Algebraic Signal Processing Theory: Cooley-Tukey Type Algorithms for DCTs and DSTs

Markus Püschel, *Senior Member, IEEE,* and José M. F. Moura, *Fellow, IEEE*

***Abstract*— This paper presents a systematic methodology to derive and classify fast algorithms for linear transforms. The approach is based on the algebraic signal processing theory. This means that the algorithms are not derived by manipulating the entries of transform matrices, but by a stepwise decomposition of the associated signal models, or polynomial algebras. This decomposition is based on two generic methods or algebraic principles that generalize the well-known Cooley-Tukey FFT and make the algorithms' derivations concise and transparent. Application to the 16 discrete cosine and sine transforms yields a large class of fast general radix algorithms, many of which have not been found before.**

***Index Terms*— Fast Fourier transform, discrete Fourier transform, discrete cosine transform, discrete sine transform, DFT, DCT, DST, polynomial algebra, Chinese remainder theorem, representation theory**

I. Introduction

There is a large body of literature on fast transform algorithms. With few exceptions these algorithms are derived by clever and often lengthy manipulation of the transform coefficients. These derivations are hard to grasp, and provide little insight into the structure of the resulting algorithm. Further, it is hard to determine if all relevant classes of algorithms have been found. This is not just an academic problem as the variety of different implementation platforms and application requirements makes a thorough knowledge of the algorithm space crucial. For example, state-of-the-art implementations of the discrete Fourier transform (DFT) heavily rely on various variants of general radix algorithms to adapt the implementation to the memory hierarchy [1], [2], [3] or to optimize it for vector instructions and multiple threads [4], [5], [6], [7]

In this paper, we derive fast algorithms for linear transforms algebraically. This means that we do not manipulate the actual transform to obtain an algorithm, but decompose a *polynomial algebra* associated with the transform. In this spirit, we first present two general decomposition theorems for polynomial algebras and show that they generalize the well-known Cooley-Tukey fast Fourier transform (FFT). Then, we apply these theorems to the discrete cosine and sine transforms (DCTs and DSTs) and derive a large new class of recursive general radix Cooley-Tukey type algorithms for these transforms, only special cases of which have been known. In particular, these new fast algorithms that we present are the first to provide a *general* radix decomposition for the DCTs and DSTs.

This work was supported by NSF through awards 9988296, 0310941, and 0634967.

Markus Püschel and José M. F. Moura are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh. E-mail: {pueschel,moura}@ece.cmu.edu; ph:(412)268-6341; fax:(412)268-3890.

Our algorithm derivation is a natural application of the algebraic signal processing theory.

Algebraic signal processing theory. In [8], [9], [10], we proposed a new approach to *linear* signal processing (henceforth just referred to as signal processing or SP), called algebraic signal processing theory (ASP). ASP is a general, axiomatic approach to SP that is built from the concept of a *signal model*, defined as a triple $(\mathcal{A}, \mathcal{M}, \Phi)$, where $\mathcal{A}$ is the filter space (an algebra), $\mathcal{M}$ the signal space (an $\mathcal{A}$-module), and Φ generalizes the concept of the z-transform. Once a signal model is given, other concepts such as convolution, spectrum, Fourier transform are automatically defined but take different forms for different models. For example, discrete infinite and finite (finite number of samples) 1-D time are signal models with associated z-transform and finite z-transform (defined in [8]) and the DTFT and DFT as associated Fourier transforms, respectively. Beyond that, we identified the signal models associated with practically all 1-D trigonometric transforms [8], [9], [10]. This includes the so-called 1-D space models for which the DCTs and DSTs are Fourier transforms. In each case, filter and signal space are given by a polynomial algebra, which ASP hence identifies as a key structure in SP.

Algebraic theory of transform algorithms. As we show in this paper, knowing the polynomial algebra associated with a transform is also the key to understanding and concisely deriving its fast algorithms. Using a few general theorems operating on polynomial algebras, many known and novel transform algorithms can be derived. In this paper, we consider the DCTs and DSTs extending our preliminary results from [11], [12]. In [13], we use the same approach for the derivation of real FFTs, and in [14] we show how the theory extends to the nonseparable 2-D transforms introduced by ASP in [15].

The theory in this paper does not cover all existing classes of DCT/DST algorithms. More precisely, we will not consider orthogonal algorithms (i.e., algorithms built from rotations such as [16]), "prime-factor type" algorithms [17], and "Rader-type" algorithms (e.g., [18]). The algebraic approach to these algorithms will be the subject of a future paper.

Besides explaining the theory and derivation of the fast algorithms, we also made an effort to present the results in concise, self-contained tables suitable for readers only interested in the actual algorithms. For these readers, we suggest to start with Table XV, which lists for each DCT and DST a reference to the best algorithms in this paper and their operations count. Also, further details on all algorithms can be found in a longer version of this paper available at [19].

Related work. The approach taken in this paper to derive fast algorithms using polynomial algebras builds on and extends early work on DFT algorithms. The known interpretation

of the DFT_n in terms of the polynomial algebra $\mathbb{C}[x]/(x^n-1)$ was used to derive and explain the (radix-2) Cooley-Tukey FFT by Auslander, Feig, and Winograd [20] using the Chinese remainder theorem (CRT). Equivalently, Nicholson [21] explains DFT and FFT using group theory; so does Beth [22] and generalizes the approach to more general groups. Winograd's DFT algorithms [23], [24] and his results in complexity theory make heavy use of polynomial algebras and the CRT. So do extensions of the above work by Burrus et al. [25], [26]. The first book on FFTs by Nussbaumer uses polynomial algebras as a framework for algorithm derivation [27].

For the DFT, it turns out that to derive the most important FFTs it is not necessary to work with polynomial algebras, but sufficient to work with index arithmetic modulo n. This approach is used in [28] as a consistent FFT framework. However, this arithmetic approach provides no insight into how to derive algorithms for other transforms. In contrast, our algebraic approach provides one theory that explains both DFT and DCT/DST algorithms (and algorithms for other transforms).

Further, the algorithm theory is a natural application and extension of ASP, which shows that polynomial algebras are a natural structure from an SP point of view [8].

The only (implicit) use of polynomial algebras for DCT or DST algorithms we found in the literature is in the derivation of a DCT (type 3) algorithm by Steidl [29], [30]. These papers provided valuable hints for developing the work in this paper.

We provide many more references to existing DCT/DST algorithms later and place them into our algebraically derived algorithm space.

Organization of the paper. Section II explains the relationship between polynomial algebras and transforms and connects to the notion of signal model in ASP. Most relevant for this paper are the polynomial algebras associated with the DFT and DCTs/DSTs. Section III introduces the notation we use to represent algorithms as products of structured matrices. Two general algebraic methods to derive algorithms from a polynomial algebra are explained in Section IV using the DFT as an example. Then we apply these methods to derive numerous Cooley-Tukey type algorithms for the DCTs and DSTs in Sections V and VI. These algorithms are analyzed and discussed in Section VII. We conclude in Section VIII.

II. Background: Polynomial Algebras and Transforms

In this section, we explain the relationship between polynomial algebras and transforms. Knowing the polynomial algebra associated with a transform is the key to understanding and deriving many of its fast algorithms as we show later.

Polynomial algebra. An *algebra* $\mathcal{A}$ is a vector space where also the multiplication of elements is defined such that the distributivity law holds. Examples of algebras include $\mathbb{C}$ (complex numbers) and $\mathbb{C}[x]$ (set of polynomials with complex coefficients).

Particularly important in signal processing (as shown below) are *polynomial algebras* $\mathcal{A} = \mathbb{C}[x]/p(x)$ with a suitable polynomial $p(x)$. They are defined as

$$\mathbb{C}[x]/p(x) = \{q(x) \mid \deg(q) < \deg(p)\}.$$

In words, given $p(x)$, $\mathbb{C}[x]/p(x)$ is the set of all polynomials of degree smaller than $\deg p$ with addition and multiplication modulo p. If $\deg(p) = n$, then $\dim(\mathbb{C}[x]/p(x)) = n$.

Polynomial transforms. Given a polynomial algebra $\mathcal{A} = \mathbb{C}[x]/p(x)$ and assuming that the zeros of $p(x)$ are pairwise distinct, given by $\alpha = (\alpha_0, \ldots, \alpha_{n-1})$, $\mathcal{A}$ can be decomposed using the *Chinese remainder theorem (CRT)* (see Appendix I) as follows:

$$\begin{array}{rccl} \mathcal{F}: & \mathbb{C}[x]/p(x) & \to & \bigoplus_{0\le k<n} \mathbb{C}[x]/(x-\alpha_k), \\ & s(x) & \mapsto & (s(\alpha_0), \ldots, s(\alpha_{n-1})). \end{array} \tag{1}$$

This mapping is linear. Hence, if we fix a basis $b = (p_0, \ldots, p_{n-1})$ in $\mathcal{A}$ and choose bases (of length 1) $(x^0) = (1)$ in each $\mathbb{C}[x]/(x-\alpha_k)$, $\mathcal{F}$ is represented by the matrix

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

We call $\mathcal{P}_{b,\alpha}$ the *polynomial transform* for $\mathcal{A}$ with basis b. It is possible to choose a different basis (β_k), $\beta_k \in \mathbb{C}$ in each $\mathbb{C}[x]/(x-\alpha_k)$, in which case we obtain the *scaled polynomial transform*

$$\mathcal{F} = \mathrm{diag}(1/\beta_0, \ldots, 1/\beta_{n-1}) \cdot \mathcal{P}_{b,\alpha}. \tag{3}$$

Connection to ASP. ASP [8] uses the concept of *signal model* to capture different SP frameworks. It is defined as a triple $(\mathcal{A}, \mathcal{M}, \Phi)$, where $\mathcal{A}$ is the algebra of filters, $\mathcal{M}$ the module of signals, and Φ generalized the concept of a z-transform. Once a signal model is given, other concepts, such as filtering, spectrum, and Fourier transform follow.

ASP asserts that if a signal model is for finite-length 1-D signals $\mathbf{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{C}^n$ and supports shift-invariance, then $\mathcal{A}$ has to be a polynomial algebra $\mathbb{C}[x]/p(x)$. Conversely, if $\mathbb{C}[x]/p(x)$ is given with basis $b = (p_0, \ldots, p_{n-1})$, then $\mathcal{A} = \mathcal{M} = \mathbb{C}[x]/p(x)$ with

$$\Phi: \ \mathbb{C}^n \to \mathcal{M}, \quad \mathbf{s} \mapsto s = s(x) = \sum_{0\le\ell<n} s_\ell p_\ell, \tag{4}$$

defines a signal model; Φ is the "z-transform" in this model. Once the model is given, other key SP concepts are automatically defined. For example, for the above model, filtering is the multiplication $h(x)s(x) \bmod p(x)$. The spectral decomposition of $s \in \mathcal{M}$ with respect to this model is given by (1), and $\mathcal{F}$ defined in (1) or in matrix form (2) or (3) (scaled or unscaled polynomial transform) is the Fourier transform for this model.

Example: finite time model. As an example consider the signal model $\mathcal{A} = \mathcal{M} = \mathbb{C}[x]/(x^n-1)$ with basis $b = (x^0, \ldots, x^{n-1})$ in $\mathcal{M}$ and thus, for $\mathbf{s} = (s_0, \ldots, s_{n-1})^T \in \mathbb{C}^n$,

$$\Phi: \ \mathbf{s} \mapsto s = s(x) = \sum_{0\le\ell<n} s_\ell x^\ell \in \mathbb{C}[x]/(x^n-1), \tag{5}$$

which we call the *finite z-transform* [8]. After applying the model, filtering is defined for $h = h(x) \in \mathcal{A}$ and $s = s(x) \in \mathcal{M}$ as

$$h(x)s(x) \bmod (x^n-1),$$

which is equivalent to computing the circular convolution of the coefficient sequences $\mathbf{h}$ and $\mathbf{s}$ [27].

TABLE I
SIGNAL MODELS ASSOCIATED TO THE DFTS.

$\mathcal{F}$	$p(x)$	b	$f(\alpha_k)$	(k,ℓ) entry of $\mathcal{F}$
DFT-1 = DFT	x^n-1	x^ℓ	1	$\omega_n^{k\ell}$
DFT-2	x^n-1	x^ℓ	$\alpha_k^{1/2}$	$\omega_n^{k(\ell+1/2)}$
DFT-3	x^n+1	x^ℓ	1	$\omega_n^{(k+1/2)\ell}$
DFT-4	x^n+1	x^ℓ	$\alpha_k^{1/2}$	$\omega_n^{(k+1/2)(\ell+1/2)}$
DFT(a)	x^n-a	x^ℓ	1	$\omega_n^{k\ell}\ \sqrt[n]{a}^\ell$

The zeros of x^n-1 are $\alpha=(\omega_n^0,\dots,\omega_n^{n-1})$ with $\omega_n=e^{-2\pi j/n}$. Hence, the Fourier transform for this model is given by

$$\mathcal{F}:\ \mathbb{C}[x]/(x^n-1)\to\bigoplus_{0\le k<n}\mathbb{C}[x]/(x-\omega_n^k)$$

and in matrix form

$$\mathcal{F}=\mathcal{P}_{b,\alpha}=[\omega_n^{k\ell}]_{0\le k,\ell<n}=\mathrm{DFT}_n \tag{6}$$

is precisely the discrete Fourier transform. This explains why we call this signal model the *finite time model*.

A. Signal Models for DFTs and DTTs

In this section we list the signal models (and hence polynomial algebras) for 4 types of DFTs and all the 16 DCTs and DSTs introduced in [31]. We refer to the DCTs and DSTs collectively as DTTs (discrete trigonometric transforms) even though this class is actually larger (e.g., including the discrete Hartley transform other real discrete Fourier transforms). Further, we define 4 types of skew DTTs, which were introduced in [9], and which are necessary to derive a complete set of algorithms.

Each of these transforms is a Fourier transform for a finite shift-invariant regular 1-D signal model. These models are uniquely determined by $p(x)$, which defines $\mathcal{A}=\mathcal{M}=\mathbb{C}[x]/p(x)$, and the basis b of $\mathcal{M}$, which defines Φ in (4). The model in turn uniquely determines the associated *polynomial* Fourier transform $\mathcal{P}_{b,\alpha}$ in (2). To characterize an *arbitrary* Fourier transform, we need to specify in addition the diagonal matrix in (3). We do this in the following by providing a function f such that the diagonal matrix is given by

$$D_f=\mathrm{diag}_{0\le k<n}(f(\alpha_k)),$$

where the α_k are, as before, the zeros of $p(x)$.

A derivation and explanation of these signal models can be found in [8], [9].

DFTs. We consider the DFTs of type 1–4 following [32], and a parameterized $\mathrm{DFT}(a)$. Each one of those is a Fourier transform for a variant of the finite time model as shown in Table I. In particular, DFT-1 $=\mathrm{DFT}(1)=$ DFT. Both type 1 and 3 are polynomial transforms and special cases of $\mathrm{DFT}(a)$ briefly discussed next.

Consider the signal model given by $\mathcal{A}=\mathcal{M}=\mathbb{C}[x]/(x^n-a)$ and $\Phi:\ \mathbf{s}\mapsto\sum_{0\le\ell<n}s_\ell x^\ell$. The zeros of x^n-a are the n nth roots of a and thus straightforward computation yields as polynomial Fourier transform

$$\mathrm{DFT}(a)=\mathcal{P}_{b,\alpha}=\mathrm{DFT}_n\,\mathrm{diag}_{0\le\ell<n}(\sqrt[n]{a}^\ell), \tag{7}$$

TABLE II
SIGNAL MODELS ASSOCIATED TO THE 16 DTTS (DCTS AND DSTS).

$\mathcal{F}$	$p=p(x)$	b	$f(\alpha_k),\ \alpha_k=\cos\theta$
T-group			
DCT-3	T_n	T_ℓ	1
DST-3	T_n	U_ℓ	$\sin(\theta)$
DCT-4	T_n	V_ℓ	$\cos(\theta/2)$
DST-4	T_n	W_ℓ	$\sin(\theta/2)$
U-group			
DCT-1	$(x^2-1)U_{n-2}$	T_ℓ	1
DST-1	U_n	U_ℓ	$\sin(\theta)$
DCT-2	$(x-1)U_{n-1}$	V_ℓ	$\cos(\theta/2)$
DST-2	$(x+1)U_{n-1}$	W_ℓ	$\sin(\theta/2)$
V-group			
DCT-7	$(x+1)V_{n-1}$	T_ℓ	1
DST-7	V_n	U_ℓ	$\sin(\theta)$
DCT-8	V_n	V_ℓ	$\cos(\theta/2)$
DST-8	$(x+1)V_{n-1}$	W_ℓ	$\sin(\theta/2)$
W-group			
DCT-5	$(x-1)W_{n-1}$	T_ℓ	1
DST-5	W_n	U_ℓ	$\sin(\theta)$
DCT-6	$(x-1)W_{n-1}$	V_ℓ	$\cos(\theta/2)$
DST-6	W_n	W_ℓ	$\sin(\theta/2)$

where $\sqrt[n]{a}=|a|^{1/n}e^{\nu j/n}$ for $a=|a|e^{\nu j}$.

DTTs. The 16 DTTs are Fourier transforms for finite *space* models, which are defined in Table II. Space in ASP means that the shift operator on which the models are based operates undirected versus the directed operation of the time shift. In contrast to the time models, the basis polynomials are now Chebyshev polynomials of the first (T_ℓ), second (U_ℓ), third (V_ℓ), or fourth (W_ℓ) kind instead of x^ℓ. Appendix II shows their definitions and their properties that we will use in this paper.

