathrm{D}_{\frac{\mathrm{N}}{2}} \\ \left(-\mathrm{C}_{\frac{\mathrm{N}}{2}} + \mathrm{D}_{\frac{\mathrm{N}}{2}}\right)_{\mathrm{R}} \\ \mathrm{C}_{\frac{\mathrm{N}}{2}} - \mathrm{D}_{\frac{\mathrm{N}}{2}} \\ \left(-\mathrm{C}_{\frac{\mathrm{N}}{2}} - \mathrm{D}_{\frac{\mathrm{N}}{2}}\right)_{\mathrm{R}} \end{bmatrix}. \tag{28}$$

Looking at the form of these two sets of sequences, it can be inferred that the inverse transform should be of the form

$$\begin{bmatrix} \mathrm{P}_{\mathrm{N}} + \mathrm{Q}_{\mathrm{N}} \\ (\mathrm{P}_{\mathrm{N}} - \mathrm{Q}_{\mathrm{N}})_{\mathrm{R}} \end{bmatrix}. \tag{29}$$

Since $\mathbf{A}^{\mathbf{cs}}$ windows out the $(\mathrm{P}_{\mathrm{N}} - \mathrm{Q}_{\mathrm{N}})_{\mathrm{R}}$ part, we do not need to compute $\mathrm{P}_{\mathrm{N}} + \mathrm{Q}_{\mathrm{N}}$. If $\mathrm{P_e}$, $\mathrm{P_o}$, $\mathrm{Q_e}$, and $\mathrm{Q_o}$ refer to the even and odd samples of P and Q, they are noted to be as follows:

$$\mathrm{P} = \begin{bmatrix} \mathrm{C}_1^{-1}\mathrm{C}(=\mathrm{P_e}) \\ \mathrm{C}_2^{-1}\mathrm{D}(=\mathrm{P_o}) \end{bmatrix}, \quad \mathrm{Q} = \begin{bmatrix} \mathrm{S}_1^{-1}\mathrm{E}(=\mathrm{Q_e}) \\ \mathrm{S}_2^{-1}\mathrm{F}(=\mathrm{Q_o}) \end{bmatrix} \tag{30}$$

where C_1^{-1}, C_2^{-1}, S_1^{-1}, and S_2^{-1} are the inverse transforms of the type DCT-I, DCT-II, DST-I, and DST-II, respectively. Since the inverse DST-II can be computed using the inverse DCT-II [9], the inverse Hartley transform of order 2N actually needs the same order of computations as a forward transform of length N. $\mathbf{E}_{\mathrm{H}}^{\mathbf{cs}}(\mathrm{k})$ can be computed in exactly a similar way as $\mathbf{W}_{\mathrm{H}}^{\mathbf{cs}}(\mathrm{k})$, as $\mathbf{e}^{\mathrm{cs}}$ is similar in structure to $\mathbf{w}^{\mathrm{cs}}$. The other inverse transform follows the same pattern as the one mentioned above.

V. DCT and DFT Comparisons

The DFT can be implemented either with an aim to reduce memory, execution time, or a combination of both in mind. The algorithm to be used for the DFT depends on the application

