AAECC manuscript No.
(will be inserted by the editor)

Markus Püschel and Martin Rötteler

Algebraic Signal Processing Theory: Cooley-Tukey Type Algorithms on the 2-D Hexagonal Spatial Lattice

the date of receipt and acceptance should be inserted later

It is with great sadness that the authors contribute this paper to this special issue in memory of their former PhD advisor Thomas Beth. Beth was an extremely versatile researcher with contributions in a wide range of disciplines. However, one pervading theme can be identified in all of his work: the belief that mathematics, and in particular abstract algebra, was the language and key to uncovering the structure in many real world problems. One testament to this vision is his seminal habilitation thesis on the theory of Fourier transform algorithms, which ingeniously connects one of the principal tools in signal processing with group theory to open up an entirely new field of research. The authors deeply regret that Beth's untimely death prevented him from seeing the Algebraic Signal Processing Theory, a body of work, including the present paper, that develops an axiomatic approach to and generalization of signal processing based on the representation theory of algebras. The theory is a logical continuation of Beth's ideas and would not exist without him and his influence as PhD advisor. The authors like to think that he would have approved of this work and wish to dedicate this paper to him and his memory.

Abstract Recently, we introduced the framework for signal processing on a non-separable 2-D hexagonal spatial lattice including the associated notion of Fourier transform called discrete triangle transform (DTT). Spatial means that the lattice is undirected in contrast to earlier work by Mersereau introducing hexagonal discrete Fourier transforms. In this paper we derive a general-radix algorithm for the DTT of an $n \times n$ 2-D signal, focusing on the radix-2×2 case. The runtime of the algorithm is $O(n^2 \log(n))$, which is the same as for commonly used separable 2-D transforms. The DTT algorithm derivation is based on the algebraic signal processing theory. This means that instead of manipulating transform co-

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

Markus Püschel is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh. Martin Rötteler is with NEC Laboratories America, Inc., 4 Independence Way, Suite 200, Princeton, NJ 08540. E-mail: pueschel@ece.cmu.edu, mroetteler@neclabs.com

Address(es) of author(s) should be given

Fig. 1	Fourier transform	Polynomial algebra	Basis	Separable
(a)	2-D DFT	$\mathbb{C}[x,y]/\langle x^n-1, y^n-1\rangle$	$\{x^i y^j\}$	yes
(b)	2-D DCT, type 3	$\mathbb{C}[x,y]/\langle T_n(x), T_n(y)\rangle$	$\{T_i(x)T_j(y)\}$	yes
(c)	DTT	$\mathbb{C}[x,y]/\langle T_{n,0}(x,y), T_{0,n}(x,y)\rangle$	$\{T_{i,j}(x,y)\}$	no

Table 1 Correspondence between the signal structures in Fig. 1, associated Fourier transforms, and polynomial algebra with fixed basis.

efficients, the algorithm is derived through a stepwise decomposition of its underlying polynomial algebra based on a general theorem that we introduce. The theorem shows that the obtained DTT algorithm is the precise equivalent of the well-known Cooley-Tukey fast Fourier transform, which motivates the title of this paper.

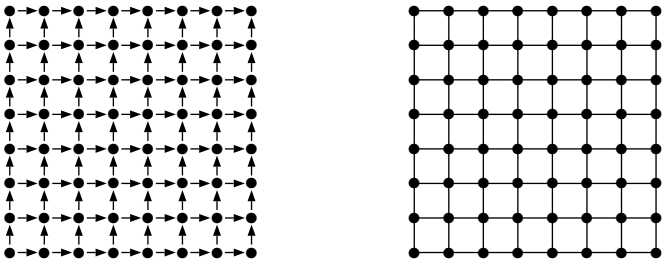
1 Introduction

In [1], we introduced the framework for signal processing on a spatial hexagonal lattice in two dimensions (2-D). The derivation of this framework is an application of the algebraic signal processing theory that we recently introduced [2, 3]. Namely, from basic assumptions on the desired geometry (in this case a hexagonal lattice) to be imposed on the signal we derived the proper polynomial algebra and a suitable basis to support this geometry. The polynomial algebra then provides the proper notions of signal space, filter space, shift operators, convolution, spectrum, and Fourier transform. The latter we termed the discrete triangle transform (DTT).

The DTT, and the polynomial algebra framework in general, is best understood by visualizing the associated geometry of the signal domain imposed by the DTT and comparing it to the geometry imposed by other transforms: see Fig. 1 and Table 1, which provides the associated polynomial algebras and basis. More details are provided in the paper.

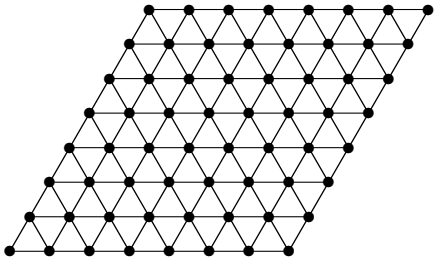
For example, the 2-D discrete Fourier transform (DFT) assumes the signal (given by a 2-D array of numbers) resides on the rectangular lattice shown in Fig. 1(a). The lattice is directed and the boundary conditions (not shown) are periodic, which makes the domain a torus. The 2-D DFT diagonalizes the adjacency matrix of this torus, which makes the connection between transform and lattice rigorous. The polynomial algebra and basis underlying the torus are shown in Table 1.

Next, the 2-D discrete cosine and sine transforms (DCTs/DSTs) impose a similar structure shown in Fig. 1(b), but now the lattice is undirected, which we call *spatial*. The boundary conditions are symmetric or antisymmetric, depending on the type of DCT or DST. Again, the adjacency matrix of this graph is diagonalized by the respective 2-D DCT or DST. The polynomial algebra and basis for the special case of a DCT, type 3, is shown in Table 1. In contrast to the previous case, both algebra and basis are built from Chebyshev polynomials $T_i(x)$ of the first kind in one variable [4, 5].



(a) 2-D DFT

(b) 2-D DCT/DST



(c) DTT

Fig. 1 Visualizations of the signal structure imposed by (a) the 2-D DFT (cyclic boundary conditions are omitted); (b) the 2-D DCTs/DSTs (symmetric/antisymmetric boundary conditions are omitted); and (c) the DTTs (boundary conditions are omitted).

In both cases, the signal domain is separable, which means a direct product of line graphs. As a consequence the transforms are tensor or Kronecker products of 1-D transforms (DFT or DCT/DST in this case).

The signal structure assumed by the DTT, again without boundary conditions, is shown in Fig. 1(c). The lattice is spatial and hexagonal and hence nonseparable. Its adjacency matrix is diagonalized by the DTT. The polynomial algebra (Table 1) is now built from the lesser known Chebyshev polynomials $T_{i,j}(x, y)$ of the first kind in *two* variables [1, 6].

In this paper we first show that the DTT possesses general radix Cooley-Tukey type algorithms. By this we mean algorithms that are based on the same algebraic principle as the Cooley-Tukey fast Fourier transform (FFT). Namely, we derive these algorithms by a stepwise decomposition of the underlying polynomial algebra rather than the transform itself using only one general theorem. In [7, 8] we have used this technique to derive a large class of Cooley-Tukey type algorithms for trigonometric 1-D transforms. Here, we generalize this theorem to 2-D and apply it to the DTT. Further, we perform a detailed derivation for the radix-2×2 case, completing our preliminary results from [9], and show that the resulting algorithm has $O(n^2 \log(n))$ runtime, just like the best known algorithms for the separable 2-D DFT and 2-D DCTs/DSTs.

Related work. Hexagonal lattices, their associated Fourier transforms, and their fast algorithms where considered before as part of the seminal work on multidimensional signal processing by Mersereau [10,11]. See also the book [12] which contains an extensive treatment of hexagonal lattices in image processing and applications and the paper [13] and the references given therein.

However, the above work considers exclusively *directed* hexagonal lattices, i.e., those of the type in Fig. 1(a). This implies a notion of Fourier transform different from the DTT. Intuitively, a spatial, or *undirected*, lattice should make more sense for many image processing applications. Indeed, for standard rectangular images, the 2-D DCT, associated with the spatial lattice in Fig. 1(b), is often used instead of the 2-D DFT. A prominent example is JPEG image compression.

One contribution of this paper is in providing a fast algorithm for the DTT. The other is our technique for deriving this algorithm, namely, by working with the polynomial algebra instead of the transform. This approach can be viewed as an extension of Nussbaumer's derivations of FFTs and convolution algorithms in the first book on FFTs [14]. Namely, he also used polynomial algebras and factorization properties of polynomials. The same holds for the radix-2 fast DCT algorithm derived in [15]. In contrast, we use decomposition properties of polynomials and have shown that these underly general-radix algorithms [4,8].

Another technique for deriving fast algorithms for 1-D transforms is presented in [16,17] and requires that the underlying polynomial algebra has a basis of orthogonal polynomials. The technique only uses the three-term recurrence, which is a defining property of these polynomials, and is thus very general. However, for the 1-D DCTs and DSTs, built from (orthogonal) Chebyshev polynomials, it yields suboptimal algorithms with $O(n \log^2(n))$ operations versus $O(n \log(n))$ for the best known ones. The same suboptimality is likely for a potential 2-D extension of that method (which does not exist yet) when applied to the DTT.

Finally, we want to characterize our algorithm derivation in more algebraic terms. Namely, a polynomial algebra, viewed as regular module, is an induction of the trivial regular module [18]. Our stepwise decomposition is equivalent to a decomposition of the induction into two steps. The same method, applied to group algebras, has been used to derive fast Fourier transforms on groups, an area pioneered by Beth in his seminal book [19] and further developed by Clausen [20–22], and by Rockmore, Maslen, and others [23–25].

