Matrix Expansions for Computing the Discrete Hartley Transform

R. C. de Oliveira
Amazon State University, UEA
Manaus, Amazonas, Brazil
rcorrea.oliveira@gmail.com

R.M. Campello de Souza and H.M. de Oliveira
Federal University of Pernambuco, UFPE
Recife, Pernambuco, Brazil
{ricardo,hno}@ufpe.br

***Abstract*— A new fast algorithm for computing the discrete Hartley transform (DHT) is presented, which is based on the expansion of the transform matrix. The algorithm presents a better performance, in terms of multiplicative complexity, than previously known fast Hartley transform algorithms. A detailed description of the computation of DHTs with blocklengths 8 and 12 is shown. The algorithm is very attractive for blocklengths *N* ≥ 128.**

***Index Terms*— Discrete Hartley transform, fast Hartley transform.**

I. INTRODUCTION

Regarded, for many years, essentially as a technique for computing Fourier transforms, the Hartley transforms, continuous and discrete, became very important tools with many applications in several fields of Engineering [1]. In particular, the discrete Hartley transform pair is defined, for a length-N sequence $h(n), 0 \le n \le N-1$, by Equations (1) and (2) [2],

$$H(k) := \sum_{n=0}^{N-1} h(n) cas(\tfrac{2\pi}{N} kn), \quad 0 \le k \le N\text{-}1, \qquad (1)$$

$$h(n) = \tfrac{1}{N} \sum_{k=0}^{N-1} H(k) cas(\tfrac{2\pi}{N} kn), \; 0 \le n \le N\text{-}1, \qquad (2)$$

where $cas(.)$ denotes the *cosine and sine* function defined as $cas(i) := \cos(i) + \sin(i)$. The DHT, as its continuous counterpart, is real and the symmetry of the transform pair is a valuable feature for its implementation.

With the advent of VLSI and the development of the digital signal processor (DSP) to implement signal processing techniques, discrete transforms, such as the discrete Fourier transform (DFT) and the DHT, became attractive tools for performing spectrum evaluation. The cost reduction of DSPs and the astonishing capacity achieved by up to date processors has made real-time applications feasible for several types of signals. In this scenario, the successful application of transform techniques is mainly due to the existence of the so-called fast transform algorithms.

Over the years, fast algorithms, in terms of multiplicative complexity, were introduced for computing the DHT [3-7]. This paper proposes a new fast algorithm for computing DHTs of sequences of lengths $N \equiv 0 \,(\text{mod}\, 4)$. The paper is organized as follows. In Section II the DHT transform matrix is expressed an expansion involving matrices. In Section III the new fast Hartley transform algorithm (FHT) introduced in this paper is described and, to illustrate the technique, complete examples of the algorithm, to compute blocklength $N = 8$ and 12 DHTs, are shown in Sections IV and V, respectively. The paper closes with some concluding remarks on Section VI.

II. EXPANDING THE DHT TRANSFORM MATRIX

The first step towards the FHT proposed in this paper is to rewrite Equation (1) in matrix form,

$$\begin{bmatrix} H_0 \\ H_1 \\ H_2 \\ \vdots \\ H_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & cas(1) & cas(2) & \dots & cas(N-1) \\ 1 & cas(2) & cas(4) & \dots & cas(2.(N-1)) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & cas(N-1) & cas(2(N-1)) & \dots & cas((N-1).(N-1)) \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_{N-1} \end{bmatrix},$$

or $H(k)$=[DHT] $h(n)$, where, for the sake of simplicity, we denote $cas(\frac{2\pi}{N} kn)$ by $cas(kn)$. Considering that, for l and r integers, $cas(\frac{2\pi}{N} l) = cas(\frac{2\pi}{N}(l + rN))$, there are only N distinct arguments of $cas(.)$ in the transform matrix. For every even blocklength, $N = 2m$, there exists an argument ki yielding the eigenvalues of the DHT, i.e., {1, -1}. These terms do not contribute to the multiplicative complexity, because

$$cas(0) = cas(N/4) = 1$$

and

$$cas(N/2) = cas(3N/4) = -1.$$

The arguments of $cas(.)$ in the above expression generate a set of two points that lie on the real axis. This fact is associated with the class

C_0:={0, N/2, N/4, 3N/4}.

The set of distinct arguments of $cas(.)$, $Z_N = \{0,1,2,..., N-1\}$, is then partitioned into $N/4$ disjoint classes $C_m := \{i \in Z_N \mid 4i \equiv 4m \, (\bmod\, N)\}$, where $m = 0,1,2,..., \frac{N}{4}-1$.

