TR-94-001

# An Efficient Algorithm for the 2-D Discrete Cosine Transform[1]

PeiZong Lee[2], Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C.
Gau-Shin Liu, Dept. of Electrical Engineering, Chung-Cheng Univ., Chia-Yi, Taiwan, R.O.C.

# An Efficient Algorithm for the 2-D Discrete Cosine Transform[1]

PeiZong Lee[2], Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C.

Gau-Shin Liu, Dept. of Electrical Engineering, Chung-Cheng Univ., Chia-Yi, Taiwan, R.O.C.

**Abstract**

In this paper, we present a new algorithm for computing the 2-D discrete cosine transform (DCT). First, we arrange the input data matrix in a certain order. Second, we formulate the output data matrix to a block-structure matrix multiplication, and show that the block-structure matrix has the same properties as the 1-D DCT kernel matrix. Therefore, the 2-D DCT problem is reduced to the 1-D DCT problem. We thus can provide a procedure for computing the 2-D DCT which is similar to that of the 1-D DCT.

If we are only interested in the number of scalar multiplications used in an algorithm, then the proposed algorithm requires $\frac{1}{2}N^2 \log N$ scalar multiplications for computing an $(N \times N)$-point 2-D DCT. If compared to the conventional row-column approach which requires $N^2 \log N$ scalar multiplications, or to some recursive algorithms which require $\frac{3}{4}N^2 \log N$ scalar multiplications, or to some algorithms based on polynomial transform techniques which require $\frac{1}{2}N^2 \log N$ scalar multiplications, our algorithm is one of the best. The methodology, which deals with general $(N \times N)$-point 2-D DCT's, will be illustrated by designing three lower order examples.

**Index Terms:** 2-dimensional discrete cosine transform, recursive algorithm, signal-flow graph.

*** A preliminary version of this technical report is accepted to be presented at the **IEEE Data Compression Conference**, Snowbird, Utah, U.S.A., March 29–31, 1994.

[1]This work was partially supported by the NSC under Grant NSC 82-0408-E-001-016.

[2]PeiZong Lee: leepe@iis.sinica.edu.tw, TEL: +886 (2) 788-3799, FAX: +886 (2) 782-4814.

# 1 Introduction

This paper is concerned with designing a fast algorithm for computing the 2-D discrete cosine transform (DCT). Because the DCT performs much like the theoretically optimal Karhunen-Loeve transform for the first-order Markov stationary random data, the DCT has found wide applications in speech and image processing, as well as telecommunication signal processing for the purpose of data compression, feature extraction, image reconstruction, and filtering. As these applications are key techniques used in multi-media research, DCT algorithms thus play an important role in many multi-media applications. A complete survey of DCT algorithms can be seen elsewhere [11].

For the fast computation of the 2-D DCT, the conventional approach is the row-column method, which requires evaluating $2N$ sets of $N$-point 1-D DCT's for an $(N \times N)$-point 2-D DCT. Kamangar and Rao [8], who arranged the 2-D input data and output data into 1-D arrays in lexicographical order, wrote the needed 2-D transform coefficients as the Kronecker product of the two 1-D DCT coefficient matrices and yielded the sparse matrix factorization for that 2-D coefficient matrix. Haque [6], Chan and Ho [1], and Lee and Huang [10] derived 2-D DCT recursive algorithms based on Lee's [9], Hou's [7], and Lee and Huang's [10] 1-D DCT algorithms, respectively.

Vetterli mapped the 2-D DCT into a 2-D DFT plus a number of rotations, and the 2-D DFT was then computed by polynomial transform techniques [12]. His algorithm, while achieving substantial computational savings, had a rather involved structure. Duhamel and Guillemot provided a direct 2-D DCT algorithm based on polynomial transform techniques [5]. They showed that the resulting signal-flow graphs were simple and repetitive; however, they also admitted that its derivation was complicated and it was fairly difficult to obtain a closed-form expression for the arithmetic complexity of such an algorithm.

Cho and Lee [2] [3] claimed a fast 2-D DCT algorithm, in which an $(N \times N)$-point 2-D DCT could be obtained by computing $N$ sets of $N$-point 1-D DCT's as well as some additions. That is, the number of multiplications required for the proposed algorithm is the same as that required for computing $N$ sets of $N$-point 1-D DCT's. So the complexity of their algorithm is greatly reduced. However, according to their comments [4], the signal-flow graph for the post-addition

stage seems very complicated and the order of the output index is seemingly irregular, because the post-addition stage was not based on the mathematical expressions. Consequently, derivation of the signal-flow graph becomes complicated as the transform size increases.

Recently, Cho, Yun, and Lee elaborated their previous work and presented systematic expressions for the post-addition stage [4]. The structure of the signal-flow graph shows a regularity similar to that of Hou's 1-D DCT algorithm [7]. However, their algorithm still has a minor complication. First, before performing 1-D DCT's, they need to calculate a pre-addition stage, in which the plus operations and the minus operations cross each other, and therefore, the pre-addition stage is seemingly irregular. Second, they divide the post-addition stage into two parts and solve them using two methods independently. They also did not provide a closed-form expression for the arithmetic complexity of their algorithm.

It is our goal in this paper to develop a fast and simple 2-D DCT algorithm. We will first arrange the input data matrix in a certain order. Then, we formulate the output data matrix to a block-structure matrix multiplication, and show that the block-structure matrix has the same properties as the 1-D DCT kernel matrix. Hence, the 2-D DCT problem is reduced to the 1-D DCT problem. We thus can provide a procedure for computing the 2-D DCT which is similar to that of the 1-D DCT.

The rest of this paper is organized as follows. In Section 2, we discover intrinsic properties of the 2-D DCT. In Section 3, we provide a new and fast 2-D DCT algorithm. In Section 4, we present three lower order examples. In Section 5, we show that the 2-D DST can be computed by the 2-D DCT. Finally, some concluding remarks are given in Section 6.

## 2 Properties of the 2-D DCT

For a given input data matrix $\mathbf{x}_{N\times N} = [x_{i,j}]$, $0 \le i,\ j \le N-1$, the 2-D DCT output matrix $\mathbf{y}_{N\times N} = [y_{m,n}]$, $0 \le m,\ n \le N-1$, which is defined by

$$y_{m,n} = \frac{2}{N}\,\epsilon(m)\,\epsilon(n) \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_{i,j}\,\cos\left(\frac{(2i+1)m}{2N}\pi\right)\cos\left(\frac{(2j+1)n}{2N}\pi\right), \qquad (1)$$

where

$$\epsilon(k) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } k = 0 \\ 1, & \text{for } 1 \leq k \leq N-1 \end{cases}$$

and $k = m$ or $n$.

In this paper, we want to derive a recursive formula for the 2-D DCT. For convenience, we assume throughout this paper that $N$ has a value of 2 to a power and $N \geq 2$. We will remove $\epsilon(m)$, $\epsilon(n)$, and also the normalization factor $\frac{2}{N}$ in Equation (1), since they can be done in a separate step. Therefore, from now on, we deal with a simplified version of Equation (1):

$$y_{m,n} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{i,j} \cos(\frac{(2i+1)m}{2N}\pi) \cos(\frac{(2j+1)n}{2N}\pi). \tag{2}$$

Based on the trigonometric identity: $\cos A \cos B = \frac{1}{2}[\cos(A+B) + \cos(A-B)]$,

$$y_{m,n} = \frac{1}{2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x_{i,j} [\cos(\frac{(2i+1)m + (2j+1)n}{2N}\pi) + \cos(\frac{(2i+1)m - (2j+1)n}{2N}\pi)]. \tag{3}$$

We now study the relationship between $(2i+1)$ and $(2j+1)$, for $0 \leq i, j \leq N-1$. For every pair of $(i, j)$, there exists an unique $p$, such that exactly one of the following conditions holds:

$$a: \quad (2j+1) = p(2i+1) - 2Nq_a \tag{4}$$

$$b: \quad (2j+1) = 2Nq_b - p(2i+1), \tag{5}$$

where $p$ is an odd integer, $p \in W = \{1, 3, 5, \cdots, N-1\}$, $q_a$ and $q_b$ are integers depending on $p$ and $i$. Hence, $j$ is a function of $p$ and $i$ depending on Equations (4) and (5).

We now further elaborate Equations (4) and (5). We differentiate the cases when $q_a$ or $q_b$ is even and when $q_a$ or $q_b$ is odd. In (4), if $q_a$ is even, we write the equation:

$$(2j_{(p,a,i,e)} + 1) = p(2i+1) - 2Nq_{(p,a,i,e)}; \tag{6}$$

if $q_a$ is odd, we write the equation:

$$(2j_{(p,a,i,o)} + 1) = p(2i+1) - 2Nq_{(p,a,i,o)}. \tag{7}$$

In (5), if $q_b$ is even, we write the equation:

$$(2j_{(p,b,i,e)} + 1) = 2Nq_{(p,b,i,e)} - p(2i+1); \tag{8}$$

if $q_b$ is odd, we write the equation:

$$(2j_{(p,b,i,o)} + 1) = 2Nq_{(p,b,i,o)} - p(2i + 1). \tag{9}$$

Note that, for every pair of $(p, i)$, $q_b = q_a + 1$. Therefore, if $q_a$ is even then $q_b$ is odd; if $q_a$ is odd then $q_b$ is even. Hence, either both Equations (6) and (9) hold and both Equations (7) and (8) fail, or both Equations (6) and (9) fail and both Equations (7) and (8) hold.