Organization. Section 2 provides the necessary background on polynomial algebras in one and two variables and their associated transforms. This includes the DTT that we introduce and define. We also briefly discuss the connection to signal processing. Section 3 then explains how to derive fast transform algorithms using polynomial algebras, first focusing on one variable with the DFT and DCT as examples, and then generalizing the technique to two variables to make it applicable to the DTT. In Section 4 we then perform the actual derivation of a general-radix DTT algorithm with emphasis on the radix-2×2 case. We conclude with Section 5.

2 Polynomial Algebras and Their Transforms

In this section we provide the necessary background on polynomial algebras and their associated polynomial transforms. These cover many of the transforms used in signal processing including the DTT, which is defined in this section. Later, we

derive DTT algorithms by decomposing the underlying polynomial algebra rather than working with the transform itself.

2.1 Polynomial Algebras

Recall that an *algebra* $\mathcal{A}$ is a vector space that is also a ring, i.e., it is also equipped with a multiplication for its elements [26]. Important examples for algebras are the complex numbers $\mathbb{C}$ and the sets $\mathbb{C}[x]$, $\mathbb{C}[x,y]$ of polynomials in one or two variables, respectively.

One variable. Let $p(x) \in \mathbb{C}[x]$ be a polynomial of degree n. Then we can define another important class of algebras as follows: the elements are the polynomials of degree less than n and the multiplication of two polynomials is computed modulo $p(x)$. This algebra is denoted by

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

It has dimension $\dim(\mathcal{A}) = \deg(p)$.

More formally, $\langle p(x)\rangle = p(x)\mathbb{C}[x] \lhd \mathbb{C}[x]$ is the ideal [27,28] of $\mathbb{C}[x]$ generated by $p(x)$ and $\mathbb{C}[x]/p(x)$ is the associated quotient ring or factor ring.

An example is $\mathcal{A} = \mathbb{C}[x]/(x^2-1)$. In $\mathcal{A}$, for example, $x^2 = 1 \bmod (x^2-1)$ and we can compute $x(x+1) = x^2 + x = x + 1 \bmod (x^2-1)$. A basis of $\mathcal{A}$ is $b = (1, x)$ (we use parentheses since the order of basis elements matters in this paper).

Two variables. Similarly, we can consider two polynomials $p(x,y)$ and $q(x,y)$ in two variables and construct the polynomial algebra

$$\mathcal{A} = \mathbb{C}[x,y]/\langle p(x,y), q(x,y)\rangle. \tag{1}$$

Again, $\langle p(x,y), q(x,y)\rangle = p(x,y)\mathbb{C}[x,y] + q(x,y)\mathbb{C}[x,y] \trianglelefteq \mathbb{C}[x,y]$ denotes the ideal of $\mathbb{C}[x,y]$ generated by $p(x,y)$ and $q(x,y)$ (in short: by p and q). $\mathcal{A}$ is equipped with ordinary addition and multiplication and all computations are carried out modulo p and q. Computation modulo two polynomials is no longer a simple matter as before in the case of univariate algebras. It very much depends on the actual choice of p and q whether it is possible or not to obtain a *unique* reduced normal form by successively performing reductions modulo p and q. Technically, the condition to guarantee unique normal forms is that p and q form a Gröbner basis [28,27]. We will not digress any further into this theory but note that all examples considered in this paper actually have the property that p and q form a Gröbner basis, i. e., computing modulo p and q is always well-defined (see also Appendix B for further discussion).

Also note that $\mathcal{A}$ in (1) is not necessarily of finite dimension. However, the cases considered in this paper will be.

As an example, consider $\mathcal{A} = \mathbb{C}[x,y]/\langle x^2-1, y^2-1\rangle$. In $\mathcal{A}$, we can compute, for example, $x(xy+1) = x^2y + x = x + y \bmod \langle x^2-1, y^2-1\rangle$. Using similar reductions, one finds that $\dim(\mathcal{A}) = 4$; for example, $b = (1, x, y, xy)$ is a basis.

A generalization of (1) to n variables is straightforward. However, in this paper, we restrict ourselves to the special case of two variables.

2.2 Polynomial Transforms

With each polynomial algebra we can associate a polynomial transform as explained next.

One variable. Let $\mathcal{A} = \mathbb{C}[x]/p(x)$ be a polynomial algebra in one variable. We assume that the zeros $\alpha = (\alpha_0, \ldots, \alpha_{n-1})$ of p are pairwise distinct. The Chinese Remainder Theorem (CRT) for polynomials [29] decomposes $\mathcal{A}$ into one-dimensional algebras as follows:

$$\begin{aligned} \Delta :\ \mathbb{C}[x]/p(x) &\to \textstyle\bigoplus_{0\le k<n} \mathbb{C}[x]/(x-\alpha_k), \\ s(x) &\mapsto (s(\alpha_0), \ldots, s(\alpha_{n-1})). \end{aligned} \tag{2}$$

Here, $\oplus$ is the direct sum of algebras defined as their Cartesian product with componentwise operation. Δ is an isomorphism of algebras (even an $\mathcal{A}$-module homomorphism) and, in particular, a bijective linear mapping. Let $b = (p_0, \ldots, p_{n-1})$ be a basis of $\mathbb{C}[x]/p(x)$ and choose the basis $(x^0) = (1)$ in each one-dimensional $\mathbb{C}[x]/(x-\alpha_k)$. Then Δ can be expressed by an invertible $n \times n$ matrix which we refer to as the corresponding *polynomial transform* for $\mathcal{A}$ with basis b. This matrix is denoted with $\mathcal{P}_{b,\alpha}$, since it depends on the basis b and the zeros of p. $\mathcal{P}_{b,\alpha}$ is obtained by evaluating all basis elements at all zeros of p:

$$\mathcal{P}_{b,\alpha} = [p_\ell(\alpha_k)]_{0\le k,\ell<n}. \tag{3}$$

An important example is the discrete Fourier transform DFT_n. Here the algebra is $\mathcal{A} = \mathbb{C}[x]/(x^n - 1)$ with basis polynomials $p_\ell = x^\ell$ and the zeros are $\alpha = (\omega_n^i \mid 0 \le i < n)$, where $\omega_n = \exp(-2\pi \mathrm{i}/n)$. We confirm that

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

is the polynomial transform in this case (see [14, 2]).

Furthermore, it can be shown that practically all trigonometric 1-D transforms used in signal processing are polynomial transforms (or slight generalizations thereof) [2, 4, 8]. For example, If T_k denotes the kth Chebyshev polynomial of the first kind and degree k (see Appendix A) and we set $\mathcal{A} = \mathbb{C}[x]/T_n(x)$ and $b = (T_0(x), \ldots, T_{n-1}(x))$, then

$$\mathcal{P}_{b,\alpha} = [T_\ell(\alpha_k)]_{0\le k,\ell<n} = [\cos \tfrac{(k+1/2)\ell\pi}{n}]_{0\le k,\ell<n}. \tag{4}$$

is the discrete cosine transform (DCT) of type 3. This was already mentioned (for the transpose, i.e., the DCT of type 2) in the original derivation in [30].

Two variables. We consider an algebra

$$\mathcal{A} = \mathbb{C}[x, y]/\langle p(x, y), q(x, y)\rangle \tag{5}$$

where for simplicity and because of its relevance for the following we assume that the total degrees are $\deg(p) = \deg(q) = n$. The total degree of $p(x, y)$ is the largest $k + \ell$ over all nontrivial summands $\beta_{k,\ell} x^k y^\ell$ of $p(x, y)$.

Furthermore, we assume that the common zeros of $p(x, y)$ and $q(x, y)$ constitute a finite set of precisely n^2 distinct zeros,[1] given by the pairs $\alpha = ((\alpha_k, \beta_k) \mid 0 \le k < n^2)$, with each pair in $\mathbb{C}^2$.

[1] Note that for "generic" polynomials this is true, which follows from Bézout's theorem [27].

Again we invoke the CRT and obtain a decomposition into n^2 one-dimensional algebras:

$$\begin{array}{l} \Delta:\ \mathcal{A} \to \bigoplus_{0\le k<n^2} \mathbb{C}[x,y]/\langle x-\alpha_k, y-\beta_k\rangle, \\ s(x,y) \mapsto (s(\alpha_k,\beta_k) \mid 0 \le k < n^2). \end{array} \tag{6}$$

This decomposition also implies that $\dim(\mathcal{A}) = n^2$. We express Δ as a matrix by choosing a vector space basis $b = (p_\ell(x,y) \mid 0 \le \ell < n^2)$ for $\mathcal{A}$ and the basis $(x^0y^0) = (1)$ in each $\mathbb{C}[x,y]/\langle x-\alpha_k, y-\beta_k\rangle$. The resulting matrix is the polynomial transform (now in two variables) for $\mathcal{A}$ with respect to b:

$$\mathcal{P}_{b,\alpha} = [p_\ell(\alpha_k,\beta_k)]_{0\le k,\ell<n^2}. \tag{7}$$

This is a direct generalization of (3). The DTT discussed later is a polynomial transform of the form (7).

A particularly simple example, but relevant for applications, is a *separable* polynomial algebra in two variables. This means that $p(x,y) = p(x)$ and $q(x,y) = q(y)$. In this case, $\mathcal{A}$ in (5) is to isomorphic to the tensor product of two polynomial algebras in one variable:

$$\mathbb{C}[x,y]/\langle p(x), q(y)\rangle \cong \mathbb{C}[x]/p(x) \otimes \mathbb{C}[x]/q(y).$$

More constructively, the tensor product provides for any $\mathcal{A}' = \mathbb{C}[x]/p(x)$ with basis $b' = (p_0, \ldots, p_{n-1})$ a counterpart in two variables, namely

$$\mathcal{A} = \mathcal{A}' \otimes \mathcal{A}' = \mathbb{C}[x,y]/\langle p(x), p(y)\rangle \tag{8}$$

with basis $b = b' \otimes b' = (p_i(x)p_j(x) \mid 0 \le i,j < n)$. Let $\mathcal{P}(b',\alpha')$ denote the polynomial transform for $\mathcal{A}'$, where α' denotes the zeros of $p(x)$. Computing $\mathcal{P}_{b,\alpha}$ using (7) shows that it is the tensor or Kronecker product (of matrices)

$$\mathcal{P}_{b,\alpha} = \mathcal{P}_{b',\alpha'} \otimes \mathcal{P}_{b',\alpha'},$$

where α is given by all possible pairs of elements of α' and

$$A \otimes B = [a_{k,\ell}B]_{k,\ell} \text{ for } A = [a_{k,\ell}]_{k,\ell}. \tag{9}$$

As an example, we consider $\mathcal{A} = \mathbb{C}[x,y]/\langle x^n-1, y^n-1\rangle$ with basis $(x^iy^j \mid 0 \le i,j < n)$. The above discussion shows that

$$\mathcal{P}_{b,\alpha} = \mathrm{DFT}_n \otimes \mathrm{DFT}_n$$

is precisely the 2D-DFT.

