Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

ELA
http:/repository.uwyo.edu/ela

SIGNAL PROCESSING BASED ON STABLE RADIX-2 DCT I-IV ALGORITHMS HAVING ORTHOGONAL FACTORS*

SIRANI M. PERERA†

Abstract. This paper presents stable, radix-2, completely recursive discrete cosine transform algorithms DCT-I and DCT-III solely based on DCT-I, DCT-II, DCT-III, and DCT-IV having sparse and orthogonal factors. Error bounds for computing the completely recursive DCT-I, DCT-II, DCT-III, and DCT-IV algorithms having sparse and orthogonal factors are addressed. Signal flow graphs are demonstrated based on the completely recursive DCT-I, DCT-II, DCT-III, and DCT-IV algorithms having orthogonal factors. Finally image compression results are presented based on the recursive 2D DCT-II and DCT-IV algorithms for image size 512×512 pixels with transfer block sizes 8×8, 16×16, and 32×32 with 93.75% absence of coefficients in each transfer block.

Key words. Discrete cosine transform, Sparse and orthogonal factors, Radix-2 algorithms, Recursive algorithms, Stable algorithms, Arithmetic cost, Error bounds, Image reconstruction, Signal flow graphs.

AMS subject classifications. 15A23, 15B10, 65F35, 65F50, 65T50, 65Y05, 65Y20, 94A08, 94A12.

1. Introduction.

The Fast Fourier Transform is used to efficiently compute the Discrete Fourier Transform (DFT) and its inverse. The DFTs are widely used in numerous applications in applied mathematics and electrical engineering [3, 19, 23, 24, 27, 30], etc.

The DFT uses complex arithmetic. The DFT of an n-input sequence $\{x_k\}_{k=0}^{n-1}$ is the n-output sequence $\{y_k\}_{k=0}^{n-1}$ defined via

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix} = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \cdots & \omega_n^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & \omega_n^{n-1} & \cdots & \omega_n^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix}, \tag{1.1}$$

where $\omega_n = e^{-\frac{2\pi i}{n}}$. There exist real analogues of the DFT, namely the Discrete Cosine Transforms and Discrete Sine Transforms, the main types are from I to IV. Similar to (1.1), the I-IV variants of cosine and sine matrices transform an n-input sequence into the n-output sequence via the transform matrices stated in Table 1.1, where for DCT-I $j, k = 0, 1, \ldots, n$, DST-I $j, k = 0, 1, \ldots, n-2$, DCT and DST II-IV $j, k = 0, 1, \ldots, n-1$, $\epsilon_n(0) = \epsilon_n(n) = \frac{1}{\sqrt{2}}$, $\epsilon_n(j) = 1$ for $j \in \{1, 2, \ldots, n-1\}$ and $n \geq 2$ is an even integer. Among DCT I-IV transforms, C_{n+1}^I was introduced in [31], C_n^{II} and its inverse C_n^{III} were introduced in [1], and C_n^{IV} was introduced into digital signal processing in [9]. Moreover, among DST I-IV transforms, S_{n-1}^I

*Received by the editors on January 2, 2016. Accepted for publication on April 20, 2016. Handling Editor: James G. Nagy.

†Department of Mathematics, Embry-Riddle Aeronautical University, Daytona Beach, Florida 32114, USA (pereras2@erau.edu).

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

ELA
http:/repository.uwyo.edu/ela

Cosine and Sine Transforms	Inverse Transforms
$C^I_{n+1} = \sqrt{\frac{2}{n}}\left[\epsilon_n(j)\,\epsilon_n(k)\cos\frac{jk\pi}{n}\right]$	$\left[C^I_{n+1}\right]^{-1} = C^I_{n+1}$
$C^{II}_n = \sqrt{\frac{2}{n}}\left[\epsilon_n(j)\cos\frac{j(2k+1)\pi}{2n}\right]$	$\left[C^{II}_n\right]^{-1} = C^{III}_n$
$C^{III}_n = \sqrt{\frac{2}{n}}\left[\epsilon_n(k)\cos\frac{(2j+1)k\pi}{2n}\right]$	$\left[C^{III}_n\right]^{-1} = C^{II}_n$
$C^{IV}_n = \sqrt{\frac{2}{n}}\left[\cos\frac{(2j+1)(2k+1)\pi}{4n}\right]$	$\left[C^{IV}_n\right]^{-1} = C^{IV}_n$
$S^I_{n-1} = \sqrt{\frac{2}{n}}\left[\sin\frac{(j+1)(k+1)\pi}{n}\right]$	$\left[S^I_{n-1}\right]^{-1} = S^I_{n-1}$
$S^{II}_n = \sqrt{\frac{2}{n}}\left[\epsilon_n(j+1)\sin\frac{(j+1)(2k+1)\pi}{2n}\right]$	$\left[S^{II}_n\right]^{-1} = S^{III}_n$
$S^{III}_n = \sqrt{\frac{2}{n}}\left[\epsilon_n(k+1)\sin\frac{(2j+1)(k+1)\pi}{2n}\right]$	$\left[S^{III}_n\right]^{-1} = S^{II}_n$
$S^{IV}_n = \sqrt{\frac{2}{n}}\left[\sin\frac{(2j+1)(2k+1)\pi}{4n}\right]$	$\left[S^{IV}_n\right]^{-1} = S^{IV}_n$

Table 1.1: Cosine and sine transform matrices.

and S^{IV}_n were introduced in [9, 10] and S^{II}_n and its inverse S^{III}_n were introduced in [14]. These classifications were also stated in [19, 30].

It has been stated, in e.g. [21, 22, 24], that these cosine and sine matrices of types I-IV are orthogonal. Strang, in [24], proved that the column vectors of each cosine matrix are eigenvectors of a symmetric second difference matrix under different boundary conditions, and are hence orthogonal. Later Britanak, Yip, and Rao in [3] followed very closely the presentation made by Strang's [24] to point out that the column vectors of each cosine and sine matrix of types I-VIII are eigenvectors of a symmetric second difference matrix. Due to these DCT and DST properties, it was shown by many authors (see e.g. [2, 3, 4, 6, 7, 11, 12, 13, 14, 15, 16, 17, 24, 28, 29]) that these symmetric and asymmetric (rarely used) versions of DCT and DST can be widely used in image processing, signal processing, finger print enhancement, quick response code (QR code), etc.

To obtain real, fast DCT or DST algorithms one can mainly use a polynomial arithmetic technique or a matrix factorization technique. In the polynomial arithmetic technique (see e.g. [25]), components of $C_n\,\mathbf{x}$ or $S_n\,\mathbf{x}$ are interpreted as the nodes of a degree n polynomial, and then one applies the divide and conquer technique to reduce the degree of the polynomial. The matrix factorization technique is the direct factorization of the DCT or DST matrices into the product of sparse matrices (see e.g. [3, 18, 19, 30, 32]). Later it was found (see e.g. [26]) that if the factorization for DCT or DST does not preserve orthogonality the resulting DCT or DST algorithms lead to inferior numerical stability. The matrix factorization for DST-I in [32] used the results in [5] to decompose DST-I into DCT and DST. Also the decomposition for DCT-II in [30] is a slightly different version of the result in [5]. Though one can find orthogonal matrix factorizations for DCT and DST in [30], the resulting algorithms in [30] are not completely recursive, and hence do not lead to simple recursive algorithms. Moreover, [3] has used the same factorization for DST-II and DST-IV as in [30]. On the other hand, one can use these [3, 24, 30] results to derive recursive, stable

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

algorithms as stated in [18, 19].

However, [19] has offered stable, recursive DCT-II and DCT-IV algorithms, based on DCT-II and DCT-IV. Thus, this paper completes the picture and provides completely recursive, stable, radix-2 DCT-I and DCT-III algorithms that are solely defined via DCT I-IV, having sparse and orthogonal factors. The paper also addresses the error bounds on computing completely recursive algorithms for DCT I-IV. Moreover, this paper elaborates signal transform designs and image compression results (absence of 93.75% coefficients in each transfer block) based on the completely recursive DCT I-IV algorithms.

In Section 2, we derive factorizations for DCT-I and DCT-III having orthogonal and sparse matrices, and state completely recursive DCT I-IV algorithms solely defined via DCT I-IV having sparse, orthogonal, and rotation/rotation-reflection matrices. Next, in Section 3, we present the arithmetic cost of computing these algorithms. In Section 4, we derive error bounds in computing these algorithms and discuss the stability. Finally, in Sections 5 and 6 respectively, we demonstrate signal flow graphs and image compression results based on these completely recursive DCT I-IV algorithms.