As an example consider the most commonly used DCT-2$_n$. The associated model is given from Table II by $\mathcal{A}=\mathcal{M}=\mathbb{C}[x]/(x-1)U_{n-1}$ and Φ is the finite V-transform mapping $\mathbf{s}\mapsto\sum_{0\le\ell<n}s_\ell V_\ell$. The zeros of $(x-1)U_{n-1}$ are given by $\alpha_k=\cos(k\pi/n)$, $0\le k<n$ (see Table XVI in Appendix II). Thus the unique *polynomial* Fourier transform for the model is given by

$$\mathcal{P}_{b,\alpha}=[V_\ell(\alpha_k)]_{0\le k,\ell<n}=\left[\frac{\cos\frac{k(\ell+1/2)\pi}{n}}{\cos\frac{k\pi}{2n}}\right]_{0\le k,\ell<n}. \tag{8}$$

Multiplying $\mathcal{P}_{b,\alpha}$ from the left by the scaling diagonal

$$\mathrm{diag}_{0\le k<n}(\cos(k\pi/(2n)))=\mathrm{diag}_{0\le k<n}(\cos(\mathrm{acos}(\alpha_k)/2))$$

cancels the denominator to yield

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

which identifies DCT-2 as a Fourier transform of the form (3) for the specified signal model.

The definitions of all 16 DTTs are given in Table III. Types 1, 4, 5, 8 are symmetric; types 2, 3 and 6, 7 are transposes of each other, respectively.

TABLE III

8 TYPES OF DCTS AND DSTS OF SIZE n. THE ENTRY AT ROW k AND COLUMN ℓ IS GIVEN FOR $0 \le k, \ell < n$.

type	DCTs	DSTs
1	$\cos k\ell\frac{\pi}{n-1}$	$\sin(k+1)(\ell+1)\frac{\pi}{n+1}$
2	$\cos k(\ell+\frac{1}{2})\frac{\pi}{n}$	$\sin(k+1)(\ell+\frac{1}{2})\frac{\pi}{n}$
3	$\cos(k+\frac{1}{2})\ell\frac{\pi}{n}$	$\sin(k+\frac{1}{2})(\ell+1)\frac{\pi}{n}$
4	$\cos(k+\frac{1}{2})(\ell+\frac{1}{2})\frac{\pi}{n}$	$\sin(k+\frac{1}{2})(\ell+\frac{1}{2})\frac{\pi}{n}$
5	$\cos k\ell\frac{\pi}{n-\frac{1}{2}}$	$\sin(k+1)(\ell+1)\frac{\pi}{n+\frac{1}{2}}$
6	$\cos k(\ell+\frac{1}{2})\frac{\pi}{n-\frac{1}{2}}$	$\sin(k+1)(\ell+\frac{1}{2})\frac{\pi}{n+\frac{1}{2}}$
7	$\cos(k+\frac{1}{2})\ell\frac{\pi}{n-\frac{1}{2}}$	$\sin(k+\frac{1}{2})(\ell+1)\frac{\pi}{n+\frac{1}{2}}$
8	$\cos(k+\frac{1}{2})(\ell+\frac{1}{2})\frac{\pi}{n+\frac{1}{2}}$	$\sin(k+\frac{1}{2})(\ell+\frac{1}{2})\frac{\pi}{n-\frac{1}{2}}$

TABLE IV

4 TYPES OF SKEW DTTS AND ASSOCIATED SIGNAL MODELS. THE PARAMETER r IS IN $0 \le r \le 1$. FOR $r = 1/2$ THEY REDUCE TO THE T-GROUP DTTS.

$\mathcal{F}$	$p = p(x)$	b	$f = f(\alpha_k),\ \alpha_k = \cos\theta$
DCT-3(r)	$T_n - \cos r\pi$	T_ℓ	1
DST-3(r)	$T_n - \cos r\pi$	U_ℓ	$\sin(\theta)$
DCT-4(r)	$T_n - \cos r\pi$	V_ℓ	$\cos(\theta/2)$
DST-4(r)	$T_n - \cos r\pi$	W_ℓ	$\sin(\theta/2)$

Every DTT has a corresponding polynomial transform, which we write as $\overline{\text{DTT}}$. For example $\overline{\text{DCT-2}}_n$ is the matrix in (8). For the DCTs of types 1, 3, 5, 7, the scaling function is 1 (see Table II) and hence $\overline{\text{DTT}} = \text{DTT}$ in these cases.

We will later see that, in some cases, the polynomial DTTs have a lower arithmetic cost than the corresponding DTTs, which makes them suitable choices in applications in which the transform output is scaled anyway.

We divide the DTTs into 4 groups, called T-, U-, V-, and W-group depending on p as shown in Table II. Within each group, the algebra and module are (almost) the same. This leads to sparse relationships (conversion using sparse matrices) between DTTs in one group as we have shown in [9]; examples we will use are in Appendix III.

Further, within a group, the DTTs are pairwise *dual* (they have flipped associated boundary conditions [9]), which means that they can be translated into each other without additional arithmetic operations (see (54) in Appendix III).

Skew DTTs. We introduced the skew DTTs in [9] since their associated signal models are also reasonable space models, but, more importantly, because they are important building blocks of Cooley-Tukey type algorithms as we will show in this paper. There are 4 types of skew DTTs, each parameterized by $0 < r < 1$. They generalize the four T-group DTTs (DCT/DST of type 3/4) and have the same scaling functions as those. The models that define these transforms are shown in Table IV. The corresponding polynomial versions are again denoted using a bar as in $\overline{\text{DCT-3}}_n(r)$.

To obtain the exact form of these transforms, we need the zeros of the polynomial $T_n - \cos r\pi$ and choose a fixed order of these zeros. This is done in the following lemma.

Lemma 1 Let $0 \le r \le 1$. We have the factorization

$$T_n - \cos r\pi = 2^{n-1} \prod_{0 \le i < n} (x - \cos \tfrac{r+2i}{n}\pi), \tag{9}$$

which determines the zeros of $T_n - \cos r\pi$. We order the zeros as $\alpha = (\cos r_0\pi, \ldots, \cos r_{n-1}\pi)$, such that $0 \le r_i \le 1$, and $r_i < r_j$ for $i < j$. The list of the r_k is given by the concatenation

$$(r_k)_{0 \le k < n} = \bigcup_{0 \le i < n/2} (\tfrac{r+2i}{n}, \tfrac{2-r+2i}{n})$$

for n even, and by

$$(r_k)_{0 \le k < n} = \Big(\bigcup_{0 \le i < \frac{n-1}{2}} (\tfrac{r+2i}{n}, \tfrac{2-r+2i}{n}) \Big) \cup (\tfrac{r+n-1}{n})$$

for n odd. In the particular case of $r = 1/2$ or $\cos r\pi = 0$, we thus have $\alpha = (\cos(k+1/2)\pi/n)_{0 \le k < n}$ as in Table XVI in Appendix II.

For example, the DCT-3$_n(r)$ is given by the matrix

$$\text{DCT-3}_n(r) = [\cos r_k \ell\pi]_{0 \le k, \ell < n},$$

where the r_k are provided by Lemma 1.

Relationships between the skew DTTs and skew and non-skew DTTs are shown in Appendix III.

III. BACKGROUND: FAST TRANSFORM ALGORITHMS

In this section, we explain the notation that we use to represent and manipulate transform algorithms.

Representation of algorithms. We discuss two representations for transforms[1] and their algorithms. Traditionally, transforms in SP are written as summation like

$$y_k = \sum_{0 \le \ell < n} t_{k,\ell} s_\ell, \tag{10}$$

where $\mathbf{s} = (s_0, \ldots, s_{n-1})^T$ is the input signal, $\mathbf{y} = (y_0, \ldots, y_{n-1})^T$ the output signal, and $t_{k,\ell}$ the transform coefficients. This representation is usually adopted because these transforms are thought of as truncated versions of infinite series expansions. Correspondingly, algorithms are written as sequences of such summations, cleverly organized to reduce the operations count.

A different approach, equivalent in content, represents transforms as matrix-vector products

$$\mathbf{y} = T\mathbf{s}, \quad \text{where } T = [t_{k,\ell}]_{0 \le k, \ell < n}. \tag{11}$$

The transform matrix is T, and transform algorithms correspond to factorizations of T into a product of sparse structured matrices. This approach was adopted for the DFT in [33], [28] (and as early as [21]), but also for other transforms in various research papers on fast transform algorithms.

In ASP, we adopt the second approach for two reasons. First, in ASP, a transform is a decomposition of a signal model into its spectral components, e.g., as in (1). This decomposition is a base change and hence represented by a matrix. Further, we

[1]By "transforms," we mean here those computing some sort of spectrum of finite length discrete signals like the DFT or DTTs.

TABLE V
AUXILIARY MATRICES USED IN THIS PAPER.

$$I_n = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix},\ J_n = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix},\ S_n = \begin{bmatrix} 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \\ & & & 1 \end{bmatrix},$$

$$F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},\ Z_n = \begin{bmatrix} & & & 0 \\ & \iddots & 0 & 1 \\ & \iddots & \iddots & \\ 0 & 1 & & \end{bmatrix},\ \overline{Z}_n = \begin{bmatrix} & & 1 & 0 \\ & \iddots & \iddots & \\ 1 & 0 & & \\ 0 & & & \end{bmatrix}$$

derive later fast algorithms by performing this decomposition in steps, where the steps correspond to sparse base changes, i.e., structured, sparse matrices.

Second, there are other advantages of the matrix representation from an algorithmic and implementation point of view. Namely, it reveals the structure of the algorithm and makes it is easy to manipulate it to derive variants.

Notation. We use the following notation to represent structured matrices.

As basic matrices, we use the ones shown in Table V. F_2 is the *butterfly matrix*.

Further, we use permutation matrices defined by their corresponding permutations

$$P:\ i \mapsto f(i), \quad 0 \le i < n,$$

which means that the matrix P has in row i the entry 1 at position $f(i)$ and the entry 0 elsewhere. In this paper, all matrix indices start with 0. Most important is the $n \times n$ *stride* permutation matrix, which can be defined for $m|n$ by

$$L^n_m:\ i_2 \tfrac{n}{m} + i_1 \mapsto i_1 m + i_2 \tag{12}$$

for $0 \le i_1 < \frac{n}{m},\ 0 \le i_2 < m$. This definition shows that L^n_m transposes an $\frac{n}{m} \times m$ matrix stored in row-major order. Alternatively, we can write

$$\begin{aligned} L^n_m:\quad & i \mapsto im \bmod n-1, \quad \text{for } 0 \le i < n-1, \\ & n-1 \mapsto n-1. \end{aligned}$$

Since the last point $n-1$ is fixed, we can define an *odd* stride permutation $\widehat{L}$ for $m \mid n+1$ as the restriction of L^{n+1}_m to the first n points,

$$\widehat{L}^n_m:\ i \mapsto im \bmod n. \tag{13}$$

Analogous to the stride permutation, $(\widehat{L}^n_m)^{-1} = \widehat{L}^n_{(n+1)/m}$, and ($\oplus$ is defined right below)

$$L^n_m = \widehat{L}^{n-1}_m \oplus I_1.$$

Diagonal matrices are written as $\mathrm{diag}(\alpha_0, \ldots, \alpha_{n-1})$.

Further, we use matrix operators, like the product of matrices, the direct sum

$$A \oplus B = \begin{bmatrix} A & \\ & B \end{bmatrix},$$

and the Kronecker or tensor product

$$A \otimes B = [a_{k,\ell} B]_{k,\ell}, \quad \text{for } A = [a_{k,\ell}].$$

Since $I_k \otimes A = A \oplus \ldots \oplus A$, we will write the direct sum of different $m \times m$ matrices A_i, $0 \le i < k$, as

$$I_k \otimes_i A_i = A_0 \oplus \ldots \oplus A_{k-1}.$$

We will occasionally also construct a larger matrix as a matrix of matrices, e.g.,

$$\begin{bmatrix} A & B \\ B & A \end{bmatrix}.$$

Transposition and inversion. If an algorithm for a transform is given as a product of sparse matrices built from the constructs above, then an algorithm for the transpose or inverse of the transform can be readily derived using mathematical properties including

$$\begin{aligned} &(AB)^T = B^T A^T, && (AB)^{-1} = B^{-1} A^{-1}, \\ &(A \oplus B)^T = A^T \oplus B^T, && (A \oplus B)^{-1} = A^{-1} \oplus B^{-1}, \\ &(A \otimes B)^T = A^T \otimes B^T, && (A \otimes B)^{-1} = A^{-1} \otimes B^{-1}. \end{aligned} \tag{14}$$

Permutation matrices are orthogonal, i.e., $P^T = P^{-1}$. The transposition or inversion of diagonal matrices is obvious. Note that in general the inverse of a sparse matrix is not sparse.

Arithmetic cost. We will analyze the number of operations of the algorithms using the notation of a triple (a, m, m_2), where a is the number of additions or subtractions, m_2 the number of multiplications by a 2-power not equal to 1, and m the number of remaining multiplications by constants not equal to -1. The total operations count is then given by $c = a + m + m_2$.

In many SP publications the term complexity is used for the operations count or arithmetic cost. In a strict sense this is not correct, since complexity is a property of a problem (like computing a DFT), not of an algorithm (like a specific FFT). Thus we will use the term cost.

IV. ALGEBRAIC DERIVATION OF FAST TRANSFORM ALGORITHMS

In this section, we algebraically derive Fourier transform algorithms, where the term "Fourier transform" is meant in the general sense of the algebraic signal processing theory (e.g., including the DCTs, DSTs, and other trigonometric transforms).

Overview. We consider finite shift-invariant regular signal models, i.e., models of the form $\mathcal{A} = \mathcal{M} = \mathbb{C}[x]/p(x)$ and

$$\Phi:\ \mathbb{C}^n \to \mathcal{M}, \quad \mathbf{s} \mapsto \sum_{0 \le \ell < n} s_\ell p_\ell,$$

where $b = (p_0, \ldots, p_{n-1})$ is a basis for $\mathcal{M}$. Further, we assume that p has pairwise different zeros, which causes the spectrum to consist of distinct one-dimensional submodules. The Fourier transform in these cases is given by the CRT (1) and as a matrix $\mathcal{F}$ takes the form of a polynomial transform in (2) or a scaled polynomial transform in (3).

Assume a transform $\mathcal{F}$ is given. The basic idea of the algebraic approach is to derive algorithms for $\mathcal{F}$ by manipulating the associated signal model $(\mathcal{A}, \mathcal{M}, \Phi)$, instead of manipulating the matrix entries of $\mathcal{F}$. Namely, (1) shows that $\mathcal{F}$ decomposes $\mathbb{C}[x]/p(x)$ into one-dimensional polynomial

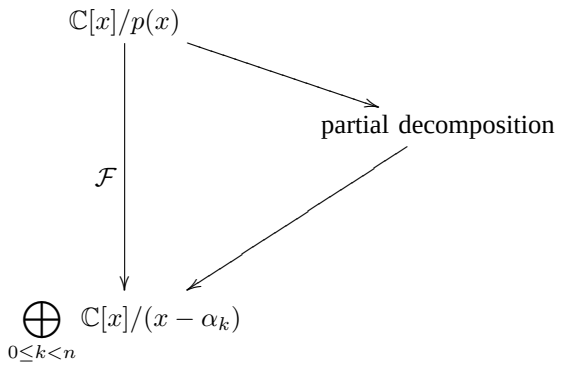


Fig. 1. Basic idea behind the algebraic derivation of Cooley-Tukey type algorithms for a Fourier transform $\mathcal{F}$.

algebras, i.e., its spectrum. Fast algorithms are obtained, as we will show, by performing this decomposition *in steps* using an intermediate subalgebra. Fig. 1 visualizes this approach.

The advantage of the algebraic derivation is that it identifies a few general principles that account for many different algorithms when instantiated for different transforms. Further, the derivation is often greatly simplified, since the only task required is to determine the base change matrices when instantiating the general theorems.

In this paper, we focus on explaining and deriving "Cooley-Tukey type" algorithms as we will call them. As the name suggests, these algorithms will include, as well as generalize, the equally named algorithms for the DFT. The latter will serve as examples in this section. Our main focus in the remainder of this paper will then be the derivation of analogous algorithms for the DCTs and DSTs, most of which have not been reported in the literature. All these new algorithms are non-orthogonal, i.e., are not constructed exclusively from butterflies and 2×2 rotations. Orthogonal algorithms do exist and will be captured algebraically in a future paper. Also "Rader" type algorithms, which apply when the above decomposition methods fail (for the DFT in the case of a prime size), will be explained in a future paper.

The existence and usefulness of Cooley-Tukey type algorithms for the above signal model depends on properties of both $p(x)$ and b. Specifically, algorithms may arise from two different basic principles, which manifest themselves as a property of p:

1) *Cooley-Tukey type (factorization)*: $p(x) = q(x) \cdot r(x)$ factorizes; and
2) *Cooley-Tukey type (decomposition)*: $p(x) = q(r(x))$ decomposes.

Clearly, 1) is always possible (if we consider the basefield $\mathbb{C}$), but 2) is a special property of p.

In both cases, as we will show, we obtain a matrix factorization of $\mathcal{F}$ containing smaller transforms, i.e., the algorithm for $\mathcal{F}$ is recursive. Its usefulness as a *fast* algorithm, however, depends on the basis b. In the remainder of this section, we derive the general form of these two algorithms. We focus on Fourier transforms that are polynomial transforms $\mathcal{P}_{b,\alpha}$. Since any Fourier transform has the form $\mathcal{F} = D\mathcal{P}_{b,\alpha}$ in (3), where D is a diagonal matrix, any algorithm for $\mathcal{P}_{b,\alpha}$ also yields an algorithm for $\mathcal{F}$.