In contrast to this example, the DTT defined next is nonseparable.

2.3 Discrete Triangle Transform (DTT)

In this section we define the discrete triangle transform (DTT), introduced in [31, 1], as a polynomial transform for a suitable polynomial algebra in two variables. Hence, the DTT is a special case of (7). The definition requires the Chebyshev polynomials $T_{k,\ell}(x,y)$, $k,\ell \in \mathbb{Z}$, in two variables. The reader is invited to read Appendix B at this point, which collects all the necessary basic information about these polynomials.

DTT definition. Let $\mathcal{A} = \mathbb{C}[x,y]/\langle T_{n,0}, T_{0,n}\rangle$ with basis $b_n = (T_{k,\ell} \mid 0 \le k,\ell < n)$. We denote the n^2 solutions of $T_{n,0} = T_{0,n} = 0$ with $\alpha = ((\alpha_{i,j},\beta_{i,j}) \mid 0 \le i,j < n)$. These are determined by

$$(u_i, v_j) = (\omega_n^i, \omega_{3n}^{1+3j}), \quad 0 \le i,j < n, \tag{10}$$

in the power form of $T_{n,0},\ T_{0,n}$ (see (57) in Appendix B).

The DTT is the associated polynomial transform; namely, the $n^2 \times n^2$ matrix

$$\mathrm{DTT}_{n\times n} = [T_{k,\ell}(\alpha_{i,j},\beta_{i,j})]_{0\le i,j<n,\ 0\le k,\ell<n}. \tag{11}$$

The double index (i,j) is the row index, and (k,ℓ) is the column index, both ordered lexicographically. To compute the matrix entries, the polynomials $T_{k,\ell}$ are evaluated at the zeros of $T_{n,0} = T_{0,n} = 0$ using (10) and the power form of $T_{k,\ell}$ (see (56) in Appendix B). The result is

$$\begin{aligned} T_{k,\ell}(\alpha_{i,j},\beta_{i,j}) = \tfrac{1}{6}\big(&\omega_{3n}^{3ki-3\ell j-\ell} + \omega_{3n}^{3kj-3\ell i+k} + \omega_{3n}^{3ki+3\ell i+3\ell j+\ell} \\ &+ \omega_{3n}^{3\ell i+3kj+3\ell j+k+\ell} + \omega_{3n}^{-3ki-3\ell i-3kj-k} + \omega_{3n}^{-3ki-3kj-3\ell j-k-\ell}\big). \end{aligned}$$

Example: DTT of size 2×2. As an example we consider $n\times n = 2\times 2$, which implies that the DTT is a 4×4 matrix. Here we have to evaluate the polynomials $T_{0,0} = 1, T_{0,1} = y, T_{1,0} = x, T_{1,1} = \frac{1}{2}(3xy-1)$ at the zeros of the equation $T_{2,0} = T_{0,2} = 0$. The solutions are given by the four points

$$(\tfrac{2}{3},\tfrac{2}{3}), (0,0), (\tfrac{2}{3}\omega_3, \tfrac{2}{3}\omega_3^2), (\tfrac{2}{3}\omega_3^2, \tfrac{2}{3}\omega_3). \tag{12}$$

Hence $\mathrm{DTT}_{2\times 2}$ is the 4×4 matrix

$$\mathrm{DTT}_{2\times 2} = \begin{bmatrix} 1 & \frac{2}{3} & \frac{2}{3} & \frac{1}{6} \\ 1 & 0 & 0 & -\frac{1}{2} \\ 1 & \frac{2}{3}\omega_3^2 & \frac{2}{3}\omega_3 & \frac{1}{6} \\ 1 & \frac{2}{3}\omega_3 & \frac{2}{3}\omega_3^2 & \frac{1}{6} \end{bmatrix}. \tag{13}$$

Origin of the DTT. The motivation for the definition of the DTT is explained in detail in [1] and briefly in the next section where we establish the connection to signal processing.

2.4 Connection to Signal Processing: Signal Model

Even though it is not crucial for this paper, we very briefly explain the connection between polynomial algebras, transforms and signal processing as established by the algebraic signal processing theory [4].

Finite shift-invariant 1-D signal models. Given a polynomial algebra $\mathcal{A} = \mathbb{C}[x]/p(x)$ of dimension $\deg(p) = n$ and a basis $b = (p_0, \ldots, p_{n-1})$ of $\mathcal{A}$, we consider $\mathcal{A}$ also as a regular $\mathcal{A}$-module $\mathcal{M} = \mathcal{A}$. We set $\mathbf{s} = (s_0, \ldots, s_{n-1})^T \in \mathbb{C}^n$ and define

$$\begin{aligned} \Phi :\ \mathbb{C}^n &\to \mathcal{M} \\ \mathbf{s} &\mapsto s(x) = \textstyle\sum_{0 \le \ell < n} s_\ell p_\ell. \end{aligned}$$

The triple $(\mathcal{A}, \mathcal{M}, \Phi)$ is called a *signal model* for $\mathbb{C}^n$ in algebraic signal processing. The idea is that the signal model assigns the structure of the $\mathcal{A}$-module $\mathcal{M}$ to the vector space $\mathbb{C}^n$ of finite 1-D signals via the bijective mapping Φ. Once the signal model is chosen, the basic signal processing concepts are all defined. For example, $\mathcal{A}$ is the filter space, $\mathcal{M}$ the signal space, and Φ is the "z-transform" associated with this model. The multiplication of polynomials $h(x) \in \mathcal{A}$ and $s(x) \in \mathcal{M}$ (i.e., the multiplication modulo $p(x)$) is the filtering or convolution for this model. The special filter $x \in \mathcal{A}$ is the shift operator and, since $\mathcal{A}$ is commutative, the signal model supports *shift-invariant* signal processing, i.e., $xh(x) = h(x)x$ for $h(x) \in \mathcal{A}$. The CRT in (2) yields the spectral decomposition of the signal space $\mathcal{M}$ (i.e., the decomposition into irreducible $\mathcal{A}$-modules) and $\mathcal{P}_{b,\alpha}$ is the Fourier transform (in coordinate form) associated with this model.

For example, the signal model associated with the DFT is given by $\mathcal{A} = \mathcal{M} = \mathbb{C}[x]/(x^n - 1)$ and the finite z-transform

$$\Phi :\ \mathbf{s} \mapsto \sum_{0 \le \ell < n} s_\ell x^\ell. \tag{14}$$

The convolution associated with this model is the multiplication of polynomials $h(x), s(x)$ modulo $x^n - 1$. This is equivalent to circular convolution of the coordinate sequences $\mathbf{h}, \mathbf{s}$, as expected.

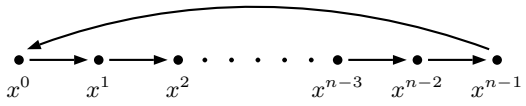
Similarly, the signal model associated with the DCT, type 3 is given by $\mathcal{A} = \mathcal{M} = \mathbb{C}[x]/T_n(x)$ and the finite T-transform $\Phi :\ \mathbf{s} \mapsto \sum_{0 \le \ell < n} s_\ell T_\ell(x)$.

The differences between signal models are best understood through their visualizations. The visualization is a graph that has the basis elements of b (fixed by Φ) as nodes and the edges are determined by the operation of the shift x on b. In algebraic terms, the adjacency matrix of this graph is $\phi(x)$, where ϕ is the representation of $\mathcal{A}$ afforded by $\mathcal{M}$ with basis b. Basic algebra asserts that $\phi(x)$ is diagonalized by the Fourier transform $\mathcal{P}_{b,\alpha}$.

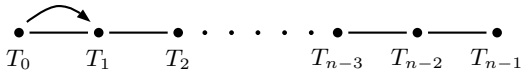
Two examples are shown in Fig. 2. Fig. 2(a) is a directed graph, since $x \cdot x^\ell = x^{\ell+1}$ and thus is called a *time* model. Fig. 2(b) is undirected since $x \cdot T_\ell = (T_{\ell-1} + T_{\ell+1})/2$ (see (49) in Appendix A) and hence is called a *space* model.

Intuitively, the visualization shows the structure imposed on $\mathbf{s} \in \mathbb{C}^n$ by the signal model including the boundary conditions. For example, Fig. 2(a) shows the cyclic boundary condition associated with the DFT arising from $x^n - 1 = 0$ (in $\mathcal{M}$), i.e., $x^n = 1 = x^0$.