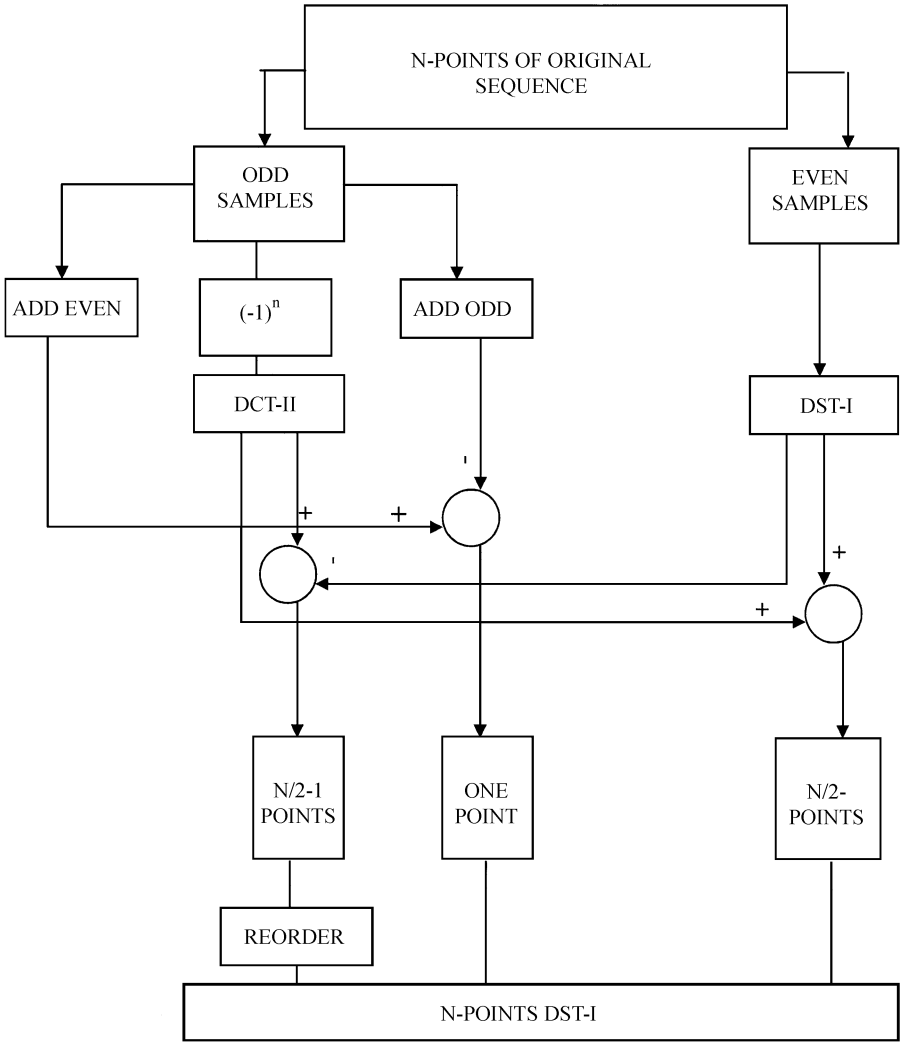


Fig. 2. Block diagram for finding the N-point DST-I of a given sequence. Circular blocks represent adders.

at hand, more so, because a particular algorithm could cater to the needs of that application better. With this in mind, a general choice for the DFT of FBNLMS to compare with the DCT of HBNLMS is a choice dependent on the specifics of the application. Instead of going into the specifics of algorithms and applications, we use the radix-2 versions of DCT and DFT to illustrate the advantages of the two implementation schemes. This does not in anyway imply that the advantages are obtainable only with the radix-2 versions. Our aim is only to show the general trend in computational benefits of the HBNLMS with respect to FBNLMS.

In addition, since we have assumed real inputs and weights, it would be unfair to compare a complex FFT with a real DCT, and hence, in our analysis, we use specialized FFT algorithms that work with real data [23]. However, our choice for the real algorithms is restricted to the radix-2 counterparts on both sides since our motive is in showing the relative advantages rather than comparing the individual algorithms.

Comparing the performance of radix-2 DCT-II and radix-2 DFT, we need, in general, for a DCT-II [21] of order N, $(\mathrm{N}/2\log_2 \mathrm{N})$ multiplications and $(3/2\mathrm{N}\log_2 \mathrm{N} - \mathrm{N} + 1)$ additions. For a real-valued radix-2 DFT [23] of order N, we need $(3/4\mathrm{N}\log_2 \mathrm{N} - 5/2\mathrm{N} + 4)$ multiplications and $(7/4\mathrm{N}\log_2 \mathrm{N} - 7/2\mathrm{N} + 6)$ additions. Table I gives the computation counts of DCT-II and DFT for different N.