Proposition 1. The classes C_m induce a partition of Z_N.

Proof. Suppose that there exists a pair $m \neq m'$ such that $C_m \cap C_{m'} \neq \varnothing$. Therefore, there is a common element $i \in C_m$ and $i \in C_{m'}$ such that $4i \equiv 4m \, (\bmod\, N)$ and $4i \equiv 4m' (\bmod\, N)$. Therefore $4m \equiv 4m' (\bmod\, N)$, which is the same as $m \equiv m' (\bmod\, N/4)$, a contradiction. The cardinality of a set C_m for each m is $|C_m| = 4$. There are N/4 disjoint classes, therefore $\left| \bigcup_{m=0}^{(N/4)-1} C_m \right| = 4.(N/4) = N$ and the proof is complete. ■

Let us introduce the matrix of arguments of $cas(.)$ in Equation (1), an $N \times N$ matrix $A := (a_{kn})$, where $a_{kn} = (kn (\bmod\, N))$, and an operator B_l over $N \times N$ matrix, for each l = 0,1,2,…, N-1, which yields an $N \times N$ binary matrix whose elements are $\left(\delta_{l, m_{k,n}}\right)$, where δ is the Kronecker symbol.

Associated with each class C_m we define the matrix M_m as

$$M_m := \sum_{l \in C_m} \mathrm{sgn}(cas(l)) \mathrm{B}_l(A), \qquad (3)$$

where $\mathrm{sgn}(x)$ returns the sign of x. Thus, for instance, $m = 0$ corresponds to the additive part of the DHT transform matrix,

$$M_0 = \mathrm{B}_0(A) - \mathrm{B}_{N/2}(A) + \mathrm{B}_{N/4}(A) - \mathrm{B}_{3N/4}(A).$$

The following proposition shows the symmetries of the $cas(.)$ function, which are important in the construction of the fast algorithm described in this paper.

Proposition 2. i) $cas(m + \frac{N}{2}) = -cas(m)$.

ii) $cas(m + \frac{N}{4}) = cas(N - m)$.

Proof:

i) $cas(m + \frac{N}{2}) := cas(\frac{2\pi}{N}(m + \frac{N}{2})) = cas(\frac{2\pi m}{N} + \pi) =$
$= \cos(\frac{2\pi m}{N} + \pi) + \sin(\frac{2\pi m}{N} + \pi) = -cas(m)$.

ii) $cas(m + \frac{N}{4}) := cas(\frac{2\pi}{N}(m + \frac{N}{4})) = cas(\frac{2\pi m}{N} + \frac{\pi}{2}) =$
$= \cos(\frac{2\pi m}{N} + \frac{\pi}{2}) + \sin(\frac{2\pi m}{N} + \frac{\pi}{2}) = cas(N - m)$.

Proposition 2 implies that, from a given value $cas(m)$, four different values of $cas(.)$ can be obtained. Therefore, only $N/4$ terms $cas(m)$ are required to compute $H(k)$.

From the matrices M_m, $m = 0,1,2,..., \frac{N}{4}-1$, the DHT transform matrix can be expressed by following expansions:

i) For N an even multiple of 4,

$$[DHT] = \sum_{m=0}^{(N/4)-1} M_m cas(m). \qquad (4)$$

ii) For N an odd multiple of 4,

$$[DHT] = \sum_{m=0}^{(N/4)-1} [C_c(M_m) + C_s(M_m)] cas(m), \qquad (5)$$

where the matrices $C_c(M_m)$ and $C_s(M_m)$ are defined as

$$C_c(M_m) = \mathrm{B}_m(A) - \mathrm{B}_{m+\frac{N}{2}}(A),$$

$$C_s(M_m) = \mathrm{B}_{m+\frac{N}{4}}(A) - \mathrm{B}_{m+\frac{3N}{4}}(A).$$

The multiplicative complexity of the algorithms (Equations (4) and (5)) can be computed, respectively, by Equation (6) and (7).

$$\sum_{m=0}^{(N/4)-1} rank(M_m) \qquad (6)$$

$$\sum_{m=0}^{(N/4)-1} rank(C_c(M_m)). \qquad (7)$$

The procedure to compute the DHT can be summarized as follows:

1. Compute the matrix of arguments (A);
2. Compute the matrix of classes;
3. Repeat for all classes:
   3.1. Compute the binary matrix given Equation (4)/(5);
   3.2. Compute the binary matrix in standard echelon form (SEF), referred here as *rref* (*r*ow-*r*educed *e*chelon *f*orm).
   3.3. Compute the floating point multiplications in the SEF binary matrix;
   3.4. Compute the additions to calculate the DHT components.