2. Completely recursive radix-2 DCT algorithms having orthogonal factors. This section introduces sparse and orthogonal factorizations for DCT-I and DCT-III matrices. In the meantime, we present completely recursive, radix-2 DCT I-IV algorithms solely defined via DCT I-IV, having sparse, orthogonal, and butterfly matrices. One can observe a variant of the DCT-II and DCT-IV algorithms having almost orthogonal factors in [19].

The following notations and sparse matrices are used frequently in this paper. Denote an involution matrix $\tilde{I}_n$ by $\tilde{I}_n\,\mathbf{x} = [x_{n-1}, x_{n-2}, \ldots, x_0]^T$, a diagonal matrix D_n by $D_n\,\mathbf{x} = \operatorname{diag}\left((-1)^k\right)_{k=0}^{n-1}\mathbf{x}$, an even-odd permutation matrix P_n $(n \geq 3)$ by

$$P_n\,\mathbf{x} = \begin{cases} [x_0, x_2, \ldots, x_{n-2}, x_1, x_3, \ldots, x_{n-1}]^T \text{ even } n, \\ [x_0, x_2, \ldots, x_{n-1}, x_1, x_3, \ldots, x_{n-2}]^T \text{ odd } n, \end{cases}$$

for any $\mathbf{x} = [x_j]_{j=0}^{n-1}$, and orthogonal matrices $(n \geq 4)$ by

$$\breve{H}_{n+1} = \frac{1}{\sqrt{2}} \begin{bmatrix} I_{\frac{n}{2}} & & \widetilde{I}_{\frac{n}{2}} \\ & \sqrt{2} & \\ I_{\frac{n}{2}} & & -\widetilde{I}_{\frac{n}{2}} \end{bmatrix}, \; H_n = \frac{1}{\sqrt{2}} \begin{bmatrix} I_{\frac{n}{2}} & \widetilde{I}_{\frac{n}{2}} \\ I_{\frac{n}{2}} & -\widetilde{I}_{\frac{n}{2}} \end{bmatrix},$$

$$U_n = \begin{bmatrix} 1 & & & \\ & \frac{1}{\sqrt{2}} \begin{bmatrix} I_{\frac{n}{2}-1} & I_{\frac{n}{2}-1} \\ I_{\frac{n}{2}-1} & -I_{\frac{n}{2}-1} \end{bmatrix} & & \\ & & & -1 \end{bmatrix} \begin{bmatrix} I_{\frac{n}{2}} & \\ & D_{\frac{n}{2}} \widetilde{I}_{\frac{n}{2}} \end{bmatrix},$$

$$R_n = \begin{bmatrix} I_{\frac{n}{2}} & \\ & D_{\frac{n}{2}} \end{bmatrix} \begin{bmatrix} \operatorname{diag} C_{\frac{n}{2}} & \left(\operatorname{diag} S_{\frac{n}{2}}\right) \widetilde{I}_{\frac{n}{2}} \\ -\widetilde{I}_{\frac{n}{2}} \left(\operatorname{diag} S_{\frac{n}{2}}\right) & \operatorname{diag}\left(\widetilde{I}_{\frac{n}{2}} C_{\frac{n}{2}}\right) \end{bmatrix},$$

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

where for $k = 0, 1, \ldots, \frac{n}{2} - 1$,

$$C_{\frac{n}{2}} = \left[\cos\frac{(2k+1)\pi}{4n}\right] \quad \text{and} \quad S_{\frac{n}{2}} = \left[\sin\frac{(2k+1)\pi}{4n}\right].$$

DCT-II and DCT-IV algorithms are the keys for the completely recursive procedure, so for a given vector $\mathbf{x} \in \mathbb{R}^n$, we present algorithms in order $\mathbf{y} = C_n^{II}\,\mathbf{x}$, $\mathbf{y} = C_n^{IV}\,\mathbf{x}$, $\mathbf{y} = C_n^{III}\,\mathbf{x}$, and $\mathbf{y} = C_{n+1}^{I}\,\mathbf{x}$. Following the matrix factorizations for DCT-II and DCT-IV in [19], let us first state recursive DCT-II and DCT-IV having orthogonal factors via algorithms (2.1) and (2.2), respectively.

ALGORITHM 2.1. $(\mathbf{cos2}(\mathbf{x}, \mathbf{n}))$
Input: $n = 2^t (t \geq 1)$, $n_1 = \frac{n}{2}$, $\mathbf{x} \in \mathbb{R}^n$.

1. *If* $n = 2$, *then*
$$\mathbf{y} := \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \mathbf{x}.$$
2. *If* $n \geq 4$, *then*
$$[u_j]_{j=0}^{n-1} := H_n\,\mathbf{x},$$
$$\mathbf{z1} := \mathbf{cos2}\left([u_j]_{j=0}^{n_1-1}, n_1\right),$$
$$\mathbf{z2} := \mathbf{cos4}\left([u_j]_{j=n_1}^{n-1}, n_1\right),$$
$$\mathbf{y} := P_n^T \left(\mathbf{z1}^T, \mathbf{z2}^T\right)^T.$$

Output: $\mathbf{y} = C_n^{II}\mathbf{x}$.

ALGORITHM 2.2. $(\mathbf{cos4}(\mathbf{x}, \mathbf{n}))$
Input: $n = 2^t (t \geq 1)$, $n_1 = \frac{n}{2}$, $\mathbf{x} \in \mathbb{R}^n$.

1. *If* $n = 2$, *then*
$$\mathbf{y} := \begin{bmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} \end{bmatrix} \mathbf{x}.$$
2. *If* $n \geq 4$, *then*
$$[u_j]_{j=0}^{n-1} := R_n\,\mathbf{x},$$
$$\mathbf{z1} := \mathbf{cos2}\left([u_j]_{j=0}^{n_1-1}, n_1\right),$$
$$\mathbf{z2} := \mathbf{cos2}\left([u_j]_{j=n_1}^{n-1}, n_1\right),$$
$$\mathbf{w} := U_n \left(\mathbf{z1}^T, \mathbf{z2}^T\right)^T,$$
$$\mathbf{y} := P_n^T \mathbf{w}.$$

Output: $\mathbf{y} = C_n^{IV}\mathbf{x}$.

By using the well known transpose property between DCT-II and DCT-III we can state an algorithm for DCT-III via (2.3). This algorithm executes recursively with the DCT-II and DCT-IV algorithms.

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

ALGORITHM 2.3. ($\mathbf{cos3}(\mathbf{x}, \mathbf{n})$)
Input: $n = 2^t (t \geq 1)$, $n_1 = \frac{n}{2}$, $\mathbf{x} \in \mathbb{R}^n$.

1. *If* $n = 2$, *then*
$$\mathbf{y} := \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \mathbf{x}.$$
2. *If* $n \geq 4$, *then*
$$[u_j]_{j=0}^{n-1} := P_n \, \mathbf{x},$$
$$\mathbf{z1} := \mathbf{cos3}\left([u_j]_{j=0}^{n_1 - 1}, n_1 \right),$$
$$\mathbf{z2} := \mathbf{cos4}\left([u_j]_{j=n_1}^{n-1}, n_1 \right),$$
$$\mathbf{y} := H_n^T \left(\mathbf{z1}^T, \mathbf{z2}^T \right)^T.$$

Output: $\mathbf{y} = C_n^{III} \mathbf{x}$.

Before stating the algorithm for DCT-I let us derive a sparse and orthogonal factorization for DCT-I.

LEMMA 2.4. *Let $n \geq 4$ be an even integer. The matrix C_{n+1}^I can be factored in the form*

$$C_{n+1}^I = P_{n+1}^T \left[\begin{array}{c|c} C_{\frac{n}{2}+1}^I & 0 \\ \hline 0 & C_{\frac{n}{2}}^{III} \end{array} \right] \breve{H}_{n+1}. \tag{2.1}$$

Proof. Let's apply P_{n+1} to C_{n+1}^I to permute rows and then partition the resultant matrix. So, (1,1) block becomes

$$\sqrt{\frac{2}{n}} \left[\epsilon_n(2j) \epsilon_n(k) \cos \frac{2jk\pi}{n} \right]_{j,k=0}^{\frac{n}{2}},$$

(1,2) block becomes

$$\begin{aligned} &\sqrt{\frac{2}{n}} \left[\epsilon_n(2j) \epsilon_n \left(\frac{n}{2} + k + 1 \right) \cos \frac{j(n + 2k + 2)\pi}{n} \right]_{j,k=0}^{\frac{n}{2}, \frac{n}{2}-1} \\ &= \sqrt{\frac{2}{n}} \left[\epsilon_n(2j) \epsilon_n \left(\frac{n}{2} + k + 1 \right) \cos \frac{j(n - 2k - 2)\pi}{n} \right]_{j,k=0}^{\frac{n}{2}, \frac{n}{2}-1}, \end{aligned}$$