A. Cooley-Tukey Type Algorithms: Factorization

A simple way to decompose $\mathbb{C}[x]/p(x)$ in steps is to use a factorization $p(x) = q(x) \cdot r(x)$ of p. Namely, let $k = \deg(q)$ and $m = \deg(r)$, then

$$\begin{aligned} & \mathbb{C}[x]/p(x) \\ \to\ & \mathbb{C}[x]/q(x) \oplus \mathbb{C}[x]/r(x) && (15)\\ \to\ & \bigoplus_{0\le i<k} \mathbb{C}[x]/(x-\beta_i) \oplus \bigoplus_{0\le j<m} \mathbb{C}[x]/(x-\gamma_j) && (16)\\ \to\ & \bigoplus_{0\le i<n} \mathbb{C}[x]/(x-\alpha_i). && (17)\end{aligned}$$

Here the β_i are the zeros of q and the γ_j are the zeros of r, which implies that both are a subset of the zeros α_i of p. Both steps (15) and (16) use the Chinese remainder theorem, whereas (17) is just a reordering of the spectrum. The corresponding factorization of the Fourier transform is provided in the following theorem.

Theorem 1 (Cooley-Tukey Type Algorithm by Factorization) Let $p(x) = q(x) \cdot r(x)$, and let c and d be bases of $\mathbb{C}[x]/q(x)$ and $\mathbb{C}[x]/r(x)$, respectively. Further, denote with β and γ the lists of zeros of q and r, respectively. Then

$$\mathcal{P}_{b,\alpha} = P(\mathcal{P}_{c,\beta} \oplus \mathcal{P}_{d,\gamma})B.$$

The matrix B corresponds to (15), which maps the basis b to the concatenation (c, d) of the bases c and d, and P is the permutation matrix mapping the concatenation (β, γ) to the list of zeros α in (17).

Note that the factorization of $\mathcal{P}_{b,\alpha}$ in Theorem 1 is useful as a fast algorithm, i.e., reduces the arithmetic cost, only if B is sparse or can be multiplied with efficiently. Referring to Fig. 1, the "partial decomposition" is step (15) corresponding to B.

Example: DFT. The DFT is a (polynomial) Fourier transform for the regular signal model given by $\mathcal{A} = \mathcal{M} = \mathbb{C}[x]/(x^n - 1)$ with basis $b = (1, x, \ldots, x^{n-1})$ as shown in (6). We assume $n = 2m$ and use the factorization $x^{2m} - 1 = (x^m - 1)(x^m + 1)$. Applying Theorem 1 yields the following decomposition steps:

$$\begin{aligned} & \mathbb{C}[x]/(x^n - 1) \\ \to\ & \mathbb{C}[x]/(x^m - 1) \oplus \mathbb{C}[x]/(x^m + 1) && (18)\\ \to\ & \bigoplus_{0\le i<m} \mathbb{C}[x]/(x - \omega_n^{2i}) \oplus \bigoplus_{0\le i<m} \mathbb{C}[x]/(x - \omega_n^{2i+1}) && (19)\\ \to\ & \bigoplus_{0\le i<n} \mathbb{C}[x]/(x - \omega_n^{i}). && (20)\end{aligned}$$

As bases in the smaller modules $\mathbb{C}[x]/(x^m - 1)$ and $\mathbb{C}[x]/(x^m + 1)$, we choose $c = d = (1, x, \ldots, x^{m-1})$. We note that from this point on the derivation of the algorithm is entirely mechanical.

First, we derive the base change matrix B corresponding to (18). To do so, we have to express the base elements $x^\ell \in b$ in the basis (c, d) (concatenation); the coordinate vectors are

the columns of B. For $0 \le \ell < m$, x^ℓ is actually contained in c and d, so the first m columns of B are

$$B = \begin{bmatrix} I_m & * \\ I_m & * \end{bmatrix},$$

where the entries $*$ are determined next. For the base elements $x^{m+\ell}$, $0 \le \ell < m$, we have

$$\begin{aligned} x^{m+\ell} &\equiv x^\ell \bmod (x^m - 1), \\ x^{m+\ell} &\equiv -x^\ell \bmod (x^m + 1), \end{aligned}$$

which yields the final result

$$B = \begin{bmatrix} I_m & I_m \\ I_m & -I_m \end{bmatrix} = \mathrm{DFT}_2 \otimes I_m.$$

Next, we consider step (19). $\mathbb{C}[x]/(x^m-1)$ is decomposed by DFT_m and $\mathbb{C}[x]/(x^m+1)$ by DFT-3$_m$ (see Table II). Finally, the permutation in step (20) is the perfect shuffle L^{2m}_m, which interleaves the even and odd spectral components (even and odd exponents of ω_n). The final algorithm obtained is

$$\mathrm{DFT}_{2m} = L^n_m(\mathrm{DFT}_m \oplus \text{DFT-3}_m)(\mathrm{DFT}_2 \otimes I_m).$$

Using DFT-3$_m = \mathrm{DFT}_m D_m$ with $D_m = \mathrm{diag}_{0 \le i < m}(\omega_n^i)$, we get the better known form

$$\begin{aligned} \mathrm{DFT}_{2m} &= L^n_m(\mathrm{DFT}_m \oplus \mathrm{DFT}_m D_m)(\mathrm{DFT}_2 \otimes I_m) \\ &= L^n_m(I_2 \otimes \mathrm{DFT}_m)(I_m \oplus D_m)(\mathrm{DFT}_2 \otimes I_m). \end{aligned}$$

The last expression is the radix-2 decimation-in-frequency Cooley-Tukey FFT. The corresponding decimation-in-time version is obtained by transposition using (14) and that the DFT is symmetric. The entries of the diagonal matrix $I_m \oplus D_m$ are commonly called *twiddle factors*.

Remarks. Theorem 1 is well-known, as it is the CRT for polynomials expressed in matrix form. The above DFT example is equivalent to the derivation in [27] or [20]. Theorem 1 is also used as the first step in the derivation of Winograd DFT algorithms [24]. There, the polynomial $x^n - 1$ is completely factored over the rational numbers, and the DFT decomposed accordingly.

An algorithm based on Theorem 1 is naturally implemented recursively and requires the availability of algorithms for the smaller transforms.

The algorithm derivation method in Theorem 1 is always applicable if the basefield is $\mathbb{C}$, but in general the base change matrix B will be dense and without useful structure. Otherwise, every polynomial transform would have a fast algorithm, which by the current state of knowledge is not the case. The subsequent method is different in that respect: it requires a special property of $p(x)$, and only this property leads to the typical general radix Cooley-Tukey FFT structure.

B. Cooley-Tukey Type Algorithms: Decomposition

A more interesting factorization of $\mathcal{F} = \mathcal{P}_{b,\alpha}$ can be derived if $p(x)$ *decomposes* into two polynomials, $p(x) = q(r(x))$. If $\deg(q) = k$ and $\deg(r) = m$, then $\deg(p) = n = km$, i.e., the degree of p is necessarily composite. In this case, the polynomial $r(x)$ generates a subalgebra $\mathcal{B}$ of $\mathcal{A} = \mathbb{C}[x]/p(x)$ consisting of all polynomials in $r(x)$. Setting $y = r(x)$ makes the structure of $\mathcal{B}$ evident: $\mathcal{B} = \mathbb{C}[y]/q(y)$.

TABLE VI
ALGEBRAS OCCURING IN THE ALGORITHM DERIVATION BASED ON THE DECOMPOSITION $p(x) = q(y)$, $y = r(x)$.

algebra	basis	zeros
$\mathcal{A} = \mathbb{C}[x]/p(x)$	$b = (p_0, \ldots, p_{n-1})$	$\alpha = (\alpha_0, \ldots, \alpha_{n-1})$
$\mathcal{B} = \mathbb{C}[y]/q(y)$	$c = (q_0, \ldots, q_{k-1})$	$\beta = (\beta_0, \ldots, \beta_{k-1})$
$\mathcal{C}_i = \mathbb{C}[x]/r(x) - \beta_i$	$d = (r_0, \ldots, r_{m-1})$	$\gamma_i = (\gamma_{i,0}, \ldots, \gamma_{i,m-1})$

Let $\beta = (\beta_0, \ldots, \beta_{k-1})$ be the zeros of q and let $\gamma_i = (\gamma_{i,0}, \ldots, \gamma_{i,m-1})$ be the zeros of $r(x) - \beta_i$, $0 \le i < k$. Then

$$p(x) = \prod_{0 \le i < k} (r(x) - \beta_i) = \prod_{0 \le i < k} \prod_{0 \le j < m} (x - \gamma_{i,j}).$$

In particular, each $\gamma_{i,j}$ is a zero α_ℓ of p. Now we decompose $\mathbb{C}[x]/p(x)$ in the following steps:

$$\begin{aligned} \mathbb{C}[x]/p(x) &\to \mathbb{C}[x]/q(r(x)) && (21) \\ &\to \bigoplus_{0 \le i < k} \mathbb{C}[x]/(r(x) - \beta_i) && (22) \\ &\to \bigoplus_{0 \le i < k} \bigoplus_{0 \le j < m} \mathbb{C}[x]/(x - \gamma_{i,j}) && (23) \\ &\to \bigoplus_{0 \le i < n} \mathbb{C}[x]/(x - \alpha_i). && (24) \end{aligned}$$

Steps (22) and (23) use the Chinese remainder theorem. To derive the corresponding factorization of $\mathcal{P}_{b,\alpha}$ into four factors, we first choose bases in the occuring algebras as shown in Table VI. Note that in each $\mathcal{C}_i$ we choose the same basis d.

In the first step (21), we do not change $\mathcal{A}$ but only make a base change in $\mathcal{A}$ from the given basis b to the new basis

$$\begin{aligned} b' = (&r_0 q_0(r), \ldots, r_{m-1} q_0(r), \\ &\ldots \\ &r_0 q_{k-1}(r), \ldots, r_{m-1} q_{k-1}(r)), \end{aligned} \tag{25}$$

which is a product of the "coarse" basis of the subalgebra $\mathcal{B} \le \mathcal{A}$ with the "fine" common basis of the $\mathcal{C}_i$. We denote the base change matrix for $b \to b'$ with B.

Next, we compute the base change matrix M corresponding to the coarse decomposition (22) of $\mathcal{A}$ with basis b' and the basis d in each $\mathcal{C}_i$ on the right hand side. Let $r_\ell(x) q_j(r(x)) \in b'$. Then

$$r_\ell(x) q_j(r(x)) \equiv r_\ell(x) q_j(\beta_i) \bmod (r(x) - \beta_i),$$

which is $q_j(\beta_i)$ times the ℓth base vector $r_\ell(x)$ in d. Thus,

$$M = [q_j(\beta_i) \cdot I_m]_{0 \le i,j < k} = \mathcal{P}_{c,\beta} \otimes I_m.$$

The third step (23) decomposes the summands in (22) by their respective Fourier transforms $\mathcal{P}_{d,\gamma_i}$.

The final step (24) reorders the one-dimensional summands by a suitable permutation P. We summarize the resulting factorization in the following theorem.

Theorem 2 (Cooley-Tukey Type Algorithms by Decomposition) Let $p(x) = q(r(x))$. Using the notation in Table VI,

$$\mathcal{P}_{b,\alpha} = P(I_k \otimes_i \mathcal{P}_{d,\gamma_i})(\mathcal{P}_{c,\beta} \otimes I_m)B,$$

where B is the base change matrix mapping b to b', and P is the permutation matrix mapping the concatenation of the γ_i onto α in (24).

As in Theorem 1, the usefulness of this factorization as fast algorithm depends on the base change matrix B. Referring to Fig. 1, the "partial decomposition" is step (22).

Example: DFT. Let $\mathcal{A} = \mathcal{M} = \mathbb{C}[x]/(x^n - 1)$ with basis $b = (1, x, \ldots, x^{n-1})$ be the signal model associated with the DFT_n. Further, assume that $n = km$. Then the polynomial $p(x) = x^n - 1$ decomposes as

$$x^n - 1 = (x^m)^k - 1, \tag{26}$$

i.e., $p(x) = q(r(x))$ with $q(y) = y^k - 1$ and $r(x) = x^m$. Thus Theorem 2 is applicable. The zeros of $q(y)$ are $\beta_i = \omega_k^i$, $0 \le i < k$. Further, we choose $c = (1, y, \ldots, y^{k-1})$ as basis in $\mathcal{B} = \mathbb{C}[x]/q(y)$ and $d = (1, x, \ldots, x^{m-1})$ as basis in each $\mathcal{C}_i = \mathbb{C}[x]/(x^m - \omega_k^i)$. We find that $b' = b$ in (25), which implies $B = I_n$.

Thus, the matrix $\mathrm{DFT}_k \otimes I_m$ performs the following coarse decomposition corresponding to (22):

$$\mathbb{C}[x]/(x^n - 1) \to \bigoplus_{0 \le i < k} \mathbb{C}[x]/(x^m - \omega_k^i).$$

Each $\mathbb{C}[x]/(x^m - \omega_k^i)$ is further decomposed as

$$\mathbb{C}[x]/(x^m - \omega_k^i) \to \bigoplus_{0 \le j < m} \mathbb{C}[x]/(x - \omega_n^{jk+i})$$

and the associated polynomial transform is a special case of (7):

$$\mathrm{DFT}_m(\omega_k^i) = \mathrm{DFT}_m \cdot \mathrm{diag}_{j=0}^{m-1}(\omega_n^{ij}).$$

At this point, corresponding to (23), $\mathbb{C}[x]/p(x)$ is completely decomposed, but the spectrum is ordered according to $jk + i$, $0 \le i < m$, $0 \le j < k$ (j runs faster). The desired order is $im + j$. Thus, we need to apply the permutation $jk + i \mapsto im + j$, which is exactly the stride permutation L_m^n in (12).

In summary, we obtain the Cooley-Tukey decimation-in-frequency FFT with arbitrary radix:

$$\begin{aligned} & L_m^n \Big(I_k \otimes_i \mathrm{DFT}_m \cdot \mathrm{diag}_{j=0}^{m-1}(\omega_n^{ij}) \Big) (\mathrm{DFT}_k \otimes I_m) \\ = \quad & L_m^n (I_k \otimes \mathrm{DFT}_m) T_m^n (\mathrm{DFT}_k \otimes I_m), \end{aligned} \tag{27}$$

where the matrix T_m^n is diagonal and usually called the *twiddle matrix*. Transposition of (27) using (14) yields the corresponding decimation-in-time version.

Again, we note that after recognizing the decomposition property (26), the derivation is completely mechanical.

Remarks. Theorem 2 makes use of the CRT in (22) and (23), but it is the decomposition property of $x^n - 1$ that produces the general radix structure. The previous work on the algebraic derivation of this FFT did not make use of decompositions. As we briefly discuss the next section, the decomposition is a special case of a more general algebraic principle.

An algorithm based on Theorem 2 is naturally implemented recursively, where the smaller transform are called inside loops coresponding to the tensor product and direct sum, respectively. The structure of the algorithm also makes it a candidate for efficient vectorization and parallelization [5], [6], [33], [28].

C. Remarks on Algebraic Principles

The algorithms derived in this section are based on the factorization or decomposition of the polynomial $p(x)$ in the signal model provided by $\mathbb{C}[x]/p(x)$ (and basis b). This is pleasantly simple, but it is also of interest to identify the (more general) principle from the representation theory of algebras that lies behind that. This is important, as other signal models may not be regular or may not be represented by a polynomial algebra in one variable, but the algebraic principle may still apply.

We focus on the decomposition property of $p(x)$ and be brief, assuming some familiarity with representation theory. The key concept underlying Theorem 2 is *induction* as implicit in step (21). Namely, $r(x)$ generates a subalgebra $\mathcal{B} = \langle r(x) \rangle \le \mathcal{A}$, which is equal (setting $y = r(x)$) to $\mathbb{C}[y]/q(y)$. Further, $d = (r_0, \ldots, r_{m-1})$ is a transversal of $\mathcal{B}$ in $\mathcal{A}$, which means $\mathcal{A}$ is a direct sum of the vector spaces $r_i\mathcal{B}$:

$$\mathcal{A} = r_0\mathcal{B} \oplus \ldots \oplus r_{m-1}\mathcal{B}. \tag{28}$$

This shows that the regular $\mathcal{A}$-module is an induction of the regular $\mathcal{B}$-module with transversal d: $\mathcal{A} = \mathcal{B} \uparrow_d \mathcal{A}$. The natural basis of this induction is b' in (25), which reflects the structure of (28). The purpose of step (21) is to make this induction explicit, and Theorem 2 is a decomposition theorem for inductions of (regular modules of) polynomial algebras.

This is a satisfying insight since it connects to related work on group FFTs that uses, among other techniques, the same principle for algorithm derivation (e.g., [22], [34], [35], [36]; compare Theorem 2 to [36, Th. 2 in the appendix]). In our own prior work [36], [37] we extended the group techniques to automatically find algorithms for a given transform. Application to the DCTs and DSTs yielded in some cases *orthogonal* algorithms that are different from the ones in this paper.

Further, we have used already a different generalization of Theorem 2, namely to polynomial algebras in *two* variables (which provide two-dimensional signal models) to derive a Cooley-Tukey type algorithm in [38] for the discrete triangle transform introduced in [39].

V. Cooley-Tukey Type DTT Algorithms (Factorization)

In this section we derive recursive DTT algorithms by applying Theorem 1, i.e., by factorizing the polynomial p in the module $\mathbb{C}[x]/p(x)$ associated to a given DTT. To do so, we will use the following *rational* factorizations of Chebyshev polynomials.

Lemma 2 The following factorizations hold for the Chebyshev polynomials T, U, V, W:

i) $T_3 = x(4x^2 - 3)$

ii) $U_{2n-1} = 2U_{n-1}T_n$.
iii) $U_{2n} = V_n W_n$.
iv) $V_{3n+1} = 2V_n(T_{2n+1} - 1/2)$.
v) $W_{3n+1} = 2W_n(T_{2n+1} + 1/2)$.

Proof: Follows from the closed form of the polynomials given in Table XVI and trigonometric identities. ■

The factorizations in Lemma 2 give rise to size 3 algorithms for DTTs in the T-group and recursive algorithms for DTTs in the U-, V-, and W-groups. The derivation is in each case straightforward using Theorem 1, hence we give only one detailed example.