Finding the DCT-I from the DCT-IIs requires $(1/2\mathrm{N}\log_2 \mathrm{N} - \mathrm{N} + 1)$ multiplications and $(3/2\mathrm{N}\log_2 \mathrm{N} + \log_2 \mathrm{N} - 4\mathrm{N} + 4) + (3/2\mathrm{N} - 2 - \log_2 \mathrm{N})$ additions. To find the DCT-I of $\mathrm{x(n)}(-1)^{\mathrm{n}}$ and the DCT-I of the 2N-point sequence, an additional (3/2 N) additions are required. Therefore, for finding the DCT-I of the first sequence, we need $(1/2\mathrm{N}\log_2 \mathrm{N} - \mathrm{N} + 1)$ multiplications and $(3/2\mathrm{N}\log_2 \mathrm{N} - \mathrm{N} + 2)$ additions.

To find the L-point DST-II of the sequence at each stage from the corresponding L-point DCT-II, we need L additions. Similarly, to get the 2L-point DST-I from the corresponding DST-II, we need 2L additions. Summing these contributions, we require $(2\mathrm{N} - 4)$ additions. Here, an additional count of (3/2 N) is required to obtain the DST-I of the original 2N-point sequence. By an easy manipulation, the 2N additions that need to be performed to get $\mathbf{W}_{\mathrm{H}}^{\mathbf{cs}}(\mathrm{k})$ from the DCTs and DSTs can be reduced to unit additions.

Removing redundant additions, we find that to compute the 2N-point Hartley transform from the original sequence, we require $(1/2\mathrm{N}\log_2 \mathrm{N} - \mathrm{N} + 1)$ multiplications and $(3/2\mathrm{N}\log_2 \mathrm{N} +$

TABLE I
COMPARISON OF COMPUTATIONAL COMPLEXITIES OF RADIX-2 DCT-II AND RADIX-2 RFFT ALGORITHMS

N	RADIX-2 DCT–II		RADIX-2 REAL DFT	
	MULTIPLICATIONS	ADDITIONS	MULTIPLICATIONS	ADDITIONS
8	12	29	2	20
16	32	81	12	62
32	80	209	44	174
64	192	513	132	454
128	448	1217	356	1126
256	1024	2817	900	2694
512	2304	6401	2180	6278
1024	5120	14337	5124	14342
2048	11264	31745	11780	32262
4096	24576	69633	26628	71686

TABLE II
COMPARISON OF COMPUTATIONAL COMPLEXITIES OF HBNLMS AND FBNLMS ALGORITHMS

N	FBNLMS		HBNLMS		
	MULTIPLICATIONS	ADDITIONS	MULTIPLICATIONS	ADDITIONS	SAVINGS IN %
8	10	100	25	246	-150
16	60	310	85	618	-41.7
32	220	870	245	1482	-11.4
64	660	2270	645	3450	2.27
128	1780	5630	1605	7866	9.83
256	4500	13470	3845	17658	14.56
512	10900	31390	8965	39162	17.75
1024	25620	71710	20485	86010	20.04
2048	58900	161310	46085	187386	21.76
4096	133140	358430	102400	405498	23.08

$5/2\mathrm{N}-2)$ additions. For the N-point DFT, however, we need $(3/4\mathrm{N}\log_2\mathrm{N}-5/2\mathrm{N}+4)$ multiplications and $(7/4\mathrm{N}\log_2\mathrm{N}-7/2\mathrm{N}+6)$ additions. Thus, if we compare the DCT system used here with the real-DFT system used in the FBNLMS, we find that for $\mathrm{N}\geq 64$, the number of multiplications is lesser for the DCT system. However, in some modern DSP's, additions and multiplications are performed at almost the same speed. For $\mathrm{N}\geq 32\,000$, the total number of computations of the DCT system is lower than the real-DFT system. Thus, for large adaptive filters, the implementation proposed has a distinct advantage with respect to computations. Table II shows the computation counts of the FFT's and the FCT's involved in the FBNLMS and HBNLMS algorithms, respectively. We assume the same computation counts for the inverse transform and the forward transform. The savings in Table II are computed with respect to multiplications, as they are more prominent for small N.