(2,1) block becomes

$$\sqrt{\frac{2}{n}} \left[\epsilon_n(k) \cos \frac{(2j+1)k\pi}{n} \right]_{j,k=0}^{\frac{n}{2}-1, \frac{n}{2}},$$

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

and (2,2) block becomes

$$\sqrt{\frac{2}{n}}\left[\epsilon_n\left(\frac{n}{2}+k+1\right)\cos\frac{(2j+1)(n+2k+2)\pi}{2n}\right]_{j,k=0}^{\frac{n}{2}-1}$$
$$=\sqrt{\frac{2}{n}}\left[-\epsilon_n\left(\frac{n}{2}+k+1\right)\cos\frac{(2j+1)(n-2k-2)\pi}{2n}\right]_{j,k=0}^{\frac{n}{2}-1}.$$

Hence,

$$P_{n+1}C_{n+1}^{I}=\frac{1}{\sqrt{2}}\left[\begin{array}{c|c} C_{\frac{n}{2}+1}^{I}\begin{bmatrix} I_{\frac{n}{2}} & 0\\ 0 & \sqrt{2}\end{bmatrix} & C_{\frac{n}{2}+1}^{I}\begin{bmatrix}\tilde{I}_{\frac{n}{2}}\\ 0\end{bmatrix}\\ \hline C_{\frac{n}{2}}^{III}\begin{bmatrix} I_{\frac{n}{2}} & 0\end{bmatrix} & -C_{\frac{n}{2}}^{III}\,\tilde{I}_{\frac{n}{2}}\end{array}\right]$$
$$=\left[\begin{array}{c|c} C_{\frac{n}{2}+1}^{I} & 0\\ \hline 0 & C_{\frac{n}{2}}^{III}\end{array}\right]\breve{H}_{n+1}. \quad \square$$

Thus, an algorithm for DCT-I can be stated via (2.5), which executes recursively with DCT II-IV algorithms.

ALGORITHM 2.5. ($\mathbf{cos1}(\mathbf{x},\mathbf{n}+\mathbf{1})$)
Input: $n=2^t(t\geq 1)$, $n_1=\frac{n}{2}$, $\mathbf{x}\in\mathbb{R}^{n+1}$.

1. *If* $n=2$, *then*
$$\mathbf{y}:=\tfrac{1}{2}\begin{bmatrix}1 & 1 & 0\\ 0 & 0 & \sqrt{2}\\ 1 & -1 & 0\end{bmatrix}\begin{bmatrix}1 & 0 & 1\\ 0 & \sqrt{2} & 0\\ 1 & 0 & -1\end{bmatrix}\mathbf{x}.$$
2. *If* $n\geq 4$, *then*
$$[u_j]_{j=0}^{n}:=\breve{H}_{n+1}\,\mathbf{x},$$
$$\mathbf{z1}:=\mathbf{cos1}\left([u_j]_{j=0}^{n_1},n_1+1\right),$$
$$\mathbf{z2}:=\mathbf{cos3}\left([u_j]_{j=n_1+1}^{n},n_1\right),$$
$$\mathbf{y}:=P_{n+1}^{T}\left(\mathbf{z1}^T,\mathbf{z2}^T\right)^T.$$

Output: $\mathbf{y}=C_{n+1}^{I}\mathbf{x}$.

3. Arithmetic cost of computing DCT algorithms. We first calculate the arithmetic cost of computing DCT I-IV algorithms. Let's denote the number of additions and multiplications required to compute, say a length n DCT-II algorithm: $\mathbf{y}=C_n^{II}\,\mathbf{x}$ by $\#a(\text{DCT-II},n)$ and $\#m(\text{DCT-II},n)$. Note that the multiplication of ± 1 and permutations are not counted. Once the cost is computed we show numerical results for the speed improvement factor of these algorithms.

3.1. Number of additions and multiplications in computing DCT I-IV algorithms. Here we calculate the arithmetic cost of computing the DCT I-IV algorithms in order (2.1), (2.2), (2.3) and (2.5). The cost of addition in computing DCT-II and DCT-IV

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

algorithms is the same as in [19], but the cost of multiplication is different from [19]. The latter is because in this paper, not only DCT-I and DCT-III algorithms but also DCT-II and DCT-IV algorithms have orthogonal vectors instead of almost orthogonal vectors. Let us first derive explicitly the number of multiplications required to compute DCT-II and DCT-IV algorithms and then the arithmetic cost of DCT-III and DCT-I algorithms, respectively.

LEMMA 3.1. *Let $n = 2^t (t \geq 2)$ be given. Using algorithms* (2.1) *and* (2.2)*, the arithmetic cost of computing length n DCT-II algorithm is given by*

$$\begin{aligned} \#a(DCT\text{-}II, n) &= \frac{4}{3}nt - \frac{8}{9}n - \frac{1}{9}(-1)^t + 1, \\ \#m(DCT\text{-}II, n) &= \frac{5}{3}nt - \frac{10}{9}n + \frac{1}{9}(-1)^t + 1. \end{aligned} \tag{3.1}$$

Proof. Following algorithms (2.1) and (2.2)

$$\begin{aligned} \#m(\text{DCT-II}, n) &= \#m\left(\text{DCT-II}, \frac{n}{2}\right) + \#m\left(\text{DCT-IV}, \frac{n}{2}\right) + \#m\left(H_n\right), \\ \#m\left(\text{DCT-IV}, n\right) &= \#m\left(U_n\right) + 2 \cdot \#m\left(\text{DCT-II}, \frac{n}{2}\right) + \#m\left(R_n\right). \end{aligned} \tag{3.2}$$

By referring to the structures of H_n, U_n, and R_n

$$\begin{aligned} \#a\left(H_n\right) = n, \ \ \#m\left(H_n\right) = n, \\ \#a\left(U_n\right) = n-2, \ \ \#m\left(U_n\right) = n-2, \\ \#a\left(R_n\right) = n, \ \ \#m\left(R_n\right) = 2n. \end{aligned} \tag{3.3}$$

Thus,

$$\#m(\text{DCT-II}, n) = \#m\left(\text{DCT-II}, \frac{n}{2}\right) + 2 \cdot \#m\left(\text{DCT-II}, \frac{n}{4}\right) + \frac{5}{2}n - 2.$$

Since $n = 2^t$ we can obtain the second order linear difference equation with respect to $t(\geq 3)$

$$\#m(\text{DCT-II}, 2^t) - \#m\left(\text{DCT-II}, 2^{t-1}\right) - 2 \cdot \#m\left(\text{DCT-II}, 2^{t-2}\right) = 5 \cdot 2^{t-1} - 2.$$

If $\#m(\text{DCT-II}, 2^t) = \alpha^t$(where $\alpha \neq 0$) is a solution then the above follows

$$\alpha^t - \alpha^{t-1} - 2\left(\alpha^{t-2}\right) = 5 \cdot 2^{t-1} - 2. \tag{3.4}$$

The homogeneous solution of the above is given by solving the characteristic equation

$$\alpha^{t-2}(\alpha^2 - \alpha - 2) = 0.$$

From which we get

$$\#m(\text{DCT-II}, 2^t) = r_1 2^t + r_2(-1)^t + \text{particular solution},$$

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

where r_1 and r_2 are constants. Let $\alpha^t = r_3 + r_4 t \cdot 2^t$ (where r_3 and r_4 are constants) be the particular solution. Substituting this potential equation into (3.4) and equating the coefficients we can find that

$$\#m(\text{DCT-II}, 2^t) = r_1 2^t + r_2(-1)^t + \frac{5}{3} \cdot t \cdot 2^t + 1.$$

Using the initial conditions $\#m\,(\text{DCT-II}, 2) = 2$ and $\#m\,(\text{DCT-II}, 4) = 10$, we can determine the general solution

$$\#m(\text{DCT-II}, 2^t) = \frac{5}{3} \cdot t \cdot 2^t - \frac{10}{9} 2^t + \frac{1}{9}(-1)^t + 1. \tag{3.5}$$

Thus, substituting $n = 2^t$ we can obtain the number of multiplications required to compute DCT-II algorithm as stated in (3.1).