***T*-Group DTT algorithms for Size 3.** We derive algorithms based on Lemma 2, i), i.e., for DTTs in the T-group (DTTs of type 3 and 4) of size 3. As an example, we consider a DCT-4$_3$. We start with the polynomial version $\overline{\text{DCT-4}}_3$, which is a polynomial transform for $\mathbb{C}[x]/T_3$ with V-basis $(V_0, V_1, V_2) = (1, 2x-1, 4x^2-2x-1)$. The zeros of T_3 are $(\sqrt{3}/2, 0, -\sqrt{3}/2)$. The factorization $T_3 = x(4x^2-3)$ yields the stepwise decomposition

$$\begin{aligned}
& \mathbb{C}[x]/T_3 \\
\to\ & \mathbb{C}[x]/x \oplus \mathbb{C}[x]/(x^2 - \tfrac{3}{4}) && (29)\\
\to\ & \mathbb{C}[x]/x \oplus \big(\mathbb{C}[x]/(x - \tfrac{\sqrt{3}}{2}) \oplus \mathbb{C}[x]/(x + \tfrac{\sqrt{3}}{2})\big) && (30)\\
\to\ & \mathbb{C}[x]/(x - \tfrac{\sqrt{3}}{2}) \oplus \mathbb{C}[x]/x \oplus \mathbb{C}[x]/(x + \tfrac{\sqrt{3}}{2}). && (31)
\end{aligned}$$

We start with the partial decomposition in (29) and choose in all three algebras a V-basis. The base change matrix B is computed by mapping (V_0, V_1, V_2) and expressing it in the basis of the direct sum in (29). The coordinate vectors are the columns of B. The first column is $(1,1,0)^T$. Because of $V_1 = 2x - 1 \equiv -1 \bmod x$, the second column is $(-1,0,1)^T$. The last column is obtained from $V_2 = 4x^2 - 2x - 1 \equiv -1 \bmod x$ and $4x^2 - 2x - 1 \equiv -2x + 2 = -V_1 + V_0 \bmod 4x^2 - 3$ as $(-1,1,-1)^T$. Step (30) requires polynomial transforms for $\mathbb{C}[x]/x$ and $\mathbb{C}[x]/(x^2 - 3/4)$ with V-bases, which are given by

$$[1] \quad \text{and} \quad \begin{bmatrix} V_0(\frac{\sqrt{3}}{2}) & V_1(\frac{\sqrt{3}}{2}) \\ V_0(-\frac{\sqrt{3}}{2}) & V_1(-\frac{\sqrt{3}}{2}) \end{bmatrix} = \begin{bmatrix} 1 & \sqrt{3}-1 \\ 1 & -\sqrt{3}-1 \end{bmatrix},$$

respectively. Finally, we have to exchange the first two spectral components in (31). The result is

$$\overline{\text{DCT-4}}_3 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \sqrt{3}-1 \\ 0 & 1 & -\sqrt{3}-1 \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 \\ 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}.$$

The corresponding algorithm for DCT-4$_3$ is obtained by scaling from the left with the diagonal matrix $\text{diag}(\cos(\pi/12), \cos(3\pi/12), \cos(5\pi/12))$ to get

$$\text{DCT-4}_3 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & \cos\frac{\pi}{12} & \frac{1}{\sqrt{2}} \\ 0 & \cos\frac{5\pi}{12} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 \\ 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}.$$

Similarly, we get algorithms for the other DTTs of size 3 in the T-group. Those, which are among the best known ones, are collected in Table X in Section VI-A.

***U*-Group DTT algorithms.** The factorizations in Lemma 2, ii) and iii), yield a complete set of recursive algorithms for DTTs in the U-group. We derived these already in [11] and restate them in Table VII(a) and (b) for completeness.

The algorithms in Table VII(a) and (b) appeared first in the literature (to our best knowledge) in [40], [41], [16], and [42], respectively. Combining Table VII(a) with the many ways of translating DTTs into each other given by duality or base change (see Appendix III) gives a large number of different recursions, many of them, however, with suboptimal arithmetic cost. Examples include [43], [44], [45].

One application of Table VII(b) is in obtaining DTT algorithms for small sizes, where the smaller DTTs of type 5–8 are base cases. As a simple example, we get

$$\begin{aligned}
& \text{DCT-2}_3 \\
=\ & \widehat{L}^3_2(\text{DCT-6}_2 \oplus \text{DCT-8}_1)B_3 \\
=\ & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 1 \\ \frac{1}{2} & -1 \end{bmatrix} \oplus \frac{\sqrt{3}}{2} I_1 \right) \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & -1 \end{bmatrix}. \quad (32)
\end{aligned}$$

Transposition yields a DCT-3$_3$ algorithm, equivalent to the one obtainable from Lemma 2(i).

***V*- and *W*-Group DTT algorithms.** The factorizations in Lemma 2, iv) and v), yield recursive algorithms for the DTTs in the V- and W-group, i.e., for all DTTs of type 5–8. Since the second factor in these factorizations is $T_{2n+1} \pm 1/2 = T_{2n+1} \pm \cos(\pi/3)$, the skew DTTs (see Section II-A) come into play. The resulting algorithms are shown in Table VII(c) and (d) and are novel.

One may ask what is the "natural" size of the DTTs of types 5–8, i.e., the sizes that yield the best decomposition. In each case, the occurring skew DTT is of odd size $2m+1$. Hence, it can be decomposed best using the algorithms shown later in Table XI if $2m+1$ is a 3-power. This implies that $3m+2$ and $3m+1$ are of the forms $(3^k+1)/2$ and $(3^k-1)/2$, respectively.

Polynomial DTTs. Every DTT in Table VII is decomposed into two DTTs that have the same associated basis b. Thus they have the same scaling function (see Table II: b and f are connected), which is the reason why we see no scaling factors in the equations. As an important consequence, we get algorithms corresponding to Table VII for the polynomial transforms $\overline{\text{DTT}}$.

As an example, we derive the polynomial equivalent of (32):

$$\overline{\text{DCT-2}}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 1 \\ 1 & -2 \end{bmatrix} \oplus I_1 \right) \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & -1 \end{bmatrix},$$

where $\text{DCT-2}_3 = \text{diag}(1, \frac{\sqrt{3}}{2}, \frac{1}{2}) \cdot \overline{\text{DCT-2}}_3$. The algorithm requires 4 additions and 1 multiplication and is thus 1 multiplication cheaper than its non-polynomial equivalent (32).

Final remarks. The algorithms given in this section are based on Lemma 2, which provides factorizations of the Chebyshev polynomials T, U, V, W. Since all these polynomial factorizations are rational, the associated matrix factorizations are also rational (the occuring transforms may of course still contain irrational entries). In Lemma 2, ii) and iii), the factors are again Chebyshev polynomials, and thus the smaller transforms in the decomposition are again DTTs.

TABLE VII

DTT ALGORITHMS BASED ON FACTORIZATION PROPERTIES OF THE CHEBYSHEV POLYNOMIALS. TRANSPOSITION YIELDS A DIFFERENT SET OF ALGORITHMS. REPLACING EACH TRANSFORM BY ITS POLYNOMIAL COUNTERPART YIELDS ALGORITHMS FOR THE POLYNOMIAL DTTS. THE BASE CASES ARE GIVEN IN TABLE IX. THE OCCURING MATRICES ARE DEFINED IN TABLE VIII.

(a) U-group: Based on $U_{2n-1} = 2U_{n-1}T_n$

$$\begin{aligned}
\text{DCT-1}_{2m+1} &= \widehat{L}^{2m+1}_{m+1}(\text{DCT-1}_{m+1} \oplus \text{DCT-3}_m)B_{2m+1} \\
\text{DST-1}_{2m-1} &= \widehat{L}^{2m-1}_{m}(\text{DST-3}_m \oplus \text{DST-1}_{m-1})B_{2m-1} \\
\text{DCT-2}_{2m} &= L^{2m}_{m}(\text{DCT-2}_m \oplus \text{DCT-4}_m)B_{2m} \\
\text{DST-2}_{2m} &= L^{2m}_{m}(\text{DST-4}_m \oplus \text{DST-2}_m)B_{2m}
\end{aligned}$$

(b) U-group: Based on $U_{2n} = V_nW_n$

$$\begin{aligned}
\text{DCT-1}_{2m} &= L^{2m}_{m}(\text{DCT-5}_m \oplus \text{DCT-7}_m)B_{2m} \\
\text{DST-1}_{2m} &= L^{2m}_{m}(\text{DST-7}_m \oplus \text{DST-5}_m)B_{2m} \\
\text{DCT-2}_{2m+1} &= \widehat{L}^{2m+1}_{m+1}(\text{DCT-6}_{m+1} \oplus \text{DCT-8}_m)B_{2m+1} \\
\text{DST-2}_{2m+1} &= \widehat{L}^{2m+1}_{m+1}(\text{DST-8}_{m+1} \oplus \text{DST-6}_m)B_{2m+1}
\end{aligned}$$

(c) V-group: Based on $V_{3n+1} = 2(T_{2n+1} - 1/2)V_n$

$$\begin{aligned}
\text{DCT-7}_{3m+2} &= P^{3m+2}_{m}(\text{DCT-3}_{2m+1}(\tfrac{1}{3}) \oplus \text{DCT-7}_{m+1})B^{(C7)}_{3m+2} \\
\text{DST-7}_{3m+1} &= \widehat{P}^{3m+1}_{m}(\text{DST-3}_{2m+1}(\tfrac{1}{3}) \oplus \text{DST-7}_m)B^{(S7)}_{3m+1} \\
\text{DCT-8}_{3m+1} &= \widehat{P}^{3m+1}_{m}(\text{DCT-4}_{2m+1}(\tfrac{1}{3}) \oplus \text{DCT-8}_m)B^{(C8)}_{3m+1} \\
\text{DST-8}_{3m+2} &= P^{3m+2}_{m}(\text{DST-4}_{2m+1}(\tfrac{1}{3}) \oplus \text{DST-8}_{m+1})B^{(S8)}_{3m+2}
\end{aligned}$$

(d) W-group: Based on $W_{3n+1} = 2W_n(T_{2n+1} + 1/2)$

$$\begin{aligned}
\text{DCT-5}_{3m+2} &= Q^{3m+2}_{m}(\text{DCT-5}_{m+1} \oplus \text{DCT-3}_{2m+1}(\tfrac{2}{3}))B^{(C5)}_{3m+2} \\
\text{DST-5}_{3m+1} &= \widehat{Q}^{3m+1}_{m}(\text{DST-5}_m \oplus \text{DST-3}_{2m+1}(\tfrac{2}{3}))B^{(S5)}_{3m+1} \\
\text{DCT-6}_{3m+2} &= Q^{3m+2}_{m}(\text{DCT-6}_{m+1} \oplus \text{DCT-4}_{2m+1}(\tfrac{2}{3}))B^{(C6)}_{3m+2} \\
\text{DST-6}_{3m+1} &= \widehat{Q}^{3m+1}_{m}(\text{DST-6}_m \oplus \text{DST-4}_{2m+1}(\tfrac{2}{3}))B^{(S6)}_{3m+1}
\end{aligned}$$

TABLE VIII

MATRICES USED IN THE ALGORITHMS IN TABLE VII.

(a) Base change matrices in Table VII(a) and (b)

$$B_{2m} = \begin{bmatrix} I_m & J_m \\ I_m & -J_m \end{bmatrix} = (\text{DFT}_2 \otimes I_m)(I_m \oplus J_m), \qquad B_{2m+1} = \begin{bmatrix} I_m & 0 & J_m \\ 0 & 1 & 0 \\ I_m & 0 & -J_m \end{bmatrix}$$

(b) Base change matrices in Table VII(c); from left to right: $B^{(C7)}_{3m+2}$, $B^{(S7)}_{3m+1}$, $B^{(C8)}_{3m+1}$, $B^{(S8)}_{3m+2}$.

$$\left[\begin{array}{ccc|ccc} & I_{2m+1} & & 1/2 & & I_m \\ & & & & & -J_m \\ \hline 1 & & & -1 & & \\ & I_m & -J_m & & & -I_m \end{array}\right], \quad \left[\begin{array}{ccc|ccc} & I_{2m+1} & & I_m \\ & & & J_m \\ & & & 0 & \cdots & 0 \\ \hline I_m & J_m & \begin{matrix}0\\ \vdots \\ 0\end{matrix} & -I_m \end{array}\right], \quad \left[\begin{array}{ccc|ccc} & I_{2m+1} & & I_m \\ & & & 0 & \cdots & 0 \\ & & & -J_m \\ \hline I_m & \begin{matrix}0\\ \vdots \\ 0\end{matrix} & -J_m & -I_m \end{array}\right], \quad \left[\begin{array}{ccc|ccc} & I_{2m+1} & & I_m & & \\ & & & & & 2 \\ & & & J_m & & \\ \hline I_m & & J_m & -I_m & & \\ & 1 & & & & -1 \end{array}\right]$$

(c) Base change matrices in Table VII(d); from left to right: $B^{(C5)}_{3m+2}$, $B^{(S5)}_{3m+1}$, $B^{(C6)}_{3m+2}$, $B^{(S6)}_{3m+1}$.

$$\left[\begin{array}{ccc|ccc} 1 & & & 1 & & \\ & I_m & J_m & & & I_m \\ \hline & & & -1/2 & & \\ & I_{2m+1} & & & & -I_m \\ & & & & & -J_m \end{array}\right], \quad \left[\begin{array}{ccc|ccc} I_m & & -J_m & \begin{matrix}0\\ \vdots \\ 0\end{matrix} & & I_m \\ \hline & I_{2m+1} & & & & -I_m \\ & & & & & J_m \\ & & & 0 & \cdots & 0 \end{array}\right], \quad \left[\begin{array}{ccc|ccc} I_m & & J_m & I_m & & \\ & 1 & & & & 1 \\ \hline & I_{2m+1} & & -I_m & & \\ & & & & -2 & \\ & & & -J_m & & \end{array}\right], \quad \left[\begin{array}{ccc|ccc} & & & 0 & & \\ I_m & \vdots & -J_m & & & I_m \\ & 0 & & & & \\ \hline & I_{2m+1} & & & & -I_m \\ & & & 0 & \cdots & 0 \\ & & & & & J_m \end{array}\right]$$

(d) Permutations in Table VII(c) and (d); note that $\hat{P}$ and $\hat{Q}$ are implicitly defined by dropping a fixpoint from P and Q, respectively.

$$\begin{aligned}
P^{3m+2}_{m} &= i_1 + 3i_2 \mapsto \begin{cases} 2i_2, & \text{for } i_1 = 0; \\ i_2 + 2m + 1, & \text{for } i_1 = 1; \\ 2i_2 + 1, & \text{for } i_1 = 2; \end{cases} \\
&= \widehat{L}^{3m+2}_{m+1} \begin{bmatrix} I_{m+1} & & \\ & & I_{m+1} \\ & I_m & \end{bmatrix} (\widehat{L}^{2m+1}_{2} \oplus I_{m+1}) \\
&= \widehat{P}^{3m+1}_{m} \oplus I_1
\end{aligned}$$

$$\begin{aligned}
Q^{3m+2}_{m} &= i_1 + 3i_2 \mapsto \begin{cases} i_2, & \text{for } i_1 = 0; \\ 2i_2 + m + 1, & \text{for } i_1 = 1; \\ 2i_2 + m + 2, & \text{for } i_2 = 2. \end{cases} \\
&= \widehat{L}^{3m+2}_{m+1} \begin{bmatrix} & & I_{m+1} \\ I_{2m+1} & & \end{bmatrix} (\widehat{L}^{2m+1}_{2} \oplus I_{m+1}) \\
&= I_1 \oplus \widehat{Q}^{3m+1}_{m}
\end{aligned}$$

TABLE IX

BASE CASES FOR THE ALGORITHMS IN TABLE VII. THE T-GROUP DTT BASE CASES ARE PROVIDED IN TABLE X.

$\text{DCT-1}_2 = F_2$	$\overline{\text{DCT-1}}_2 = \text{DCT-1}_2$
$\text{DST-1}_1 = I_1$	$\overline{\text{DST-1}}_1 = I_1$
$\text{DCT-2}_2 = \text{diag}(1, \frac{1}{\sqrt{2}}) \cdot F_2$	$\overline{\text{DCT-2}}_2 = F_2$
$\text{DST-2}_2 = \text{diag}(\frac{1}{\sqrt{2}}, 1) \cdot F_2$	$\overline{\text{DST-2}}_2 = F_2$

$\text{DCT-7}_2 = \left[\begin{smallmatrix} 1 & 1/2 \\ 1 & -1 \end{smallmatrix}\right]$	$\overline{\text{DCT-7}}_2 = \text{DCT-7}_2$
$\text{DST-7}_1 = \frac{\sqrt{3}}{2} I_1$	$\overline{\text{DST-7}}_1 = I_1$
$\text{DCT-8}_1 = \frac{\sqrt{3}}{2} I_1$	$\overline{\text{DCT-8}}_1 = I_2$
$\text{DST-8}_2 = \left[\begin{smallmatrix} 1/2 & 1 \\ 1 & -1 \end{smallmatrix}\right]$	$\overline{\text{DCT-8}}_2 = \left[\begin{smallmatrix} 1 & 2 \\ 1 & -1 \end{smallmatrix}\right]$

$\text{DCT-5}_2 = \left[\begin{smallmatrix} 1 & 1 \\ 1 & -1/2 \end{smallmatrix}\right]$	$\overline{\text{DCT-5}}_2 = \text{DCT-5}_2$
$\text{DST-5}_1 = \frac{\sqrt{3}}{2} I_1$	$\overline{\text{DST-5}}_1 = I_1$
$\text{DCT-6}_2 = \left[\begin{smallmatrix} 1 & 1 \\ 1/2 & -1 \end{smallmatrix}\right]$	$\overline{\text{DCT-6}}_2 = \left[\begin{smallmatrix} 1 & 1 \\ 1 & -2 \end{smallmatrix}\right]$
$\text{DST-6}_1 = \frac{\sqrt{3}}{2} I_1$	$\overline{\text{DCT-6}}_1 = I_1$

In Lemma 2, iv) and v), the second factor $T_{2n+1}-1/2$ leads to skew DTTs (see Table IV). The complete rational factorization of the Chebyshev polynomials T_n, U_n for arbitrary n is given in [46]. The rational factorization of V_n and W_n can be derived using [46] and Lemma 2, iii). These factorizations can be used to decompose a DTT, but the smaller transforms obtained are in general no DTTs or skew DTTs.