We then replace $(2j + 1)$ in (3) with the equations in (6)–(9), Equation (3) becomes

$$y_{m,n} = \tfrac{1}{2}\textstyle\sum_{p\in W}\sum_{i=0}^{N-1}[(x_{i,j_{(p,a,i,e)}} + x_{i,j_{(p,b,i,e)}}) + (-1)^n(x_{i,j_{(p,a,i,o)}} + x_{i,j_{(p,b,i,o)}})]$$
$$\times [\cos(\tfrac{(2i+1)(m+np)}{2N}\pi) + \cos(\tfrac{(2i+1)(m-np)}{2N}\pi)],$$

where $x_{i,j_{(p,a,i,e)}} = x_{i,j}$ if $j$ satisfies (6), otherwise $x_{i,j_{(p,a,i,e)}} = 0$. Similarly, $x_{i,j_{(p,a,i,o)}} = x_{i,j}$ if $j$ satisfies (7), otherwise $x_{i,j_{(p,a,i,o)}} = 0$; $x_{i,j_{(p,b,i,e)}} = x_{i,j}$ if $j$ satisfies (8), otherwise $x_{i,j_{(p,b,i,e)}} = 0$; and $x_{i,j_{(p,b,i,o)}} = x_{i,j}$ if $j$ satisfies (9), otherwise $x_{i,j_{(p,b,i,o)}} = 0$. Let

$$\begin{aligned} A_{p,i} &= (x_{i,j_{(p,a,i,e)}} + x_{i,j_{(p,b,i,e)}}) \\ B_{p,i} &= (x_{i,j_{(p,a,i,o)}} + x_{i,j_{(p,b,i,o)}}). \end{aligned}$$

Then, if $n$ is even,

$$y_{m,n} = \frac{1}{2}\sum_{p\in W}\sum_{i=0}^{N-1}(A_{p,i} + B_{p,i})\,[\cos(\frac{(2i+1)(m+np)}{2N}\pi) + \cos(\frac{(2i+1)(m-np)}{2N}\pi)]; \tag{10}$$

if $n$ is odd,

$$y_{m,n} = \frac{1}{2}\sum_{p\in W}\sum_{i=0}^{N-1}(A_{p,i} - B_{p,i})\,[\cos(\frac{(2i+1)(m+np)}{2N}\pi) + \cos(\frac{(2i+1)(m-np)}{2N}\pi)]. \tag{11}$$

Define $f_{(p-1)/2,r}$ to be the 1-D DCT output sequence of a given input data sequence $A_{p,i}$, where $0 \le i,\ r \le N-1$. That is,

$$f_{(p-1)/2,r} = \sum_{i=0}^{N-1} A_{p,i}\cos(\frac{(2i+1)r}{2N}\pi).$$

Similarly, define $f_{N-1-(p-1)/2,r}$ to be the 1-D DCT output sequence of a given input data sequence $B_{p,i}$, where $0 \le i,\ r \le N-1$. Let

$$m + np = 2Nk_1 + r_1; \tag{12}$$

$$m - np = 2Nk_2 + r_2, \tag{13}$$

where $-(N-1) \le r_1,\ r_2 \le N$, $k_1$ and $k_2$ are integers. Then,

$$\sum_{i=0}^{N-1} A_{p,i} \cos(\frac{(2i+1)(m+np)}{2N}\pi) = \begin{cases} (-1)^{k_1} f_{(p-1)/2,|r_1|} & \text{if } r_1 \neq N \\ 0 & \text{if } r_1 = N \end{cases}$$

$$\sum_{i=0}^{N-1} A_{p,i} \cos(\frac{(2i+1)(m-np)}{2N}\pi) = \begin{cases} (-1)^{k_2} f_{(p-1)/2,|r_2|} & \text{if } r_2 \neq N \\ 0 & \text{if } r_2 = N \end{cases}$$

$$\sum_{i=0}^{N-1} B_{p,i} \cos(\frac{(2i+1)(m+np)}{2N}\pi) = \begin{cases} (-1)^{k_1} f_{N-1-(p-1)/2,|r_1|} & \text{if } r_1 \neq N \\ 0 & \text{if } r_1 = N \end{cases}$$

$$\sum_{i=0}^{N-1} B_{p,i} \cos(\frac{(2i+1)(m-np)}{2N}\pi) = \begin{cases} (-1)^{k_2} f_{N-1-(p-1)/2,|r_2|} & \text{if } r_2 \neq N \\ 0 & \text{if } r_2 = N. \end{cases}$$

Define $f_{i,N} = 0$ for $0 \le i \le N-1$. Then, if $n$ is even, from (10),

$$y_{m,n} = \frac{1}{2} \sum_{p \in W} [(-1)^{k_1} f_{(p-1)/2,|r_1|} + (-1)^{k_2} f_{(p-1)/2,|r_2|} + (-1)^{k_1} f_{N-1-(p-1)/2,|r_1|} + (-1)^{k_2} f_{N-1-(p-1)/2,|r_2|}]; \tag{14}$$

if $n$ is odd, from (11),

$$y_{m,n} = \frac{1}{2} \sum_{p \in W} [(-1)^{k_1} f_{(p-1)/2,|r_1|} + (-1)^{k_2} f_{(p-1)/2,|r_2|} - (-1)^{k_1} f_{N-1-(p-1)/2,|r_1|} - (-1)^{k_2} f_{N-1-(p-1)/2,|r_2|}]. \tag{15}$$

Let $\underline{\mathbf{y}}_n$ represent the $n$-th column entries of $\mathbf{y}_{N \times N}$. Then, $\underline{\mathbf{y}}_n = (y_{0,n}, y_{1,n}, \cdots, y_{N-1,n})^t$. Let $\underline{\mathbf{f}}_i$ represent the sequence of $f_{i,r}$, for $0 \le i,\ r \le N-1$. Then, $\underline{\mathbf{f}}_i = (f_{i,0}, f_{i,1}, \cdots, f_{i,N-1})^t$. Since the size of the vectors $\underline{\mathbf{y}}_n$ and $\underline{\mathbf{f}}_i$ is $N$, we can represent Equations (14) and (15) as a block-structure matrix multiplication:

$$\mathbf{Y}_{N^2 \times 1} = (\mathbf{E}_N)_{N^2} \mathbf{F}_{N^2 \times 1}, \tag{16}$$

where

$$\mathbf{Y}_{N^2 \times 1} = \begin{bmatrix} \underline{\mathbf{y}}_0 \\ \underline{\mathbf{y}}_1 \\ \vdots \\ \underline{\mathbf{y}}_{N-1} \end{bmatrix}, \qquad \mathbf{F}_{N^2 \times 1} = \begin{bmatrix} \underline{\mathbf{f}}_0 \\ \underline{\mathbf{f}}_1 \\ \vdots \\ \underline{\mathbf{f}}_{N-1} \end{bmatrix},$$

and $(\mathbf{E}_N)_{N^2} =$

$$\begin{bmatrix} (G_{0,1})_N & (G_{0,3})_N & \cdots & (G_{0,N-1})_N & (G'_{0,N-1})_N & \cdots & (G'_{0,3})_N & (G'_{0,1})_N \\ (G_{1,1})_N & (G_{1,3})_N & \cdots & (G_{1,N-1})_N & -(G'_{1,N-1})_N & \cdots & -(G'_{1,3})_N & -(G'_{1,1})_N \\ (G_{2,1})_N & (G_{2,3})_N & \cdots & (G_{2,N-1})_N & (G'_{2,N-1})_N & \cdots & (G'_{2,3})_N & (G'_{2,1})_N \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ (G_{N-1,1})_N & (G_{N-1,3})_N & \cdots & (G_{N-1,N-1})_N & -(G'_{N-1,N-1})_N & \cdots & -(G'_{N-1,3})_N & -(G'_{N-1,1})_N \end{bmatrix}. \tag{17}$$

In Equations (16) and (17), $(G_{n,p})_N$ and $(G'_{n,p})_N$ represent $N \times N$ matrices, $(\mathbf{E}_N)_{N^2}$ represents an $N^2 \times N^2$ matrix, $\mathbf{Y}_{N^2 \times 1}$ and $\mathbf{F}_{N^2 \times 1}$ represent vectors of size $N^2$.