Again by algorithms (2.1) and (2.2) together with (3.3), we can state

$$\#a(\text{DCT-II}, n) = \#a\left(\text{DCT-II}, \frac{n}{2}\right) + 2 \cdot \#a\left(\text{DCT-II}, \frac{n}{4}\right) + 2n - 2.$$

Since $n = 2^t$ we can obtain the second order linear difference equation with respect to $t(\geq 3)$

$$\#a(\text{DCT-II}, 2^t) - \#a\left(\text{DCT-II}, 2^{t-1}\right) - 2 \cdot \#a\left(\text{DCT-II}, 2^{t-2}\right) = 2^{t+1} - 2.$$

As derived analogously in the cost of multiplication, we can solve the above equation under the initial conditions $\#a\,(\text{DCT-II}, 2) = 2$ and $\#a\,(\text{DCT-II}, 4) = 8$ to obtain

$$\#a(\text{DCT-II}, n) = \frac{4}{3} nt - \frac{8}{9} n - \frac{1}{9}(-1)^t + 1. \quad \square \tag{3.6}$$

COROLLARY 3.2. *Let $n = 2^t$ $(t \geq 2)$ be given. Using algorithms* (2.2) *and* (2.1)*, the arithmetic cost of computing length n DCT-IV algorithm is given by*

$$\begin{aligned} \#a(\textit{DCT-IV}, n) &= \frac{4}{3} nt - \frac{2}{9} n + \frac{2}{9}(-1)^t, \\ \#m(\textit{DCT-IV}, n) &= \frac{5}{3} nt + \frac{2}{9} n - \frac{2}{9}(-1)^t. \end{aligned} \tag{3.7}$$

Proof. The number of multiplications required to compute DCT-IV algorithm can be found by substituting (3.5) at $\frac{n}{2} (= 2^{t-1})$ into the equation (3.2)

$$\#m(\text{DCT-IV}, n) = n - 2 + 2\left(\frac{5}{3} \cdot \frac{n}{2}(t-1) - \frac{10}{9} \cdot \frac{n}{2} + \frac{1}{9}(-1)^{t-1} + 1\right) + 2n.$$

Simplifying the above gives the cost of multiplication

$$\#m(\text{DCT-IV}, n) = \frac{5}{3} nt + \frac{2}{9} n - \frac{2}{9}(-1)^t.$$

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

Similarly, the number of additions required to compute DCT-IV algorithm can be found by substituting (3.6) at $\frac{n}{2}(= 2^{t-1})$ to

$$\begin{aligned}\#a\,(\text{DCT-IV}, n) &= \#a\,(U_n) + 2 \cdot \#a\left(\text{DCT-II}, \frac{n}{2}\right) + \#a\,(R_n)\\ &= 2 \cdot \#a\left(\text{DCT-II}, \frac{n}{2}\right) + 2n - 2.\end{aligned}$$

Simplifying the above yields

$$\#a(\text{DCT-IV}, n) = \frac{4}{3}nt - \frac{2}{9}n + \frac{2}{9}(-1)^t. \quad \square$$

The DCT-III algorithm (2.3) was stated using the transpose property of matrices so the following corollary is trivial.

COROLLARY 3.3. *Let $n = 2^t$ $(t \geq 2)$ be given. If DCT-III could be computed by using algorithms* (2.3)*,* (2.2)*, and* (2.1)*, then the arithmetic cost of computing a length n DCT-III algorithm is given by*

$$\begin{aligned}\#a(\textit{DCT-III}, n) &= \frac{4}{3}nt - \frac{8}{9}n - \frac{1}{9}(-1)^t + 1,\\ \#m(\textit{DCT-III}, n) &= \frac{5}{3}nt - \frac{10}{9}n + \frac{1}{9}(-1)^t + 1.\end{aligned} \tag{3.8}$$

REMARK 3.4. By using the DCT-III algorithm (2.3) and the arithmetic cost of computing the DCT-IV algorithm (in corollary (3.2)), one can obtain the same results as in corollary (3.3).

Let us state the arithmetic cost of computing the DCT-I algorithm (2.5).

LEMMA 3.5. *Let $n = 2^t$ $(t \geq 2)$ be given. Using algorithms* (2.5)*,* (2.3)*,* (2.2) *and* (2.1)*, the arithmetic cost of a DCT-I algorithm of length $n+1$ is given by*

$$\begin{aligned}\#a\,(\textit{DCT-I}, n+1) &= \frac{4}{3}nt - \frac{14}{9}n + \frac{1}{18}(-1)^t + t + \frac{7}{2},\\ \#m\,(\textit{DCT-I}, n+1) &= \frac{5}{3}nt - \frac{22}{9}n - \frac{1}{18}(-1)^t + t + \frac{11}{2}.\end{aligned} \tag{3.9}$$

Proof. Referring to the DCT-I algorithm (2.5)

$$\#a\,(\text{DCT-I}, n+1) = \#a\,\left(\text{DCT-I}, \frac{n}{2} + 1\right) + \#a\,\left(\text{DCT-III}, \frac{n}{2}\right) + \#a\,\left(\breve{H}_{n+1}\right). \tag{3.10}$$

The structure of $\breve{H}_{n+1}$ leads to $\#a\left(\breve{H}_{n+1}\right) = n$. This together with the arithmetic cost of computing DCT-III (3.8) algorithm, we can rewrite (3.10)

$$\#a\,(\text{DCT-I}, n+1) = \#a\,\left(\text{DCT-I}, \frac{n}{2} + 1\right) + \frac{2}{3}nt - \frac{1}{9}n + \frac{1}{9}(-1)^t + 1.$$

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

Since $n = 2^t$ we can obtain the first order linear difference equation with respect to $t(\geq 2)$

$$\#a\,(\text{DCT-I}, 2^t + 1) - \#a\,\left(\text{DCT-I}, 2^{t-1} + 1\right) = \frac{2}{3}t \cdot 2^t - \frac{1}{9}2^t + \frac{1}{9}(-1)^t + 1. \tag{3.11}$$

We can obtain the number of additions required to compute the DCT-I algorithm by solving (3.11) with initial condition $\#a\ (\text{DCT-I}, 3) = 4$. Analogously, one can solve the first order linear difference equation with initial condition $\#m\ (\text{DCT-I}, 3) = 5$ to obtain the number of multiplications. □

3.2. Speed improvement factor of DCT I-IV algorithms. Based on the results in lemmas 3.1, 3.5 and corollaries 3.2, 3.3, we graph the speed improvement factor of DCT I-IV algorithms having orthogonal factors. It is known that the speed improvement factor plays a critical role in the DFT algorithms as it gives us an idea about the processing speed of the algorithms. We should recall here that this factor increases with the size of matrix.

In our case, the speed improvement factor says the ratio between the number of additions and multiplications required to compute the DCT I-IV algorithms, and the direct computation cost of computing these algorithms which is $2n^2 - n$ for DCT II-IV, and $2n^2 + 3n + 1$ for DCT-I. Figure 3.1 shows the speed improvement factor corresponding to the DCT I-IV algorithms with respect to the size of matrix. These numerical data correspond to MATLAB (R2014a version) with machine precision 2.2×10^{-16}.

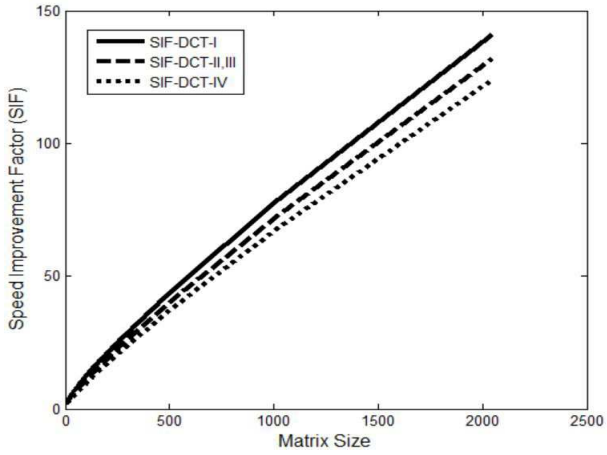


Fig. 3.1: Speed improvement factor of DCT I-IV algorithms.

4. Error bounds and stability of DCT algorithms. Error bounds and stability of computing the DCT I-IV algorithms are the main concern in this section. Here, to verify the stability, we will use error bounds (using perturbation of the product of matrices stated in [8]) in computing these algorithms. Let us assume that the computed trigonometry functions ($d_r := \sin \frac{r\pi}{4n}$ or $\cos \frac{r\pi}{4n}$ are the entries of the butterfly matrix) $\widehat{d}_r$ are used and satisfy

$$\widehat{d}_r = d_r + \epsilon_r, \quad |\epsilon_r| \leq \mu \tag{4.1}$$

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

ELA
http:/repository.uwyo.edu/ela

for all $r = 1, 3, 5, \ldots, n-1$, where $\mu := O(u)$ and u is the unit roundoff.