All algorithms in Table VII can be manipulated in numerous ways using the identities in Appendix III or transposition to obtain different algorithms.

VI. Cooley-Tukey Type DTT Algorithms (Decomposition)

In this section, we derive DTT algorithms that are based on Theorem 2, i.e., on a decomposition $p(x) = q(r(x))$ of the polynomial p in the associated algebra $\mathbb{C}[x]/p(x)$. These algorithms are structurally and in a precise mathematical sense the equivalent of the Cooley-Tukey FFT (27), which we derived based on the decomposition $x^n - 1 = (x^m)^k - 1$.

We will see that all 16 DTTs possess such algorithms, and that in many cases there are several reasonable variants with different characteristics to choose from. Some of these algorithms generalize the ones we introduced in Section V.

Each of these DTT algorithms exhibits the same flexible recursion and regular and versatile structure that has been the success of the Cooley-Tukey FFT. As a consequence, one may expect that many FFT variants optimized for, e.g., parallel or vector computation will have counterparts for the 16 DTTs. See [33], [28], [5], [6] for more details on FFT variants.

Only very few special cases of these algorithms have been found before. Again, our algebraic methods show their power: all algorithms are based on the single Theorem 2 and the derivation is comparatively easy since only base changes have to be computed. In contrast, a derivation based on matrix entries would become hopelessly complicated and does not provide a guideline on how to obtain an algorithm at all.

Decomposition of Chebyshev polynomials. The DTT algorithms are based on the following lemma.

Lemma 3 The Chebyshev polynomials T, U, V, W have the following decomposition properties:

i) $T_{km} = T_k(T_m)$; $T_{km} - a = T_k(T_m) - a$, $a \in \mathbb{C}$.
ii) $U_{km-1} = U_{m-1} \cdot U_{k-1}(T_m)$.
iii) $V_{(k-1)/2+km} = V_m \cdot V_{(k-1)/2}(T_{2m+1})$.
iv) $W_{(k-1)/2+km} = W_m \cdot W_{(k-1)/2}(T_{2m+1})$.
v) $T_{km+m/2} = T_{m/2} \cdot V_k(T_m)$.
vi) $U_{km+m/2-1} = U_{m/2-1} \cdot W_k(T_m)$.

Proof: Straightforward using the closed form of T_n from Table XVI. In particular, property i) is well-known in the literature [47]. ∎

Inspecting the identities in Lemma 3, we observe that only i) provides a pure decomposition; the other identities are a decomposition up to a factor. Thus, in these cases, the algorithm derivation requires us to first apply Theorem 1 and then Theorem 2.

Also, we observe that Lemma 3 provides decompositions of *all four types* of Chebyshev polynomials. Thus we can expect Cooley-Tukey type algorithms for all 16 DTTs. Looking at Lemma 3, Theorem 2, and its derivation in (21)–(24), we see that the algebras in (22), will always have the form

$$\mathbb{C}[x]/(T_n - \cos r\pi).$$

Thus the decomposition (23) will require skew DTTs, which motivates their introduction in [9]. Of course, this poses the question how to further decompose the skew DTTs for non-trivial sizes. This question is answered by the second identity in Lemma 3, i): $T_n - a$ decomposes exactly as T_n does, which establishes a one-to-one correspondence between algorithms for the DTTs in the T-group and their skew counterparts.

We will focus on the derivation of algorithms for T-group DTTs, and, due to space limitations, comment only very briefly on the others. Additional details are in [19].

A. T-Group DTT Algorithms

In this section, we derive Cooley-Tukey algorithms for the four DTTs in the T-group based on the decomposition $T_n = T_k(T_m)$. These algorithms are then summarized in Table XI.

We start with a fixed DTT in the T-group with associated algebra $\mathbb{C}[x]/T_n$ and C-basis[2] $b = (C_0, \ldots, C_{n-1})$, where $C \in \{T, U, V, W\}$ depends on the chosen DTT. We assume $n = km$, and use the decomposition $T_n = T_k(T_m)$. The decomposition steps (21)–(24) of Theorem 2 take the form

$$\begin{aligned}
\mathbb{C}[x]/T_n &\to \mathbb{C}[x]/T_k(T_m) && (33)\\
&\to \bigoplus_{0\le i<k} \mathbb{C}[x]/(T_m - \cos\tfrac{i+1/2}{k}\pi) && (34)\\
&\to \bigoplus_{0\le i<k} \bigoplus_{0\le j<m} \mathbb{C}[x]/(x - \cos r_{i,j}\pi) && (35)\\
&\to \bigoplus_{0\le i<n} \mathbb{C}[x]/(x - \cos\tfrac{i+1/2}{n}\pi), && (36)
\end{aligned}$$

where the $r_{i,j}$ are determined by Lemma 1.

In the first step (33), we change bases in $\mathbb{C}[x]/T_n = \mathbb{C}[x]/T_k(T_m)$, from the given C-basis b to the basis b' given in (25). The question arises, which basis to choose in the subalgebra $\mathcal{B} = \mathbb{C}[y]/T_k$, and which common basis to choose in the "skew" algebras $\mathcal{C}_i = \mathbb{C}[x]/\big(T_m - \frac{\cos(i+1/2)\pi}{k}\big)$. In the latter ones, we always choose the same C-basis as in the original algebra. For the subalgebra, it turns out that we have two reasonable choices: a T-basis or a U-basis. We consider both cases, starting with the U-basis.

***U*-basis.** We choose, independently of C, a U-basis in $\mathbb{C}[y]/T_k$. Note, that this makes the corresponding DTT a DST-3$_m$ (see Table II). The basis b' in (25) is then given by

$$\begin{aligned}
b' &= (C_0U_0(T_m), \ldots, C_{m-1}U_0(T_m),\\
&\quad \ldots\\
&\quad C_0U_{k-1}(T_m), \ldots, C_{m-1}U_{k-1}(T_m))\\
&= (C_jU_i(T_m) \mid 0 \le i < k,\ 0 \le j < m).
\end{aligned} \qquad (37)$$

We order double indices always lexicographically $(i, j) = (0, 0), (0, 1), \ldots$.

[2] C-basis does not mean that the basefield is $\mathbb{C}$ but that it consists of Chebyshev polynomials.

We denote the corresponding base change matrix $b \to b'$ in (33) with $\overline{B}^{(*)}_{k,m}$. Here, and in the following, the $* \in \{C3, S3, C4, S4\}$ in the superscript means that the matrix depends on the DTT that is decomposed.

We show DTT = DCT-3 as an example. The matrix $\overline{B}^{(*)}_{k,m} = \overline{B}^{(C3)}_{k,m}$ in (40) performs in this case in $\mathbb{C}[x]/T_n$ a base change from the T-basis to the basis b' in (37) with $C = T$. To compute $\overline{B}^{(C3)}_{k,m}$ we have to express every element T_i in b as a linear combination of elements in b'. To do this, we first write b as

$$b = (T_{im+j} \mid 0 \le i < k,\ 0 \le j < m).$$

We did not change b, but only decomposed the index into a radix-m representation. The basis b' is a special case of (37):

$$b' = (T_j U_i(T_m) \mid 0 \le i < k,\ 0 \le j < m).$$

First, we consider the case $j = 0$. From Table XVII, we know that $T_i = (U_i - U_{i-2})/2$ and thus

$$T_{im} = T_i(T_m) = \tfrac{1}{2} U_i(T_m) - \tfrac{1}{2} U_{i-2}(T_m) \tag{38}$$

is the desired representation in b'.

Now, let $j \neq 0$, i.e., $1 \le j < m$. We claim that

$$T_{im+j} = T_j U_i(T_m) - T_{m-j} U_{i-1}(T_m). \tag{39}$$

To prove it, we define the recursion

$$\begin{aligned} p_0 &= T_{j-m} = T_{m-j}, \\ p_1 &= T_j, \\ p_{i+1} &= 2T_m p_i - p_{i-1}. \end{aligned}$$

First, because of (53) in Appendix III we see that

$$p_{i+1} = T_{im+j},$$

which is the left hand side of (39). On the other hand, using (52) in Appendix II with T_m playing the role of x in (52) shows that p_{i+1} is also the right hand side of (39), as desired.

The equations (38) and (39) define the columns of the base change matrix, which is thus given by

$$\overline{B}^{(C3)}_{k,m} = \begin{bmatrix} 1 & & & -\frac{1}{2} & & & \\ & I_{m-1} & -J_{m-1} & & \ddots & & \\ & & \frac{1}{2} & & \ddots & -\frac{1}{2} & \\ & & & \ddots & -J_{m-1} & & \\ & & & & \frac{1}{2} & & \\ & & & & & I_{m-1} & -J_{m-1} \\ & & & & & & \frac{1}{2} \\ & & & & & & I_{m-1} \end{bmatrix}$$

For example, all rows with an index that is a multiple of m are determined by (38) and thus contain the numbers $1/2$.

Using

$$C_{im+j} = C_j U_i(T_m) - C_{j-m} U_{i-1}(T_m),$$

which generalizes (39), yields the base change matrices in the other three cases $* \in \{C4, S3, S4\}$.

After the base change, the decomposition follows steps (34)–(36) and Theorem 2. The coarse decomposition in step (34) is obtained with the matrix $\overline{\text{DST-3}}_k \otimes I_m$, since Theorem 2 requires us to choose a polynomial transform for the coarse decomposition. For step (35), we need a direct sum of skew DTTs: $I_k \otimes_i \text{DTT}_m(\frac{i+1/2}{k})$. These are of the same type as the DTT we started with, since they have the same C-basis as the DTT to be decomposed.

Finally, we order the one-dimensional summands in step (36) using a permutation. This permutation does not depend on the basis, but only on the zeros of T_k and T_m. Thus it is the same in all four cases of DTTs in the T-group, and, using Lemma 1, it takes the form

$$K^n_m = (I_k \oplus J_k \oplus I_k \oplus J_k \oplus \dots) L^n_m.$$

This permutation is the T-group DTT equivalent of the stride permutation L^n_m in the Cooley-Tukey FFT.

In summary, we obtain

$$\text{DTT}_n = K^n_m \big(I_k \otimes_i \text{DTT}_m(\tfrac{i+1/2}{k})\big)(\overline{\text{DST-3}}_k \otimes I_m)\overline{B}^{(*)}_{k,m}. \tag{40}$$

The question that remains is how to decompose the smaller transforms: the skew DTT_m's and the polynomial $\overline{\text{DST-3}}_k$. However, this poses no problem. Since for any $a \in \mathbb{C}$, $T_n - a$ decomposes exactly as T_n, we derive in a completely analogous way the "skew version" of (40) as

$$\text{DTT}_n(r) = K^n_m \big(I_k \otimes_i \text{DTT}_m(r_i)\big)\big(\overline{\text{DST-3}}_k(r) \otimes I_m\big)\overline{B}^{(*)}_{k,m}, \tag{41}$$

which generalizes (40); namely, (40) is (41) for $r = 1/2$. The numbers r_i are computed from r using Lemma 1. The matrix K^n_m neither depends on the type of DTT, nor on r; the matrix $\overline{B}^{(*)}_{k,m}$ does depend on the type of DTT, but not on r, since the bases b and b' are independent of r.

Further, since DTTs and skew DTTs have the same scaling function (Tables II and IV), we obtain corresponding algorithms for the polynomial version of the transforms by just replacing each DTT by its polynomial counterpart:

$$\overline{\text{DTT}}_n(r) = K^n_m \big(I_k \otimes_i \overline{\text{DTT}}_m(r_i)\big)\big(\overline{\text{DST-3}}_k(r) \otimes I_m\big)\overline{B}^{(*)}_{k,m}.$$

All details of the above algorithms are in Table XI.

Next, we derive the analogue of the above algorithms, if a T-basis, instead of a U-basis is chosen in the subalgebra $\mathbb{C}[x]/T_k$.

***T*-basis.** In distinction to the above, we choose this time, independently of C, a T-basis in $\mathbb{C}[y]/T_k$. Thus, the corresponding DTT is a DCT-3_m. The basis b' in (25) is now given by

$$\begin{aligned} b' &= (C_0 T_0(T_m), \dots, C_{m-1} T_0(T_m), \\ &\quad \dots \\ &\quad C_0 T_{k-1}(T_m), \dots, C_{m-1} T_{k-1}(T_m)) \\ &= (C_{j-im}/2 + C_{j+im}/2 \mid 0 \le i < k, 0 \le j < m), \end{aligned} \tag{42}$$

using (53) in Appendix II. We denote the base change matrix for $b \to b'$ by $B^{(*)}_{k,m}$. We omit the derivation, which is similar to the U-basis case above. Details are in [19].

The coarse decomposition in step (34) is now performed by the matrix $\text{DCT-3}_k \otimes I_m$ (note that DCT-3 is a polynomial transform). The remaining steps (35) and (36) are equal to what we had before.

As a result, we obtain

$$\mathrm{DTT}_n = K^n_m\big(I_k \otimes_i \mathrm{DTT}_m(\tfrac{i+1/2}{k})\big)(\text{DCT-3}_k \otimes I_m)B^{(*)}_{k,m}, \tag{43}$$

and its generalization to the skew DTTs

$$\mathrm{DTT}_n(r) = K^n_m\big(I_k \otimes_i \mathrm{DTT}_m(r_i)\big)\big(\text{DCT-3}_k(r) \otimes I_m\big)B^{(*)}_{k,m}. \tag{44}$$

Again, $B^{(*)}_{k,m}$ only depends on the type of DTT, and not on r.

The polynomial version is again given by simply replacing all DTTs by their polynomial counterparts:

$$\overline{\mathrm{DTT}}_n(r) = K^n_m\big(I_k \otimes_i \overline{\mathrm{DTT}}_m(r_i)\big)\big(\text{DCT-3}_k(r) \otimes I_m\big)B^{(*)}_{k,m}.$$

We mentioned above that choosing a U-basis in the subalgebra $\mathbb{C}[x]/T_k$ leads to base change matrices $\overline{B}_{k,m}$ that are sparse. For the T-basis, this is somewhat different. In fact, inspecting (42) shows that the inverse base change $b' \to b$, i.e., $B^{-1}_{k,m}$ is sparse (with at most two entries in each column). For this reason, we will also consider the inverse of (43) and (44). The sparsity of $B_{k,m}$ depends on k; the best case is $k = 2$ and the only one we consider in this paper.

All the details are in Table XI.

***T*-basis inverted.** To express the inverse, we need the inverse skew DTTs (Appendix III). The inverse of (44) will take, after minor simplifications, in each case the general form

$$\mathrm{iDTT}_n(r) = C^{(*)}_{k,m}(\text{iDCT-3}_k(r) \otimes I_m) \big(I_k \otimes_i \mathrm{iDTT}_m(r_i)\big)M^n_k, \tag{45}$$

where $M^n_k = (K^n_m)^{-1} = L^n_k(I_k \oplus J_k \oplus I_k \oplus J_k \oplus \dots)$, and $C^{(*)}_{k,m}$ is closely related to $(B^{(*)}_{k,m})^{-1}$. (45) provides algorithms for the DTTs of type 2 and 4 (the inverses of the DTTs in the T-group). See Table XI for the exact form.

Variants. The algorithms derived above can be further manipulated to obtain variants. We saw already an example: the inversion of (44) to obtain (45). One obvious manipulation is transposition, which turns each T-group DTT algorithm into an algorithm for a DCT or DST of type 2 or 4 (the transposes of the T-group DTTs).

More interestingly, each of the above algorithms has a corresponding "twiddle version," which is obtained by translating skew DTTs into their non-skew counterparts using (56) in Appendix III. For example, the twiddle version of (43) is given by

$$\mathrm{DTT}_n = K^n_m(I_k \otimes \mathrm{DTT}_m)D_{k,m}(\text{DCT-3}_k \otimes I_m)B_{k,m}, \tag{46}$$

where $D_{k,m} = I_k \otimes_i X^{(*)}_m(\frac{i+1/2}{k})$ is a direct sum of the x-shaped matrices in (56) (Appendix III).

The twiddle version seems more appealing; however, we will later see that at least in the 2-power case $n = 2^k$ they incur a higher arithmetic cost. The reason is that skew and non-skew DTTs can be computed with the same cost in this case. For other sizes, the twiddle version may not incur any penalty. Most state of the art software implementations [2], [3] fuse the twiddle factors with the subsequent loop incurred by the tensor product anyway to achieve better locality.

Base cases. We provide the base cases for the above algorithms for size $n = 2,\ 3$ in Table X. The size 2 cases follow from the definition; most of the size 3 cases are derived as explained in in Section V.

TABLE X
BASE CASES FOR NORMAL AND SKEW T-GROUP DTTS OF SIZE 2 AND 3.