For $\mathrm{N}=32\,768$, using the FBNLMS, we require 3 727 390 additions and 1 433 620 multiplications, for a total of 5 161 010 computations. Using the HBNLMS, we require 3 981 306 additions and 1 064 965 multiplications for a total of 5 046 271 computations. Thus, we can see the inherent advantage of the HBNLMS as N increases.

As N increases asymptotically to infinity, we find that the computational savings are of the order of $(1/4\mathrm{N}\log_2\mathrm{N}/3/4\mathrm{N}\log_2\mathrm{N}=33\%)$ if multiplications are seen to be the primary computation operations and $(1/2\mathrm{N}\log_2\mathrm{N}/5/2\mathrm{N}\log_2\mathrm{N}=20\%)$ when multiplications and additions are seen to be equal contributors to computations.

VI. MEMORY REQUIREMENTS

Computation count is just one of the many parameters that reflect the 'desiredness' of an algorithm. The algorithm would not serve much purpose if the memory requirements of the same were to far exceed that of the original algorithm. The FBNLMS requires three forward transforms and two inverse transforms. The memory requirements of the FBNLMS are the incoming data, $\mathbf{U^c}(\mathrm{n})$ (length N) and $\mathbf{w^c}$ (length M), two buffer registers of length N to store the intermediate results (like forward transforms etc.), a desired data register of length B, and twiddle factors that can be stored in a register of length less than N if we use their complex conjugate symmetry. Unit register locations are needed to store the adaptation constants and the variance estimator. Thus, we would require, in all, an order of $(4\mathrm{N}+\mathrm{M}+\mathrm{B})$ memory units for the FBNLMS.

The HBNLMS, on the other hand, has to incorporate the inherent symmetry of the data sequences if we want to reduce the number of memory units used. We require length N and M registers for the data sequences viz. $\mathbf{U^c}(\mathrm{n})$ and $\mathbf{w^c}$, respectively. Since the 2N-point Hartley transforms are of the form as in (25), it would be easier if we store the two N/2-point sequences, $\mathrm{T}_{\mathrm{N}/2}$ and $\mathrm{U}_{\mathrm{N}/2}$. Four length N buffer registers are needed (in comparison with two in FBNLMS) as we would have to store $\mathbf{H}\ \mathbf{w}_1^{\mathbf{cs}}$, $\mathbf{H}\ \mathbf{w}_2^{\mathbf{cs}}$, $\mathbf{U}_{1\mathrm{H}}^{\mathbf{cs}}(\mathrm{k})$ and their products. The desired data needs a length B register and twiddle factors of the DCT-II require a register of length less than N. Unit memory units for adaptation constants follow the same pattern as in FBNLMS. Thus, the HBNLMS requires an order of $(6\mathrm{N}+\mathrm{M}+\mathrm{B})$ memory units. However, the computational benefits achieved by the HBNLMS overweigh the 2N extra memory units that it needs in the process.

VII. CONCLUSION

In this paper, the HBNLMS has been implemented using the DCT-DST symmetric decomposition. The HBNLMS is an exact equivalent of the FBNLMS and forms a part of the broad class of Transform domain real LMS algorithms [7], [8]. The HBNLMS implemented using the cosine-sine symmetric decomposition, reduces the number of computations (multiplications and additions) by a large amount, especially if the order of the filter is large, thus enabling direct implementation of many audio and video estimation problems efficiently. Many computation reduction techniques that can be employed with the DFT, like higher-radix and split-radix algorithms, pruning [24]–[26], block transforms [27], and slide transforms [28] can be easily extended to the DHT, DCT and DST, thereby resulting in an almost equivalent reduction in complexity of the HBNLMS over the FBNLMS with such techniques. Besides, the memory requirements of the HBNLMS are of the same order as that of the FBNLMS.