Let's recall the perturbation of the product of matrices stated in [8, Lemma 3.7], i.e., if $A_k + \Delta A_k \in \mathbb{R}^{n\times n}$ satisfies $|\Delta A_k| \le \delta_k |A_k|$ for all k, then

$$\left| \prod_{k=0}^{m} (A_k + \Delta A_k) - \prod_{k=0}^{m} A_k \right| \le \left(\prod_{k=0}^{m} (1+\delta_k) - 1 \right) \prod_{k=0}^{m} |A_k|,$$

where $|\delta_k| < u$. Moreover, recall $\prod_{k=1}^{n} (1+\delta_k)^{\pm 1} = 1 + \theta_n$ where $|\theta_n| \le \frac{nu}{1-nu} =: \gamma_n$ and $\gamma_k + u \le \gamma_{k+1}$, $\gamma_k + \gamma_j + \gamma_k\gamma_j \le \gamma_{k+j}$ from [8, Lemmas 3.1 and 3.3].

We explicitly derive an error bound for computing the DCT-I algorithm (as the rest follow analogously).

THEOREM 4.1. *Let $\widehat{\mathbf{y}} = fl(C^I_{n+1}\mathbf{x})$, where $n = 2^t (t \ge 2)$, be computed using the algorithms* (2.5), (2.3), (2.2), (2.1), *and assume that (4.1) holds. Then*

$$\frac{\|\mathbf{y} - \widehat{\mathbf{y}}\|_2}{\|\mathbf{y}\|_2} \le \frac{\gamma_7 t}{1 - \gamma_7 t}. \tag{4.2}$$

Proof. Using the algorithms (2.5), (2.3), (2.2), (2.1), and the computed matrices $\widehat{\mathbf{B}}_k$ (in terms of the computed $\widehat{d_r}$) for $k = 2, 3, \cdots, t-2$:

$$\begin{aligned}
\widehat{\mathbf{y}} &= fl\left(\mathbf{A}_0\, \mathbf{A}_1 \cdots \mathbf{A}_{t-2}\, \mathbf{C}_{t-1}\, \widehat{\mathbf{B}}_{t-2} \cdots \widehat{\mathbf{B}}_2 \mathbf{B}_1 \mathbf{B}_0\, \mathbf{x}\right) \\
&= (\mathbf{A}_0 + \Delta\mathbf{A}_0) \cdots (\mathbf{A}_{t-2} + \Delta\mathbf{A}_{t-2})\, (\mathbf{C}_{t-1} + \Delta\mathbf{C}_{t-1}) \\
&\quad (\widehat{\mathbf{B}}_{t-2} + \Delta\widehat{\mathbf{B}}_{t-2}) \cdots (\widehat{\mathbf{B}}_2 + \Delta\widehat{\mathbf{B}}_2)(\mathbf{B}_1 + \Delta\mathbf{B}_1)(\mathbf{B}_0 + \Delta\mathbf{B}_0)\mathbf{x}.
\end{aligned}$$

Each $\mathbf{A}_k$ is formed containing a combination of matrices $P^T_{\frac{n}{2^k}+1}$, $P^T_{\frac{n}{2^k}}$, $H^T_{\frac{n}{2^k}}$ and $U_{\frac{n}{2^k}}$ except $\mathbf{A}_0 = P^T_{n+1}$ and $\mathbf{A}_1 = \text{blkdiag}\left(P^T_{\frac{n}{2}+1}, H^T_{\frac{n}{2}}\right)$. Using the fact that each row in $\mathbf{A}_k$ has at most two non-zero entries with mostly ones per row:

$$|\Delta\mathbf{A}_0| = 0, \qquad |\Delta\mathbf{A}_k| \le \gamma_2\, |\mathbf{A}_k| \quad \text{for} \quad k = 1, 2, \ldots, t-2.$$

Also each $\mathbf{B}_k$ is formed containing a combination of matrices $\breve{H}_{\frac{n}{2^k}+1}$, $H_{\frac{n}{2^k}}$, $P_{\frac{n}{2^k}}$ and $R_{\frac{n}{2^k}}$ except $\mathbf{B}_0 = \breve{H}_{n+1}$ and $\mathbf{B}_1 = \text{blkdiag}\left(\breve{H}_{\frac{n}{2}+1}, P_{\frac{n}{2}}\right)$. Using the fact that each row in $\mathbf{B}_k$ has at most two non-zero entries per row:

$$|\Delta\mathbf{B}_0| \le \gamma_2\ |\mathbf{B}_0|\,, \quad |\Delta\mathbf{B}_1| \le \gamma_2\ |\mathbf{B}_1|\,, \quad \left|\Delta\widehat{\mathbf{B}}_k\right| \le \gamma_3\ \left|\widehat{\mathbf{B}}_k\right| \ \text{ for } \ k = 2, 3, \ldots, t-2.$$

$\mathbf{C}_{t-1}$ is a block diagonal matrix containing C^I_1, C^{II}_2, C^{III}_2 and C^{IV}_2 hence

$$|\Delta\mathbf{C}_{t-1}| \le \gamma_3\ |\mathbf{C}_{t-1}|\,.$$

Using direct call of computing trigonometric functions, i.e., the view of (4.1),

$$\widehat{\mathbf{B}}_k = \mathbf{B}_k + \Delta\mathbf{B}_k, \quad |\Delta\mathbf{B}_k| \le \mu|\mathbf{B}_k|.$$

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

Thus, overall

$$\widehat{\mathbf{y}} = (\mathbf{A}_0 + \Delta\mathbf{A}_0)\cdots(\mathbf{A}_{t-2} + \Delta\mathbf{A}_{t-2})\,(\mathbf{C}_{t-1} + \Delta\mathbf{C}_{t-1})$$
$$(\mathbf{B}_{t-2} + \mathbf{E}_{t-2})\cdots(\mathbf{B}_2 + \mathbf{E}_2)(\mathbf{B}_1 + \Delta\mathbf{B}_1)(\mathbf{B}_0 + \Delta\mathbf{B}_0)\mathbf{x},$$

$$|\mathbf{E}_k| \le (\mu + \gamma_3(1+\mu))|\mathbf{B}_k| \le \gamma_5|\mathbf{B}_k|.$$

Hence,

$$|\mathbf{y} - \widehat{\mathbf{y}}| \le \left[(1+\gamma_2)^t(1+\gamma_3)(1+\gamma_5)^{t-3} - 1\right] |\mathbf{A}_0|\,|\mathbf{A}_1|\cdots|\mathbf{A}_{t-2}|$$
$$|\mathbf{C}_{t-1}|\,|\mathbf{B}_{t-2}|\,|\mathbf{B}_{t-3}|\cdots|\mathbf{B}_0|\,|\mathbf{x}|,$$

where

$$(1+\gamma_2)^t(1+\gamma_3)(1+\gamma_5)^{t-3} - 1 \le (1+\gamma_2)^t(1+\gamma_5)^{t-2} - 1 \le (1+\gamma_7)^t - 1 \le \frac{\gamma_7 t}{1-\gamma_7 t}.$$

Since $\mathbf{A}_k, \mathbf{C}_{t-1}, \mathbf{B}_k$ are orthogonal matrices, $\|\mathbf{A}_k\|_2 = \|\mathbf{C}_{t-1}\|_2 = \|\mathbf{B}_k\|_2 = 1$. By orthogonality of C_{n+1}^I, $\|\mathbf{y}\|_2 = \|\mathbf{x}\|_2$. Hence

$$\|\mathbf{y} - \widehat{\mathbf{y}}\|_2 \le \frac{\gamma_7 t}{1-\gamma_7 t}\,\|\mathbf{y}\|_2 . \quad \square$$

COROLLARY 4.2. $\mathbf{y} = C_{n+1}^I\,\mathbf{x}$ *is forward and backward stable.*

Proof. The above theorem says that radix 2 DCT-I yields a tiny forward error provided that $\sin\frac{r\pi}{4n}$ and $\cos\frac{r\pi}{4n}$ are computed stably. It immediately follows that the computation is backward stable because $\widehat{\mathbf{y}} = \mathbf{y} + \Delta\mathbf{y} = C_{n+1}^I\mathbf{x} + \Delta\mathbf{y}$ implies $\widehat{\mathbf{y}} = C_{n+1}^I(\mathbf{x} + \Delta\mathbf{x})$ with $\frac{\|\Delta\mathbf{x}\|_2}{\|\mathbf{x}\|_2} = \frac{\|\Delta\mathbf{y}\|_2}{\|\mathbf{y}\|_2}$. If we form $\mathbf{y} = C_{n+1}^I\mathbf{x}$ by using exact C_{n+1}^I, then $|\mathbf{y} - \widehat{\mathbf{y}}| \le \gamma_{n+1}\,\left|C_{n+1}^I\right|\,|\mathbf{x}|$ so $\|\mathbf{y} - \widehat{\mathbf{y}}\|_2 \le \gamma_{n+1}\,\|\mathbf{y}\|_2$. As μ is of order u, the C_{n+1}^I has an error bound smaller than that for usual multiplication by the same factor as the reduction in complexity of the method, so DCT-I is perfectly stable. $\square$

A similar analogue as in theorem 4.1 results in the error bound for computing the DCT-II algorithm.

