COOLEY-TUKEY FFT LIKE ALGORITHMS FOR THE DCT

Markus Püschel

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, U.S.A.

ABSTRACT

The Cooley-Tukey FFT algorithm decomposes a discrete Fourier transform (DFT) of size $n = km$ into smaller DFTs of size k and m. In this paper we present a theorem that decomposes a polynomial transform into smaller polynomial transforms, and show that the FFT is obtained as a special case. Then we use this theorem to derive a new class of recursive algorithms for the discrete cosine transforms (DCTs) of type II and type III. In contrast to other approaches, we manipulate polynomial algebras instead of transform matrix entries, which makes the derivation transparent, concise, and gives insight into the algorithms' structure. The derived algorithms have a regular structure and, for 2-power size, minimal arithmetic cost (among known DCT algorithms).

1. INTRODUCTION

The celebrated Cooley-Tukey fast Fourier transform (FFT) algorithm [1] decomposes a discrete Fourier transform (DFT) of size $n = km$ into smaller DFTs of size k and m. The algorithm's inherent versatility—due to the degree of freedom in factoring n and due to its structure—has proven very useful for its implementation on a variety of diverse computing platforms. A recent example is automatic platform adaptation of FFT software through algorithmic search [2, 3, 4].

In this paper we derive a new class of algorithms, analogous to the FFT, for the discrete cosine transforms (DCTs) of type II and III. The discovery of this class and its derivation is made possible through an approach using *polynomial algebras*.

Algebraic Approach to the DCTs. There is a large number of publications on fast DCT algorithms. With few exceptions (including [5, 6, 7, 8]) these algorithms have been found by clever manipulation of matrix entries, which provides little insight into their structure or the reason for their existence. In [9, 10] we developed an algebraic approach to the 16 DCTs and DSTs to remedy this situation. We associate to each DCT (or DST) a polynomial algebra, and derive known algorithms by manipulating these algebras instead of transform matrix entries. This approach makes the derivation transparent and provides the mathematical underpinning for these algorithms.

A New Class of DCT Algorithms. In this paper we present a general theorem for decomposing polynomial algebras and show that the Cooley-Tukey FFT is obtained as a special case. Then we use this theorem to derive the analogous class of algorithms for the DCTs of type II and III. Thus, the term "analogous" is in a strict mathematical sense.

All algorithms in this class have a large degree of parallelism, low arithmetic cost, and a regular, flexible, recursive structure, which makes them suitable for efficient hardware implementation or for automatic software generation and adaptation [2, 3]. Only few special cases in this class are known from the literature. We present the mathematical framework in Section 2; the algorithm derivation and analysis is in Section 3.

2. POLYNOMIAL ALGEBRAS AND TRANSFORMS

A vector space $\mathcal{A}$ that permits multiplication of elements such that the distributive law holds is called an *algebra*. Examples include $\mathbb{C}$ and the set $\mathbb{C}[x]$ of polynomials with complex coefficients.

Polynomial Algebra. Let $p(x)$ be a polynomial of degree $\deg(p) = n$. Then, $\mathcal{A} = \mathbb{C}[x]/p(x) = \{q(x) \mid \deg(q) < n\}$, the set of residue classes modulo p, is an n-dimensional algebra with respect to the addition of polynomials, and the polynomial multiplication modulo p. We call $\mathcal{A}$ a *polynomial algebra*.

Polynomial Transform. For the remainder of this paper, we assume that the polynomial $p(x)$ in $\mathcal{A} = \mathbb{C}[x]/p(x)$ has pairwise distinct zeros $\alpha = (\alpha_0, \dots, \alpha_{n-1})$. Then, the Chinese Remainder Theorem decomposes $\mathcal{A}$ into a Cartesian product of one-dimensional algebras as

$$\begin{aligned} \mathbb{C}[x]/p(x) &\to \mathbb{C}[x]/(x-\alpha_0) \oplus \ldots \oplus \mathbb{C}[x]/(x-\alpha_{n-1}), \\ q(x) &\mapsto (q(\alpha_0), \dots, q(\alpha_{n-1})). \end{aligned} \tag{1}$$