$\overline{\text{DCT-3}}_2 = F_2 \cdot \mathrm{diag}(1, 1/\sqrt{2})$
$\overline{\text{DST-3}}_2 = F_2 \cdot \mathrm{diag}(1, \sqrt{2})$
$\overline{\text{DCT-4}}_2 = F_2 \cdot \begin{bmatrix} 1 & -1 \\ 0 & \sqrt{2} \end{bmatrix}$
$\overline{\text{DST-4}}_2 = F_2 \cdot \begin{bmatrix} 1 & 1 \\ 0 & \sqrt{2} \end{bmatrix}$
$\text{DCT-3}_2 = \overline{\text{DCT-3}}_2$
$\text{DST-3}_2 = F_2 \cdot \mathrm{diag}(1/\sqrt{2}, 1)$
$\text{DCT-4}_2 = \mathrm{diag}(\cos\frac{\pi}{8}, \sin\frac{\pi}{8}) \cdot F_2 \cdot \begin{bmatrix} 1 & -1 \\ 0 & \sqrt{2} \end{bmatrix}$
$\text{DST-4}_2 = \mathrm{diag}(\sin\frac{\pi}{8}, \cos\frac{\pi}{8}) \cdot F_2 \cdot \begin{bmatrix} 1 & 1 \\ 0 & \sqrt{2} \end{bmatrix}$
$\overline{\text{DCT-3}}_2(r) = F_2 \cdot \mathrm{diag}(1, \cos\frac{r}{2}\pi)$
$\overline{\text{DST-3}}_2(r) = F_2 \cdot \mathrm{diag}(1, 2\cos\frac{r\pi}{2})$
$\overline{\text{DCT-4}}_2(r) = F_2 \cdot \begin{bmatrix} 1 & -1 \\ 0 & 2\cos\frac{r\pi}{2} \end{bmatrix}$
$\overline{\text{DST-4}}_2(r) = F_2 \cdot \begin{bmatrix} 1 & 1 \\ 0 & 2\cos\frac{r\pi}{2} \end{bmatrix}$
$\text{DCT-3}_2(r) = \overline{\text{DCT-3}}_2(r)$
$\text{DST-3}_2(r) = F_2 \cdot \mathrm{diag}(\sin\frac{r\pi}{2}, \sin r\pi)$
$\text{DCT-4}_2(r) = \mathrm{diag}(\cos\frac{r\pi}{4}, \sin\frac{r\pi}{4}) \cdot F_2 \cdot \begin{bmatrix} 1 & -1 \\ 0 & 2\cos\frac{r\pi}{2} \end{bmatrix}$
$\text{DST-4}_2(r) = \mathrm{diag}(\sin\frac{r\pi}{4}, \cos\frac{r\pi}{4}) \cdot F_2 \cdot \begin{bmatrix} 1 & 1 \\ 0 & 2\cos\frac{r\pi}{2} \end{bmatrix}$
$\text{iDCT-3}_2(r) = \mathrm{diag}(1, \frac{1}{2\cos\frac{r}{2}\pi}) \cdot F_2,$
$\text{iDST-3}_2(r) = \mathrm{diag}(\frac{1}{2\sin\frac{r\pi}{2}}, \frac{1}{\sin r\pi}) \cdot F_2$
$\text{iDCT-4}_2(r) = \begin{bmatrix} 1 & 1 \\ 0 & \frac{1}{2\cos\frac{r\pi}{2}} \end{bmatrix} \cdot F_2 \cdot \mathrm{diag}(\frac{1}{2\cos\frac{r\pi}{4}}, \frac{1}{2\sin\frac{r\pi}{4}})$
$\text{iDCT-4}_2(r) = \begin{bmatrix} 1 & -1 \\ 0 & \frac{1}{2\cos\frac{r\pi}{2}} \end{bmatrix} \cdot F_2 \cdot \mathrm{diag}(\frac{1}{2\sin\frac{r\pi}{4}}, \frac{1}{2\cos\frac{r\pi}{4}})$
$\overline{\text{DCT-3}}_3 = \begin{bmatrix} 1&0&1 \\ 0&1&0 \\ 1&0&-1 \end{bmatrix} \begin{bmatrix} 1&0&1/2 \\ 1&0&-1 \\ 0&\sqrt{3}/2&0 \end{bmatrix}$
$\overline{\text{DST-3}}_3 = \begin{bmatrix} 0&1&1 \\ 1&0&0 \\ 0&1&-1 \end{bmatrix} \begin{bmatrix} 1&0&-1 \\ 1&0&2 \\ 0&\sqrt{3}&0 \end{bmatrix}$
$\overline{\text{DCT-4}}_3 = \begin{bmatrix} 0&1&\sqrt{3}-1 \\ 1&0&0 \\ 0&1&-\sqrt{3}-1 \end{bmatrix} \begin{bmatrix} 1&-1&-1 \\ 1&0&1 \\ 0&1&-1 \end{bmatrix}$
$\overline{\text{DST-4}}_3 = \begin{bmatrix} 0&1&\sqrt{3}+1 \\ 1&0&0 \\ 0&1&-\sqrt{3}+1 \end{bmatrix} \begin{bmatrix} 1&1&-1 \\ 1&0&1 \\ 0&1&1 \end{bmatrix}$
$\text{DCT-3}_3 = \overline{\text{DCT-3}}_3$
$\text{DST-3}_3 = \begin{bmatrix} 1&0&1 \\ 0&1&0 \\ 1&0&-1 \end{bmatrix} \begin{bmatrix} 1/2&0&1 \\ 1&0&-1 \\ 0&\sqrt{3}/2&0 \end{bmatrix}$
$\text{DCT-4}_3 = \begin{bmatrix} 1&-1&0 \\ 0&0&1 \\ 1&1&0 \end{bmatrix} \begin{bmatrix} 1&0&0 \\ 0&1&-1 \\ 0&-2&-1 \end{bmatrix} \mathrm{diag}(\sqrt{\frac{3}{2}}, \sqrt{\frac{1}{8}}, \sqrt{\frac{1}{2}}) \begin{bmatrix} 1&0&1 \\ -1&0&1 \\ 0&1&0 \end{bmatrix}$
$\text{DST-4}_3 = \begin{bmatrix} 1&-1&0 \\ 0&0&1 \\ 1&1&0 \end{bmatrix} \begin{bmatrix} 1&0&0 \\ 0&1&-1 \\ 0&2&1 \end{bmatrix} \mathrm{diag}(\sqrt{\frac{3}{2}}, \sqrt{\frac{1}{8}}, \sqrt{\frac{1}{2}}) \begin{bmatrix} 1&0&1 \\ 1&0&-1 \\ 0&1&0 \end{bmatrix}$
$\overline{\text{DCT-3}}_3(r) = \begin{bmatrix} 1&1&1 \\ 1&-1&0 \\ 1&0&-1 \end{bmatrix} \left(I_1 \oplus \begin{bmatrix} \cos(\frac{1+r}{3}\pi) & \cos(\frac{1-2r}{3}\pi) \\ \cos(\frac{1-r}{3}\pi) & \cos(\frac{1+2r}{3}\pi) \end{bmatrix}\right)$
$\overline{\text{DST-3}}_3(r) = \begin{bmatrix} 1&1&1 \\ 1&-1&0 \\ 1&0&-1 \end{bmatrix} \left(I_1 \oplus 2\begin{bmatrix} \cos(\frac{1+r}{3}\pi) & \cos(\frac{1-2r}{3}\pi) \\ \cos(\frac{1-r}{3}\pi) & \cos(\frac{1+2r}{3}\pi) \end{bmatrix}\right) \begin{bmatrix} 1&0&1 \\ 0&1&0 \\ 0&0&1 \end{bmatrix}$
$\overline{\text{DCT-4}}_3(r) =$ by definition
$\overline{\text{DST-4}}_3(r) =$ by definition
$\text{DCT-3}_3(r) = \overline{\text{DCT-3}}_3(r)$
$\text{DST-3}_3(r) = \mathrm{diag}(\sin\frac{r}{3}\pi, \sin\frac{2-r}{3}\pi, \sin\frac{2+r}{3}\pi)\overline{\text{DST-3}}_3(r)$
$\text{DCT-4}_3(r) =$ by definition
$\text{DST-4}_3(r) =$ by definition

Special case. We briefly discuss the special case of (48) in Table XI for DTT = DCT-3 and $k = 2$. $B^{(C3)}_{2,m}$ shonw in Table XII(c) incurs multiplications by 2, which can be fused

TABLE XI

GENERAL RADIX COOLEY-TUKEY ALGORITHMS BASED ON $T_n = T_k(T_m)$ FOR THE DTTS IN THE T-GROUP (AND THEIR TRANSPOSES). IN EACH CASE $\mathrm{DTT} \in \{\text{DCT-3}, \text{DST-3}, \text{DCT-4}, \text{DST-4}\}$. THE EXACT FORM OF THE OCCURRING MATRICES IS GIVEN IN TABLE XII.

$$U\text{-basis:}\quad \mathrm{DTT}_n = \mathrm{DTT}_n(1/2), \qquad \mathrm{DTT}_{km}(r) = K_m^n \big(I_k \otimes_i \mathrm{DTT}_m(r_i)\big) \big(\overline{\text{DST-3}}_k(r) \otimes I_m\big) \overline{B}_{k,m}^{(*)} \tag{47}$$

$$T\text{-basis:}\quad \mathrm{DTT}_n = \mathrm{DTT}_n(1/2), \qquad \mathrm{DTT}_{km}(r) = K_m^n \big(I_k \otimes_i \mathrm{DTT}_m(r_i)\big) \big(\text{DCT-3}_k(r) \otimes I_m\big) B_{k,m}^{(*)} \tag{48}$$

$$\text{Inverse of (48):}\quad \mathrm{DTT}_n^T = \mathrm{iDTT}_n(1/2), \quad \mathrm{iDTT}_{km}(r) = C_{k,m}^{(*)} \big(\text{iDCT-3}_k(r) \otimes I_m\big) \big(I_k \otimes_i \mathrm{iDTT}_m(r_i)\big) M_k^n \tag{49}$$

TABLE XII

MATRICES USED IN TABLE XI.

(a) Permutations.

$$K_m^n = (I_k \oplus J_k \oplus I_k \oplus J_k \oplus \dots) L_m^n, \quad M_k^n = (K_m^n)^{-1} = L_k^n (I_k \oplus J_k \oplus I_k \oplus J_k \oplus \dots)$$

(b) Base change matrices for (47); from left to right: $\overline{B}^{(C3)}$, $\overline{B}^{(S3)}$, $\overline{B}^{(C4)}$, $\overline{B}^{(S4)}$.

$$\begin{bmatrix} 1 & & & -\frac{1}{2} & & \\ & I_{m-1} & -J_{m-1} & & & \\ & & & \ddots & & \\ & \frac{1}{2} & & \ddots & & -\frac{1}{2} \\ & & \ddots & & -J_{m-1} & \\ & & & \frac{1}{2} & & \\ & & & & I_{m-1} & -J_{m-1} \\ & & & & \frac{1}{2} & \\ & & & & & I_{m-1} \end{bmatrix}, \begin{bmatrix} I_m & \overline{Z}_m & & & \\ & I_m & \overline{Z}_m & & \\ & & \ddots & \ddots & \\ & & & I_m & \overline{Z}_m \\ & & & & I_m \end{bmatrix}, \begin{bmatrix} I_m & -J_m & & & \\ & I_m & -J_m & & \\ & & \ddots & \ddots & \\ & & & I_m & -J_m \\ & & & & I_m \end{bmatrix}, \begin{bmatrix} I_m & J_m & & & \\ & I_m & J_m & & \\ & & \ddots & \ddots & \\ & & & I_m & J_m \\ & & & & I_m \end{bmatrix}.$$

(c) Base change matrices for (48) in the case $k = 2$.

$$B_{2,m}^{(C3)} = (I_m \oplus \mathrm{diag}(1, 2, \dots, 2)) \begin{bmatrix} I_m & -Z_m \\ & I_m \end{bmatrix}, \quad B_{2,m}^{(S3)} = \begin{bmatrix} I_m & \overline{Z}_m \\ & 2I_m \end{bmatrix}, \quad B_{2,m}^{(C4)} = \begin{bmatrix} I_m & -J_m \\ & 2I_m \end{bmatrix}, \quad B_{2,m}^{(S4)} = \begin{bmatrix} I_m & J_m \\ & 2I_m \end{bmatrix}.$$

(d) Base change matrices for (49); from left to right: $C^{(C3)}$, $C^{(S3)}$, $C^{(C4)}$, $C^{(S4)}$.

$$\begin{bmatrix} I_m & Z_m & & & \\ & I_m & Z_m & & \\ & & \ddots & \ddots & \\ & & & I_m & Z_m \\ & & & & I_m \end{bmatrix}, \begin{bmatrix} I_{m-1} & -J_{m-1} & & & & \\ & \frac{1}{2} & & & -\frac{1}{2} & \\ & & I_{m-1} & -J_{m-1} & & \ddots \\ & & & \frac{1}{2} & \ddots & & -\frac{1}{2} \\ & & & & \ddots & -J_{m-1} & \\ & & & & & \frac{1}{2} & \\ & & & & & & I_{m-1} \\ & & & & & & & 1 \end{bmatrix}, \begin{bmatrix} I_m & J_m & & & \\ & I_m & J_m & & \\ & & \ddots & \ddots & \\ & & & I_m & J_m \\ & & & & I_m \end{bmatrix}, \begin{bmatrix} I_m & -J_m & & & \\ & I_m & -J_m & & \\ & & \ddots & \ddots & \\ & & & I_m & -J_m \\ & & & & I_m \end{bmatrix}.$$

with the multiplications incurred by the adjacent $\text{DCT-3}_2(r)$. Namely, using $\text{DCT-3}_2(r) = F_2 \cdot \mathrm{diag}(1, \cos\frac{r}{2}\pi)$ (see Table X), we can manipulate (48) in Table XI to take the form

$$\text{DCT-3}_n(r) = K_m^n \big(\text{DCT-3}_m(\tfrac{r}{2}) \oplus \text{DCT-3}_m(\tfrac{2-r}{2})\big) (F_2 \otimes I_m) E_{2,m}, \tag{50}$$

where

$$E_{2,m} = \begin{bmatrix} I_m & -Z_m \\ & \cos\frac{r}{2}\pi (I_1 \oplus 2I_{m-1}) \end{bmatrix}. \tag{51}$$

B. Alternative T-Group DTT Algorithms

It is also possible to derive T-group DTT algorithms using the decomposition $T_{km+m/2} = T_{m/2} \cdot V_k(T_m)$ in Lemma 3, v). One application is in obtaining algorithms for size $5 = 2 \cdot 2 + 2/2$. For the DCT-3 this yields the algorithm in Table XIII. The cost can be read off as $(12, 6, 1)$. Transposition yields an algorithm for DCT-2_5 with identical cost, which is only slightly worse than the $(13, 5, 0)$ algorithm in [48].

C. $U/V/W$-group DTT Algorithms

Lemma 3, ii)–iv), yields general radix algorithms for the other DTTs in the $U/V/W$-groups. These are generalizations of the algorithms in Table VII. Since the most important DCT-2 is already covered by the transposes of the DCT-3 algorithms, we are very brief and give one representative example from each group in Table XIV. The occurring matrices and more details for the other DTTs can be found in [19].

VII. ANALYSIS

In this section we analyze the algorithms presented in this paper in Tables VII and XI with respect to arithmetic cost and other characteristics. We also identify the special cases that can be found in the literature.

Cost analysis. We focus the discussion on the most important cases. Table XV shows a summary of the achievable costs for all 16 DTTs including the algorithms that achieve them.

TABLE XIII

ALGORITHM FOR DCT-3_5 WITH COST (12, 6, 1). TRANSPOSITION YIELDS A DCT-2_5 ALGORITHM OF EQUAL COST.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \left(I_1 \oplus \left(F_2 \operatorname{diag}(1, \cos\tfrac{\pi}{5}) \oplus F_2 \operatorname{diag}(1, \cos\tfrac{3\pi}{5}) \right) \begin{bmatrix} I_2 & \operatorname{diag}(\cos\tfrac{\pi}{5}, 2\cos\tfrac{\pi}{5}) \\ I_2 & \operatorname{diag}(\cos\tfrac{3\pi}{5}, 2\cos\tfrac{3\pi}{5}) \end{bmatrix} \right) \begin{bmatrix} 1 & 0 & -1 & 0 & 1 \\ 1 & 0 & 1/2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

TABLE XIV

EXAMPLES OF COOLEY-TUKEY TYPE ALGORITHMS FOR DTTS IN THE $U/V/W$-GROUP, BASED ON THE RESPECTIVE DECOMPOSITIONS $U_{km-1} = U_{k-1}(T_m) \cdot U_{m-1}$, $V_{(k-1)/2+km} = V_{(k-1)/2}(T_{2m+1}) \cdot V_m$, $W_{(k-1)/2+km} = W_m \cdot W_{(k-1)/2}(T_{2m+1})$. THE POLYNOMIAL VERSIONS ARE OBTAINED BY REPLACING ALL TRANSFORMS BY THEIR POLYNOMIAL COUNTERPARTS.

$$\begin{aligned}
\text{DST-1}_{km-1} &= P^{(S1)}_{k,m} \Big(\big(I_{k-1} \otimes_i \text{DST-3}_m(\tfrac{i+1}{k}) \big) \big(\overline{\text{DST-1}}_{k-1} \otimes I_m \big) \oplus \text{DST-1}_{m-1} \Big) B^{(S1)}_{k,m} \\
\text{DST-7}_{km+(k-1)/2} &= P^{(S7)}_{k,m} \Big(\big(I_{(k-1)/2} \otimes_i \text{DST-3}_{2m+1}(\tfrac{2i+1}{k}) \big) \big(\overline{\text{DST-7}}_{\frac{k-1}{2}} \otimes I_{2m+1} \big) \oplus \text{DST-7}_m \Big) B^{(S7)}_{k,m} \\
\text{DST-5}_{km+(k-1)/2} &= P^{(S5)}_{k,m} \Big(\text{DST-5}_m \oplus \big(I_{(k-1)/2} \otimes_i \text{DST-3}_{2m+1}(\tfrac{2i+2}{k}) \big) \big(\overline{\text{DST-5}}_{\frac{k-1}{2}} \otimes I_{2m+1} \big) \Big) B^{(S5)}_{k,m}
\end{aligned}$$

For the T-group DTTs, we consider the algorithms in Table XI; (49) is considered transposed. Transposition yields 2-power algorithms of equal cost for DCT and DST of type 2.