Note that, in Equations (12) and (13), $r_1$ and $r_2$ will not be equal to $N$ at the same time. If $m = 0$ or $n = 0$, then $|k_1| = |k_2|$ and $|r_1| = |r_2|$; if $m \neq 0$ and $n \neq 0$, then $|r_1| \neq |r_2|$. Therefore, from Equations (14) and (15), we can observe that each of the rows of $(G_{n,p})_N$ or $(G'_{n,p})_N$ has one or two nonzero elements, and the location of the nonzero elements is dependent on $np$, where $0 \leq n \leq N-1$ and $p \in W$. Specifically, in both $(G_{n,p})_N$ and $(G'_{n,p})_N$, for $0 \leq m,\ r \leq N-1$,

the element in the position of $(m, r)$

$$= \begin{cases} \frac{1}{2} & \text{if either } \cos\frac{m+np}{2N}\pi = \cos\frac{r}{2N}\pi \text{ or } \cos\frac{m-np}{2N}\pi = \cos\frac{r}{2N}\pi; \\ -\frac{1}{2} & \text{if either } \cos\frac{m+np}{2N}\pi = -\cos\frac{r}{2N}\pi \text{ or } \cos\frac{m-np}{2N}\pi = -\cos\frac{r}{2N}\pi; \\ 1 & \text{if } \cos\frac{m+np}{2N}\pi = \cos\frac{m-np}{2N}\pi = \cos\frac{r}{2N}\pi; \\ -1 & \text{if } \cos\frac{m+np}{2N}\pi = \cos\frac{m-np}{2N}\pi = -\cos\frac{r}{2N}\pi; \\ 0 & \text{otherwise.} \end{cases}$$

Therefore,

$$(G_{n,p})_N = (G'_{n,p})_N.$$

For convenience, without any confusion, for $N \geq 2$, we will frequently represent $(\mathbf{E}_N)_{N^2}$ only by its left half matrix:

$$(\mathbf{E}_N)_{N^2} = [(G_{n,p})_N]_{\substack{0 \leq n \leq N-1 \\ p \in \{1, 3, \cdots, N-1\}}} = [(G_{n,2i+1})_N]_{\substack{0 \leq n \leq N-1 \\ 0 \leq i \leq N/2-1}}$$

In the following, we want to show that the block-structure matrix $(\mathbf{E}_N)_{N^2}$ has the same properties as the $N$-point 1-D DCT kernel matrix. First, the block-structure matrix $(\mathbf{E}_N)_{N^2}$ in Equation (17) has the same structure as the $N$-point 1-D DCT kernel matrix shown below.

$$\text{(1-D DCT)}_N = \mathbf{C}_N$$

$$= \begin{bmatrix} c(0,1) & c(0,3) & \cdots & c(0,N-1) & c(0,N+1) & \cdots & c(0,2N-3) & c(0,2N-1) \\ c(1,1) & c(1,3) & \cdots & c(1,N-1) & c(1,N+1) & \cdots & c(1,2N-3) & c(1,2N-1) \\ c(2,1) & c(2,3) & \cdots & c(2,N-1) & c(2,N+1) & \cdots & c(2,2N-3) & c(2,2N-1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c(N-1,1) & c(N-1,3) & \cdots & c(N-1,N-1) & c(N-1,N+1) & \cdots & c(N-1,2N-3) & c(N-1,2N-1) \end{bmatrix}$$

$$= \begin{bmatrix} c(0,1) & c(0,3) & \cdots & c(0,N-1) & c(0,N-1) & \cdots & c(0,3) & c(0,1) \\ c(1,1) & c(1,3) & \cdots & c(1,N-1) & -c(1,N-1) & \cdots & -c(1,3) & -c(1,1) \\ c(2,1) & c(2,3) & \cdots & c(2,N-1) & c(2,N-1) & \cdots & c(2,3) & c(2,1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ c(N-1,1) & c(N-1,3) & \cdots & c(N-1,N-1) & -c(N-1,N-1) & \cdots & -c(N-1,3) & -c(N-1,1) \end{bmatrix},$$

where $c(n,p)$ represents $\cos \frac{n \times p}{2N}\pi$.

Second, since the elements in $(G_{n,p})_N$ only depend on $np$, therefore, for $0 \le n,\ n' \le N-1$ and $p,\ p' \in W$, if $np = n'p'$ then $(G_{n,p})_N = (G_{n',p'})_N$. Define

$$(H_{np})_N = (G_{n,p})_N,$$

for $0 \le n \le N-1$ and $p \in W$. Then, for $i \ge 0$,

the element $(m,r)$ of $(H_i)_N$

$$= \begin{cases} \frac{1}{2} & \text{if either } \cos\frac{m+i}{2N}\pi = \cos\frac{r}{2N}\pi \text{ or } \cos\frac{m-i}{2N}\pi = \cos\frac{r}{2N}\pi \\ -\frac{1}{2} & \text{if either } \cos\frac{m+i}{2N}\pi = -\cos\frac{r}{2N}\pi \text{ or } \cos\frac{m-i}{2N}\pi = -\cos\frac{r}{2N}\pi \\ 1 & \text{if } \cos\frac{m+i}{2N}\pi = \cos\frac{m-i}{2N}\pi = \cos\frac{r}{2N}\pi \\ -1 & \text{if } \cos\frac{m+i}{2N}\pi = \cos\frac{m-i}{2N}\pi = -\cos\frac{r}{2N}\pi \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

In addition, $(H_i)_N$ has the following properties, which are the same as $\cos\frac{i}{2N}\pi$.

| Property | $(H_i)_N$ | $\cos\frac{i}{2N}\pi$ |
|---|---|---|
| 1 | $(H_{i-2Nk})_N = (-1)^k (H_i)_N$ for $i,\ i-2Nk \ge 0$ | $\cos\frac{i-2Nk}{2N}\pi = (-1)^k \cos\frac{i}{2N}\pi$ for $i,\ i-2Nk \ge 0$ |
| 2 | $(H_{2Nk-i})_N = (-1)^k (H_i)_N$ for $i,\ 2Nk-i \ge 0$ | $\cos\frac{2Nk-i}{2N}\pi = (-1)^k \cos\frac{i}{2N}\pi$ for $i,\ 2Nk-i \ge 0$ |
| 3 | $2(H_i)_N(H_j)_N = \begin{cases} (H_{\lvert i-j\rvert})_N, & \text{if } i+j = N \\ (H_{\lvert i-j\rvert})_N + (H_{i+j})_N, & \text{if } i+j \ne N \end{cases}$ for $0 \le i,\ j \le N-1$ | $2\cos\frac{i}{2N}\pi \cos\frac{j}{2N}\pi = \begin{cases} \cos\frac{\lvert i-j\rvert}{2N}\pi, & \text{if } i+j = N \\ \cos\frac{\lvert i-j\rvert}{2N}\pi + \cos\frac{i+j}{2N}\pi, & \text{if } i+j \ne N \end{cases}$ for $0 \le i,\ j \le N-1$ |
| 4 | $(H_i)_N(H_j)_N = (H_j)_N(H_i)_N$ for $0 \le i,\ j \le N-1$ | $\cos\frac{i}{2N}\pi \cos\frac{j}{2N}\pi = \cos\frac{j}{2N}\pi \cos\frac{i}{2N}\pi$ for $0 \le i,\ j \le N-1$ |

(19)

Property 1 holds because of

$$\begin{aligned} \cos\frac{m+(i-2Nk)}{2N}\pi &= (-1)^k \cos\frac{m+i}{2N}\pi \\ \cos\frac{m-(i-2Nk)}{2N}\pi &= (-1)^k \cos\frac{m-i}{2N}\pi. \end{aligned}$$

Property 2 holds because of

$$\cos\frac{m+(2Nk-i)}{2N}\pi = (-1)^k\cos\frac{m-i}{2N}\pi$$
$$\cos\frac{m-(2Nk-i)}{2N}\pi = (-1)^k\cos\frac{m+i}{2N}\pi.$$

Property 3 was proved by [4]. Property 4 follows immediately from Property 3.

The purpose of showing that the block-structure matrix $(\mathbf{E}_N)_{N^2}$ has the same properties as the $N$-point 1-D DCT kernel matrix $\mathbf{C}_N$, is that we can compute the 2-D DCT using a procedure similar to that of the 1-D DCT.

# 3 The 2-D DCT Algorithm

In [10], we derived a recursive algorithm for computing the 1-D DCT $\mathbf{Z}_N = \mathbf{C}_N\,\mathbf{z}_N$. The derivation was based on the four cosine properties listed in (19). Because our 2-D DCT algorithm is based on this recursive algorithm, we describe its procedure due to its relevance to deriving our 2-D DCT algorithm.

## 3.1 A Recursive Algorithm for Computing the 1-D DCT

Let $\begin{bmatrix} Z_e \\ Z_o \end{bmatrix}$ be $(Z_0, Z_2, \ldots, Z_{N-2}, Z_1, Z_3, \ldots, Z_{N-1})^t$ and $\begin{bmatrix} z_e \\ z_{\bar{o}} \end{bmatrix}$ be $(z_0, z_2, \ldots, z_{N-2}, z_{N-1}, \cdots, z_3, z_1)^t$. We will use the subindex $'e'$ to represent the even entries of a vector, $'o'$ the odd entries, and $'\bar{o}'$ the odd entries but in reversed order.

Let $P_N$, $S_N$, $L_N$, and $Q_{\frac{N}{2}}$ be the same matrices defined in [10]. $P_N$ is a column permutation matrix; $S_N$ is a row permutation matrix for performing a shuffle exchange operation; $L_N$ is a regular lower triangular matrix whose nonzero entries are either $\pm 1$ or $\pm 2$; and $Q_{\frac{N}{2}}$ is a diagonal matrix whose diagonal elements are cosine coefficients $\cos\frac{4n+1}{2N}\pi$, for $0 \le n \le \frac{N}{2}-1$. Define $\hat{\mathbf{C}}_N = S_N^t\,\mathbf{C}_N\,P_N$. Then, $\hat{\mathbf{C}}_N$ has a recursive structure.