If we choose $b = (p_0, \dots, p_{n-1})$, $\deg(p_i) < n$, as a basis for $\mathcal{A}$, and $b_k = (x^0)$ as the basis in each $\mathbb{C}[x]/(x - \alpha_k)$, then the decomposition in (1) is given by the *polynomial transform*

$$\mathcal{P}_{b,\alpha} = [p_\ell(\alpha_k)]_{0 \le k,\ell < n}. \tag{2}$$

As an example, every Vandermonde matrix $[\alpha_k^\ell]_{0 \le k,\ell < n}$ is known to be a polynomial transform by choosing $p_\ell = x^\ell$.

Fast Algorithm. If $p(x)$ decomposes into two polynomials, $p(x) = q(r(x))$, then $\mathcal{P}_{b,\alpha}$ can be factorized as briefly explained next (see [9, 10] for details). We assume $\deg(q) = k$, $\deg(r) = m$, i.e., $n = km$, and denote with $\beta = (\beta_0, \dots, \beta_{k-1})$ the zeros of q and with $\alpha_i' = (\alpha_{i,0}, \dots, \alpha_{i,m-1})$ the zeros of $r(x) - \beta_i$, i.e., each $\alpha_{i,j}$ is a zero α_ℓ of p.

Then $\mathbb{C}[x]/p(x)$ decomposes in the following steps.

$$\begin{aligned} \mathbb{C}[x]/p(x) &\to \bigoplus_{0 \le i < k} \mathbb{C}[x]/(r(x) - \beta_i) & (3) \\ &\to \bigoplus_{0 \le i < k} \bigoplus_{0 \le j < m} \mathbb{C}[x]/(x - \alpha_{i,j}) & (4) \\ &\to \bigoplus_{0 \le \ell < n} \mathbb{C}[x]/(x - \alpha_\ell). & (5) \end{aligned}$$

This work was supported by NSF through award 9988296.

0-7803-7663-3/03/$17.00 ©2003 IEEE

Further, we choose a basis $c = (q_0, \dots, q_{k-1})$ for $\mathbb{C}[x]/q(x)$, and a basis $d = (r_0, \dots, r_{m-1})$ for each $\mathbb{C}[x]/(r(x) - \beta_i)$ in (3). Then

$$\begin{aligned} b' \; = \; & (r_0 q_0(r), \dots, r_{m-1} q_0(r), \\ & \dots \\ & r_0 q_{k-1}(r), \dots, r_{m-1} q_{k-1}(r)) \end{aligned} \tag{6}$$

is a basis of $\mathbb{C}[x]/p(x)$. The final factorization of $\mathcal{P}_{b,\alpha}$ is given by the following theorem, in which we use the tensor (or Kronecker) product and the direct sum of matrices A, B, respectively defined by

$$A \otimes B = [a_{k,\ell} \cdot B], \quad \text{for } A = [a_{k,\ell}], \quad A \oplus B = \left[\begin{smallmatrix} A & \\ & B \end{smallmatrix}\right].$$

Theorem 1 Using previous notation,

$$\mathcal{P}_{b,\alpha} = P\Big(\bigoplus_{0 \le i < k} \mathcal{P}_{d,\alpha'_i}\Big)(\mathcal{P}_{c,\beta} \otimes \mathrm{I}_m)B, \tag{7}$$

where B is the base change matrix mapping b to b', and the remaining three factors in (7) correspond (from right to left) to steps (3), (4), and (5), respectively. In particular, P is a permutation matrix mapping the concatenation of the α'_i onto α in (5).

The usefulness of (7) as a fast algorithm for $\mathcal{P}_{b,\alpha}$ depends on the matrix B; the other factors are sparse. The following example shows that the Cooley-Tukey FFT is obtained as a special case of (7).