For a 2-power n, the costs in each case are independent of the chosen recursive split strategy and are equal for the skew (and inverse skew) and non-skew versions since they have the same recursions *and* the base cases in Table X have equal cost. The polynomial DTTs save multiplications since they have cheaper base cases (except for the DCT-3 $= \overline{\text{DCT-3}}$).

For a 3-power n, the skew DTTs are more expensive than their non-skew counterparts, because the size 3 base cases have this property. Also, the stated costs for 3-powers in Table XV(a) can be further improved with the algorithms in this paper. For example, we can slightly improve a DCT-3 of a 3-power size n using first the transpose of Table VII(b) and then decompose the occuring DCTs of type 7 and 8 with Tables VII(c) and VII(d) to get a cost of

$$\begin{aligned}
&(\tfrac{8}{3} n \log_3(n) - 2n + 2, \tfrac{4}{3} n \log_3(n) - \tfrac{7}{4} n + \tfrac{1}{2} \log_3(n) + \tfrac{7}{4}, \\
&\tfrac{1}{4} n + \tfrac{1}{2} \log_3(n) - \tfrac{1}{4}) = 4n \log_3(n) - \tfrac{7}{2} n + \log_3(n) + \tfrac{7}{2}
\end{aligned}$$

while sacrificing some regularity in structure. For example, for $n = 9$, Table XV(a) yields $(32, 12, 4) = 48$ and the above $(32, 11, 3) = 46$. The same costs apply to a DCT-2 by transposing the algorithms. Reference [48] provides an $(34, 8, 2) = 44$ algorithm (proven optimal with respect to non-rational multiplications) with no obvious structure. Any of these size 9 algorithms can also be used as base case for a larger 3-power size. Using (55) and (56), the cost for a 3-power size DCT-4 can also be reduced.

For an arbitrary p-power n, we can compute T-group DTTs (and their transposes) using the twiddle versions of the T-group algorithms such as (46). For example, a DCT-2_{p^t} computed with (49) then requires, independently of the split strategy, $\frac{n}{p} \log_p(n)$ DCT-2_p's and

$$2(1 - 1/p) n \log_p(n) - 2n + 2$$

additions and multiplications, respectively. For a given DCT-2_p kernel (e.g., the transpose of Table XIII for $p = 5$ or [48] for $p = 5, 7$), it is now easy to compute a precise cost. The other T-group DTTs can be decomposed analogously.

Further comments.

- The algorithms in (48) in Table XI for $k = 2$ have the appealing feature that all multiplications occur in parallel with additions on the same operands as shown for the DCT-3 in (51). Further, their transposes are a good choice if the output is to be pruned, i.e., only, say, the first half of the output matters. This was used in [49] for the DCT-2.
- The algorithms (49) involve iDTTs and hence inverse cosines (from the base cases of the iDTTs in Table X). This may cause numerical instability.
- Transposition of the algorithms in Tables VII and XI yields algorithms for the respective transposed DTTs with equal cost. The reason for this is that all occurring matrices have this property.
- If a non-skew DTT is decomposed using any of the algorithms in Table XI, and n is odd, then (the middle) one of the occurring skew DTTs in the direct sum has $r = 1/2$, i.e., is non-skew.
- Any odd-size DCT of type 2 or 3 can be translated into an RDFT without incurring operations [48].
- Again, we note that the algorithms in this section are not all the available ones. One example are orthogonal algorithms, which are due to other algebraic principles such as [36].
- All the algorithms for DTTs of types 2–4 have a total cost of $2n \log_2(n) + O(n)$ for a 2-power size n. This can be improved by roughly 5% with the recent paper [50] to $\frac{17}{9} n \log_2(n) + O(n)$ at the expense of some regularity.

Literature. Algorithm (48) for the DCT-3 of 2-power size in the special case $k = 2$ was derived in [51] and in [29]; the latter also considered 3-powers and $k = 3$. For arbitrary p-powers (p prime) and $k = p$, the derivation is in [30]. The above references also used Chebyshev polynomials in their derivation, but they do not use the algebraic framework, and they present the algorithms in an iterative form only, which avoids the definition of skew DTTs. For software

TABLE XV

ARITHMETIC COSTS ACHIEVABLE FOR THE 16 DTTS WITH THE ALGORITHMS IN THIS PAPER.

(a) T-group DTTs of 2-power and 3-power sizes n. All the 3-power size costs can be slightly improved upon (see Section VII). For 2-power sizes n, the polynomial versions of the type 4 DTTs require n multiplications less and the polynomial DST-3 requires $n/2$ multiplications less.

Transform	Cost (adds, mults, 2-power mults)	Total cost	Achieved by
2-power n			
DCT-3$_n$	$(\frac{3}{2}n\log_2(n) - n + 1,\ \frac{1}{2}n\log_2(n),\ 0)$	$2n\log_2(n) - n + 1$	(47) = (50), (49)^T, Table VII(a)^T
DST-3$_n$	same as DCT-3$_n$	$2n\log_2(n) - n + 1$	duality (54), Table VII(a)^T
DCT-4$_n$	$(\frac{3}{2}n\log_2(n),\ \frac{1}{2}n\log_2(n) + n,\ 0)$	$2n\log_2(n) + n$	(47), (48), (49), (55), and their transposes
DST-4$_n$	same as DCT-4$_n$	$2n\log_2(n) + n$	(47), (48), (49), duality (54)
DCT-3$_n(r)$	same as DCT-3$_n$	$2n\log_2(n) - n + 1$	(48)
DST-3$_n(r)$	$(\frac{3}{2}n\log_2(n) - n + 1,\ \frac{1}{2}n\log_2(n) + \frac{1}{2}n,\ 0)$	$2n\log_2(n) - \frac{1}{2}n + 1$	(47)
DCT-4$_n(r)$	same as DCT-4$_n$	$2n\log_2(n) + n$	(47), (48)
DST-4$_n(r)$	same as DCT-4$_n$	$2n\log_2(n) + n$	(47), (48)
3-power n			
DCT-3$_n$	$(\frac{8}{3}n\log_3(n) - 2n + 2,\ \frac{4}{3}n\log_3(n) - \frac{3}{2}n,\ \frac{1}{2}n - \frac{1}{2})$	$4n\log_3(n) - 3n + 3$	(48), (49)^T, see also discussion in SectionVII
DST-3$_n$	same as DCT-3$_n$	$4n\log_3(n) - 3n + 3$	duality (54)
DCT-4$_n$	$(\frac{8}{3}n\log_3(n) - n + 1,\ \frac{4}{3}n\log_3(n) - \frac{1}{2}n,\ \frac{1}{2}n - \frac{1}{2})$	$4n\log_3(n) - n + 2$	(55)
DST-4$_n$	same as DCT-4$_n$	$4n\log_3(n) - n + 2$	duality (54)
DCT-3$_n(r)$	$(\frac{8}{3}n\log_3(n) - n + 1,\ \frac{4}{3}n\log_3(n),\ 0)$	$4n\log_3(n) - n + 1$	(48)
DST-3$_n(r)$	$(\frac{8}{3}n\log_3(n) - n + 1,\ \frac{4}{3}n\log_3(n) + \frac{1}{2}n + \frac{1}{2},\ \frac{1}{2}n - \frac{1}{2})$	$4n\log_3(n) + 1$	(56)
DCT-4$_n(r)$	$(\frac{8}{3}n\log_3(n),\ \frac{4}{3}n\log_3(n) + \frac{1}{2}n - \frac{1}{2},\ \frac{1}{2}n - \frac{1}{2})$	$4n\log_3(n) + n$	(57)
DST-4$_n(r)$	same as DCT-4$_n(r)$	$4n\log_3(n) + n$	the equivalent to (57)

(b) $U/V/W$-group DTTs. The size of DCT-1 is $n = 2^k + 1$, the size of DST-1 is $n = 2^k - 1$, the sizes of DCT-2 and DST-2 is $n = 2^k$, the size of DCT-5, DCT-6, DCT-7, DST-8 is $n = (3^k + 1)/2$, and the size of DST-5, DST-6, DST-7, DCT-8 is $n = (3^k - 1)/2$. The polynomial DTTs of type 2 are $n - 1$ multiplications cheaper.

Transform	Cost (adds, mults, 2-power mults)	Total cost	Achieved by
DCT-1$_n$	$(\frac{3}{2}n\log_2(n-1) - 2n - \frac{1}{2}\log_2(n-1) + 6,$ $\frac{1}{2}n\log_2(n-1) - n - \frac{1}{2}\log_2(n-1) + 2,\ 0)$	$2n\log_2(n-1) - 3n$ $-\log_2(n-1) + 8$	Table VII(a)
DST-1$_n$	$(\frac{3}{2}n\log_2(n+1) - 2n + \frac{5}{2}\log_2(n+1) + 2,$ $\frac{1}{2}n\log_2(n+1) - n + \frac{1}{2}\log_2(n+1),\ 0)$	$2n\log_2(n+1) - 3n$ $+3\log_2(n+1) + 2$	Table VII(a)
DCT-2$_n$	$(\frac{3}{2}n\log_2(n) - n + 1,\ \frac{1}{2}n\log_2(n),\ 0)$	$2n\log_2(n) - n + 1$	Table VII(a), (48)^T = (50)^T, (49)
DST-2$_n$	same as DCT-2	$2n\log_2(n) - n + 1$	Table VII(a), duality (54)^T
DCT-7$_n$	$(\frac{8}{3}n\log_3(2n-1) - 3n - \frac{1}{3}\log_3(2n-1) + 3,$ $\frac{4}{3}n\log_3(2n-1) - 2n - \frac{2}{3}\log_3(2n-1) + 2,\ \log_3(2n-1))$	$4n\log_3(2n-1) - 5n + 5$	Table VII(c)
DST-7$_n$	$(\frac{8}{3}n\log_3(2n+1) - 3n + \frac{1}{3}\log_3(2n+1),$ $\frac{4}{3}n\log_3(2n+1) - \frac{3}{2}n + \frac{7}{6}\log_3(2n+1),\ \frac{1}{2}n - \frac{1}{2}\log_3(2n+1))$	$4n\log_3(2n+1) - 4n$ $+\log_3(2n+1)$	Table VII(c)
DCT-8$_n$	same as DST-7		duality (54)
DST-8$_n$	same as DCT-7		duality (54)
DCT-5$_n$	same as DCT-7		Table VII(d) and its transpose
DST-5$_n$	same as DST-7		Table VII(d) and its transpose
DCT-6$_n$	same as DCT-7		duality (54), Table VII(c)^T
DST-6$_n$	same as DST-7		duality (54), Table VII(c)^T

implementations, it is crucial to have a recursive form as presented here. Further, the derivation for $p > 2$ in [30] produced suboptimal cost compared to our algorithms.

Special cases of (48) for the DCT-3 with the reverse split, i.e., for $n = p^t$, $k = p^{t-1}$ and $m = p$, may not be practical because of the long critical path for computing $B_{k,m}$. Their discovery, however, is more straightforward, since they do not require large skew DCTs, which are unexpected without the algebraic approach. The case $p = 2$ was reported in [52], $p = 3, 6$ in [53], the case of a general p in [54] with examples $p = 3, 5, 7, 9$.

Algorithm (48) for DST-3 and for 2-powers and $k = 2$ was found in [55]. The only special case of (48) for DCT-4 we found in the literature (again for 2-powers and $k = 2$ only) was derived implicitly in [51], where the DCT-4 is called "odd DCT."

The only case of (49) we found in the literature is for DCT-2 and $n = 2^t$, $m = 2$, in which case the skew DCTs become trivial [56].

All other cases in Table XI are to our best knowledge novel.

VIII. CONCLUSIONS

We presented an algebraic approach to deriving fast transform algorithms as an extension to the algebraic signal processing theory (ASP). In particular, we identified the general principle behind the Cooley-Tukey FFT and applied it to derive

"Cooley-Tukey type" algorithms for all 16 DCTs and DSTs. In doing so, we explain many existing algorithms and discover an even larger number of new algorithms that were not found with previous methods. In particular, general radix algorithms for the DCTs and DSTs were not known before. The availability of a flexible radix algorithm helps, as for the DFT, with the optimization of its implementation for computers with a deep memory hierarchy, vector instruction sets, and multiple processors.

From a theoretical point of view, our approach also explains why these algorithms exist, makes the derivation comparatively easy, and explains their structure. The key is to associate with each transform a polynomial algebra, and to derive algorithms by manipulating this algebra rather than the transform itself. That polynomial algebras play such an important role is not surprising as explained by ASP [8]: they provide the structure for finite, shift-invariant SP. This means, the signal and filter spaces are polynomial algebras and the associated Fourier transform is provided by the Chinese remainder theorem. Thus, in ASP, the signal processing theory naturally connects with the theory of fast transform algorithms.

REFERENCES

[1] M. Frigo and S. G. Johnson, "FFTW: An adaptive software architecture for the FFT," in *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1998, vol. 3, pp. 1381–1384.

[2] Matteo Frigo and Steven G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, 2005, special issue on "Program Generation, Optimization, and Adaptation".

[3] Markus Püschel, José M. F. Moura, Jeremy Johnson, David Padua, Manuela Veloso, Bryan W. Singer, Jianxin Xiong, Franz Franchetti, Aca Gačić, Yevgen Voronenko, Kang Chen, Robert W. Johnson, and Nick Rizzolo, "SPIRAL: Code generation for DSP transforms," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 232–275, 2005, special issue on "Program Generation, Optimization, and Adaptation".

[4] "FFTW web site," 1998, `www.fftw.org`.

[5] F. Franchetti and M. Püschel, "A SIMD vectorizing compiler for digital signal processing algorithms," in *Proc. IPDPS*, 2002, pp. 20–26.

[6] F. Franchetti, Y. Voronenko, and M. Püschel, "FFT program generation for shared memory: SMP and multicore," in *Proc. Supercomputing (SC)*. 2006, ACM, New York.

[7] "Spiral web site," 1998, `www.spiral.net`.

[8] M. Püschel and J. M. F. Moura, "Algebraic signal processing theory: Foundation and 1-D time," submitted for publication, the material is also available as part of [10].

[9] M. Püschel and J. M. F. Moura, "Algebraic signal processing theory: 1-D space," submitted for publication, the material is also available as part of [10].

[10] M. Püschel and J. M. F. Moura, "Algebraic signal processing theory," available at http://arxiv.org/abs/cs.IT/0612077, material from this manuscript is submitted as [8] and [9].

[11] M. Püschel and J. M. F. Moura, "The algebraic approach to the discrete cosine and sine transforms and their fast algorithms," *SIAM Journal of Computing*, vol. 32, no. 5, pp. 1280–1316, 2003.

[12] M. Püschel, "Cooley-Tukey FFT like algorithms for the DCT," in *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003, vol. 2, pp. 501–504.

[13] Y. Voronenko and M. Püschel, "Algebraic derivation of general radix Cooley-Tukey algorithms for the real discrete Fourier transform," in *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2006, vol. 3, pp. 876–879.

[14] M. Püschel and M. Rötteler, "Algebraic signal processing theory: Cooley-Tukey type algorithms on the 2-D hexagonal spatial lattice," *Applicable Algebra in Engineering, Communication and Computing*, special issue in the memory of Thomas Beth, to appear.

[15] M. Püschel and M. Rötteler, "Algebraic signal processing theory: 2-D hexagonal spatial lattice," *IEEE Transactions on Image Processing*, vol. 16, no. 6, pp. 1506–1521, 2007.

[16] W. H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. on Communications*, vol. COM-25, no. 9, pp. 1004–1009, 1977.

[17] P. P. N. Yang and M. J. Narasimha, "Prime factor decomposition of the discrete cosine transform," in *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1985, pp. 772–775.

[18] P. Duhamel and H. H'Mida, "New 2^n algorithms suitable for VLSI implementations," in *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1987, pp. 1805–1808.

[19] M. Püschel and J. M. F. Moura, "Algebraic signal processing theory: Cooley-Tukey type algorithms for DCTs and DSTs," extended version of this paper; available at http://arxiv.org/abs/cs.IT/0702025.

[20] L. Auslander, E. Feig, and S. Winograd, "Abelian semi-simple algebras and algorithms for the discrete Fourier transform," *Advances in Applied Mathematics*, vol. 5, pp. 31–55, 1984.

[21] P. J. Nicholson, "Algebraic theory of finite Fourier transforms," *Journal of Computer and System Sciences*, vol. 5, no. 5, pp. 524–547, 1971.

[22] Th. Beth, *Verfahren der Schnellen Fouriertransformation [Methods for the Fast Fourier Transform]*, Teubner, 1984.

[23] S. Winograd, "On the multiplicative complexity of the discrete Fourier transform," *Advances in Mathematics*, vol. 32, pp. 83–117, 1979.

[24] L. Auslander, E. Feig, and S. Winograd, "The multiplicative complexity of the discrete Fourier transform," *Advances in Applied Mathematics*, vol. 5, pp. 87–109, 1984.

[25] H. W. Johnson and C. S. Burrus, "On the structure of efficient DFT algorithms," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-33, no. 1, pp. 248–254, 1985.

[26] M. T. Heideman and C. S. Burrus, "On the number of multiplications necessary to compute a length-2^n DFT," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, no. 1, pp. 91–95, 1986.

[27] H. J. Nussbaumer, *Fast Fourier Transformation and Convolution Algorithms*, Springer, 2nd edition, 1982.

[28] R. Tolimieri, M. An, and C. Lu, *Algorithms for Discrete Fourier Transforms and Convolution*, Springer, 2nd edition, 1997.

[29] G. Steidl and M. Tasche, "A polynomial approach to fast algorithms for discrete Fourier-cosine and Fourier-sine transforms," *Mathematics of Computation*, vol. 56, no. 193, pp. 281–296, 1991.

[30] G. Steidl, "Fast radix-p discrete cosine transform," *Applicable Algebra in Engineering, Communication and Computing*, vol. 3, no. 1, pp. 39–46, 1992.