**1-D DCT recursive algorithm [10] :**

**Step 1:** Evaluate $\begin{bmatrix} z_e \\ z_{\bar{o}} \end{bmatrix} = P_N^t \, \mathbf{z}$;

**Step 2:** /* Recursively compute $\begin{bmatrix} Z_e \\ Z_o \end{bmatrix} = \hat{\mathbf{C}}_N \begin{bmatrix} z_e \\ z_{\bar{o}} \end{bmatrix}$. */

**2.1** if $N$ is 2, then compute $Z_e = (z_e + z_{\bar{o}})$ and $Z_o = \cos \frac{N/2}{2N} \pi \, (z_e - z_{\bar{o}})$ directly;

**2.2** else recursively perform $Z_e = S_{\frac{N}{2}} \, \hat{\mathbf{C}}_{\frac{N}{2}} \, (z_e + z_{\bar{o}})$ and $Z_o = L_{\frac{N}{2}} \, S_{\frac{N}{2}} \, \hat{\mathbf{C}}_{\frac{N}{2}} \, Q_{\frac{N}{2}} \, (z_e - z_{\bar{o}})$;

**Step 3:** calculate $\mathbf{Z} = S_N \begin{bmatrix} Z_e \\ Z_o \end{bmatrix}$. □

### 3.2 Our 2-D DCT Algorithm

The procedure of our 2-D DCT algorithm is similar to the 1-D DCT algorithm described in the last subsection. We start from Equation (16). Suppose that $N'$ also has a value of 2 to a power and $2 \leq N' \leq N$. First, let $\begin{bmatrix} Y_e \\ Y_o \end{bmatrix}$ be $(\underline{\mathbf{y}}_0^t, \underline{\mathbf{y}}_2^t, \ldots, \underline{\mathbf{y}}_{N-2}^t, \underline{\mathbf{y}}_1^t, \underline{\mathbf{y}}_3^t, \ldots, \underline{\mathbf{y}}_{N-1}^t)^t$ and $\begin{bmatrix} F_e \\ F_{\bar{o}} \end{bmatrix}$ be $(\underline{\mathbf{f}}_0^t, \underline{\mathbf{f}}_2^t, \ldots, \underline{\mathbf{f}}_{N-2}^t, \underline{\mathbf{f}}_{N-1}^t, \ldots, \underline{\mathbf{f}}_3^t, \underline{\mathbf{f}}_1^t)^t$. Second, let $P_{N'}$, $S_{N'}$, and $L_{N'}$ be the same matrices defined in [10]. Let $(P_N)_{N'N}$, $(S_N)_{N'N}$, and $(L_N)_{N'N}$ be the block matrix versions of $P_{N'}$, $S_{N'}$, and $L_{N'}$, respectively. That is, $(P_N)_{N'N} = P_{N'} \otimes I_N$; $(S_N)_{N'N} = S_{N'} \otimes I_N$; and $(L_N)_{N'N} = L_{N'} \otimes I_N$, where $\otimes$ denotes the Kronecker product and $I_N$ is the $N \times N$ identity matrix.

Third, let $(Q_N)_{N'N}$ be a block matrix with block size $N$, where $2 \leq N' \leq N/2$, and $(Q_N)_{N'N} = diagonal[(G_{N/2N',4n+1})_N] = diagonal[(H_{N(4n+1)/2N'})_N]$, for $0 \leq n \leq N'-1$. Fourth, for $2 \leq N' \leq N$, define

$$
\begin{aligned}
(\mathbf{E}_N)_{N'N} &= \left[(G_{\frac{N}{N'}n,2i+1})_N\right] \quad \begin{matrix} 0 \leq n \leq N'-1 \\ 0 \leq i \leq N'/2-1 \end{matrix} \\
(\hat{\mathbf{E}}_N)_{N'N} &= (S_N^t)_{N'N} \, (\mathbf{E}_N)_{N'N} \, (P_N)_{N'N}.
\end{aligned}
$$

Then, $(\hat{\mathbf{E}}_N)_{N'N}$ also has a recursive structure.

In the following, we describe the procedure for computing the 2-D DCT. It is a recursive algorithm, and its derivation is based on the four properties listed in (19). Note that, the derivation does not need to be illustrated due to space constraints; however, this has been verified.

**2-D DCT algorithm:**

**Step 1:** Compute $N$ sets of 1-D DCT's for each input data sequence of $\{A_{p,i}\}_{i=0}^{N-1}$ and $\{B_{p,i}\}_{i=0}^{N-1}$, where $p \in W$; and let the output vectors be $\mathbf{f}_{(p-1)/2}$ and $\mathbf{f}_{N-1-(p-1)/2}$, respectively;

**Step 2:** evaluate $\begin{bmatrix} F_e \\ F_{\bar{o}} \end{bmatrix} = (P_N^t)_{N^2}\,\mathbf{F}$;

let $N' = N$;

**Step 3:** /* Recursively compute $\begin{bmatrix} Y_e \\ Y_o \end{bmatrix} = (\hat{\mathbf{E}}_N)_{N'N} \begin{bmatrix} F_e \\ F_{\bar{o}} \end{bmatrix}$. */

**3.1** if $N'$ is 2, then compute $Y_e = (F_e + F_{\bar{o}})$ and $Y_o = (H_{N/2})_N(F_e - F_{\bar{o}})$ directly;

**3.2** else recursively perform

$$\begin{aligned} Y_e &= (S_N)_{N'N/2}(\hat{\mathbf{E}}_N)_{N'N/2}(F_e + F_{\bar{o}}); \\ Y_o &= (L_N)_{N'N/2}(S_N)_{N'N/2}(\hat{\mathbf{E}}_N)_{N'N/2}(Q_N)_{N'N/2}(F_e - F_{\bar{o}}); \end{aligned}$$

**Step 4:** calculate $\mathbf{Y} = (S_N)_{N^2} \begin{bmatrix} Y_e \\ Y_o \end{bmatrix}$. □

In the following, we analyze the algorithm's complexity. As shown in [7] [10], it requires $\frac{1}{2}N\log N$ multiplications and $\frac{3}{2}N\log N - N + 1$ additions for computing an $N$-point 1-D DCT. In our algorithm, for computing an $(N \times N)$-point 2-D DCT, it requires evaluating $N$ sets of $N$-point 1-D DCT's and a post-addition stage, where the block version of this post-addition stage has the same structure as the 1-D DCT. Therefore, it requires $\frac{1}{2}N^2\log N$ multiplications and $\frac{7}{2}N^2\log N - 2N^2 + 2N - N\log N$ additions for computing an $(N \times N)$-point 2-D DCT. Note that, it only requires $N - 2$ additions for calculating $(H_i)_N\,\mathbf{v}_N$, where $\mathbf{v}_N$ is a vector of size $N$ and $0 < i \leq N - 1$. Table 1 shows the comparison of the computation complexities among those 2-D DCT algorithms found in the related research works.

| | # of multiplications | # of additions |
|---|---|---|
| this paper | $\frac{1}{2}N^2 \log N$ | $\frac{7}{2}N^2 \log N - 2N^2 + 2N - N \log N$ |
| [4] | $\frac{1}{2}N^2 \log N$ | $\frac{7}{2}N^2 \log N - 2N^2 + 2N - N \log N$ |
| [2] | $\frac{1}{2}N^2 \log N$ | $\frac{5}{2}N^2 \log N - 2N + 2$ |
| [12] | $\frac{1}{2}N^2 \log N + \frac{1}{3}N^2 - 2N + \frac{8}{3}$ | $\frac{5}{2}N^2 \log N + \frac{1}{3}N^2 - 6N + \frac{62}{3}$ |
| [10] | $\frac{3}{4}N^2 \log N - \frac{1}{4}N^2$ | $3N^2 \log N - 2N^2 + 2N$ |
| [1] [6] | $\frac{3}{4}N^2 \log N$ | $3N^2 \log N - 2N^2 + 2N$ |
| row-column method | $N^2 \log N$ | $3N^2 \log N - 2N^2 + 2N$ |

Table 1: Computation complexities for the 2-D DCT.

# 4 Examples

In this section, we illustrate our 2-D DCT algorithm with three lower order examples. Let $\alpha = 1/2$.

For $N = 2$,

Step 1:

$$\{A_{1,i}\}_{i=0}^{1} = \{x_{0,0},\ x_{1,1}\}, \qquad \underline{\mathbf{f}}_0 = DCT(\{A_{1,i}\}_{i=0}^{1});$$
$$\{B_{1,i}\}_{i=0}^{1} = \{x_{0,1},\ x_{1,0}\}, \qquad \underline{\mathbf{f}}_1 = DCT(\{B_{1,i}\}_{i=0}^{1}).$$

Step 2:

$$P_2^t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad (P_2^t)_4 = P_2^t \otimes I_2 = \begin{bmatrix} I_2 & 0_2 \\ 0_2 & I_2 \end{bmatrix}, \quad \begin{bmatrix} F_e \\ F_{\bar{o}} \end{bmatrix} = (P_2^t)_4 \mathbf{F} = \begin{bmatrix} \underline{\mathbf{f}}_0 \\ \underline{\mathbf{f}}_1 \end{bmatrix}.$$

$$(\mathbf{E}_2)_4 = \begin{bmatrix} (H_0)_2 & (H_0)_2 \\ (H_1)_2 & -(H_1)_2 \end{bmatrix}, \quad (H_0)_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad (H_1)_2 = \begin{bmatrix} 0 & 1 \\ \alpha & 0 \end{bmatrix}.$$

$$S_2^t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad (S_2^t)_4 = S_2^t \otimes I_2 = \begin{bmatrix} I_2 & 0_2 \\ 0_2 & I_2 \end{bmatrix}, \quad (\hat{\mathbf{E}}_2)_4 = (S_2^t)_4 (\mathbf{E}_2)_4 (P_2)_4 = (\mathbf{E}_2)_4.$$

Step 3: $N' = 2$, therefore,

$$Y_e = \underline{y}_0 = (F_e + F_{\bar{o}}) = (\underline{f}_0 + \underline{f}_1); \quad Y_o = \underline{y}_1 = (H_1)_2(F_e - F_{\bar{o}}) = (H_1)_2(\underline{f}_0 - \underline{f}_1).$$

Step 4:

$$\mathbf{Y} = \begin{bmatrix} \underline{y}_0 \\ \underline{y}_1 \end{bmatrix} = (S_2)_4 \begin{bmatrix} Y_e \\ Y_o \end{bmatrix} = \begin{bmatrix} (\underline{f}_0 + \underline{f}_1) \\ (H_1)_2(\underline{f}_0 - \underline{f}_1) \end{bmatrix}.$$

Figs 1-(a) and 1-(b) show the signal-flow graph and the abstract signal-flow graph for $N = 2$.