III. AN FHT OF BLOCKLENGTH N =8

For N=8, we start by gathering the arguments in the class {0, 4, 2, 6}, which are not associated with multiplications. This corresponds to the set C_0. The matrix A with the arguments of the terms in the DHT matrix is

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 2 & 4 & 6 & 0 & 2 & 4 & 6 \\ 0 & 3 & 6 & 1 & 4 & 7 & 2 & 5 \\ 0 & 4 & 0 & 4 & 0 & 4 & 0 & 4 \\ 0 & 5 & 2 & 7 & 4 & 1 & 6 & 3 \\ 0 & 6 & 4 & 2 & 0 & 6 & 4 & 2 \\ 0 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \end{bmatrix}.$$

There are only N/4=2 classes, namely, C_0 = (0, 4, 2, 6) and C_1 = (1, 5).

In this particular case, the greatest index is (N/4)-1=1. Indeed, C_0 and C_1 are a partition of {0,1,2,…,7}, as expected. We observe that the class C_1 has only two values of arguments, because *cas(3)=cas(7)=0*.

It is straightforward to observe that given C_0, the elements of other classes can be derived as follows,

for m = 0 to (N/4)-1 do
for i = 0 to 3 do
If i < 2 then

$$C_{i,m} = mod(C_0 + m, N)$$
$$\text{If } i >= 2 \text{ then}$$
$$C_{i,m} = mod(C_0 - m, N).$$

The operations involving products by the eigenvalues (elements of C_0) must not be considered as floating-point multiplications.

The matrices of interest in the algorithm are:

❶ $M_0 = B_0(A) - B_4(A) + B_2(A) - B_6(A).$

The additive matrix M_0 is

$$M_0 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \end{bmatrix},$$

which furnishes $rank(M_0)$=6. In SEF

$$rref(M_0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

❷ $M_1 = B_1(A) - B_5(A).$ Then,

$$M_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

and $rank(M_1)$=2, the SEF of which is

$$rref(M_1) := \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}$$

In order to evaluate the multiplicative complexity of the FHT of blocklength 8, we determine $rank(M_m)$.

The two preaddition matrices associated with the multiplicative branches of the algorithm are:

$$rref(M_1) := \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}$$

Figure 1 shows a block diagram for implementing the FHT algorithm. The total multiplicative complexity is two floating-point multiplications, which meets the Heideman lower bound [8]. From Equation (4), the DHT transform matrix expansion is $[DHT] = M_0 + M_1 cas(1)$, i.e.,

$$[DHT] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} cas(1).$$

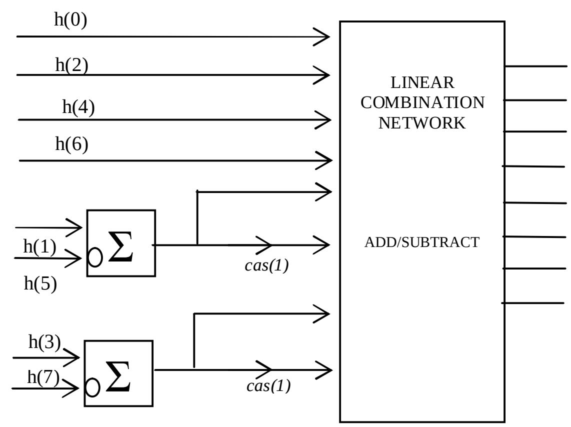

Figure 1. Scheme for the computation of a DHT with blocklength N=8. The small circles into the Σ-box denote subtraction.

The pseudo-code presented below describes the calculations, showing the floating-point multiplications. Moreover, we can see that the linear combination of adders involves only multiplications by 1 and -1, which are trivial.

Computing the array which executes floating-point multiplications:

```
For i = 0 to (N-1)
        VM(i) = 0;
For i = 0 to (N-1) do
        For j = 0 to (N-1) do
                IF MRed(i,j) = 1 then
                        For j = j to (N-1) do
                                Tmp = Tmp + MRed(i,j);
                        VM(i) = Tmp * Cas(m)
                End
        End
End
```

Computing the array for the final additions:

```
For i = 0 to (N-1) do
        For j = 0 to (N – 1) do
                MC(j,i) = MC(j,i) + M(.)(j,i) * VM(i)
```

Final additions for calculating the DHT:

```
For i = 0 to (N – 1) do
        For j = 0 to (N – 1) do
                H(i) = H(i) + MC(i,j)
```

IV. AN FHT OF BLOCKLENGTH $N = 12$

For $N = 12$, there are $\frac{N}{4} = 3$ classes, namely,

$C_0 = (0, 3, 6, 9)$,
$C_1 = (1, 4, 7, 10)$,
$C_2 = (11, 2, 5, 8)$.