THEOREM 4.3. *Let* $\widehat{\mathbf{y}} = fl(C_n^{II}\mathbf{x})$*, where* $n = 2^t (t \ge 2)$*, be computed using the algorithms* (2.1)*,* (2.2)*, and assume that (4.1) holds, then*

$$\frac{\|\mathbf{y} - \widehat{\mathbf{y}}\|_2}{\|\mathbf{y}\|_2} \le \frac{\gamma_7(t-1)}{1-\gamma_7(t-1)}. \tag{4.3}$$

Following the equivalent lines as in corollary 4.2, one can confirm the stability of the DCT-II algorithm.

COROLLARY 4.4. $\mathbf{y} = C_n^{II}\,\mathbf{x}$ *is forward and backward stable.*

Due to the relationship between the DCT-II and DCT-III algorithms, the following results are trivial.

COROLLARY 4.5. *Let* $\widehat{\mathbf{y}} = fl(C_n^{III}\mathbf{x})$*, where* $n = 2^t (t \ge 2)$*, be computed using the algorithms* (2.3)*,* (2.2)*,* (2.1)*, and assume that (4.1) holds, then*

$$\frac{\|\mathbf{y} - \widehat{\mathbf{y}}\|_2}{\|\mathbf{y}\|_2} \le \frac{\gamma_7(t-1)}{1-\gamma_7(t-1)}. \tag{4.4}$$

COROLLARY 4.6. $\mathbf{y} = C_n^{III}\,\mathbf{x}$ *is forward and backward stable.*

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

Analogue of the results of theorem 4.1 and corollary 4.2 lead to:

THEOREM 4.7. *Let* $\widehat{\mathbf{y}} = fl(C_n^{IV}\mathbf{x})$, *where* $n = 2^t (t \geq 2)$, *be computed using the algorithms* (2.2), (2.1), *and assume that (4.1) holds, then*

$$\frac{\|\mathbf{y} - \widehat{\mathbf{y}}\|_2}{\|\mathbf{y}\|_2} \leq \frac{\gamma_7 t}{1 - \gamma_7 t}. \tag{4.5}$$

COROLLARY 4.8. $\mathbf{y} = C_n^{IV}\,\mathbf{x}$ *is forward and backward stable.*

5. Signal flow graphs for DCT algorithms. Signal flow graphs commonly represent the realization of systems such as electronic devices in electrical engineering, control theory, system engineering, theoretical computer science, etc. Simply put, the objective is to build a device to implement or realize an algorithm, using devices that implement the algebraic operations used in these recursive algorithms. These building blocks are shown next in Figure 5.1.

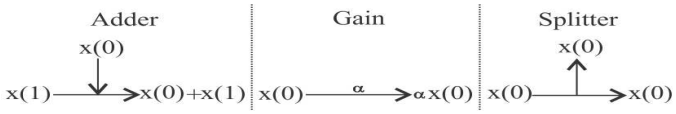


Fig. 5.1: Signal flow graphs building blocks.

This section presents signal flow graphs for 9-point DCT-I and 8-point DCT II-IV algorithms via Figures 5.2, 5.3, 5.4, and 5.5. As shown in the flow graphs, in each graph signal flows from the left to the right. These signal flow graphs are corresponding to the decimation-in-frequency algorithms. However, one can convert these decimation-in-frequency DCT algorithms into decimation-in-time DCT algorithms. In each Figure (5.2, 5.3, 5.4, and 5.5), $\epsilon := \frac{1}{\sqrt{2}}$, $C_{i,j} := \cos\frac{i\pi}{2^j}$, and $S_{i,j} = \sin\frac{i\pi}{2^j}$.

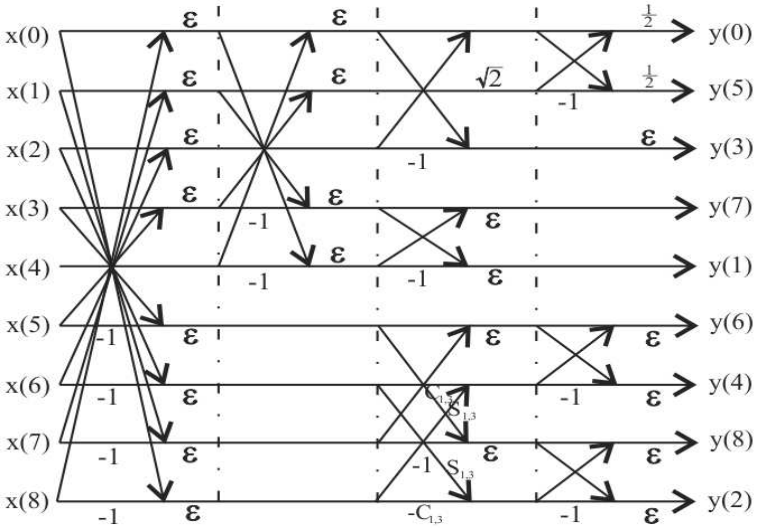


Fig. 5.2: Flow graph for 9-point DCT-I algorithm.

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

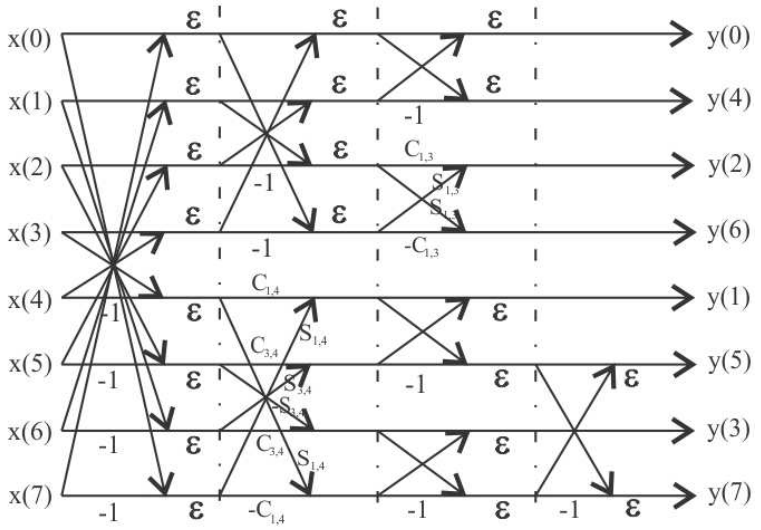


Fig. 5.3: Flow graph for 8-point DCT-II algorithm.

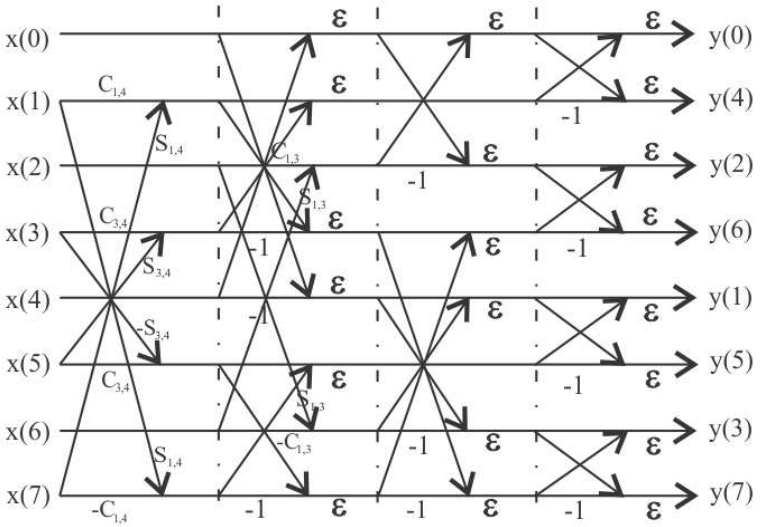


Fig. 5.4: Flow graph for 8-point DCT-III algorithm.

As shown in Figures 5.3, 5.4, and 5.5, the input signals are in order: $\mathbf{x} = \{x(0), x(1), \ldots, x(7)\}$ and output signals are in bit-reversed order: $\mathbf{y} = \{y(0), y(4), y(2), y(6), y(1), y(5), y(3), y(7)\}$. In bit-reversed order, each output index is represented as a binary number and the indices' bits are reversed. Say for 8-point DCT II, the sequential order of the input indices' bits is $\{000, 001, 010, 011, 100, 101, 110, 111\}$. Then reversing these input signal bits yields $\{000, 100, 010, 110, 001, 101, 011, 111\}$ which is the output signal.