For $N = 4$,

Step 1:

$$\begin{aligned} \{A_{1,i}\}_{i=0}^{3} &= \{x_{0,0},\ x_{1,1},\ x_{2,2},\ x_{3,3}\}, & \underline{f}_0 &= DCT(\{A_{1,i}\}_{i=0}^{3}); \\ \{A_{3,i}\}_{i=0}^{3} &= \{x_{0,1},\ x_{1,3},\ x_{2,0},\ x_{3,2}\}, & \underline{f}_1 &= DCT(\{A_{3,i}\}_{i=0}^{3}); \\ \{B_{3,i}\}_{i=0}^{3} &= \{x_{0,2},\ x_{1,0},\ x_{2,3},\ x_{3,1}\}, & \underline{f}_2 &= DCT(\{B_{3,i}\}_{i=0}^{3}); \\ \{B_{1,i}\}_{i=0}^{3} &= \{x_{0,3},\ x_{1,2},\ x_{2,1},\ x_{3,0}\}, & \underline{f}_3 &= DCT(\{B_{1,i}\}_{i=0}^{3}). \end{aligned}$$

Step 2:

$$P_4^t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad (P_4^t)_{16} = P_4^t \otimes I_4 = \begin{bmatrix} I_4 & 0_4 & 0_4 & 0_4 \\ 0_4 & 0_4 & I_4 & 0_4 \\ 0_4 & 0_4 & 0_4 & I_4 \\ 0_4 & I_4 & 0_4 & 0_4 \end{bmatrix}, \quad \begin{bmatrix} F_e \\ F_{\bar{o}} \end{bmatrix} = (P_4^t)_{16}\mathbf{F} = \begin{bmatrix} \underline{f}_0 \\ \underline{f}_2 \\ \underline{f}_3 \\ \underline{f}_1 \end{bmatrix}.$$

$$(\mathbf{E}_4)_{16} = \begin{bmatrix} (H_0)_4 & (H_0)_4 & (H_0)_4 & (H_0)_4 \\ (H_1)_4 & (H_3)_4 & -(H_3)_4 & -(H_1)_4 \\ (H_2)_4 & -(H_2)_4 & -(H_2)_4 & (H_2)_4 \\ (H_3)_4 & -(H_1)_4 & (H_1)_4 & -(H_3)_4 \end{bmatrix}.$$

$$(H_0)_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (H_1)_4 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \alpha & 0 & \alpha & 0 \\ 0 & \alpha & 0 & \alpha \\ 0 & 0 & \alpha & 0 \end{bmatrix},$$

$$(H_2)_4 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & \alpha & 0 & \alpha \\ \alpha & 0 & 0 & 0 \\ 0 & \alpha & 0 & -\alpha \end{bmatrix}, \quad (H_3)_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & \alpha & 0 \\ 0 & \alpha & 0 & -\alpha \\ \alpha & 0 & -\alpha & 0 \end{bmatrix}.$$

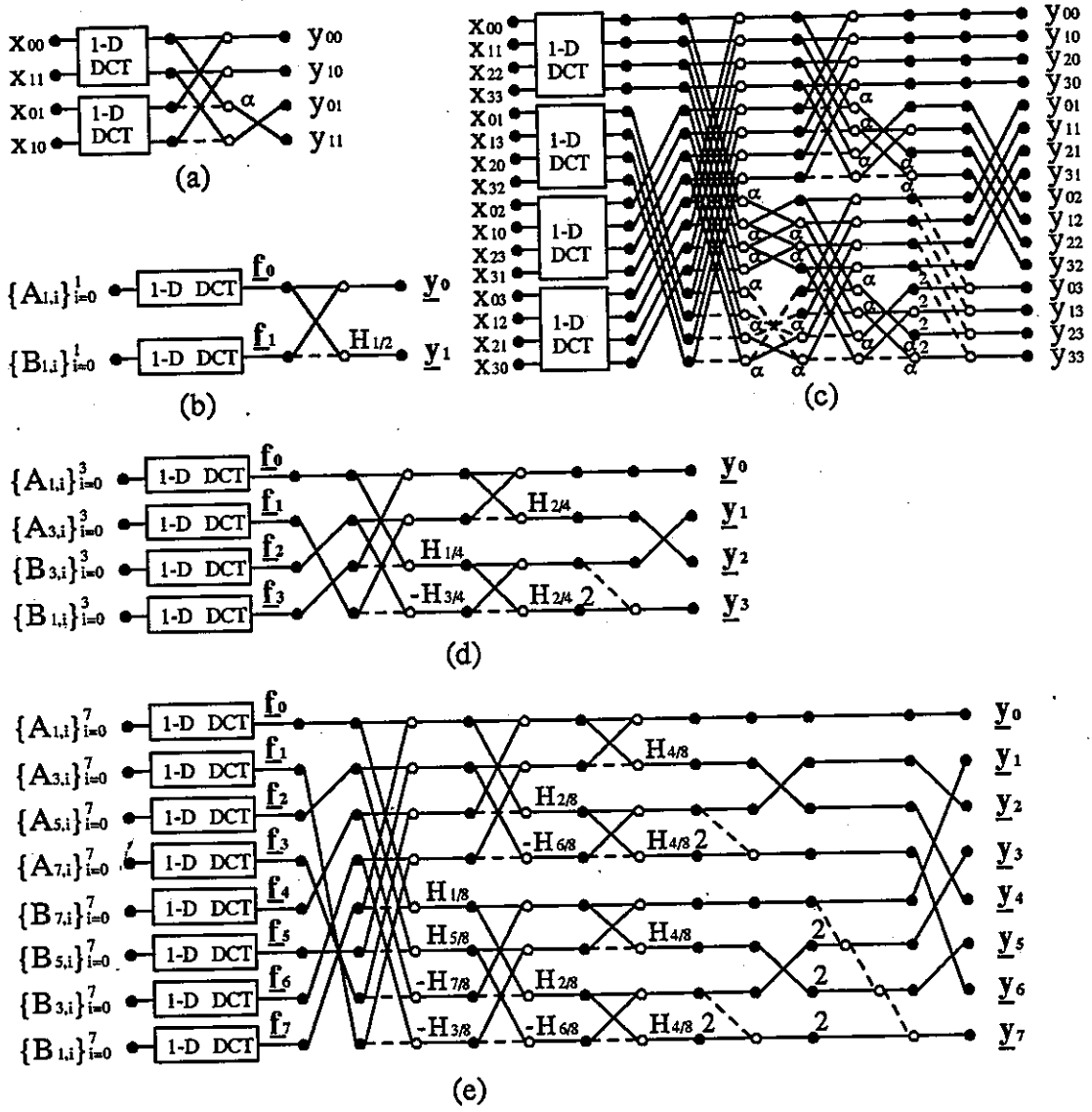


Figure 1: The 2-D DCT signal-flow graphs and abstract signal-flow graphs for (a) (b) $N = 2$, (c) (d) $N = 4$, and (e) $N = 8$. Solid lines represent transfer factor 1, while dashed lines represent transfer factor $-1$. Circles o represent adders. $H\,i/j$ means $(H_i)_j$. $\alpha = 1/2$.

$$S_4^t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (S_4^t)_{16} = S_4^t \otimes I_4 = \begin{bmatrix} I_4 & 0_4 & 0_4 & 0_4 \\ 0_4 & 0_4 & I_4 & 0_4 \\ 0_4 & I_4 & 0_4 & 0_4 \\ 0_4 & 0_4 & 0_4 & I_4 \end{bmatrix}.$$

$$(\hat{\mathbf{E}}_4)_{16} = (S_4^t)_{16}\,(\mathbf{E}_4)_{16}\,(P_4)_{16} = \begin{bmatrix} (H_0)_4 & (H_0)_4 & (H_0)_4 & (H_0)_4 \\ (H_2)_4 & -(H_2)_4 & (H_2)_4 & -(H_2)_4 \\ (H_1)_4 & -(H_3)_4 & -(H_1)_4 & (H_3)_4 \\ (H_3)_4 & (H_1)_4 & -(H_3)_4 & -(H_1)_4 \end{bmatrix}.$$