The matrix A with the arguments of the terms in the DHT matrix is

$$A = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&2&3&4&5&6&7&8&9&10&11\\ 0&2&4&6&8&10&0&2&4&6&8&10\\ 0&3&6&9&0&3&6&9&0&3&6&9\\ 0&4&8&0&4&8&0&4&8&0&4&8\\ 0&5&10&3&8&1&6&11&4&9&2&7\\ 0&6&0&6&0&6&0&6&0&6&0&6\\ 0&7&2&9&4&11&6&1&8&3&10&5\\ 0&8&4&0&8&4&0&8&4&0&8&4\\ 0&9&6&3&0&9&6&3&0&9&6&3\\ 0&10&8&6&4&2&0&10&8&6&4&2\\ 0&11&10&9&8&7&6&5&4&3&2&1 \end{bmatrix}$$

The relevant matrices are

❶ $M_0 = B_0(A) - B_6(A) + B_3(A) - B_9(A)$.

This additive matrix M_0 is then separated into its C_c and C_s components,

$$C_c(M_0) = B_0(A) - B_6(A),$$

$$C_C(M_0) = \begin{bmatrix} 1&1&1&1&1&1&1&1&1&1&1&1\\ 1&0&0&0&0&0&-1&0&0&0&0&0\\ 1&0&0&-1&0&0&1&0&0&-1&0&0\\ 1&0&-1&0&1&0&-1&0&1&0&-1&0\\ 1&0&0&1&0&0&1&0&0&1&0&0\\ 1&0&0&0&0&0&-1&0&0&0&0&0\\ 1&-1&1&-1&1&-1&1&-1&1&-1&1&-1\\ 1&0&0&0&0&0&-1&0&0&0&0&0\\ 1&0&0&1&0&0&1&0&0&1&0&0\\ 1&0&-1&0&1&0&-1&0&1&0&-1&0\\ 1&0&0&-1&0&0&1&0&0&-1&0&0\\ 1&0&0&0&0&0&-1&0&0&0&0&0 \end{bmatrix}$$

which has $rank(C_c(M_0)) = 6$ and

$$C_s(M_0) = B_3(A) - B_9(A),$$

$$C_S(M_0) = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&-1&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&-1&0&1&0&-1&0&1&0&-1\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&-1&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&-1&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&-1&0&1&0&-1&0&1&0&-1&0&1\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&-1&0&0&0&0&0&1&0&0 \end{bmatrix}$$

which has $rank(C_s(M_0)) = 2$.

❷ $M_1 = B_1(A) - B_7(A) + B_2(A) - B_8(A)$,

$$C_c(M_1) = B_1(A) - B_7(A),$$

$$C_C(M_1) = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&-1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&-1\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&-1&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&-1&0&0&0&0&0&1 \end{bmatrix}$$

and $rank(C_c(M_1)) = 2$.

$$C_s(M_1) = B_2(A) - B_8(A),$$

$$C_S(M_1) = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&-1&0\\ 0&0&1&0&0&-1&0&0&1&0&0&-1\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&-1&0&0&-1&0&0&-1&0&0&-1\\ 0&0&0&0&-1&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&-1&0&0&0\\ 0&-1&0&0&-1&0&0&-1&0&0&-1&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&-1&0&0&1&0&0&-1&0&0&1\\ 0&0&0&0&-1&0&0&0&0&0&1&0 \end{bmatrix}$$

and $rank(C_S(M_1) = 6$.

❸ $M_2 = B_{11}(A) - B_5(A) + B_4(A) - B_{10}(A)$.

$$C_c(M_2) = B_{11}(A) - B_5(A),$$

$$C_C(M_2) = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&-1&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&-1&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&-1\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&-1&0&0&0&0 \end{bmatrix}$$

so $rank(C_C(M_2)) = 2$. The SEF of this matrix is the same as that of $C_C(M_1)$.

$$C_s(M_2) = B_4(A) - B_{10}(A),$$

$$C_S(M_2)=\begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&-1&0\\ 0&0&1&0&0&-1&0&0&-1&0&0&-1\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&1&0&0&1&0&0&1&0\\ 0&0&-1&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&-1&0\\ 0&0&1&0&0&1&0&0&1&0&0&1\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&-1&0&0&1&0&0&-1&0&0&1&0\\ 0&0&-1&0&0&0&0&0&1&0&0&0 \end{bmatrix}$$

which has rank($C_S(M_2)$)$=6$. The SEF of this matrix is the same as that of $C_S(M_1)$. From Equation (5), the DHT transform matrix expansion is

$$[DHT]=M_0+M_1 cas(1)+M_2 cas(2).$$

Table 1. Complexity of the power series-based FHT algorithm in terms of the number of *real* non-trivial floating-point multiplications, compared to the radix-2 and radix-4 FHT algorithms.