Since DCTs and DSTs are used for video coding in the various MPEG standards and have been implemented in special VLSI blocks, these can be exploited to implement the HBNLMS. Recursive implementation of DCTs [22], with parallel processing blocks, can also be used, especially for large filters to reduce computations by a large amount. The stability of the HBNLMS algorithm vis-à-vis rounding off errors can

be studied equivalently by studying the stability of the various DCT algorithms in comparison with the stability of the real FFT algorithms used in FBNLMS.

REFERENCES

[1] B. Widrow, "Adaptive filters," in *Aspects of Network and System Theory*, R. Kalman and N. DeClaris, Eds. New York: Holt, Rinehart, and Winston, 1971, pp. 563–587.

[2] L. J. Griffiths, "A continuously adaptive filter implemented as a lattice structure," in *Proc. ICASSP*, Hartford, Conn., 1977, pp. 683–686.

[3] G. Carayannis, D. Manolakis, and N. Kalouptsidis, "A fast sequential algorithm for least squares filtering and prediction," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, no. 6, pp. 1394–1402, Dec. 1983.

[4] J. Cioffi and T. Kailath, "Fast recursive least squares transversal filters for adaptive filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 304–337, Apr. 1984.

[5] D. Mansour and A. H. Gray Jr., "Unconstrained frequency domain adaptive filter," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, no. 5, pp. 726–734, Oct. 1982.

[6] M. Dentino, J. McCool, and B. Widrow, "Adaptive filtering in the frequency domain," *Proc. IEEE*, vol. 66, no. 12, pp. 1658–1659, Dec. 1978.

[7] S. S. Narayan, A. M. Peterson, and M. J. Narasimha, "Transform domain LMS algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, no. 3, pp. 609–615, June 1983.

[8] R. Merched and A. H. Sayed, "An embedding approach to frequency domain and subband adaptive filtering," *IEEE Trans. Signal Processing*, vol. 48, no. 9, pp. 2607–2619, Sep. 2000.

[9] Z. Wang, "A fast algorithm for the discrete sine transform implemented by the fast cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, no. 5, pp. 814–815, Oct. 1982.

[10] T. W. Wong and C. P. Kwong, "Adaptive filtering using Hartley transform and overlap-save methods," *IEEE Trans. Signal Processing*, vol. 39, no. 7, pp. 1708–1711, July 1991.

[11] A. O. Ogunfunmi and A. M. Peterson, "On the implementation of the frequency-domain LMS adaptive filter," *IEEE Trans. Circuits Syst.*, vol. CAS-39, no. 5, pp. 318–322, May 1992.

[12] G. P. M. Egelmeers, "Real Time Realization Concepts of Large Adaptive Filters," Ph.D. Dissertation, Technical Univ., Eindhoven, The Netherlands, Nov. 1995.

[13] P. C. W. Sommen, "Adaptive Filtering Methods," Ph.D. Dissertation, Eindhoven Univ. Technol., Eindhoven, The Netherlands, June 1992.

[14] P. J. Davis, *Circulant Matrices*. New York: Wiley, 1979.

[15] R. N. Bracewell, "Discrete Hartley transform," *J. Opt. Amer.*, vol. 73, no. 12, pp. 1832–1835, Dec. 1983.

[16] H. V. Sorenson, D. L. Jones, C. S. Burrus, and M. T. Heideman, "On computing the discrete Hartley transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, no. 4, pp. 1231–1238, Oct. 1985.

[17] H. Guo, G. A. Sitton, and C. S. Burrus, "The quick Fourier transform: an FFT based on symmetries," *IEEE Trans. Signal Processing*, vol. 46, no. 2, pp. 335–341, Feb. 1998.

[18] H. S. Malvar, "Fast computations of discrete cosine and the discrete Hartley transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, no. 10, pp. 1484–1485, Oct. 1985.

[19] H. Kitajima, "A symmetric cosine transform," *IEEE Trans. Comput.*, vol. C-29, no. 4, pp. 317–323, Apr. 1980.

[20] A. K. Jain, "Fast Karhunen-Loeve transform for a class of stochastic processes," *IEEE Trans. Commun.*, vol. COM-24, no. 9, pp. 1023–1029, Sep. 1976.