Step 3: $N' = 4$, we compute $Y_e$ and $Y_o$ recursively.

$$\begin{aligned} Y_e &= \begin{bmatrix} \underline{y}_0 \\ \underline{y}_2 \end{bmatrix} = (S_4)_8\,(\hat{\mathbf{E}}_4)_8\,(F_e + F_{\bar{o}}) \\ &= \begin{bmatrix} I_4 & 0_4 \\ 0_4 & I_4 \end{bmatrix} \begin{bmatrix} (H_0)_4 & (H_0)_4 \\ (H_2)_4 & -(H_2)_4 \end{bmatrix} \left( \begin{bmatrix} \underline{f}_0 \\ \underline{f}_2 \end{bmatrix} + \begin{bmatrix} \underline{f}_3 \\ \underline{f}_1 \end{bmatrix} \right); \\ Y_o &= \begin{bmatrix} \underline{y}_1 \\ \underline{y}_3 \end{bmatrix} = (L_4)_8\,(S_4)_8\,(\hat{\mathbf{E}}_4)_8\,(Q_4)_8\,(F_e - F_{\bar{o}}) \\ &= \begin{bmatrix} I_4 & 0_4 \\ -I_4 & 2I_4 \end{bmatrix} \begin{bmatrix} I_4 & 0_4 \\ 0_4 & I_4 \end{bmatrix} \begin{bmatrix} (H_0)_4 & (H_0)_4 \\ (H_2)_4 & -(H_2)_4 \end{bmatrix} \begin{bmatrix} (H_1)_4 & 0_4 \\ 0_4 & (H_5)_4 \end{bmatrix} \left( \begin{bmatrix} \underline{f}_0 \\ \underline{f}_2 \end{bmatrix} - \begin{bmatrix} \underline{f}_3 \\ \underline{f}_1 \end{bmatrix} \right) \\ &= \begin{bmatrix} (H_1)_4 & -(H_3)_4 \\ (H_3)_4 & (H_1)_4 \end{bmatrix} \left( \begin{bmatrix} \underline{f}_0 \\ \underline{f}_2 \end{bmatrix} - \begin{bmatrix} \underline{f}_3 \\ \underline{f}_1 \end{bmatrix} \right). \end{aligned}$$

Step 4:

$$\underline{Y} = \begin{bmatrix} \underline{y}_0 \\ \underline{y}_1 \\ \underline{y}_2 \\ \underline{y}_3 \end{bmatrix} = (S_4)_{16} \begin{bmatrix} Y_e \\ Y_o \end{bmatrix} = \begin{bmatrix} I_4 & 0_4 & 0_4 & 0_4 \\ 0_4 & 0_4 & I_4 & 0_4 \\ 0_4 & I_4 & 0_4 & 0_4 \\ 0_4 & 0_4 & 0_4 & I_4 \end{bmatrix} \begin{bmatrix} \underline{y}_0 \\ \underline{y}_2 \\ \underline{y}_1 \\ \underline{y}_3 \end{bmatrix}.$$

Figs 1-(c) and 1-(d) show the signal-flow graph and the abstract signal-flow graph for $N = 4$.

For $N = 8$,

Step 1:

$$
\begin{aligned}
\{A_{1,i}\}_{i=0}^{7} &= \{x_{0,0},\ x_{1,1},\ x_{2,2},\ x_{3,3},\ x_{4,4},\ x_{5,5},\ x_{6,6},\ x_{7,7}\}, & \underline{\mathbf{f}}_0 &= DCT(\{A_{1,i}\}_{i=0}^{7});\\
\{A_{3,i}\}_{i=0}^{7} &= \{x_{0,1},\ x_{1,4},\ x_{2,7},\ x_{3,5},\ x_{4,2},\ x_{5,0},\ x_{6,3},\ x_{7,6}\}, & \underline{\mathbf{f}}_1 &= DCT(\{A_{3,i}\}_{i=0}^{7});\\
\{A_{5,i}\}_{i=0}^{7} &= \{x_{0,2},\ x_{1,7},\ x_{2,3},\ x_{3,1},\ x_{4,6},\ x_{5,4},\ x_{6,0},\ x_{7,5}\}, & \underline{\mathbf{f}}_2 &= DCT(\{A_{5,i}\}_{i=0}^{7});\\
\{A_{7,i}\}_{i=0}^{7} &= \{x_{0,3},\ x_{1,5},\ x_{2,1},\ x_{3,7},\ x_{4,0},\ x_{5,6},\ x_{6,2},\ x_{7,4}\}, & \underline{\mathbf{f}}_3 &= DCT(\{A_{7,i}\}_{i=0}^{7});\\
\{B_{7,i}\}_{i=0}^{7} &= \{x_{0,4},\ x_{1,2},\ x_{2,6},\ x_{3,0},\ x_{4,7},\ x_{5,1},\ x_{6,5},\ x_{7,3}\}, & \underline{\mathbf{f}}_4 &= DCT(\{B_{7,i}\}_{i=0}^{7});\\
\{B_{5,i}\}_{i=0}^{7} &= \{x_{0,5},\ x_{1,0},\ x_{2,4},\ x_{3,6},\ x_{4,1},\ x_{5,3},\ x_{6,7},\ x_{7,2}\}, & \underline{\mathbf{f}}_5 &= DCT(\{B_{5,i}\}_{i=0}^{7});\\
\{B_{3,i}\}_{i=0}^{7} &= \{x_{0,6},\ x_{1,3},\ x_{2,0},\ x_{3,2},\ x_{4,5},\ x_{5,7},\ x_{6,4},\ x_{7,1}\}, & \underline{\mathbf{f}}_6 &= DCT(\{B_{3,i}\}_{i=0}^{7});\\
\{B_{1,i}\}_{i=0}^{7} &= \{x_{0,7},\ x_{1,6},\ x_{2,5},\ x_{3,4},\ x_{4,3},\ x_{5,2},\ x_{6,1},\ x_{7,0}\}, & \underline{\mathbf{f}}_7 &= DCT(\{B_{1,i}\}_{i=0}^{7}).
\end{aligned}
$$

Step 2:

$$
(P_8^t)_{64} = P_8^t \otimes I_8 = \begin{bmatrix}
I_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 \\
0_8 & 0_8 & I_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 \\
0_8 & 0_8 & 0_8 & 0_8 & I_8 & 0_8 & 0_8 & 0_8 \\
0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & I_8 & 0_8 \\
0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & I_8 \\
0_8 & 0_8 & 0_8 & 0_8 & 0_8 & I_8 & 0_8 & 0_8 \\
0_8 & 0_8 & 0_8 & I_8 & 0_8 & 0_8 & 0_8 & 0_8 \\
0_8 & I_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8
\end{bmatrix}, \quad
\begin{bmatrix} F_e \\ F_{\bar{o}} \end{bmatrix} = (P_8^t)_{64}\mathbf{F} = \begin{bmatrix}
\underline{\mathbf{f}}_0 \\ \underline{\mathbf{f}}_2 \\ \underline{\mathbf{f}}_4 \\ \underline{\mathbf{f}}_6 \\ \underline{\mathbf{f}}_7 \\ \underline{\mathbf{f}}_5 \\ \underline{\mathbf{f}}_3 \\ \underline{\mathbf{f}}_1
\end{bmatrix}.
$$

$$
(\mathrm{E}_8)_{64} = \begin{bmatrix}
(H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 \\
(H_1)_8 & (H_3)_8 & (H_5)_8 & (H_7)_8 & -(H_7)_8 & -(H_5)_8 & -(H_3)_8 & -(H_1)_8 \\
(H_2)_8 & (H_6)_8 & -(H_6)_8 & -(H_2)_8 & -(H_2)_8 & -(H_6)_8 & (H_6)_8 & (H_2)_8 \\
(H_3)_8 & -(H_7)_8 & -(H_1)_8 & -(H_5)_8 & (H_5)_8 & (H_1)_8 & (H_7)_8 & -(H_3)_8 \\
(H_4)_8 & -(H_4)_8 & -(H_4)_8 & (H_4)_8 & (H_4)_8 & -(H_4)_8 & -(H_4)_8 & (H_4)_8 \\
(H_5)_8 & -(H_1)_8 & (H_7)_8 & (H_3)_8 & -(H_3)_8 & -(H_7)_8 & (H_1)_8 & -(H_5)_8 \\
(H_6)_8 & -(H_2)_8 & (H_2)_8 & -(H_6)_8 & -(H_6)_8 & (H_2)_8 & -(H_2)_8 & (H_6)_8 \\
(H_7)_8 & -(H_5)_8 & (H_3)_8 & -(H_1)_8 & (H_1)_8 & -(H_3)_8 & (H_5)_8 & -(H_7)_8
\end{bmatrix}.
$$

$$
(H_0)_8 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}, \quad
(H_1)_8 = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha & 0 & \alpha & 0 & 0 & 0 & 0 & 0 \\
0 & \alpha & 0 & \alpha & 0 & 0 & 0 & 0 \\
0 & 0 & \alpha & 0 & \alpha & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha & 0 & \alpha & 0 & 0 \\
0 & 0 & 0 & 0 & \alpha & 0 & \alpha & 0 \\
0 & 0 & 0 & 0 & 0 & \alpha & 0 & \alpha \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0
\end{bmatrix},
$$