[31] Z. Wang and B. R. Hunt, "The discrete W transform," *Applied Mathematics and Computation*, vol. 16, pp. 19–48, 1985.

[32] V. Britanak and K. R. Rao, "The fast generalized discrete Fourier transforms: A unified approach to the discrete sinusoidal transforms computation," *Signal Processing*, vol. 79, pp. 135–150, 1999.

[33] C. Van Loan, *Computational Framework of the Fast Fourier Transform*, Siam, 1992.

[34] D. Rockmore, "Fast Fourier analysis for Abelian group extensions," *Advances in Applied Mathematics*, vol. 11, pp. 164–204, 1990.

[35] D. Maslen and D. Rockmore, "Generalized FFTs – a survey of some recent results," in *Proceedings of IMACS Workshop in Groups and Computation*, 1995, vol. 28, pp. 182–238.

[36] 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.

[37] M. Püschel, "Decomposing monomial representations of solvable groups," *Journal of Symbolic Computation*, vol. 34, no. 6, pp. 561–596, 2002.

[38] M. Püschel and M. Rötteler, "Cooley-Tukey FFT like fast algorithms for the discrete triangle transform," in *Proc. 11th IEEE DSP Workshop*, 2004, pp. 158–162.

[39] M. Püschel and M. Rötteler, "The discrete triangle transform," in *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2004, vol. 3, pp. 45–48.

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

[41] P. Yip and K. R. Rao, "A fast computational algorithm for the discrete sine transform," *IEEE Trans. on Communications*, vol. COM-28, no. 2, pp. 304–307, 1980.

[42] Z. Wang, "Fast algorithms for the discrete W transform and for the discrete Fourier transform," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-32, no. 4, pp. 803–816, 1984.

[43] P. Yip and K. R. Rao, "Fast decimation-in-time algorithms for a family of discrete sine and cosine transforms," *Circuits, Systems, and Signal Processing*, vol. 3, no. 4, pp. 387–408, 1984.

[44] P. Yip and K. R. Rao, "The decimation-in-frequency algorithms for a family of discrete sine and cosine transforms," *Circuits, Systems, and Signal Processing*, vol. 7, no. 1, pp. 3–19, 1988.

[45] S. C. Chan and K. L. Ho, "Direct methods for computing discrete sinusoidal transforms," *IEE Proceedings*, vol. 137, no. 6, pp. 433–442, 1990.

[46] M. O. Rayes, V. Trevisan, and P. S. Wang, "Factorization of Chebyshev polynomials," Tech. Rep. ICM-199802-0001, Kent State University, 1998.

[47] G. Szegö, *Orthogonal Polynomials*, Amer. Math. Soc. Colloq. Publ., 4th edition, 1992.

[48] M. T. Heideman, "Computation of an odd-length DCT from a real-valued DFT of the same length," *IEEE Trans. on Signal Processing*, vol. 40, pp. 54–61, 1992.

[49] Z. Wang, "Pruning the fast discrete cosine transform," *IEEE Transactions on Communications*, vol. 39, no. 5, pp. 640–643, 1991.

[50] X. Shao and S. Johnson, "Type-II/II DCT/DST algorithms with reduced number of arithmetic operations," submitted for publication.

[51] Y. Morikawa, H. Hamada, and N. Yamane, "A fast algorithm for the cosine transform based on successive order reduction of the Chebyshev polynomial," *Electronics and Communications in Japan, Part 1*, vol. 69, no. 3, pp. 173–180, 1986.

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

[53] Y. H. Chan and W. C. Siu, "Mixed-radix discrete cosine transform," *IEEE Trans. on Signal Processing*, vol. 41, no. 11, pp. 3157–3161, 1993.

[54] 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.

[55] Z. Wang, "Fast discrete sine transform algorithms," *Signal Processing*, vol. 19, pp. 91–102, 1990.

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

[57] T. S. Chihara, *An Introduction to Orthogonal Polynomials*, Gordon and Breach, 1978.

[58] T. J. Rivlin, *The Chebyshev Polynomials*, Wiley Interscience, 1974.

APPENDIX I
CHINESE REMAINDER THEOREM

Let $\mathbb{C}[x]/p(x)$ be a polynomial algebra (see Section II) and assume that $p(x) = q(x)r(x)$ factors into two coprime polynomials, i.e., $\gcd(q, r) = 1$. Then the Chinese remainder theorem (for polynomials) states that

$$\begin{aligned} \phi:\ \mathbb{C}[x]/p(x) &\rightarrow \mathbb{C}[x]/q(x) \oplus \mathbb{C}[x]/r(x) \\ s(x) &\mapsto (s(x) \bmod q(x), s(x) \bmod r(x)) \end{aligned}$$

is an isomorphism of algebras. Formally, this implies

$$\begin{aligned} \phi(s + s') &= \phi(s) + \phi(s'), \\ \phi(s \cdot s') &= \phi(s) \cdot \phi(s'). \end{aligned}$$

Informally, this means that computing in $\mathbb{C}[x]/p(x)$ and elementwise computing in $\mathbb{C}[x]/q(x) \oplus \mathbb{C}[x]/r(x)$ is equivalent.

APPENDIX II
CHEBYSHEV POLYNOMIALS

Chebyshev polynomials are a special class of orthogonal polynomials and play an important role in many mathematical areas. Excellent introductory books are [57], [47], [58]. We only introduce the definitions and the properties of the polynomials we use in this paper.

Let $C_0(x) = 1$ and $C_1(x)$ be a polynomial of degree 1, and define $C_n(x)$ for $n > 1$ by the recurrence

$$C_n(x) = 2xC_{n-1}(x) - C_{n-2}(x).$$

TABLE XVI

FOUR SERIES OF CHEBYSHEV POLYNOMIALS. THE RANGE FOR THE ZEROS IS $0 \le k < n$. IN THE TRIGONOMETRIC CLOSED FORM $\cos\theta = x$.

	$n = 0, 1$	closed form	symmetry	zeros
T_n	$1, x$	$\cos(n\theta)$	$T_{-n} = T_n$	$\cos\frac{(k+\frac{1}{2})\pi}{n}$
U_n	$1, 2x$	$\frac{\sin(n+1)\theta}{\sin\theta}$	$U_{-n} = -U_{n-2}$	$\cos\frac{(k+1)\pi}{n+1}$
V_n	$1, 2x-1$	$\frac{\cos(n+\frac{1}{2})\theta}{\cos\frac{1}{2}\theta}$	$V_{-n} = V_{n-1}$	$\cos\frac{(k+\frac{1}{2})\pi}{n+\frac{1}{2}}$
W_n	$1, 2x+1$	$\frac{\sin(n+\frac{1}{2})\theta}{\sin\frac{1}{2}\theta}$	$W_{-n} = -W_{n-1}$	$\cos\frac{(k+1)\pi}{n+\frac{1}{2}}$

TABLE XVII

IDENTITIES AMONG THE FOUR SERIES OF CHEBYSHEV POLYNOMIALS; C_n HAS TO BE REPLACED BY T_n, U_n, V_n, W_n TO OBTAIN ROWS $1, 2, 3, 4$, RESPECTIVELY.

C_n	$C_n - C_{n-2}$	$C_n - C_{n-1}$	$C_n + C_{n-1}$
T_n	$2(x^2-1)U_{n-2}$	$(x-1)W_{n-1}$	$(x+1)V_{n-1}$
U_n	$2T_n$	V_n	W_n
V_n	$2(x-1)W_{n-1}$	$2(x-1)U_{n-1}$	$2T_n$
W_n	$2(x+1)V_{n-1}$	$2T_n$	$2(x+1)U_{n-1}$

Running this recurrence backwards yields polynomials C_{-n}, $n > 0$. Each sequence $(C_n)_{n\in\mathbb{Z}}$ of polynomials defined this way is called a sequence of Chebyshev polynomials. It is uniquely determined by $C_0 = 1$ and the choice of C_1. Four special cases are of particular importance in signal processing [9], [10] and in this paper. They are denoted by $C \in \{T, U, V, W\}$ and are called Chebyshev polynomials of the first, second, third, and fourth kind. Table XVI gives their initial conditions, their closed form, their symmetry properties, and their zeros.

For example, $T_n(x) = \cos(n\theta)$, where $\cos\theta = x$. The closed form easily yields the zeros of T_n.

We will use the following properties of Chebyshev polynomials:

1) For any sequence of Chebyshev polynomials with arbitrary initial conditions C_0, C_1, we have
$$C_n = C_1U_{n-1} - C_0U_{n-2}. \tag{52}$$
2) For any sequence of Chebyshev polynomials C_n,
$$T_kC_n = (C_{n-k} + C_{n+k})/2. \tag{53}$$
3) The identities in Table XVII hold. They are based on trigonometric identities.

APPENDIX III
RELATIONSHIPS BETWEEN DTTS

We use in this paper the following relationships between DTTs. The explanation for their existence and proofs can be found in [9], [11].

Duality. Two DTTs $\mathrm{DTT}_n, \mathrm{DTT}'_n$, which have flipped boundary conditions are called *dual* to each other. They are necessarily in the same group (see Table II). The duality property is not visible from Table II since we omitted the boundary conditions. Thus we just state the pairs: DCT-3/DST-3,

TABLE XVIII

MATRICES IN (56). FROM LEFT TO RIGHT, $X_n^{(C3)}(r)$, $X_n^{(S3)}(r)$, $X_n^{(C4)}(r)$, $X_n^{(S4)}(r)$. WE USE $c_\ell = \cos(1/2-r)\ell\pi/n$, $s_\ell = \sin(1/2-r)\ell\pi/n$, $c'_\ell = \cos(1/2-r)(2\ell+1)\pi/(2n)$, AND $s'_\ell = \sin(1/2-r)(2\ell+1)\pi/(2n)$. WHERE THE DIAGONALS CROSS, THE ELEMENTS ARE ADDED.

$$\begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & c_1 & & & s_{n-1} \\ \vdots & & \ddots & \iddots & \\ \vdots & & \iddots & \ddots & \\ 0 & s_1 & & & c_{n-1} \end{bmatrix}, \begin{bmatrix} c_1 & & & -s_{n-1} & 0 \\ & \ddots & \iddots & & \vdots \\ & \iddots & \ddots & & \vdots \\ -s_1 & & & c_{n-1} & 0 \\ 0 & \cdots & \cdots & 0 & c_n \end{bmatrix}, \begin{bmatrix} c'_0 & & & s'_{n-1} \\ & \ddots & \iddots & \\ & \iddots & \ddots & \\ s'_0 & & & c'_{n-1} \end{bmatrix}, \begin{bmatrix} c'_0 & & & -s'_{n-1} \\ & \ddots & \iddots & \\ & \iddots & \ddots & \\ -s'_0 & & & c'_{n-1} \end{bmatrix}$$

DCT-4/DST-4, the DTTs in the U-group (DTTs of type 1 and 2) are all self-dual, DCT-7/DST-8, DST-7/DCT-8, DCT-5/DCT-6, DST-5/DST-6.

The following relationship holds for dual DTTs:

$$\mathrm{diag}_{0\le i<n}((-1)^i) \cdot \mathrm{DTT}_n \cdot J_n = \mathrm{DTT}'_n. \tag{54}$$

As a consequence any DTT algorithm can be converted into a DTT' algorithm without incurring additional operations.

Base change. Two DTTs (or skew DTTs) in the same group (e.g., T-group) have (at least almost) the same associated algebra. As a consequence they can be translated into each other at the expense of $O(n)$ operations with a suitable base change using Table XVII (see [9]).

Examples include

$$\begin{aligned} \text{DCT-4}_n &= S_n \cdot \text{DCT-2}_n \cdot \tfrac{1}{2} D_n(1/2)^{-1}, \\ \text{iDCT-4}_n(r) &= S_n \cdot \text{iDCT-3}_n(r) \cdot \tfrac{1}{2} D_n(r)^{-1}. \end{aligned} \tag{55}$$

S_n is defined in Table V and $D_n(r) = \mathrm{diag}_{0\le i<n}(\cos \frac{r_i}{2}\pi)$. The r_i are computed from r using Lemma 1.

Skew and non-skew DTTs. Every skew $\mathrm{DTT}(r)$ can be translated into its non-skew counterpart DTT:

$$\begin{aligned} \mathrm{DTT}_n(r) &= \mathrm{DTT}_n \cdot X_n^{(*)}(r), \quad \text{and} \\ \overline{\mathrm{DTT}}_n(r) &= \overline{\mathrm{DTT}}_n \cdot X_n^{(*)}(r). \end{aligned} \tag{56}$$

Here, $X_n^{(*)}(r)$ depends on the DTT; the exact form is given in Table XVIII.

Combining (56) with (55) gives, for example

$$\text{DCT-4}_n(r) = S_n \cdot \text{DCT-2}_n \cdot \tfrac{1}{2} D_n(1/2)^{-1} \cdot X_n^{(C4)}(r). \tag{57}$$

The diagonal matrix can be fused with the x-shaped matrix to save multiplications.

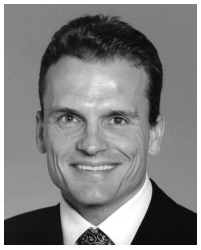
Inversion of (56) gives the corresponding identities for the $\mathrm{iDTT}(r)$'s:

$$\mathrm{iDTT}_n(r) = \left(X_n^{(*)}(r)\right)^{-1} \cdot \mathrm{DTT}_n^T. \tag{58}$$

The matrices $\left(X_n^{(*)}(r)\right)^{-1}$ have the same x-shaped structure and the same arithmetic complexity as $X_n^{(*)}(r)$ and can be readily computed because of their block structure. For example:

$$\left(X_n^{(C3)}(r)\right)^{-1} = \frac{1}{\cos(1/2-r)\pi} \begin{bmatrix} c_n & 0 & \cdots & \cdots & 0 \\ 0 & c_{n-1} & & & -s_{n-1} \\ \vdots & & \ddots & \iddots & \\ \vdots & & \iddots & \ddots & \\ 0 & -s_1 & & & c_1 \end{bmatrix}.$$

The above identities show that the complexity of a skew DTT differs from the complexity of the associated DTT by $O(n)$.



Markus Püschel (M'99–SM'05) is an Associate Research Professor of Electrical and Computer Engineering at Carnegie Mellon University. He received his Diploma (M.Sc.) in Mathematics and his Doctorate (Ph.D.) in Computer Science, in 1995 and 1998, respectively, both from the University of Karlsruhe, Germany. From 1998-1999 he was a Postdoctoral Researcher at Mathematics and Computer Science, Drexel University, Philadelphia. Since 2000 he has been with Carnegie Mellon University. He is an Associate Editor for the *IEEE Transactions on Signal Processing*, and was an Associate Editor for the *IEEE Signal Processing Letters*, a Guest Editor of the *Journal of Symbolic Computation*, and the *Proceedings of the IEEE*. His research interests include signal processing theory/software/hardware, scientific computing, compilers, applied mathematics and algebra.

José M. F. Moura (S'71–M'75–SM'90–F'94) received the engenheiro electrotécnico degree in 1969 from Instituto Superior Técnico (IST), Lisbon, Portugal, and the M.Sc., E.E., and D.Sc. degrees in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology (M.I.T.), Cambridge, in 1973 and 1975, respectively.

He is a Professor of Electrical and Computer Engineering and, by courtesy, of BioMedical Engineering at Carnegie Mellon University (CMU) where he is a founding co-director of the Center for Sensed Critical Infrastructures Research (CenSCIR). He was on the faculty at IST (1975-84) and has held visiting faculty appointments at MIT (1984-86, 1999-00, 2006-07) and as a research scholar at USC (summers of 1978-81). In 2006, he founded the Information and Communications Technologies Institute, a joint venture between CMU and Portugal, that he co-Directs, and that manages the CMU-Portugal education and research program, www.icti.cmu.edu.

His research interests include statistical and algebraic signal processing, image, bioimaging, and video processing, and digital communications. He has published over 300 technical Journal and Conference papers, is the co-editor of two books, holds six patents on image and video processing, and digital communications with the US Patent Office, and has given numerous invited seminars at US and European Universities and industrial and government Laboratories.

Dr. Moura is the *President Elect* (2006-07) for the *IEEE Signal Processing Society*(SPS), which he has served as *Vice-President for Publications* (2000-02), *Editor in Chief* for the *IEEE Transactions in Signal Processing* (1975-99), interim *Editor in Chief* for the *IEEE Signal Processing Letters* (December 2001-May 02), founding member of the *Bioimaging and Signal Processing* (BISP) Technical Committee, and member of several other Technical Committees. He was *Vice-President for Publications* for the *IEEE Sensors Council* (2000-02) and is or was on the *Editorial Board* of several Journals, including the *IEEE Proceedings*, the *IEEE Signal Processing Magazine*, and the *ACM Transactions on Sensor Networks*. He chaired the *IEEE TAB Transactions Committee* (2002-03) that joins the more than 80 Editors in Chief of the IEEE Transactions and served on the *IEEE TAB Periodicals Review Committee* (2002-06). He is on the *International Conference on Information Processing and Sensor Networks* (IPSN) and was on the *International Symposium on BioImaging* (ISBI) Steering Committees and has been on the program committee of over 35 Conferences and Workshops. He was on the *IEEE Press Board* (1991-95).

Dr. Moura is a *Fellow* of the *IEEE*, a *Fellow* of the *American Association for the Advancement of Science* (AAAS), and a corresponding member of the *Academy of Sciences of Portugal* (Section of Sciences). He was awarded the 2003 *IEEE Signal Processing Society Meritorious Service Award*, in 2000 the *IEEE Millennium Medal*, in 2006 an *IBM Faculty Award*, and in 2007 the CMU College of Engineering *Outstanding Research Award* (with Püschel). He is affiliated with several IEEE societies, Sigma Xi, AMS, AAAS, IMS, and SIAM.