Example: Cooley-Tukey FFT. The discrete Fourier transform (DFT) of size n is defined by the matrix

$$\mathrm{DFT}_n = [\omega_n^{k\ell}]_{0 \le k,\ell < n}, \quad \omega_n = e^{-2\pi\sqrt{-1}/n},$$

and, as a special Vandermonde matrix, is a polynomial transform $\mathrm{DFT}_n = \mathcal{P}_{b,\alpha}$ for the algebra $\mathcal{A} = \mathbb{C}[x]/(x^n - 1)$ w. r. t. the basis $b = (1, x, \dots, x^{n-1})$ and the list of zeros $\alpha = (\omega_n^0, \dots, \omega_n^{n-1})$.

If $n = km$ then $x^n - 1 = (x^m)^k - 1$ decomposes and we can apply (7). The zeros of the outer polynomial $q(x) = x^k - 1$ are given by $\beta = (\omega_k^0, \dots, \omega_k^{k-1})$, the zeros of $r(x) - \beta_i = x^m - \omega_k^i$ by $\alpha'_i = (\omega_n^{i+0\cdot k}, \dots, \omega_n^{i+(m-1)k})$. Next we choose bases $c = (1, x, \dots, x^{k-1})$ of $\mathbb{C}[x]/(x^k - 1)$ and $d = (1, x, \dots, x^{m-1})$ of $\mathbb{C}[x]/(x^m - \omega_k^i)$, and observe that $b' = b$ in (6) and thus, in (7), $B = \mathrm{I}_n$ is the identity matrix. In summary we obtain the factorization

$$\mathrm{DFT}_n = \mathrm{L}_m^n\Big(\bigoplus_{0 \le i < k} \mathrm{DFT}_m(\omega_k^i)\Big)(\mathrm{DFT}_k \otimes \mathrm{I}_m), \tag{8}$$

where

$$\mathrm{L}_m^n : jk + i \mapsto im + j, \quad 0 \le i < k,\ 0 \le j < m,$$

is the *stride permutation*, which orders the summands in step (5), and $\mathrm{DFT}_m(\omega_k^i)$ is the polynomial transform for $\mathbb{C}[x]/(x^m - \omega_k^i)$. Using the definition, we observe that

$$\mathrm{DFT}_m(\omega_k^i) = \mathrm{DFT}_m \cdot \mathrm{diag}_{0 \le \ell < m}(\omega_n^{i\ell}), \tag{9}$$

which yields the Cooley-Tukey FFT in its familiar form

$$\mathrm{DFT}_n = \mathrm{L}_m^n(\mathrm{I}_k \otimes \mathrm{DFT}_m)\,\mathrm{T}_m^n(\mathrm{DFT}_k \otimes \mathrm{I}_m), \tag{10}$$

where T_m^n is the diagonal twiddle matrix. By transposing (10) we obtain a different recursion

$$\mathrm{DFT}_n = (\mathrm{DFT}_k \otimes \mathrm{I}_m)\,\mathrm{T}_m^n(\mathrm{I}_k \otimes \mathrm{DFT}_m)\,\mathrm{L}_k^n \tag{11}$$

where we used $(\mathrm{L}_m^n)^T = (\mathrm{L}_m^n)^{-1} = \mathrm{L}_k^n$. Note that the degree of freedom in factorizing n in (10) and (11) when applied recursively, yields a large set of structurally different FFT algorithms. Thus, it is more accurate to speak of the set of Cooley-Tukey FFT algorithms. The degree of freedom in the recursion is used for automatic software optimization [2, 3, 4].

3. RECURSIVE DCT II AND III ALGORITHMS

In [9, 10] we have shown that all 16 types of discrete cosine and sine transforms are (possibly scaled) polynomial transforms, and used this connection to derive and explain many of their known algorithms by manipulating polynomial algebras rather than matrices. In this section we extend this approach and use Theorem 1 to derive an entire class of recursive DCT type II and III algorithms based on Theorem 1, which are thus mathematically equivalent to the Cooley-Tukey FFT (10) and (11). Only few special cases in this class are known from the literature. We start by identifying the DCT type III as polynomial transform.