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

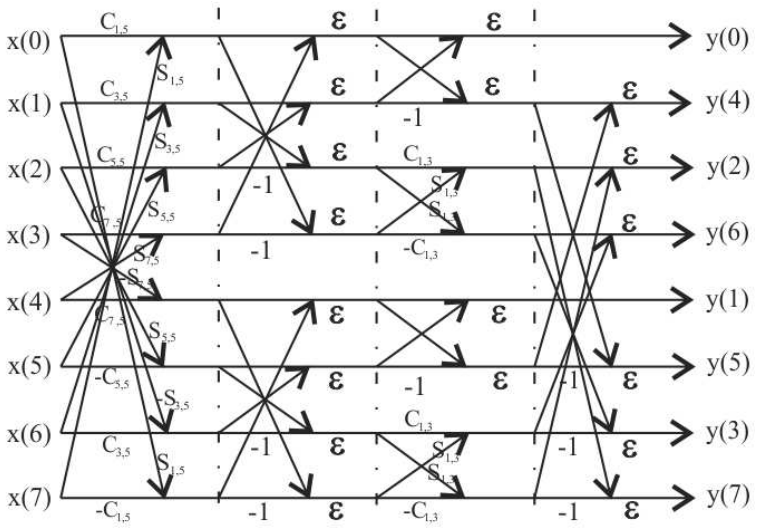


Fig. 5.5: Flow graph for 8-point DCT-IV algorithm.

6. Image compression results based on DCT algorithms. Discretized images can be considered as matrices. To compress such images one can apply different quantization techniques. In this section, we analyze quantization techniques using these recursive DCT-II and DCT-IV algorithms to compress the Polygon image of size 512×512 pixels. At first, we convert the RGB Polygon image into a grayscale scale image. Next, the image is discretized into 8×8, 16×16, and 32×32 blocks. After that, using these recursive DCT-II and DCT-IV algorithms, two-dimensional DCTs of each block in the image are computed. Then the DCT-II and DCT-IV coefficients are quantized by transforming absence of 93.75% of the DCT coefficients in each block. e.g. for 8×8 transfer block, after recursively computing two-dimensional DCT-II and DCT-IV of each 8×8 block in the image, we discard all but 4 of the 64 DCT coefficients in each block, i.e., applying the matrix $\begin{bmatrix} 1 & 1 & & \\ 1 & 1 & & \\ & & 0_{6\times 6} & \end{bmatrix}$, where $0_{6\times 6}$ is the zero matrix of order 6×6 to each transform block, and analogously for 16×16 and 32×32. Next, we reconstruct the image using the inverse two-dimensional DCT-II and DCT-IV of each blocks. Finally, putting each block back together into a single image leads to Figures 6.1 and 6.2. Simply put, we analyzed image compression results via the RGB Polygon image with 93.75% of discarded coefficients in each transfer block using these recursive DCT-II and DCT-IV algorithms.

Figure 6.1 shows images with discarded coefficients (except the top left 6.25%) in each transfer block, after applying DCT-II algorithm, and then running recursively with the DCT-IV algorithm.

Figure 6.2 shows images with discarded coefficients (except the top left 6.25%) in each transfer block after applying DCT-IV algorithm and then running recursively with the DCT-II algorithm.

Comparing to Figures 6.1 and 6.2, the image reconstruction results corresponding to DCT-II algorithm are better than that of the DCT-IV algorithm. Though the quality of reconstructed images in Figures 6.1 and 6.2 are somewhat lost, those images are clearly recognizable even though

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

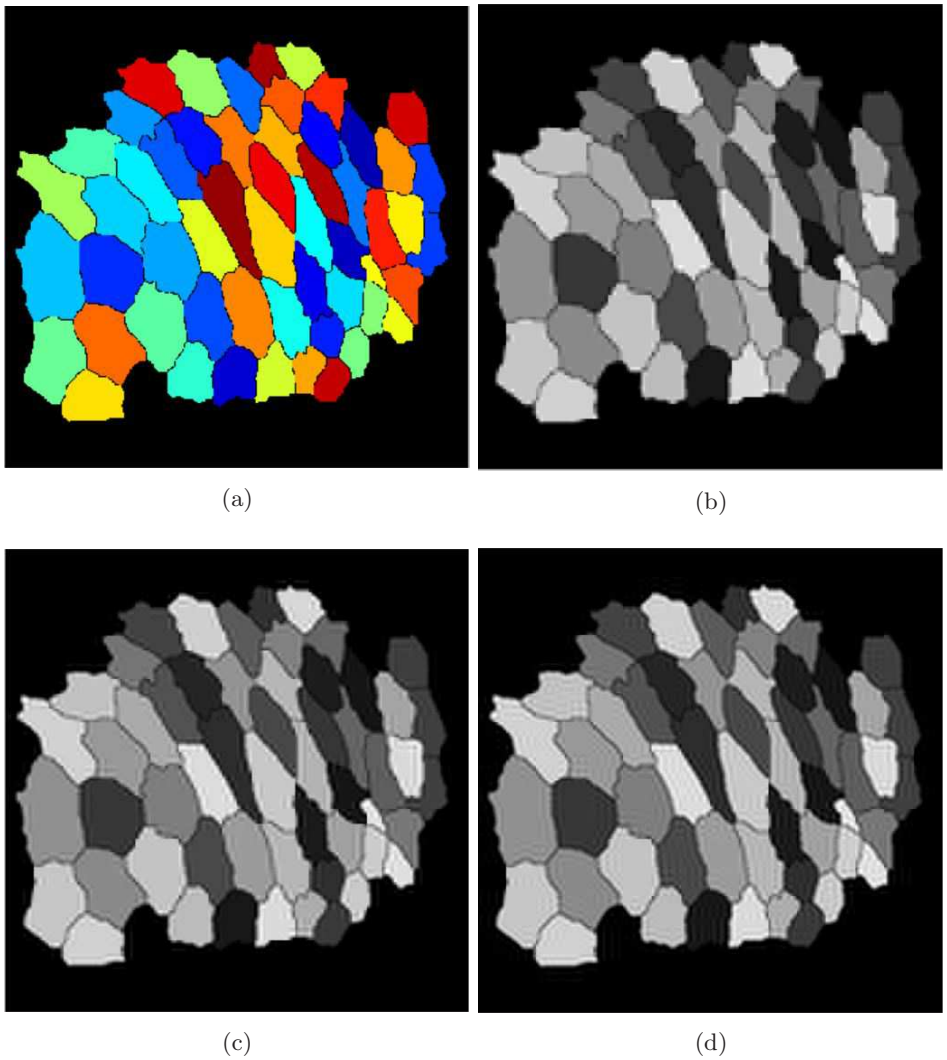
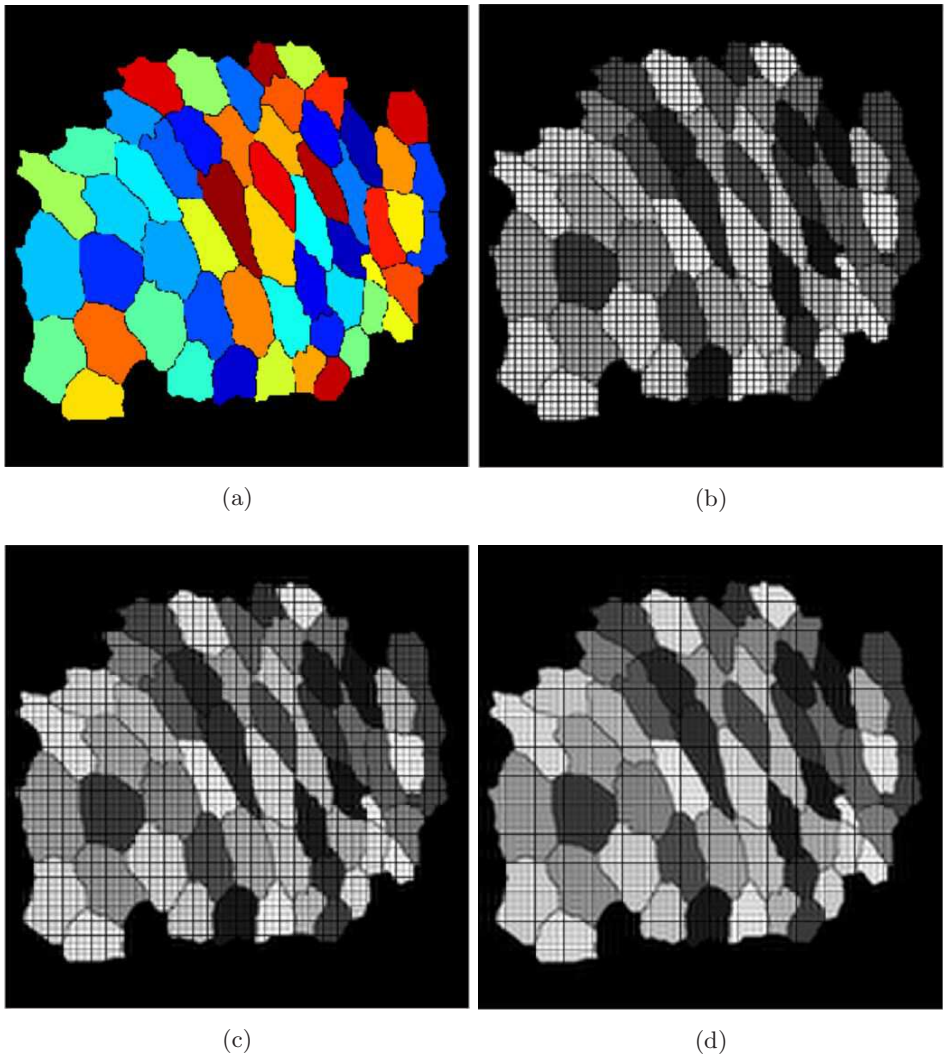