$$(H_2)_8 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha & 0 & \alpha & 0 & 0 & 0 & 0 \\ \alpha & 0 & 0 & 0 & \alpha & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 & \alpha & 0 & 0 \\ 0 & 0 & \alpha & 0 & 0 & 0 & \alpha & 0 \\ 0 & 0 & 0 & \alpha & 0 & 0 & 0 & \alpha \\ 0 & 0 & 0 & 0 & \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha & 0 & -\alpha \end{bmatrix}, \quad (H_3)_8 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha & 0 & \alpha & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 & \alpha & 0 & 0 \\ \alpha & 0 & 0 & 0 & 0 & 0 & \alpha & 0 \\ 0 & \alpha & 0 & 0 & 0 & 0 & 0 & \alpha \\ 0 & 0 & \alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha & 0 & 0 & 0 & -\alpha \\ 0 & 0 & 0 & 0 & \alpha & 0 & -\alpha & 0 \end{bmatrix},$$

$$(H_4)_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha & 0 & \alpha & 0 & 0 \\ 0 & 0 & \alpha & 0 & 0 & 0 & \alpha & 0 \\ 0 & \alpha & 0 & 0 & 0 & 0 & 0 & \alpha \\ \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 & 0 & 0 & -\alpha \\ 0 & 0 & \alpha & 0 & 0 & 0 & -\alpha & 0 \\ 0 & 0 & 0 & \alpha & 0 & -\alpha & 0 & 0 \end{bmatrix}, \quad (H_5)_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha & 0 & \alpha & 0 \\ 0 & 0 & 0 & \alpha & 0 & 0 & 0 & \alpha \\ 0 & 0 & \alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 & 0 & 0 & -\alpha \\ \alpha & 0 & 0 & 0 & 0 & 0 & -\alpha & 0 \\ 0 & \alpha & 0 & 0 & 0 & -\alpha & 0 & 0 \\ 0 & 0 & \alpha & 0 & -\alpha & 0 & 0 & 0 \end{bmatrix},$$

$$(H_6)_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha & 0 & \alpha \\ 0 & 0 & 0 & 0 & \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha & 0 & 0 & 0 & -\alpha \\ 0 & 0 & \alpha & 0 & 0 & 0 & -\alpha & 0 \\ 0 & \alpha & 0 & 0 & 0 & -\alpha & 0 & 0 \\ \alpha & 0 & 0 & 0 & -\alpha & 0 & 0 & 0 \\ 0 & \alpha & 0 & -\alpha & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (H_7)_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha & 0 & -\alpha \\ 0 & 0 & 0 & 0 & \alpha & 0 & -\alpha & 0 \\ 0 & 0 & 0 & \alpha & 0 & -\alpha & 0 & 0 \\ 0 & 0 & \alpha & 0 & -\alpha & 0 & 0 & 0 \\ 0 & \alpha & 0 & -\alpha & 0 & 0 & 0 & 0 \\ \alpha & 0 & -\alpha & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

$$(S_8^t)_{64} = S_8^t \otimes I_8 = \begin{bmatrix} I_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & I_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & 0_8 & 0_8 & I_8 & 0_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & I_8 & 0_8 \\ 0_8 & I_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & 0_8 & I_8 & 0_8 & 0_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & I_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & I_8 \end{bmatrix}.$$

$$(\hat{\mathbf{E}}_8)_{64} = (S_8^t)_{64}(\mathbf{E}_8)_{64}(P_8)_{64} =$$

$$\begin{bmatrix} (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 \\ (H_2)_8 & -(H_6)_8 & -(H_2)_8 & (H_6)_8 & (H_2)_8 & -(H_6)_8 & -(H_2)_8 & (H_6)_8 \\ (H_4)_8 & -(H_4)_8 & (H_4)_8 & -(H_4)_8 & (H_4)_8 & -(H_4)_8 & (H_4)_8 & -(H_4)_8 \\ (H_6)_8 & (H_2)_8 & -(H_6)_8 & -(H_2)_8 & (H_6)_8 & (H_2)_8 & -(H_6)_8 & -(H_2)_8 \\ (H_1)_8 & (H_5)_8 & -(H_7)_8 & -(H_3)_8 & -(H_1)_8 & -(H_5)_8 & (H_7)_8 & (H_3)_8 \\ (H_3)_8 & -(H_1)_8 & (H_5)_8 & (H_7)_8 & -(H_3)_8 & (H_1)_8 & -(H_5)_8 & -(H_7)_8 \\ (H_5)_8 & (H_7)_8 & -(H_3)_8 & (H_1)_8 & -(H_5)_8 & -(H_7)_8 & (H_3)_8 & -(H_1)_8 \\ (H_7)_8 & (H_3)_8 & (H_1)_8 & (H_5)_8 & -(H_7)_8 & -(H_3)_8 & -(H_1)_8 & -(H_5)_8 \end{bmatrix}.$$

Step 3: $N' = 8$, we compute $Y_e$ and $Y_o$ recursively.

$$
\begin{aligned}
Y_e &= \begin{bmatrix} \underline{\mathbf{y}}_0 \\ \underline{\mathbf{y}}_2 \\ \underline{\mathbf{y}}_4 \\ \underline{\mathbf{y}}_6 \end{bmatrix} = (S_8)_{32}\,(\hat{\mathbf{E}}_8)_{32}\,(F_e + F_{\bar{o}}) \\
&= \begin{bmatrix} I_8 & 0_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & I_8 & 0_8 \\ 0_8 & I_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & 0_8 & I_8 \end{bmatrix} \begin{bmatrix} (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 \\ (H_4)_8 & -(H_4)_8 & (H_4)_8 & -(H_4)_8 \\ (H_2)_8 & -(H_6)_8 & -(H_2)_8 & (H_6)_8 \\ (H_6)_8 & (H_2)_8 & -(H_6)_8 & -(H_2)_8 \end{bmatrix} \left( \begin{bmatrix} \underline{\mathbf{f}}_0 \\ \underline{\mathbf{f}}_2 \\ \underline{\mathbf{f}}_4 \\ \underline{\mathbf{f}}_6 \end{bmatrix} + \begin{bmatrix} \underline{\mathbf{f}}_7 \\ \underline{\mathbf{f}}_5 \\ \underline{\mathbf{f}}_3 \\ \underline{\mathbf{f}}_1 \end{bmatrix} \right) \\
&= \begin{bmatrix} (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 \\ (H_2)_8 & -(H_6)_8 & -(H_2)_8 & (H_6)_8 \\ (H_4)_8 & -(H_4)_8 & (H_4)_8 & -(H_4)_8 \\ (H_6)_8 & (H_2)_8 & -(H_6)_8 & -(H_2)_8 \end{bmatrix} \left( \begin{bmatrix} \underline{\mathbf{f}}_0 \\ \underline{\mathbf{f}}_2 \\ \underline{\mathbf{f}}_4 \\ \underline{\mathbf{f}}_6 \end{bmatrix} + \begin{bmatrix} \underline{\mathbf{f}}_7 \\ \underline{\mathbf{f}}_5 \\ \underline{\mathbf{f}}_3 \\ \underline{\mathbf{f}}_1 \end{bmatrix} \right);
\end{aligned}
$$

$$
\begin{aligned}
Y_o &= \begin{bmatrix} \underline{\mathbf{y}}_1 \\ \underline{\mathbf{y}}_3 \\ \underline{\mathbf{y}}_5 \\ \underline{\mathbf{y}}_7 \end{bmatrix} = (L_8)_{32}\,(S_8)_{32}\,(\hat{\mathbf{E}}_8)_{32}\,(Q_8)_{32}\,(F_e - F_{\bar{o}}) \\
&= \begin{bmatrix} I_8 & 0_8 & 0_8 & 0_8 \\ -I_8 & 2I_8 & 0_8 & 0_8 \\ I_8 & -2I_8 & 2I_8 & 0_8 \\ -I_8 & 2I_8 & -2I_8 & 2I_8 \end{bmatrix} \begin{bmatrix} I_8 & 0_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & I_8 & 0_8 \\ 0_8 & I_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & 0_8 & I_8 \end{bmatrix} \begin{bmatrix} (H_0)_8 & (H_0)_8 & (H_0)_8 & (H_0)_8 \\ (H_4)_8 & -(H_4)_8 & (H_4)_8 & -(H_4)_8 \\ (H_2)_8 & -(H_6)_8 & -(H_2)_8 & (H_6)_8 \\ (H_6)_8 & (H_2)_8 & -(H_6)_8 & -(H_2)_8 \end{bmatrix} \\
&\qquad \times \begin{bmatrix} (H_1)_8 & 0_8 & 0_8 & 0_8 \\ 0_8 & (H_5)_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & (H_9)_8 & 0_8 \\ 0_8 & 0_8 & 0_8 & (H_{13})_8 \end{bmatrix} \left( \begin{bmatrix} \underline{\mathbf{f}}_0 \\ \underline{\mathbf{f}}_2 \\ \underline{\mathbf{f}}_4 \\ \underline{\mathbf{f}}_6 \end{bmatrix} - \begin{bmatrix} \underline{\mathbf{f}}_7 \\ \underline{\mathbf{f}}_5 \\ \underline{\mathbf{f}}_3 \\ \underline{\mathbf{f}}_1 \end{bmatrix} \right) \\
&= \begin{bmatrix} (H_1)_8 & (H_5)_8 & -(H_7)_8 & -(H_3)_8 \\ (H_3)_8 & -(H_1)_8 & (H_5)_8 & (H_7)_8 \\ (H_5)_8 & (H_7)_8 & -(H_3)_8 & (H_1)_8 \\ (H_7)_8 & (H_3)_8 & (H_1)_8 & (H_5)_8 \end{bmatrix} \left( \begin{bmatrix} \underline{\mathbf{f}}_0 \\ \underline{\mathbf{f}}_2 \\ \underline{\mathbf{f}}_4 \\ \underline{\mathbf{f}}_6 \end{bmatrix} - \begin{bmatrix} \underline{\mathbf{f}}_7 \\ \underline{\mathbf{f}}_5 \\ \underline{\mathbf{f}}_3 \\ \underline{\mathbf{f}}_1 \end{bmatrix} \right).
\end{aligned}
$$