DCT as Polynomial Transform. The (unscaled) DCT of type III is defined by the matrix

$$\mathrm{DCT}_n^{\text{(III)}} = [\cos(k + 1/2)\ell\pi/n]_{0 \le k,\ell < n},$$

and the DCT of type II is its transpose, $\mathrm{DCT}_n^{\text{(II)}} = (\mathrm{DCT}_n^{\text{(III)}})^T$. The $\mathrm{DCT}_n^{\text{(III)}}$ is a polynomial transform as has been recognized already in [6]. To make this explicit we introduce the *Chebyshev polynomials* defined by the recurrence

$$T_0 = 1,\ T_1 = x, \quad T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x),\ n \ge 2.$$

The polynomial T_n can be written in the closed form $T_n(x) = \cos n\theta$, $\cos\theta = x$, which can be used to verify the following known properties of T_n.

$$T_{-n} = T_n \tag{12}$$

$$T_k T_n = (T_{n+k} + T_{n-k})/2 \tag{13}$$

$$T_{km} = T_k(T_m) \tag{14}$$

$$\text{zeros of } T_n:\ \alpha_k = \cos(k + 1/2)\pi/n,\ 0 \le k < n \tag{15}$$

$$T_n - \cos r\pi = 2^{n-1} \prod_{0 \le i < n} (x - \cos(r + 2i)\pi/n) \tag{16}$$

Now, we can readily verify that $\mathrm{DCT}_n^{\text{(III)}}$ is a polynomial transform for the algebra $\mathcal{A} = \mathbb{C}[x]/T_n$ with basis $b = (T_0, \dots, T_{n-1})$, namely

$$\mathrm{DCT}_n^{\text{(III)}} = [T_\ell(\cos(k + 1/2)\pi/n)]_{0 \le k,\ell < n}.$$

Initial Algorithm Derivation. To apply Theorem 1, we assume $n = km$ and use $T_n = T_k(T_m)$ from (14). We choose the basis $c = (T_0, \dots, T_{k-1})$ for $\mathbb{C}[x]/T_k$; the zeros of T_k are given by (15). Thus, in step (4) we need polynomial transforms for $\mathbb{C}[x]/(T_m - \cos(i + 1/2)\pi/k)$, which we define now.

Definition 1 Let $p(x) = T_n(x) - \cos r\pi$ with list of zeros $\alpha = (\cos r_1\pi, \dots, \cos r_{n-1}\pi)$, computed from (16), normalized to satisfy $0 \le r_i \le 1$, and ordered as $r_i < r_j$ for $i < j$. Further, let $d = (T_0, \dots, T_{n-1})$ be the basis of $\mathbb{C}[x]/p(x)$. Then we call $\mathrm{DCT}_n^{\text{(III)}}(r\pi) = \mathcal{P}_{d,\alpha}$ a *skew DCT of type III*. In particular, $\mathrm{DCT}_n^{\text{(III)}}(\pi/2) = \mathrm{DCT}_n^{\text{(III)}}$.

We start the derivation with the matrix $B = B_{n,k}$ in (7). The basis b' in (6) is given by

$$\begin{aligned} b' &= (T_0T_0(T_m), \dots, T_{m-1}T_0(T_m), \\ &\quad \dots \\ &\quad T_0T_{k-1}(T_m), \dots, T_{m-1}T_{k-1}(T_m)) \\ &= (T_{jm-i}/2 + T_{jm+i}/2)_{0\le i<m,\, 0\le j<k}, \end{aligned}$$

Due to the structure of b' we can easily read off the *inverse* of B_n, i.e., the base change B_n^{-1} from b' to b.

$$B_n^{-1} = C_{n,k} \cdot D_{n,k},$$

with

$$C_{n,k} = \begin{bmatrix} \mathrm{I}_m & Z_m & & & \\ & \mathrm{I}_m & Z_m & & \\ & & \ddots & \ddots & \\ & & & \mathrm{I}_m & Z_m \\ & & & & \mathrm{I}_m \end{bmatrix},$$

$$Z_m = \begin{bmatrix} & & & 0 \\ & & 0 & 1 \\ & \iddots & \iddots & \\ 0 & 1 & & \end{bmatrix}, \tag{17}$$

and the diagonal matrix

$$D_{n,k} = \mathrm{I}_m \oplus (\mathrm{I}_{k-1} \otimes \mathrm{diag}(1, 1/2, \dots, 1/2)).$$