Fig. 6.1: (6.1a) Original polygon image. (6.1b) Reconstructed image with 93.75% discarded DCT-II coefficients in each 8×8 transfer block. (6.1c) Reconstructed image with 93.75% discarded DCT-II coefficients in each 16×16 transfer block. (6.1d) Reconstructed image with 93.75% discarded DCT-II coefficients in each 32×32 transfer block.

93.75% of the DCT-II and DCT-IV coefficients are discarded in each transfer block.

7. Conclusion. This paper provided stable, completely recursive, radix-2 DCT-I and DCT-III algorithms having sparse, orthogonal and rotation/rotation-reflection matrices, defined solely via DCT I-IV algorithms. The arithmetic cost and error bounds of computing DCT I-IV algorithms are addressed. Signal flow graphs are presented for these solely based on orthogonal factorization of

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016



(a) (b)

(c) (d)

Fig. 6.2: (6.2a) Original polygon image. (6.2b) Reconstructed image with 93.75% discarded DCT-IV coefficients in each 8×8 transfer block. (6.2c) Reconstructed image with 93.75% discarded DCT-IV coefficients in each 16×16 transfer block. (6.2d) Reconstructed image with 93.75% discarded DCT-IV coefficients in each 32×32 transfer block.

DCT I-IV in decimation-of-frequency. Using the recursive DCT-II and DCT-IV algorithms with the absence of 93.75% coefficients in each transfer block in 2D DCT-II and DCT-IV, one can reconstruct 512×512 images without seriously affecting the quality.

Acknowledgment. The author would like to thank James G. Nagy and the anonymous referee for their helpful suggestions that allowed to improve the exposition of the manuscript.

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

REFERENCES

[1] H. Ahmed, T. Natarajan, and K.R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, 23:90–93, 1974.

[2] V. Britanak. New generalized conversion method of the MDCT and MDST coefficients in the frequency domain for arbitrary symmetric windowing function. *Digital Signal Processing*, 23(5):1783–1797, 2013.

[3] V. Britanak, P.C. Yip, and K.R. Rao. *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*. Academic Press, 2007.

[4] S. Chakraborty and K.R. Rao. Fingerprint enhancement by directional filtering. *9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, Thailand, 1–4, May 2012, doi: 10.1109/ECTICon.2012.6254113.

[5] W.H. Chen, C.H. Smith, and S. Fralick. A fast computational algorithm for the discrete cosine transform. *IEEE Transactions on Communications*, 25:1004–1009, 1977.

[6] D. Fan, X. Meng, Y. Wang, X. Yang, X. Peng, W. He, G. Dong, and H. Chen. Optical identity authentication scheme based on elliptic curve digital signature algorithm and phase retrieval algorithm. *Applied Optics*, 52(23):5645–5652, 2013.

[7] J. Han, A. Saxena, V. Melkote, and K. Rose. Towards jointly optimal spatial prediction and adaptive transform in video/image coding. *IEEE Transactions on Image Processing*, 21(4):1874–1884, 2012.

[8] N.J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM Publications, Philadelphia, 1996.

[9] A.K. Jain. A sinusoidal family of unitary transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(4):356–365, 1979.

[10] A.K. Jain. A fast Karhunen-Loeve transform for a class of stochastic processes. *IEEE Transactions on Communications*, 24(9):1023–1029, 1976.

[11] P. Jain, B. Kumar, and S.B. Jain. Unified recursive structure for forward and inverse modified DCT/DST/DHT. *IETE Journal of Research*, 55(4):180–191, 2009.

[12] H.B. Kekre, T.K. Sarode, and J.K. Save. Column Transform based Feature Generation for Classification of Image Database. *International Journal of Application or Innovation in Engineering and Management*, 3(7):172–181, 2014.

[13] H.B. Kekre, T. Sarode, and P. Natu. Performance Comparison of Hybrid Wavelet Transform Formed by Combination of Different Base Transforms with DCT on Image Compression. *International Journal of Innovative Research in Science, Engineering and Technology*, 3(1):8274–8281, 2014.

[14] H.B. Kekre and J.K. Solanki. Comparative performance of various trigonometric unitary transforms for transform image coding. *International Journal of Electronics*, 44(3):305–315, 1978.

[15] D.N. Kim and K.R. Rao. Two-dimensional discrete sine transform scheme for image mirroring and rotation. *Journal of Electronic Imaging*, 17(1), 2008, doi:10.1117/1.2885257.

[16] M.H. Lee, M.H.A. Khan, K.J. Kim, and D. Park. A Fast Hybrid Jacket-Hadamard Matrix Based Diagonal Block-wise Transform. *Signal Processing: Image Communication*, 29:49–65, 2014.

[17] J. Ma, G. Plonka, and M.Y. Hussaini. Compressive Video Sampling with Approximate Message Passing Decoding. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(9):1354–1364, 2012.

[18] S.M. Perera and V. Olshevsky. Fast and Stable Algorithms for Discrete Sine Transformations having Orthogonal Factors. In: M.G. Cojocaru, I. S. Kotsireas, R. N. Makarov, R. V. N. Melnik, and H. Shodiev(eds.) *Interdisciplinary Topics in Applied Mathematics, Modeling and Computational Science*, 117:347–354, Springer, 2015.

[19] G. Plonka and M. Tasche. Fast and Numerically stable algorithms for discrete cosine transforms *Linear Algebra and its Applications*, 394:309–345, 2005.

[20] D. Potts , G. Steidl, and M. Tasche. Numerical stability of fast trigonometric transforms - a worst case study. *Journal of Concrete and Applicable Mathematics*, 1:1–36, 2003.

[21] M. Puschel and J.M. Moura. The algebraic approach to the discrete cosine and sine transforms and their fast algorithms. *SIAM Journal on Computing*, 32(5):1280–1316, 2003.

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 362-380, June 2016

[22] U. Schreiber. *Fast and Numerically stable trigonometric transforms*. Thesis, University of Rostock, Germain, 1999.

[23] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, Wellesley, 1986.

[24] G. Strang. The discrete cosine transform. *SIAM Review*, 41:135–147, 1999.

[25] G. Steidl and M. Tasche. A polynomial approach to fast algorithms for discrete Fourier-cosine and Fourier-sine transforms. *Mathematics of Computation*, 56:281–296, 1991.

[26] M. Tasche and H. Zeuner Roundoff error analysis for fast trigonometry transforms. In: G. Anastassiou (editor), *Handbook of Analytic-Computational Methods in Applied Mathematics*, Chapman and Hall/CRC Press, Boca Raton, 357–406, 2000.

[27] C. Van Loan. *Computational Frameworks for the Fast Fourier Transform*. SIAM Publications, Philadelphia, 1992.

[28] R. Veerla, Z. Zhang, and K.R. Rao. Advanced image coding and its comparison with various still image codecs. *American Journal of Signal Processing*, 2(5):113–121, 2012.

[29] Y. Voronenko and M. Püschel. Algebraic Signal Processing Theory:Cooley-Tukey Type Algorithms for Real DFTs. *Transactions on Signal Processing*, 57(1):205–222, 2009.

[30] Z. Wang. Fast algorithms for the discrete W transform and the discrete Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32:803–816, 1984.

[31] Z. Wang and B.R. Hunt. The discrete cosine transform – A new version. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 8:1256–1259, 1983, doi:10.1109/ICASSP.1983.1171989.

[32] P. Yip and K.R. Rao. A fast computational algorithm for the discrete sine transform. *IEEE Transactions on Communications*, 28(2):304–307, 1980.