Finite shift-invariant 2-D signal models. The above discussion is for finite 1-D signals $\mathbf{s} \in \mathbb{C}^n$ and readily extended to 2-D using polynomial algebras in



(a) Finite discrete time model associated with the DFT



(b) Finite discrete space model associated with the DCT, type 3

Fig. 2 Visualization of two finite signal models for $\mathbb{C}^n$.

two variables. For example, the signal model underlying the 2-D DFT is given by $\mathcal{A} = \mathcal{M} = \mathbb{C}[x, y]/\langle x^n - 1, y^n - 1\rangle$ and the the finite 2-D z-transform

$$\Phi : \mathbb{C}^{n\times n} \to \mathcal{M}, \quad \mathbf{s} \mapsto \sum_{0\le k,\ell} s_{k,\ell} x^k y^\ell.$$

Note that the signals $\mathbf{s}$ are now $n \times n$ 2-D arrays. In contrast to before, we have now two shift operators x and y and letting both operate on the basis $b = (x^k y^\ell \mid 0 \le k, \ell < n)$ yields as visualization the graph that has $\phi(x) + \phi(y)$ as adjacency matrix. It is shown in Fig. 1(a), without boundary conditions. Similarly, the signal model imposed by a 2-D DCT, type 3 is shown in Fig. 1(b), again without boundary conditions. These graphs are precisely the direct products of the graphs in Fig. 2 with themselves, respectively, since the 2-D signal models are separable.

The signal model underlying the DTT is given by

$$\mathcal{A} = \mathcal{M} = \mathbb{C}[x, y]/\langle T_{n,0}(x, y), T_{0,n}(x, y)\rangle$$

and

$$\Phi : \mathbb{C}^{n\times n} \to \mathcal{M}, \quad \mathbf{s} \mapsto \sum_{0\le k,\ell} s_{k,\ell} T_{k,\ell}(x, y)$$

and is nonseparable. The operation of the shifts x and y on the $T_{k,\ell}$ follows from (54) (in Appendix B) and creates the spatial hexagonal structure shown in Fig. 1(c) without boundary conditions.

In summary, the DTT is the analog of the DCT for 2-D signals on a finite hexagonal lattice.

3 Cooley-Tukey Type Algorithms

The algebraic signal processing theory shows that polynomial algebras provide the structure for finite shift-invariant signal processing (see the brief discussion in Section 2.4) and identifies the polynomial algebras associated with many of the existing signal transforms. This makes all transforms to Fourier transforms in a

rigorous sense, namely for a suitably chosen signal model, and identifies the associated notions of spectrum, convolution, and many others. Besides that there is a second, crucial benefit in knowing the polynomial algebra underlying a transform: it provides the means to derive the transform's fast algorithms. Namely, instead of manipulating the transform itself to obtain a fast algorithm, we manipulate the underlying polynomial algebra.

The basic idea is simple. A transform decomposes a polynomial algebra via the CRT in (2) or (6). A fast algorithm is obtained by performing this decomposition *in steps*. In particular, we have shown in [7] that there is one decomposition theorem for polynomial algebras (in one variable) that spawns what we call general-radix Cooley-Tukey type algorithms for a large class of (1-D) transforms including the DFT (for which the standard Cooley-Tukey FFT is obtained), DCTs/DSTs, and the real DFT [32]. In this section we generalize this theorem to polynomial algebras in *two* variables, which makes it applicable to the DTT. Accordingly, the resulting algorithms are again "Cooley-Tukey type."

We start with introducing the matrix notation used.

Matrix notation. We use permutation matrices, most importantly for stride permutations. They are defined for integers n, m, where $m|n$, and are given by

$$L^n_m : i\frac{n}{m} + j \mapsto jm + i, \quad 0 \le j < \frac{n}{m},\ 0 \le i < m.$$

In other words, L^n_m is the transposition of an $n/m \times m$ matrix stored in a vector in row-major order.

Other permutation matrices P, are defined by their underlying permutation $\pi : i \mapsto \pi(i)$, $0 \le i < n$; namely, in row i of P there is precisely one non-zero entry 1 in the column $\pi(i)$.

We write $\mathrm{diag}(\alpha_0, \ldots, \alpha_{n-1})$ to denote an $n \times n$ diagonal matrix with diagonal entries $\alpha_0, \ldots, \alpha_{n-1}$.

For two matrices A and B, $A \oplus B$ is their (block diagonal) direct sum and $A \otimes B$ defined in (9) is their Kronecker or tensor product.

Special matrices used throughout the paper are the identity matrix I_m, the all-zero matrix 0_m, and

$$J_m = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix},\ Z_m = \begin{bmatrix} & & & 0 \\ & & 0 & 1 \\ & \iddots & \iddots & \\ 0 & 1 & & \end{bmatrix}.$$

Finally, as before, we denote the complex unit by i (in roman) to distinguish it from the summation index i (in italics).

3.1 Cooley-Tukey Type Algorithms: One Variable

We assume that a polynomial transform $\mathcal{P}_{b,\alpha}$ is given with underlying algebra $\mathcal{A} = \mathbb{C}[x]/p(x)$ and basis b. Then $\mathcal{P}_{b,\alpha}$ has a Cooley-Tukey type algorithm if $p(x) = q(r(x))$ decomposes into polynomials q and r as explained next.

We assume $\deg(q) = k$, $\deg(r) = m$, which implies $\deg(p) = n = km$. We denote the zeros of q with $\beta = (\beta_0, \ldots, \beta_{k-1})$. Now we can factor p in two steps as

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

Here, $\gamma_i = (\gamma_{i,0}, \ldots, \gamma_{i,m-1})$ are the zeros of $r(x) - \beta_i$; each $\gamma_{i,j}$ is of course also a zero α_k of p.

Using the CRT repeatedly, we obtain the following associated stepwise decomposition of $\mathcal{A}$:

$$\begin{aligned}
\mathbb{C}[x]/p(x) &\to \mathbb{C}[x]/q(r(x)) && (15)\\
&\to \bigoplus_{0 \le i < k} \mathbb{C}[x]/(r(x) - \beta_i) && (16)\\
&\to \bigoplus_{0 \le i < k} \bigoplus_{0 \le j < m} \mathbb{C}[x]/(x - \gamma_{i,j}) && (17)\\
&\to \bigoplus_{0 \le k < n} \mathbb{C}[x]/(x - \alpha_k). && (18)
\end{aligned}$$

This decomposition leads to a recursive factorization of the associated polynomial transform into a product of *four* sparse matrices corresponding to the *four* steps (15)–(18) as explained in [7,8].

Step 1: The first step (15) does not change the algebra but performs a base change B_n in $\mathbb{C}[x]/p(x)$ from the given basis b to a new basis b' which is defined by

$$\begin{aligned}
b' = (&r_0 q_0(r(x)), \ldots, r_{m-1} q_0(r(x)),\\
&\ldots\\
&r_0 q_{k-1}(r(x)), \ldots, r_{m-1} q_{k-1}(r(x))), \qquad (19)
\end{aligned}$$

where $d = (r_0, \ldots, r_{m-1})$ is a chosen basis for each of the $\mathbb{C}[x]/(r(x) - \beta_i)$ and $c = (q_0, \ldots, q_{k-1})$ is a chosen basis for $\mathbb{C}[y]/q(y)$.

Step 2: In (16), the CRT is applied to $\mathbb{C}[y]/q(y)$, $y = r(x)$. To obtain the associated matrix, every element of b' is reduced modulo $r(x) - \beta_i$ and expressed in the basis $(r_0, \ldots, r_{m-1})$. This yields the matrix $\mathcal{P}_{c,\beta} \otimes I_m$.

Step 3: In step (17), again the CRT is applied to each $\mathbb{C}[x]/(r(x) - \beta_i)$, decomposing it with $\mathcal{P}_{d,\gamma_i}$. Thus, the entire matrix is given by $\bigoplus_{0 \le i < k} \mathcal{P}_{d,\gamma_i}$.

Step 4: The final step (18) permutes the summands into the right order with a suitable permutation matrix P_n.

We summarize in the following theorem.

Theorem 1 (1-D Cooley-Tukey Type Algorithms) *Let $\mathcal{P}_{b,\alpha}$ be a polynomial transform for $\mathcal{A} = \mathbb{C}[x]/p(x)$ with basis b, where $p(x) = q(r(x))$ decomposes. Then, using the notation introduced above,*

$$\mathcal{P}_{b,\alpha} = P_n \Big(\bigoplus_{0 \le i < k} \mathcal{P}_{d,\alpha'_i} \Big) (\mathcal{P}_{c,\beta} \otimes I_m) B_n. \qquad (20)$$

Note that all four factors are guaranteed to be sparse except for B_n.

We illustrate this theorem by first deriving the general-radix Cooley-Tukey FFT, which also explains the name of Theorem 1. Then we apply it to the DCT, type 3. The latter derivation will be analogous to the algorithm derivation for the DTT shown later.

Example: DFT. The DFT is a polynomial transform for $\mathcal{A} = \mathbb{C}[x]/(x^n - 1)$ with basis $b = (1, x, \ldots, x^{n-1})$. Assuming that $n = km$, we have the decomposition $x^n - 1 = (x^m)^k - 1$, i.e., we set $q(x) = x^k - 1$, and $r(x) = x^m$. As bases we choose $c = (1, x, \ldots, x^{k-1})$ and $d = (1, x, \ldots, x^{m-1})$, i.e., $q_i = x^i$ and $r_j = x^j$.

Step 1: It turns out that $b' = b$, i.e., $B = I_n$.

Step 2: The partial decomposition is done by $\mathcal{P}_{c,\beta} \otimes I_m = \mathrm{DFT}_k \otimes I_m$. The smaller algebras in (16) are given by $\mathbb{C}[x]/(x^m - \omega_k^i)$.

Step 3: Each $\mathbb{C}[x]/(x^m - \omega_k^i)$ is decomposed by $\mathcal{P}_{d,\gamma_i} = \mathrm{DFT}_m \cdot \mathrm{diag}(\omega_n^{ij} \mid 0 \le j < m)$ as direct computation shows.

Step 4: It remains to determine the permutation matrix P_n. At this point we have the decomposition in (17), which takes the form

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

Hence, the remaining task is to reorder from $jk + i$ to $im + j$, where $0 \le i < k$ and $0 \le j < m$, to make the exponents consecutive. This implies that P_n is the stride permutation L_m^n. Overall, we obtain the radix-m decimation-in-frequency Cooley-Tukey FFT

$$\mathrm{DFT}_n = L_m^n (I_k \otimes \mathrm{DFT}_m) T_m^n (\mathrm{DFT}_k \otimes I_m), \tag{21}$$

where $T_m^n = \bigoplus_{0 \le i < k} \mathrm{diag}(\omega_n^{ij} \mid 0 \le j < m)$. Transposition of (21) yields the decimation-in-time version.