The matrices for steps (3) and (4) are straightforward; the final permutation P in (7) is given by (without proof)

$$\mathrm{K}_m^n = (\mathrm{I}_k \oplus \mathrm{J}_k \oplus \mathrm{I}_k \oplus \mathrm{J}_k \oplus \dots)\, \mathrm{L}_m^n,$$

where J_k is I_k with the order of the columns (or rows) reversed. In summary, we obtain

$$\begin{aligned} \mathrm{DCT}_n^{(\mathrm{III})} = \mathrm{K}_m^n \Big(\bigoplus_{0\le i<k} \mathrm{DCT}_n^{(\mathrm{III})}((i+1/2)\pi/k) \Big) \\ (\mathrm{DCT}_k^{(\mathrm{III})} \otimes \mathrm{I}_m) D_{n,k}^{-1} C_{n,k}^{-1}. \end{aligned} \tag{18}$$

To obtain a complete recursive algorithm, we need an algorithm for the occurring skew $\mathrm{DCT}^{(\mathrm{III})}$'s. In contrast to (9) we do not translate them into ordinary $\mathrm{DCT}^{(\mathrm{III})}$'s. Instead, we again use Theorem 1 to decompose their associated algebra $\mathbb{C}[x]/(T_m - \cos(r\pi))$, provided m is composite, since $T_m - \cos(r\pi)$ decomposes exactly as T_m. The derivation is analogous to above and yields the exactly analogous factorization

$$\begin{aligned} \mathrm{DCT}_n^{(\mathrm{III})}(r\pi) = \mathrm{K}_m^n \Big(\bigoplus_{0\le i<k} \mathrm{DCT}_m^{(\mathrm{III})}(r_i\pi) \Big) \\ (\mathrm{DCT}_k^{(\mathrm{III})}(r\pi) \otimes \mathrm{I}_m) D_{n,k}^{-1} C_{n,k}^{-1}, \end{aligned} \tag{19}$$

where r, r_i are as in Definition 1.

To determine $C_{n,k}^{-1}$, we observe that $C_{n,k}$ is a direct sum of matrices, conjugated by a suitable permutation $Q_{n,k}$ (conjugation: $A^P = P^{-1}AP$), namely

$$\begin{aligned} C_{n,k} &= (\mathrm{I}_k \oplus S_k \oplus \dots \oplus S_k)^{Q_{n,k}}, \\ &= (\mathrm{I}_k \oplus (\mathrm{I}_{m-1} \otimes S_k))^{Q_{n,k}}, \end{aligned}$$

with the bidiagonal matrix

$$S_k = \begin{bmatrix} 1 & 1 & & & \\ & 1 & 1 & & \\ & & \ddots & \ddots & \\ & & & 1 & 1 \\ & & & & 1 \end{bmatrix}. \tag{20}$$

Thus, $C_{n,k}^{-1} = (\mathrm{I}_k \oplus (\mathrm{I}_{m-1} \otimes S_k^{-1}))^{Q_{n,k}}$. The inverse of S_k can be computed with $k-1$ recursive subtractions that have to be performed in sequence, which increases the critical path of computation and thus runtime or latency (compare to S_k in (20) where all $k-1$ additions can be done in parallel). To overcome this problem, we can either consider only small values of k, or invert (19), which we do next.

Inversion. To invert (19), we define a skew $\mathrm{DCT}^{(\mathrm{II})}$, motivated by $(\mathrm{DCT}_n^{(\mathrm{III})})^{-1} = 2/n \cdot \mathrm{diag}(1/2, 1, \dots, 1) \cdot \mathrm{DCT}_n^{(\mathrm{II})}$.

Definition 2 We define the *skew DCT of type II* by

$$\mathrm{DCT}_n^{(\mathrm{II})}(r\pi) = n/2 \cdot \mathrm{diag}(2, 1, \dots, 1) \cdot (\mathrm{DCT}_n^{(\mathrm{III})}(r\pi))^{-1}.$$

In particular, $\mathrm{DCT}_n^{(\mathrm{II})} = \mathrm{DCT}_n^{(\mathrm{II})}(\pi/2)$.