N	Radix-2	Radix-4	Split-Radix	Heideman lower bound $\mu_r(N)$	Matrix Expansion FHT
8	4	-	2	2	2
16	20	14	12	10	12
32	68	-	42	32	40
64	196	142	124	84	96
128	516	-	330	198	256
256	1284	942	828	438	640
512	3076	-	1994	932	1408
1024	7172	5294	4668	1936	3328
2048	16388	-	10698	3962	7680
4096	36868	27310	24124	8034	16384

Complexity results for the matrix expansion FHT are shown in Table 1, in comparison with Heideman lower bound and standard radix-2, radix-4 and Split-Radix FHT algorithms [4]. Figure 2 shows the multiplicative complexity of the algorithms in Table 1 as a function of *N*. We must emphasize that the complexity for blocklengths *N* = (8, 32, 128, 512, 2048), for the Radix-4 algorithm, are not reported in [4].

V. CONCLUSIONS

A new fast transform algorithm for the discrete Hartley transform of length $N\equiv 0 \pmod 4$ was proposed, which is based upon a new technique to construct a matrix expansion of the transform matrix. The procedure takes advantage of the symmetries of the expansion matrices to reduce the computational load for computing the discrete Hartley spectrum. Detailed examples to illustrate the technique were presented for *N* = 8 and 12, but the entire procedure is systematic. The fast Hartley transform presented here is also easy to implement using DSP or low-cost high-speed Integrated Circuits. The algorithm presents a better performance, in terms of multiplicative complexity, than standard radix-2, radix-4 and Split-Radix Cooley-Tukey FHT algorithms.

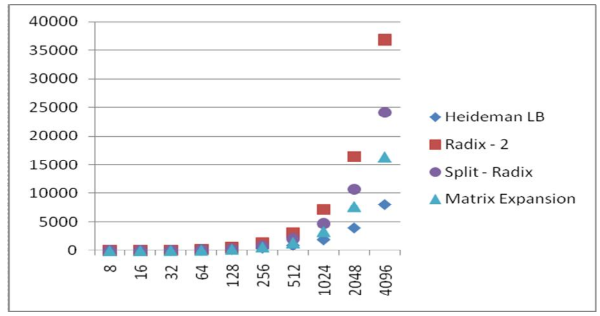

Figure 2. Multiplicative complexity for the radix-2, Split-Radix and matrix expansion FHT algorithms and Heideman lower bound, as a function of transform blocklength.

ACKNOWLEDGEMENTS

The first author was partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq).

REFERENCES

[1] Olejniczak, K. J. and Heydt, G. T. (eds.), Special Section on the Hartley Transform, *Proceedings of the IEEE*, vol. 82, No. 3, pp. 372-447, March 1994.

[2] Bracewell, R. N., "Discrete Hartley transform," *J. Opt. Soc. Am.* vol. 73, No. 12, pp. 1832–1835, 1983.

[3] Bracewell, R.N., The Fast Hartley Transform, *Proceedings of the* IEEE, vol. 72, No. 8, pp. 1010-1018, August 1984.

[4] Sorensen, S. V., Jones, D. L., Burrus, C. S. and Heideman, M. T., On Computing the Discrete Hartley Transform, *IEEE Trans. Acoust., Speech and Signal Processing*, vol. ASSP 33, No. 4, pp. 1231-1238, 1985.

[5] de Oliveira, H.M., Cintra, R.J.S., Campello de Souza, R. M., Multilevel Hadamard Decomposition of Discrete Hartley Transforms, XVIII *Brazilian Telecommunications Symposium*, SBrT'00, Gramado, RS, September, 2000.

[6] de Oliveira, H.M., Campello de Souza, R. M., A Fast Algorithm for Computing the Hartley/Fourier Spectrum, *Proceeedings of the Brazilian Academy of Science*, Rio de Janeiro, vol. 73, pp.468-468, 2001.

[7] Liu, J. G. et alli, Moment-based fast discrete Hartley transform, Signal Processing. vol. 83, No. 8, pp. 1749-1757, August 2003.

[8] Heideman, M.T. *Multiplicative Complexity, Convolution and the DFT*, Springer-Verlag, 1988.