Example: DCT, type 3. The DCT, type 3, for input size n, is denoted with DCT-3$_n$. It is the polynomial transform for the algebra $\mathbb{C}[x]/T_n(x)$ with basis $b = (T_0, \ldots, T_{n-1})$ as shown in (4). We assume $n = km$. In this case, $T_n = T_k(T_m)$ indeed decomposes and yields general-radix Cooley-Tukey algorithms [7].

We will show the derivation of the radix-2 ($k = 2$) case since it is analogous to the more involved radix-2×2 algorithm for the DTT derived later. This radix-2 algorithm was originally discovered in iterative form (which avoids the definition of skew DCTs) in [15] and derived using implicitly polynomial algebras.

We set $q(x) = T_2(x)$, $r(x) = T_m(x)$, and choose bases $c = (T_0, T_1)$ and $d = (T_0, \ldots, T_{m-1})$. The decomposition now becomes:

$$\begin{aligned}
\mathbb{C}[x]/T_n(x) &\to \mathbb{C}[x]/T_2(T_m(x)) && (22)\\
&\to \mathbb{C}[x]/(T_m(x) - \cos\tfrac{\pi}{4}) \oplus \mathbb{C}[x]/(T_m(x) - \cos\tfrac{3\pi}{4}) && (23)\\
&\to \bigoplus_{0 \le i < 2} \bigoplus_{0 \le j < m} \mathbb{C}[x]/(x - \cos r_{i,j}\pi) && (24)\\
&\to \bigoplus_{0 \le k < n} \mathbb{C}[x]/(x - \cos\tfrac{(2k+1)\pi}{2n}). && (25)
\end{aligned}$$

(a) Line graph for DCT, type 3 (b) Decomposed line graph

Fig. 3 (a) The signal domain associated with the DCT-3_n for $n = 16$. Boundary conditions are omitted. (b) Radix-2 decomposition to compute the base change in the Cooley-Tukey algorithm.

Here, $r_{0,j} = \frac{4j+2-(-1)^j}{2n}$ and $r_{1,j} = \frac{4j+2+(-1)^j}{2n}$, which follows from the general factorization (for n even)

$$T_n - \cos r\pi = 2^{n-1} \prod_{0 \le \ell < n} (x - \cos r_\ell \pi), \tag{26}$$

where the list of the zeros, ordered by increasing angle normalized to the interval $[0, \pi]$, is determined by

$$(r_\ell)_{0 \le \ell < n} = \bigcup_{0 \le i < n/2} \left(\tfrac{r+2i}{n}, \tfrac{2-r+2i}{n} \right). \tag{27}$$

Now we start the algorithm derivation.

Step 1: The basis b' is given by

$$\begin{aligned} b' &= (T_0T_0(T_m(x)), \ldots, T_{m-1}T_0(T_m(x)), T_0T_1(T_m(x)), \ldots, T_{m-1}T_1(T_m(x))) \\ &= (T_{im-j}/2 + T_{im+j}/2 \mid 0 \le i < 2, 0 \le j < m), \end{aligned}$$

where we used the property $T_kT_n = (T_{n-k} + T_{n+k})/2$ (see (53) in Appendix A) which holds for all $k, n \in \mathbb{Z}$.

Unlike in the DFT case above, the base change (22) from b to b' is no longer trivial. To determine the exact form we have to express the elements of the basis $b = (T_0, \ldots, T_{n-1})$ as a linear combination of elements of b'. Accordingly, we first split the indices into a radix-two representation:

$$b = (T_{im+j} \mid 0 \le i < 2, 0 \le j < m),$$

where we use the lexicographic ordering on pairs $(i, j) = (0, 0), (0, 1), \ldots$. Correspondingly, we get a partition of $0, \ldots, n-1$ into two regions, namely the part in which $i = 0$ and the part in which $i = 1$ (see Fig. 3).

The base change is now computed as follows.
Region I: $T_j \in b'_n$ for $0 \le j < m$.
Region II:

$$T_{m+j} = \begin{cases} T_m \cdot T_0 & : j = 0, \\ 2T_mT_j - T_{m-j} & : j \neq 0. \end{cases}$$

Hence the base change B_n takes the form

$$B_n = (I_m \oplus \operatorname{diag}(1, 2, \ldots, 2)) \begin{bmatrix} I_m & -Z_m \\ 0_m & I_m \end{bmatrix}. \tag{28}$$

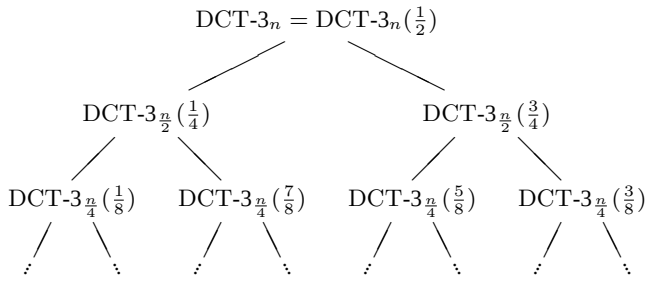


Fig. 4 The recursion tree of the radix-2 algorithm for the DCT, type 3. For input size $n = 2^d$, the depth of the tree is d.

Step 2: The partial decomposition in (23) is done with $\text{DCT-3}_2 \otimes I_m$.

Step 3: The complete decomposition in (24) is done with $\text{DCT-3}_m(\frac{1}{4}) \oplus \text{DCT-3}_m(\frac{3}{4})$, where we define $\text{DCT-3}_n(r)$ as the polynomial transform for the algebra $\mathbb{C}[x]/(T_n - \cos r\pi)$ with basis $(T_0, \ldots, T_{n-1})$ and the order of zeros shown in (27). We call $\text{DCT-3}_n(r)$ a *skew* DCT of type 3 [4].

Step 4: The final reordering of the zeros in (25) is done with the permutation

$$P_n = (I_{m/2} \otimes (I_2 \oplus J_2))L^n_m. \tag{29}$$

At this point we have the complete decomposition of DCT-3_n, namely, the computation is reduced to two skew DCTs of half the size. These, in turn, can be decomposed in exactly the same fashion, since $T_n - \cos r\pi$ decomposes if and only if T_n decomposes. In this decomposition both the base change B_n and the permutation P_n do not depend on r. The result is the following fully specified recursive radix-2 Cooley-Tukey algorithm for the DCT, type 3.

Theorem 2 (Radix-2 algorithm for DCT, type 3) *Let n be even. Then,*

$$\begin{aligned}\text{DCT-3}_n &= \text{DCT-3}_n(\tfrac{1}{2}),\\ \text{DCT-3}_n(r\pi) &= P_n \left(\text{DCT-3}_{\frac{n}{2}}(\tfrac{r}{2}) \oplus \text{DCT-3}_{\frac{n}{2}}(\tfrac{2-r}{2})\right)(\text{DCT-3}_2 \otimes I_{\frac{n}{2}})B_n,\end{aligned}$$

with base case

$$\text{DCT-3}_2(r) = \begin{bmatrix} 1 & \cos\frac{r}{2}\pi \\ 1 & -\cos\frac{r}{2}\pi \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \text{diag}(1, \cos r\pi).$$

The algorithm in Theorem 2 can be visualized by its associated recursion tree in Fig. 4, which shows the occurring skew DCTs.

By solving recurrences, the operations count of this algorithm for a two-power size n is determined as $A(n) = \frac{3}{2}n\log_2(n) - n + 1$ additions and $M(n) = \frac{1}{2}n\log_2(n)$ multiplications for a total of $2n\log_2(n) - n + 1 = O(n\log(n))$ operations.

3.2 Cooley-Tukey Type Algorithms: Two Variables

Now we extend Theorem 1 to polynomial algebras in two variables.

Let $\mathcal{A} = \mathbb{C}[x,y]/\langle p(x,y), q(x,y)\rangle$ and assume that the ideal $\langle p, q\rangle$ decomposes. This means that we can find bivariate polynomials $s,t,u,v \in \mathbb{C}[x,y]$ such that $p(x,y) = u(r(x,y), s(x,y))$ and $q(x,y) = v(r(x,y), s(x,y))$. Again, in this case a fast algorithm for the polynomial transform $\mathcal{P}_{b,\alpha}$ can be constructed by decomposing the algebra $\mathcal{A}$ in steps. For simplicity we assume that $\deg(p) = \deg(q) = n$, $\deg(u) = \deg(v) = k$, and $\deg(r) = \deg(s) = m$, which implies $n = km$. Denote the common zeros of the outer polynomials $u(x,y)$ and $v(x,y)$ by $\beta = ((\mu_i, \nu_i) \mid i = 0, \ldots, k^2 - 1)$. Next we define the common zeros $\gamma_i = ((\kappa_{i,j}, \lambda_{i,j}) \mid j = 0, \ldots, m^2 - 1)$ of the polynomials $r(x,y) - \mu_i$ and $s(x,y) - \nu_i$. Now, each $\gamma_{i,j} = (\kappa_{i,j}, \lambda_{i,j})$ is a common zero $\alpha_\ell = (\kappa_\ell, \lambda_\ell)$ of $p(x,y)$ and $q(x,y)$, where $\ell = 0, \ldots, n^2 - 1$. From the CRT we obtain the decomposition

$$\mathbb{C}[x,y]/\langle p(x,y), q(x,y)\rangle \to \mathbb{C}[x,y]/\langle u(r(x,y), s(x,y)), v(r(x,y), s(x,y))\rangle \quad (30)$$

$$\to \bigoplus_{0 \le i < k^2} \mathbb{C}[x,y]/\langle r(x,y) - \mu_i, s(x,y) - \nu_i\rangle \quad (31)$$

$$\to \bigoplus_{0 \le i < k^2} \bigoplus_{0 \le j < m^2} \mathbb{C}[x,y]/\langle (x - \kappa_{i,j}), (y - \lambda_{i,j})\rangle \quad (32)$$

$$\to \bigoplus_{0 \le \ell < n^2} \mathbb{C}[x,y]/\langle (x - \kappa_\ell), (y - \lambda_\ell)\rangle). \quad (33)$$

As in the univariate case this decomposition leads to a recursive factorization of $\mathcal{P}_{b,\alpha}$ into four matrices.