Step 4:

$$
\mathbf{Y} = \begin{bmatrix} \underline{\mathbf{y}}_0 \\ \underline{\mathbf{y}}_1 \\ \underline{\mathbf{y}}_2 \\ \underline{\mathbf{y}}_3 \\ \underline{\mathbf{y}}_4 \\ \underline{\mathbf{y}}_5 \\ \underline{\mathbf{y}}_6 \\ \underline{\mathbf{y}}_7 \end{bmatrix} = (S_8)_{64} \begin{bmatrix} Y_e \\ Y_o \end{bmatrix} = \begin{bmatrix} I_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & 0_8 & 0_8 & I_8 & 0_8 & 0_8 & 0_8 \\ 0_8 & I_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & I_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & I_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & I_8 & 0_8 \\ 0_8 & 0_8 & 0_8 & I_8 & 0_8 & 0_8 & 0_8 & 0_8 \\ 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & 0_8 & I_8 \end{bmatrix} \begin{bmatrix} \underline{\mathbf{y}}_0 \\ \underline{\mathbf{y}}_2 \\ \underline{\mathbf{y}}_4 \\ \underline{\mathbf{y}}_6 \\ \underline{\mathbf{y}}_1 \\ \underline{\mathbf{y}}_3 \\ \underline{\mathbf{y}}_5 \\ \underline{\mathbf{y}}_7 \end{bmatrix}.
$$

Fig 1-(e) shows the abstract signal-flow graph for $N = 8$.

## 5 Computation of the 2-D DST by the 2-D DCT

In this section, we show how to compute the 2-D DST by using the 2-D DCT. For a given input data matrix $[x_{i,j}]$, $0 \le i,\ j \le N-1$, the 2-D DST output matrix $\mathbf{y}^s_{N\times N} = [y^s_{u,v}]$, $0 \le u,\ v \le N-1$, is defined by

$$y^s_{u,v} = \frac{2}{N}\,\epsilon^s(u)\,\epsilon^s(v)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_{i,j}\,\sin(\frac{(2i+1)(u+1)}{2N}\pi)\,\sin(\frac{(2j+1)(v+1)}{2N}\pi), \tag{20}$$

where

$$\epsilon^s(k) = \begin{cases} 1\,, & \text{for } 0 \le k \le N-2 \\ \frac{1}{\sqrt{2}}\,, & \text{for } k = N-1 \end{cases} \tag{21}$$

and $k = u$ or $v$. Similar to the 2-D DCT case, we will also remove $\epsilon(u)$, $\epsilon(v)$, and the normalization factor $\frac{2}{N}$, and rewrite Equation (20) in a simplified version

$$y^s_{u,v} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_{i,j}\,\sin(\frac{(2i+1)(u+1)}{2N}\pi)\,\sin(\frac{(2j+1)(v+1)}{2N}\pi). \tag{22}$$

Note that, for the sake of distinction, we use the superscript '$s$' for denoting the sine transform, and '$c$' for denoting the cosine transform.

We now state how to compute the 2-D DST using the 2-D DCT. First, by substituting

$$m = N-1-u \ \text{ and } \ n = N-1-v, \ \text{ for } 0 \le u,\ v \le N-1\,,$$

into the 2-D DCT formula in (2), we have

$$\begin{aligned} y^c_{N-1-u,N-1-v} &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_{i,j}\,\cos(\frac{(2i+1)(N-1-u)}{2N}\pi)\,\cos(\frac{(2j+1)(N-1-v)}{2N}\pi) \\ &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(-1)^{i+j}\,x_{i,j}\,\sin(\frac{(2i+1)(u+1)}{2N}\pi)\,\sin(\frac{(2j+1)(v+1)}{2N}\pi. \end{aligned} \tag{23}$$

Let

$$w_{i,j} = (-1)^{i+j}\,x_{i,j}, \ \text{ for } \ 0 \le i,\ j \le N-1,$$

and the 2-D DCT output of the data matrix $[w_{i,j}]$, $0 \leq i,\ j \leq N-1$, be denoted by $W^c_{u,v}$, $0 \leq u,\ v \leq N-1$. Then, from Equation (23), we have

$$\begin{aligned} W^c_{N-1-u,N-1-v} &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(-1)^{i+j}\, w_{i,j}\, \sin(\frac{(2i+1)(u+1)}{2N}\pi)\, \sin(\frac{(2j+1)(v+1)}{2N}\pi) \\ &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_{i,j}\, \sin(\frac{(2i+1)(u+1)}{2N}\pi)\, \sin(\frac{(2j+1)(v+1)}{2N}\pi) \\ &= X^s_{u,v}. \end{aligned}$$

We use the following procedure to summarize the above description for the computation of the 2-D DST by using the 2-D DCT.

**Computation of the 2-D DST by the 2-D DCT:**

**Step 1:** Compute the data matrix $[w_{i,j}]$, for $0 \leq i,\ j \leq N-1$, where $w_{i,j} = (-1)^{i+j}x_{i,j}$;

**Step 2:** evaluate the 2-D DCT output matrix $[W^c_{u,v}]$, $0 \leq u,\ v \leq N-1$, for the data matrix $[w_{i,j}]$, $0 \leq i,\ j \leq N-1$, by using our 2-D DCT algorithm;

**Step 3:** calculate $y^s_{u,v} = W^c_{N-1-u,N-1-v}$, for $0 \leq u,\ v \leq N-1$. □

# 6 Conclusions

A systematic method for designing the 2-D DCT algorithm has been presented in this paper. The method, which is based on certain intrinsic properties of the 2-D DCT, allows us to reduce the 2-D DCT problem to the 1-D DCT problem. We thus can design a 2-D DCT algorithm similar to a 1-D DCT algorithm. We also show that the 2-D DST can be computed using the 2-D DCT.

An important contribution of this paper is to provide a clear procedure for the computation. This allows us to testify that the algorithm is simple and the corresponding signal-flow graphs are regular. It also allows us to give a closed-form expression for the arithmetic complexity of this algorithm. In practice, for some special purpose hardware devices for implementing DSP applications, such as VLSI implementations, the cost of performing a multiplication may be higher

than that of performing an addition. If we are only interested in the number of scalar multiplications used in an algorithm, then our algorithm is one of the best algorithms found in published results.

## Acknowledgements

The authors would like to thank Professor Sang Uk Lee for providing his papers and suggesting this problem be researched.

## References

[1] S. C. Chan and K. L. Ho. A new two-dimensional fast Cosine transform algorithm. *IEEE Trans. Signal Processing*, 39(2):481–485, February 1991.

[2] N. I. Cho and S. U. Lee. Fast algorithm and implementation of 2-D discrete Cosine transform. *IEEE Trans. Circuits and Syst.*, 38(3):297–305, March 1991.

[3] N. I. Cho and S. U. Lee. A fast $4 \times 4$ DCT algorithm for the recursive 2-D DCT. *IEEE Trans. Signal Processing*, 40(9):2166–2173, September 1992.

[4] N. I. Cho, I. D. Yun, and S. U. Lee. On the regular structure for the fast 2-D DCT algorithm. *IEEE Trans. Circuits and Syst.*, 40(4):259–266, April 1993.

[5] P. Duhamel and C. Guillemot. Polynomial transform computation of the 2-D DCT. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, pages 1515–1518. IEEE, April 1990.

[6] M. A. Haque. A two-dimensional fast Cosine transform. *IEEE Trans. Acoust., Speech, and Signal Processing*, ASSP-33(6):1532–1539, December 1985.

[7] H. S. Hou. A fast recursive algorithm for computing the discrete Cosine transform. *IEEE Trans. Acoust., Speech, and Signal Processing*, ASSP-35(10):1455–1461, October 1987.

[8] F. A. Kamangar and K. R. Rao. Fast algorithms for the 2-D discrete Cosine transform. *IEEE Trans. Comput.*, C-31(9):899–906, September 1982.

[9] B. G. Lee. A new algorithm to compute the discrete Cosine transform. *IEEE Trans. Acoust., Speech, and Signal Processing*, ASSP-32(6):1243–1245, December 1984.

[10] P.-Z. Lee and F. Y. Huang. Restructured recursive DCT and DST algorithms. *IEEE Trans. Signal Processing*, accepted, also Technical Report TR-93-001, Institute of Information Science, Academia Sinica, 1993.

[11] R. K. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, and Applications.* Academic Press, Inc., 1990.

[12] M. Vetterli. Fast 2-D discrete Cosine transform. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, pages 1538–1541. IEEE, March 1985.