It turns out that with this definition, by inverting (19) the multiplications cancel each other, and we get the beautifully simple form

$$\begin{aligned} \mathrm{DCT}_n^{(\mathrm{II})}(r\pi) = C_{n,k}(\mathrm{DCT}_k^{(\mathrm{II})}(r\pi) \otimes \mathrm{I}_m) \\ \Big(\bigoplus_{0\le i<k} \mathrm{DCT}_m^{(\mathrm{II})}(r_i\pi) \Big) \mathrm{M}_k^n, \end{aligned} \tag{21}$$

where $\mathrm{M}_k^n = (\mathrm{K}_m^n)^{-1} = \mathrm{L}_k^n(\mathrm{I}_k \oplus \mathrm{J}_k \oplus \mathrm{I}_k \oplus \mathrm{J}_k \oplus \dots)$.

Four Classes of Recursive Algorithms. By transposing (19) and (21), we get a total of four different recursions, two for the DCT of type III and two for type II. We list these recursions in the following using previous notation. The numbers r, r_i are related as explained in Definition 1.

For the $\mathrm{DCT}^{(\mathrm{III})}$, we have

$$\begin{aligned} \mathrm{DCT}_n^{(\mathrm{III})} &= \mathrm{DCT}_n^{(\mathrm{III})}(\pi/2), \\ \mathrm{DCT}_n^{(\mathrm{III})}(r\pi) &= \mathrm{K}_m^n \Big(\bigoplus_{0\le i<k} \mathrm{DCT}_m^{(\mathrm{III})}(r_i\pi) \Big) \\ &\quad (\mathrm{DCT}_k^{(\mathrm{III})}(r\pi) \otimes \mathrm{I}_m) D_{n,k}^{-1} C_{n,k}^{-1}, \end{aligned} \tag{22}$$

and the inverse-transpose of (22)

$$\begin{aligned} \mathrm{DCT}_n^{(\mathrm{III})} &= \mathrm{DCT}_n^{(\mathrm{II})}(\pi/2)^T, \\ \mathrm{DCT}_n^{(\mathrm{II})}(r\pi)^T &= \mathrm{K}_m^n \Big(\bigoplus_{0\le i<k} \mathrm{DCT}_m^{(\mathrm{II})}(r_i\pi)^T \Big) \\ &\quad (\mathrm{DCT}_k^{(\mathrm{II})}(r\pi)^T \otimes \mathrm{I}_m) C_{n,k}^T. \end{aligned} \tag{23}$$

For the $\mathrm{DCT}^{(\mathrm{II})}$, we have the transpose of (22)

$$\begin{aligned} \mathrm{DCT}_n^{(\mathrm{II})} &= \mathrm{DCT}_n^{(\mathrm{III})}(\pi/2)^T, \\ \mathrm{DCT}_n^{(\mathrm{III})}(r\pi)^T &= C_{n,k}^{-T} D_{n,k}^{-1} (\mathrm{DCT}_k^{(\mathrm{III})}(r\pi)^T \otimes \mathrm{I}_m) \\ &\quad \Big(\bigoplus_{0\le i<k} \mathrm{DCT}_m^{(\mathrm{III})}(r_i\pi)^T \Big) \mathrm{M}_k^n, \end{aligned} \tag{24}$$

and the inverse of (22)

$$\begin{aligned}\mathrm{DCT}_n^{(\mathrm{II})} &= \mathrm{DCT}_n^{(\mathrm{II})}(\pi/2),\\ \mathrm{DCT}_n^{(\mathrm{II})}(r\pi) &= C_{n,k}(\mathrm{DCT}_k^{(\mathrm{II})}(r\pi)\otimes \mathrm{I}_m)\\ &\qquad \Big(\bigoplus_{0\le i<k} \mathrm{DCT}_m^{(\mathrm{II})}(r_i\pi)\Big)\,\mathrm{M}_k^n\,. \qquad (25)\end{aligned}$$