Step 1: In (30) we do not decompose the algebra but rather perform a base change to the basis b' defined by

$$\begin{aligned} b' = (&r_0 u_0(r(x,y), s(x,y)), \ldots, r_{m^2-1} u_0(r(x,y), s(x,y)),\\ &\ldots\ldots\\ &r_0 u_{k^2-1}(r(x,y), s(x,y)), \ldots, r_{m^2-1} u_{k^2-1}(r(x,y), s(x,y)), \end{aligned}$$

where the same basis $d = (r_0, \ldots, r_{m^2-1})$ is chosen for each of the algebras $\mathbb{C}[x,y]/\langle r(x,y) - \mu_i, s(x,y) - \nu_i\rangle$, and $c = (u_0, \ldots, u_{k^2-1})$ is the chosen basis for $\mathbb{C}[x,y]/\langle u(x,y), v(x,y)\rangle$. We call the base change matrix B_n.

Step 2: The coarse decomposition using the CRT is performed by the matrix $\mathcal{P}_{c,\beta} \otimes I_{m^2}$ as direct computation shows.

Step 3: The complete decomposition is done by a direct sum of polynomial transforms: $\bigoplus_{0 \le i < k^2} \mathcal{P}_{d,\gamma_i}$.

Step 4: A suitable permutation P_n maps the concatenation of the γ_i onto α.

Theorem 3 (2-D Cooley-Tukey Type Algorithms) *Let $\mathcal{P}_{b,\alpha}$ be a polynomial transform for $\mathcal{A} = \mathbb{C}[x]/\langle p(x,y), q(x,y)\rangle$ with basis b, and assume that $p(x) = u(r(x), s(x))$ and $q(x) = v(r(x,y), s(x,y))$ decompose. Then, using previous notation,*

$$\mathcal{P}_{b,\alpha} = P_n \Big(\bigoplus_{0 \le i < k^2} \mathcal{P}_{d,\gamma_i} \Big) (\mathcal{P}_{c,\beta} \otimes I_{m^2}) B_n. \quad (34)$$

As in Theorem 1, all four factors are guaranteed to be sparse except for B_n.

3.3 Discussion

In the above derivation of Cooley-Tukey type algorithms we emphasized the actual construction and introduced only as much algebra as needed to follow the derivation. In particular, only rather basic polynomial computations are sufficient to arrive at the results. However, it is also desirable to understand the underlying algebraic principles at work. Among other things, this enables a comparison to related work on fast Fourier transforms for groups or group algebras. We briefly discuss this in the following.

Assume $\mathcal{A} = \mathbb{C}[x]/p(x)$ with $p(x) = q(r(x))$. This implies that $y = r(x)$ spans a subalgebra $\mathcal{B} \leq \mathcal{A}$ that is equal to $\mathbb{C}[y]/q(y)$. Further, we can choose a transversal $d = (r_0, \ldots, r_{k-1})$, $\deg(r_i) = i$, of $\mathcal{B}$ in $\mathcal{A}$ such that

$$\mathcal{A} = r_0\mathcal{B} \oplus \ldots \oplus r_{k-1}\mathcal{B}. \tag{35}$$

If we view $\mathcal{A}$ as a regular $\mathcal{A}$-module and $\mathcal{B}$ as regular $\mathcal{B}$-module, then (35) shows that $\mathcal{A}$ is the induction of $\mathcal{B}$ to $\mathcal{A}$ with transversal d, written as (see [18])

$$\mathcal{A} \cong \mathcal{A} \otimes_{\mathcal{B}} \mathcal{B} = \mathcal{B} \uparrow_d \mathcal{A}.$$

First, this explains the basis b' in (19), which is compatible with the decomposition in (35). Namely, if $c = (q_0(y), \ldots, q_{m-1}(y))$ is a basis of $\mathcal{B}$, then the $r_i q_j(y)$ form a basis of $\mathcal{A}$.

Second, Theorem 1 derives a fast algorithm by decomposing the regular $\mathcal{A}$-module into a stepwise induction; the steps determine the factorization. The same procedure, applied to group algebras, is known to produce fast algorithms for Fourier transforms on groups. This technique was used by Beth in [19] (see also [33]) to explain the Cooley-Tukey FFT as a stepwise decomposition of the group algebra for the cyclic group; then, he generalized the same technique to arbitrary solvable groups (see also [24]). For these groups he also derived an explicit recursion formula that, not surprisingly, is in structure similar to the recursions in Theorems 1 and 3. Nonsolvable groups may require additional techniques (e.g., [23, 25]), even though they are still decomposable inductions. The decomposition of non-regular group modules that afford a monomial representation was studied in [34]. Again, the derived decomposition formula looks similar to (20).

This discussion is readily extended to the bivariate case and Theorem 3.

4 Cooley-Tukey Type Algorithm for the DTT

In this section we apply Theorem 3 to derive Cooley-Tukey type algorithms for the DTT. The derivation parallels the algorithm derivation for the DCT, type 3, except for more involved calculations. To emphasize the correspondence, we follow the exact same steps as in Section 3.1. The reader is invited to switch often between that and the following section to discover the similarities.

We use notation introduced in Appendix B. In particular, we set for simplicity $T_n(x, y) = T_{n,0}(x, y)$ and denote with $\overline{p}(x, y) = p(y, x)$ the polynomial with reversed arguments. Accordingly, $\overline{T}_n(x, y) = T_{0,n}(x, y)$.

4.1 Derivation of the Fast Algorithm

The $\mathrm{DTT}_{n\times n}$ is the polynomial transform for $\mathbb{C}[x,y]/\langle T_n(x,y), \overline{T}_n(x,y)\rangle$ with basis $b_n = (T_{k,\ell} \mid 0 \le k,\ell < n)$. We assume $n = km$, which implies that $\langle T_n, \overline{T}_n\rangle$ decomposes as shown in (60) in Appendix B:

$$T_n = T_k(T_m, \overline{T}_m),\ \overline{T}_n = \overline{T}_k(T_m, \overline{T}_m).$$

Using Theorem 3, it is hence clear that the DTT possesses a general radix algorithm. The arithmetic cost of this algorithm depends on the sparsity of the initial base change matrix B. We call the general algorithm radix-$k\times k$ to emphasize the two-dimensional character of the decomposition.

In the following we will focus on the radix-2×2 case ($k = 2$) and derive the algorithm in detail. Necessarily, we assume that $n = 2m$.

Preliminaries. Before we derive the decomposition of the algebra we introduce notation to simplify the representation of the occurring zeros. Namely, we introduce functions σ and τ as

$$\begin{aligned}\sigma(r,s) &= \tfrac{1}{3}(e^{2\pi \mathrm{i} r} + e^{2\pi \mathrm{i} s} + e^{-2\pi \mathrm{i}(r+s)}),\\ \tau(r,s) &= \tfrac{1}{3}(e^{-2\pi \mathrm{i} r} + e^{-2\pi \mathrm{i} s} + e^{2\pi \mathrm{i}(r+s)}),\end{aligned}$$

where $r, s \in [0,1)$. Note that $\sigma(r,s)$ is the complex conjugate of $\tau(r,s)$. These functions will play the same role as $\cos(r\pi) = \frac{1}{2}(e^{2\pi \mathrm{i} r} + e^{-2\pi \mathrm{i} r})$ in the DCT algorithm derivation in Section 3.1. In particular, we need an equivalent to the factorization in (26), which means we have to find the simultaneous zeros of the equations

$$T_n - \sigma(r,s) = \overline{T}_n - \tau(r,s) = 0.$$

If $u = e^{2\pi \mathrm{i} r}$, $v = e^{2\pi \mathrm{i} s}$, then because of the power forms of T_n and $\overline{T}_n$ (see (57) in Appendix B), the n^2 solutions are (in u, v parameterization) given by $\{(u_i, v_j) = (e^{2\pi \mathrm{i} \frac{r+i}{n}}, e^{2\pi \mathrm{i} \frac{s+j}{n}}) \mid 0 \le i,j < n\}$. The corresponding n^2 solutions in x, y parameterization are now readily obtained by applying σ and τ. We write the result as an intersection of ideals in the following lemma.

Lemma 1 *Using previous notation,*

$$\langle T_n - \sigma(r,s), \overline{T}_n - \tau(r,s)\rangle = \bigcap_{0\le i,j<n} \langle x - \sigma(\tfrac{r+i}{n}, \tfrac{s+j}{n}), y - \tau(\tfrac{r+i}{n}, \tfrac{s+j}{n})\rangle. \quad (36)$$

In the special case $r = 0$, $s = 1/3$, we have $\sigma(r,s) = \tau(r,s) = 0$ and Lemma 1 yields the zeros of $T_n = \overline{T}_n = 0$ shown before and used to define the DTT.