[21] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, no. 6, pp. 1243–1245, Dec. 1984.

[22] H. S. Hou, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, no. 10, pp. 1455–1461, Oct. 1987.

[23] H. V. Sorenson, D. L. Jones, M. T. Heideman, and C. S. Burrus, "Real-valued fast Fourier transform algorithms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, no. 6, pp. 849–863, June 1987.

[24] H. V. Sorenson and C. S. Burrus, "Efficient computation of the DFT with only a subset of input or output points," *IEEE Trans. Signal Processing*, vol. 41, no. 3, pp. 1184–1199, Mar. 1993.

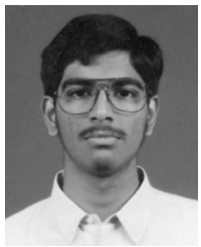
[25] D. P. Skinner, "Pruning the decimation-in-time FFT algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, no. 4, pp. 193–194, Apr. 1976.

[26] S. B. Narayanan and K. M. M. Prabhu, "Fast Hartley transform pruning," *IEEE Trans. Signal Processing*, vol. 39, no. 1, pp. 230–233, Jan. 1991.

[27] G. P. M. Egelmeers and P. C. W. Sommen, "Recursive computation of block-FFT's," in *Proc. ProRISC/IEEE Symp. Circuits, Systems Signal Processing*, Mierlo, The Netherlands, Mar. 1995, pp. 59–66.

[28] T. Springer, "Sliding FFT computes frequency spectra in real time," *EDN*, pp. 161–170, Sep. 1988.

[29] S. Haykin, *Adaptive Filter Theory*, Third ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.



Vasanthan Raghavan (S'01) received the B.Tech degree in electrical engineering from the Indian Institute of Technology, Madras, India, in 2001 and the M.S. degree in electrical and computer engineering from the University of Wisconsin, Madison, in May, 2004, where he is now working toward the Ph.D. degree.

His primary research interests are in wireless communications, communication theory, adaptive signal processing, and information theory.

K. M. M. Prabhu (SM'93) received the B.Sc. (Engg.) degree in electronics and communication engineering, the M.Sc. (Engg.) degree in applied electronics engineering, and the Ph.D. degree in digital signal processing in 1971, 1973 and 1981, respectively.

He joined the Department of Electrical Engineering, Indian Institute of Technology (IIT), Madras, in 1981, where he is currently a full Professor. While at IIT, he has been a Principal Investigator for many projects funded by the Defence Research and Development Organization. He has published more than 100 papers in international journals and international conference proceedings. Under his supervision, 17 M. S. (by research) and Ph.D. students have completed their degrees. He has conducted ten short-term courses in the area of DSP for the benefit of engineers and scientists from various organizations. He spent his sabbatical, from August 1999 to August 2000, with the Signal Processing Systems Group, Eindhoven University of Technology, Eindhoven, The Netherlands. His research interests include DSP algorithms and their implementations, digital filter structures, time-frequency signal analysis, and wavelet- based signal processing.

Dr. Prabhu has been a reviewer for a number of IEEE, IEE, and other international journals.

Piet C. W. Sommen (SM'03) received the Ingenieur degree in electrical engineering from Delft University of Technology, Delft, The Netherlands, in 1981 and the Ph.D. degree from Eindhoven University of Technology (EUT), Eindhoven, The Netherlands, in 1992.

From 1981 to 1989, he was with Philips Research Laboratories, Eindhoven, and since 1989, he has been with the faculty of Electrical Engineering, EUT, where he is currently an Associate Professor. He is involved in internal and external courses, all dealing with different basic and advanced signal processing topics. His main field of research is in adaptive array signal processing, with applications in acoustic communication systems. He is Editor of EURASIP's *Journal of Applied Signal Processing*, for which he served as Guest Editor of a special issue on "Signal Processing for Acoustic Communication Systems" in 2003.

Dr. Sommen is a member of the ProRISC board, Vice President of the IEEE Benelux Signal Processing Chapter, and Officer of the Administrative Committee of EURASIP.