The remaining problem is to find fast algorithms for the base cases, i.e., for skew DCTs of prime size. We focus on the most important case of a DCT of 2-power size n, which is eventually reduced to skew DCTs of size 2 as base cases. These are given by ($c = \cos(r_i\pi/2)$)

$$\begin{aligned}\mathrm{DCT}_2^{(\mathrm{III})}(r_i\pi) &= \mathrm{F}_2\cdot\mathrm{diag}(1,c),\\ \mathrm{DCT}_2^{(\mathrm{II})}(r_i\pi) &= \mathrm{diag}(1,1/(2c))\cdot\mathrm{F}_2\end{aligned} \qquad (26)$$

and their transposes, where $\mathrm{F}_2 = \left[\begin{smallmatrix}1 & 1\\ 1 & -1\end{smallmatrix}\right]$.

Analysis. We briefly investigate the four proposed algorithms (22)–(25).

The algorithms (23) for the $\mathrm{DCT}^{(\mathrm{III})}$ and its transpose (25) for the $\mathrm{DCT}^{(\mathrm{II})}$ have a large degree of parallelism, and a simple, very regular structure. For a 2-power n, the arithmetic cost of the algorithms is independent of the chosen recursion and given by

$$A(n) = \frac{3n}{2}\log_2(n) - n + 1, \quad M(n) = \frac{n}{2}\log_2(n) \qquad (27)$$

additions and multiplications, respectively, and is thus among the best ones known. The only possible drawback (for fixed point implementations) is the large dynamic range of the occurring inverse cosines from the base cases (26). We found only the special case $m = 2$ (in which only skew DCTs of size 2 occur) in [11] in a slightly different form. Namely, using $(A\otimes \mathrm{I}_m)^{\mathrm{L}_k^n} = (\mathrm{I}_m \otimes A)$ for any $k\times k$ matrix A, we can write (23) also as

$$\begin{aligned}\mathrm{DCT}_n^{(\mathrm{II})}(r\pi)^T &= \mathrm{R}_m^n\Big(\bigoplus_{0\le i<k}\mathrm{DCT}_m^{(\mathrm{II})}(r_i\pi)^T\Big)^{\mathrm{L}_k^n}\\ &\qquad (\mathrm{I}_m\otimes\mathrm{DCT}_k^{(\mathrm{II})}(r\pi)^T)\,\mathrm{L}_m^n\,C_{n,k}^T, \qquad (28)\end{aligned}$$

where $\mathrm{R}_m^n = (\mathrm{I}_k\oplus\mathrm{J}_k\oplus\mathrm{I}_k\oplus\mathrm{J}_k\oplus\dots)$. Similarly, (25) can be written as

$$\begin{aligned}\mathrm{DCT}_n^{(\mathrm{II})}(r\pi) &= C_{n,k}\,\mathrm{L}_k^n(\mathrm{I}_m\otimes\mathrm{DCT}_k^{(\mathrm{II})}(r\pi))\\ &\qquad \Big(\bigoplus_{0\le i<k}\mathrm{DCT}_m^{(\mathrm{II})}(r_i\pi)\Big)^{\mathrm{L}_k^n}\,\mathrm{R}_m^n\,. \qquad (29)\end{aligned}$$

Algorithms (22) and (24) suffer, as mentioned before, from the $k-1$ recursive subtractions to compute S_k^{-1}, and from the additional multiplications arising from $D_{n,k}^{-1}$, and are thus inferior to their alternatives (23) and (25), unless we choose a small value of k. As an advantage, these algorithms use only cosines as constants (since only skew $\mathrm{DCT}^{(\mathrm{III})}$'s or their transposes occur). Further, for small k, the multiplications in $D_{n,k}^{-1}$ can be fused with the adjacent skew DCTs. The case $n = q^r$, $k = q$ was derived in [12] using Chebyshev polynomials, and again in [13] by complicated manipulations of matrix entries. As an example, we consider the case $k = 2$, derived in [5] and in [6], in both cases using Chebyshev polynomials, but given only in iterative form. For $k = 2$, (22) takes after minor manipulation the form