Algorithm derivation. Following the notation from Section 3.2, we set $u = T_2$, $v = \overline{T}_2$, $r = T_m$, $s = \overline{T}_m$ and choose bases $c = (T_{0,0}, T_{0,1}, T_{1,0}, T_{1,1})$ and $d = (T_{k,\ell} \mid 0 \le k,\ell < m)$. The decomposition, following (30)-(33), now

becomes:

$$
\begin{aligned}
& \mathbb{C}[x,y]/\langle T_n, \overline{T}_n\rangle \\
\to\ & \mathbb{C}[x,y]/\langle T_2(T_m,\overline{T}_m), \overline{T}_2(T_m,\overline{T}_m)\rangle && (37)\\
\to\ & \mathbb{C}[x,y]/\langle T_m - \sigma(0,\tfrac{1}{6}), \overline{T}_m - \tau(0,\tfrac{1}{6})\rangle \\
& \oplus \mathbb{C}[x,y]/\langle T_m - \sigma(0,\tfrac{2}{3}), \overline{T}_m - \tau(0,\tfrac{2}{3})\rangle \\
& \oplus \mathbb{C}[x,y]/\langle T_m - \sigma(\tfrac{1}{2},\tfrac{1}{6}), \overline{T}_m - \tau(\tfrac{1}{2},\tfrac{1}{6})\rangle \\
& \oplus \mathbb{C}[x,y]/\langle T_m - \sigma(\tfrac{1}{2},\tfrac{2}{3}), \overline{T}_m - \tau(\tfrac{1}{2},\tfrac{2}{3})\rangle \\
& = \\
& \mathbb{C}[x,y]/\langle T_m - \tfrac{2}{3}, \overline{T}_m - \tfrac{2}{3}\rangle \\
& \oplus \mathbb{C}[x,y]/\langle T_m, \overline{T}_m\rangle \\
& \oplus \mathbb{C}[x,y]/\langle T_m - \tfrac{2}{3}\omega_3, \overline{T}_m - \tfrac{2}{3}\omega_3^2\rangle \\
& \oplus \mathbb{C}[x,y]/\langle T_m - \tfrac{2}{3}\omega_3^2, \overline{T}_m - \tfrac{2}{3}\omega_3\rangle && (38)\\
\to\ & \bigoplus_{0\le i,j<m} \mathbb{C}[x,y]/\langle x - \alpha_{2i,2j}, y - \beta_{2i,2j}\rangle \\
& \oplus \bigoplus_{0\le i,j<m} \mathbb{C}[x,y]/\langle x - \alpha_{2i,2j+1}, y - \beta_{2i,2j+1}\rangle && (39)\\
& \oplus \bigoplus_{0\le i,j<m} \mathbb{C}[x,y]/\langle x - \alpha_{2i+1,2j}, y - \beta_{2i+1,2j}\rangle \\
& \oplus \bigoplus_{0\le i,j<m} \mathbb{C}[x,y]/\langle x - \alpha_{2i+1,2j+1}, y - \beta_{2i+1,2j+1}\rangle && (40)\\
\to\ & \bigoplus_{0\le i,j<m} \mathbb{C}[x,y]/\langle x - \alpha_{i,j}, y - \beta_{i,j}\rangle. && (41)
\end{aligned}
$$

Here we have used the notation $\alpha_{i,j} = \sigma(\frac{i}{n}, \frac{j}{n})$ and $\beta_{i,j} = \tau(\frac{i}{n}, \frac{j}{n})$ for the simultaneous zeros of T_n and $\overline{T}_n$. Now we start the algorithm derivation.

Step 1: The basis b'_n is given by

$$
\begin{aligned}
b'_n = (& T_{0,0}T_{0,0}(T_m,\overline{T}_m), \dots, T_{m-1,m-1}T_{0,0}(T_m,\overline{T}_m),\\
& T_{0,0}T_{0,1}(T_m,\overline{T}_m), \dots, T_{m-1,m-1}T_{0,1}(T_m,\overline{T}_m),\\
& T_{0,0}T_{1,0}(T_m,\overline{T}_m), \dots, T_{m-1,m-1}T_{1,0}(T_m,\overline{T}_m),\\
& T_{0,0}T_{1,1}(T_m,\overline{T}_m), \dots, T_{m-1,m-1}T_{1,1}(T_m,\overline{T}_m))
\end{aligned}
$$

As for the DCT, type 3, the base change $B_{n\times n}$ from b_n to b'_n in (37) is nontrivial. Since the exact derivation of $B_{n\times n}$ is rather involved, we defer it to Section 4.2.

Step 2: The partial decomposition in (38) is done with $\mathrm{DTT}_{2\times 2}\otimes I_{m^2}$.

Step 3: The complete decomposition in (40) is done with

$$\mathrm{DTT}_{\frac{n}{2}\times\frac{n}{2}}(0,\tfrac{1}{6}) \oplus \mathrm{DTT}_{\frac{n}{2}\times\frac{n}{2}}(0,\tfrac{2}{3}) \oplus \mathrm{DTT}_{\frac{n}{2}\times\frac{n}{2}}(\tfrac{1}{2},\tfrac{1}{6}) \oplus \mathrm{DTT}_{\frac{n}{2}\times\frac{n}{2}}(\tfrac{1}{2},\tfrac{2}{3}),$$

where for $r, s \in [0,1)$ we define the *skew* transforms $\mathrm{DTT}_{n\times n}(r,s)$ as the polynomial transform for

$$\mathbb{C}[x,y]/\langle T_n - \sigma(r,s), \overline{T}_n - \tau(r,s)\rangle$$

with basis b_n and order the zeros as $((\alpha_{i,j}, \beta_{i,j}) \mid 0 \le i, j < n)$, where we use lexicographic ordering on the pairs $(i, j) = (0, 0), (0, 1), \ldots$.

Step 4: Inspection shows that the final reordering of the zeros in (41) is done with the permutation

$$P_{n\times n} = L^{n^2}_{n^2/2}(L^{2n}_n \otimes I_{n/2}). \tag{42}$$

At this point we have the complete decomposition of $\mathrm{DTT}_{n\times n}$, namely, the computation is reduced to four skew DTTs, each of one quarter of the size. These can be decomposed in exactly the same fashion, since $\langle T_n - \sigma(r,s), \overline{T}_n - \tau(r,s)\rangle$ decomposes if and only if $\langle T_n, \overline{T}_n\rangle$ decomposes. In this decomposition the permutation $P_{n\times n}$ does not depend on r, s. However, in contrast to the DCT, type 3, the base change for the skew DTT depends on r and s: $B_{n\times n} = B_{n\times n}(r, s)$. We derive it in Section 4.2. Here, we only note that it is sparse and its arithmetic cost is $O(n^2)$.

In summary, we get the fully specified recursive radix-2×2 Cooley-Tukey algorithm for the DTT.

Theorem 4 (Radix-2×2 algorithm for DTT) *Let n be even. Then,*

$$\mathrm{DTT}_{n\times n} = \mathrm{DTT}_{n\times n}(0, \tfrac{1}{3}), \tag{43}$$

$$\begin{aligned}\mathrm{DTT}_{n\times n}(r,s) = P_{n\times n}\big(&\mathrm{DTT}_{\frac{n}{2}\times\frac{n}{2}}(\tfrac{r}{2}, \tfrac{s}{2}) \oplus \mathrm{DTT}_{\frac{n}{2}\times\frac{n}{2}}(\tfrac{r}{2}, \tfrac{s+1}{2})\\ &\oplus \mathrm{DTT}_{\frac{n}{2}\times\frac{n}{2}}(\tfrac{r+1}{2}, \tfrac{s}{2}) \oplus \mathrm{DTT}_{\frac{n}{2}\times\frac{n}{2}}(\tfrac{r+1}{2}, \tfrac{s+1}{2})\big)\\ &(\mathrm{DTT}_{2\times2} \otimes I_{\frac{n^2}{4}})B_{n\times n}(r,s),\end{aligned} \tag{44}$$

with $B_{n\times n}(r, s)$ specified in Theorem 6 in Section 4.2. The base cases are

$$\mathrm{DTT}_{2\times2}(r,s) = \begin{bmatrix} 1 & \sigma(\frac{r}{2}, \frac{s}{2}) & \tau(\frac{r}{2}, \frac{s}{2}) & \frac{1}{2}\big(3\sigma(\frac{r}{2}, \frac{s}{2})\tau(\frac{r}{2}, \frac{s}{2}) - 1\big)\\ 1 & \sigma(\frac{r}{2}, \frac{s+1}{2}) & \tau(\frac{r}{2}, \frac{s+1}{2}) & \frac{1}{2}\big(3\sigma(\frac{r}{2}, \frac{s+1}{2})\tau(\frac{r}{2}, \frac{s+1}{2}) - 1\big)\\ 1 & \sigma(\frac{r+1}{2}, \frac{s}{2}) & \tau(\frac{r+1}{2}, \frac{s}{2}) & \frac{1}{2}\big(3\sigma(\frac{r+1}{2}, \frac{s}{2})\tau(\frac{r+1}{2}, \frac{s}{2}) - 1\big)\\ 1 & \sigma(\frac{r+1}{2}, \frac{s+1}{2}) & \tau(\frac{r+1}{2}, \frac{s+1}{2}) & \frac{1}{2}\big(3\sigma(\frac{r+1}{2}, \frac{s+1}{2})\tau(\frac{r+1}{2}, \frac{s+1}{2}) - 1\big)\end{bmatrix}.$$

The recursion tree associated with this algorithm is visualized in Fig. 5 and shows the occurring skew DTTs.

Assuming that $B_{n\times n}$ requires only $O(n^2)$ operations, as shown later, it is already clear that the algorithm has $O(n^2 \log(n))$ runtime. We give more detailed operation counts in Section 4.3.

It is straightforward to derive a general radix-$k \times k$ Cooley-Tukey algorithm for the DTT following Theorem 3. We give the result in the following theorem without explicitly computing B and P.