$$\begin{aligned}\mathrm{DCT}_n^{(\mathrm{III})}(r\pi) &= \mathrm{K}_m^n(\mathrm{DCT}_m^{(\mathrm{III})}(r\pi/2)\\ &\qquad \oplus\,\mathrm{DCT}_m^{(\mathrm{III})}((1-r)\pi/2))(\mathrm{F}_2\otimes\mathrm{I}_m)A_n, \qquad (30)\end{aligned}$$

with

$$A_n = \begin{bmatrix}\mathrm{I}_m & -Z_m\\ & c_r\,\mathrm{I}_m'\end{bmatrix},$$

where $c_r = \cos r\pi/2$, $I_m' = \mathrm{diag}(1,2,\dots,2)$, and Z_m is defined in (17). The arithmetic cost of (30) is equal to (27).

Summary. We derived a new class of fast algorithms for the DCT type II and III, by a stepwise decomposition of their associated polynomial algebras. The algorithms have minimal (known) arithmetic cost and a simple, regular structure. The similarity in structure to the Cooley-Tukey FFT provides the algorithms with the same versatility that was the success of the FFT, with its many variants optimized for, e.g., parallel and vector platforms [1]. Further, the algorithms are amenable to automatic software generation and adaptation as done in [2, 3, 4].

4. REFERENCES

[1] R. Tolimieri, M. An, and C. Lu, *Algorithms for Discrete Fourier Transforms and Convolution*, Springer, 1997.

[2] M. Frigo and S. G. Johnson, “FFTW: An adaptive software architecture for the FFT,” in *Proc. ICASSP*, 1998, vol. 3, pp. 1381–1384, http://www.fftw.org.

[3] J. M. F. Moura, J. Johnson, R. Johnson, D. Padua, V. Prasanna, M. Püschel, and M. M. Veloso, “SPIRAL: Automatic Library Generation and Platform-Adaptation for DSP Algorithms,” 1998, http://www.ece.cmu.edu/∼spiral.

[4] M. Püschel, B. Singer, J. Xiong, J. M. F. Moura, J. Johnson, D. Padua, M. Veloso, and R. W. Johnson, “SPIRAL: A Generator for Platform-Adapted Libraries of Signal Processing Algorithms,” 2003, to appear in *Journal of High Performance Computing and Applications*.

[5] Y. Morikawa, H. Hamada, and N. Yamane, “A Fast Algorithm for the Cosine Transform Based on Successive Order Reduction of the Chebyshev Polynomial,” *Elec. and Comm. in Japan, Part 1*, vol. 69, no. 3, pp. 173–180, 1986.

[6] G. Steidl and M. Tasche, “A Polynomial Approach to Fast Algorithms for Discrete Fourier-Cosine and Fourier-Sine Transforms,” *Mathematics in Computation*, vol. 56, no. 193, pp. 281–296, 1991.

[7] E. Feig and S. Winograd, “Fast Algorithms for the Discrete Cosine Transform,” *IEEE Trans. on Signal Processing*, vol. 40, no. 9, pp. 2174–2193, 1992.

[8] S. Egner and M. Püschel, “Automatic Generation of Fast Discrete Signal Transforms,” *IEEE Trans. on Signal Processing*, vol. 49, no. 9, pp. 1992–2002, 2001.

[9] M. Püschel and J. M. F. Moura, “The Discrete Trigonometric Transforms and Their Fast Algorithms: An Algebraic Symmetry Approach,” in *Proc. 10th IEEE DSP Workshop*, 2002.

[10] M. Püschel and J. M. F. Moura, “The Algebraic Approach to the Discrete Cosine and Sine Transforms and their Fast Algorithms,” 2003, submitted to *SIAM Journal of Computing*.

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

[12] G. Steidl, “Fast Radix-p Discrete Cosine Transform,” *Appl. Algebra Engrg. Comm. Comp.*, vol. 3, pp. 39–46, 1992.

[13] G. Bi and L. W. Yu, “DCT Algorithms for Composite Sequence Lengths,” *IEEE Trans. on Signal Processing*, vol. 46, no. 3, pp. 554–562, 1998.