Theorem 5 (Radix-$k\times k$ algorithm for DTT) *Let $n \ge 2$ and let $k|n$. Then,*

$$\mathrm{DTT}_{n\times n} = \mathrm{DTT}_{n\times n}(0, \tfrac{1}{3}), \tag{45}$$

$$\begin{aligned}\mathrm{DTT}_{n\times n}(r,s) = P^{(k)}_{n\times n} \bigoplus_{0\le i,j<k} &\mathrm{DTT}_{n/k\times n/k}\left(\frac{r+i}{k}, \frac{s+j}{k}\right)\\ &\big(\mathrm{DTT}_{k\times k} \otimes I_{n^2/k^2}\big)\, B^{(k)}_{n\times n}(r,s)\end{aligned} \tag{46}$$

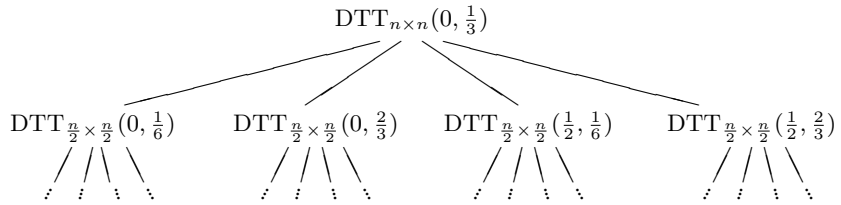


Fig. 5 The recursion tree of the radix-2×2 algorithm for the DTT. For $n=2^d$, the depth of the tree is d.

with suitably defined permutation matrix $P^{(k)}_{n\times n}$, base change matrix $B^{(k)}_{n\times n}(r,s)$, and base cases $\mathrm{DTT}_{k\times k}(r,s)$.

4.2 The Base Change for the Recursion Step

In this section we derive the precise form of the base change matrix $B_{n\times n}(r,s)$ used in Theorem 4. As mentioned before, this matrix corresponds to the base change from b_n to

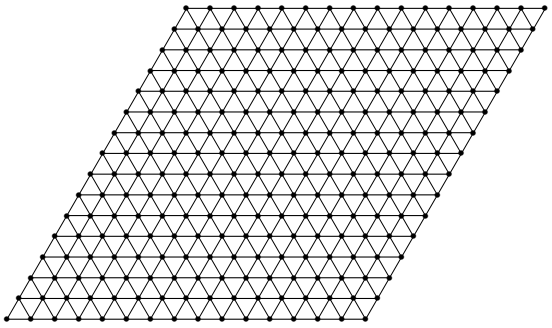
$$\begin{aligned} b'_n = (&T_{0,0}T_{0,0}(T_m,\overline{T}_m),\ldots,T_{m-1,m-1}T_{0,0}(T_m,\overline{T}_m),\\ &T_{0,0}T_{0,1}(T_m,\overline{T}_m),\ldots,T_{m-1,m-1}T_{0,1}(T_m,\overline{T}_m),\\ &T_{0,0}T_{1,0}(T_m,\overline{T}_m),\ldots,T_{m-1,m-1}T_{1,0}(T_m,\overline{T}_m),\\ &T_{0,0}T_{1,1}(T_m,\overline{T}_m),\ldots,T_{m-1,m-1}T_{1,1}(T_m,\overline{T}_m)) \end{aligned}$$

In other words we have to express every element $T_{k,\ell}\in b_n$ as a linear combination of the elements in b'_n; the coefficient vectors obtained this way are the columns of $B_{n\times n}(r,s)$. In particular, we will see that the base change matrix is sparse.

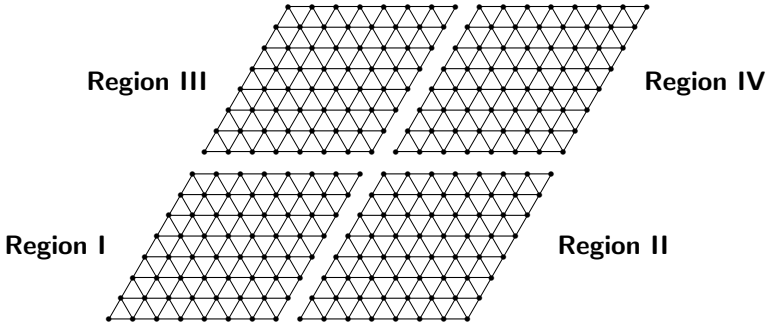
Theorem 6 *Let $n=2m$ and let $0\le k,\ell<m$. Then the following equations define the base change matrix $B_{n\times n}(r,s)$. The special case $B_{n\times n}$ is obtained by setting $r=0$ and $s=\frac{1}{3}$. The following division into regions is according to Fig. 6 and parallels Fig. 3 for the DCT, type 3.*

Region I: $T_{k,\ell}\in b'_n$ for $0\le k,\ell<m$.
Region II:

$$T_{m+k,\ell}=\begin{cases} T_{m,0} & :\ k=\ell=0\\ \frac{3}{2}T_{m,0}T_{0,\ell}-\frac{1}{2}T_{m-\ell,0} & :\ k=0,\ \ell\neq 0,\\ 3T_{m,0}T_{k,0}-2T_{m-k,k} & :\ \ell=0,\ k\neq 0,\\ 3T_{m,0}T_{k,\ell}-T_{m-\ell-k,k} & \\ \quad -T_{m-k,k+\ell} & :\ k+\ell<m; k,\ell\neq 0,\\ 3T_{m,0}T_{k,\ell}-\frac{3}{2}T_{0,m}T_{\ell,0}-\frac{1}{2}T_{0,k} & :\ k+\ell=m,\\ 3T_{m,0}T_{k,\ell}-3T_{0,m}T_{m-k,k+\ell-m} & \\ \quad +T_{\ell,2m-k-\ell} & :\ k+\ell>m. \end{cases}$$



(a) Hexagonal grid



(b) Decomposed hexagonal grid

Fig. 6 (a) The signal domain associated with the DTT for $n \times n = 16 \times 16$. Boundary conditions are omitted. (b) Radix-2×2 decomposition to compute the base change matrix in the Cooley-Tukey algorithm.

Region III: The representation of polynomials in this region is derived from Region II by applying $T_{k,m+\ell} = \overline{T}_{m+\ell,k}$. For completeness, we list them.

$$T_{k,m+\ell} = \begin{cases} T_{0,m} \ : \ k = \ell = 0 \\ \frac{3}{2}T_{0,m}T_{k,0} - \frac{1}{2}T_{0,m-k} \ : \ k = 0, \ \ell \neq 0, \\ 3T_{0,m}T_{0,\ell} - 2T_{\ell,m-\ell} \ : \ \ell = 0, \ k \neq 0, \\ 3T_{0,m}T_{k,\ell} - T_{\ell,m-\ell-k} - T_{k+\ell,m-\ell} \ : \ k+\ell < m; k, \ell \neq 0, \\ 3T_{0,m}T_{k,\ell} - \frac{3}{2}T_{m,0}T_{0,k} - \frac{1}{2}T_{\ell,0} \ : \ k+\ell = m, \\ 3T_{0,m}T_{k,\ell} - 3T_{m,0}T_{k+\ell-m,m-\ell} + T_{2m-k-\ell,k} \ : \ k+\ell > m. \end{cases}$$

Region IV: Some of the entries corresponding to basis elements from this region depend on the parameters r, s that define the skew transform. We let $\alpha = \sigma(r, s)$

and $\beta = \tau(r,s)$ and describe the dependence in terms of α and β:

$$T_{m+k,m+\ell} = \begin{cases} T_{m,m} : \ k = \ell = 0, \\ 3T_{m,m}T_{0,\ell} - 3T_{m,0}T_{\ell,m-\ell} + T_{0,\ell} : \ k = 0, \ \ell \neq 0, \\ 3T_{m,m}T_{k,0} - 3T_{0,m}T_{m-k,k} + T_{k,0} : \ \ell = 0, \ k \neq 0, \\ \begin{array}{r} 6T_{m,m}T_{k,\ell} - T_{m-\ell,m-k} + 2T_{k,\ell} \\ -3T_{m,0}T_{k+\ell,m-\ell} - 3T_{0,m}T_{m-k,k+\ell} \end{array} : \ k+\ell < m, \\ \begin{array}{r} 6T_{m,m}T_{k,\ell} + T_{k,\ell} - \frac{3}{2}T_{m,0}T_{\ell,0} \\ -\frac{3}{2}T_{0,m}T_{0,k} - \frac{3}{2}\alpha T_{0,k} - \frac{3}{2}\beta T_{\ell,0} \end{array} : \ k+\ell = m, \\ \begin{array}{r} 6T_{m,m}T_{k,\ell} + 2T_{k,\ell} \\ +3T_{m,0}T_{2m-\ell-k,k} - 3T_{m,0}T_{m-k,\ell+k-m} \\ +3T_{0,m}T_{\ell,2m-\ell-k} - 3T_{0,m}T_{\ell+k-m,m-\ell} \\ -(3\alpha+3\beta+1)T_{m-k,m-k} \end{array} : \ k = \ell = \frac{2}{3}m, \\ \begin{array}{r} 6T_{m,m}T_{k,\ell} + 3T_{m,0}T_{2m-\ell-k,k} + 2T_{k,\ell} \\ -T_{m-\ell,m-k} - 3T_{m,0}T_{m-k,\ell+k-m} \\ +3T_{0,m}T_{\ell,2m-\ell-k} - 3T_{0,m}T_{\ell+k-m,m-\ell} \\ -3\alpha T_{k+\ell-m,m-\ell} - 3\beta T_{m-k,k+\ell-m} \end{array} : \ k+\ell > m. \end{cases}$$

Proof: We focus on one particular case: we show that the entry $T_{m+k,\ell}$ for a polynomial in region II is given by the claimed formula in the case $k+\ell > m$. The other cases are shown analogously. We have to express every polynomial $T_{m+k,\ell}$, $0 \leq k, \ell < m$, in region II, in the basis b'_n. First note that using property (59) from Appendix B we obtain

$$T_{m,0}T_{k,\ell} = \tfrac{1}{3}(T_{m+k,\ell} + T_{k,\ell-m} + T_{k-m,\ell+m}),$$

which implies

$$T_{m+k,\ell} = 3T_{m,0}T_{k,\ell} - T_{k,\ell-m} - T_{k-m,\ell+m}. \tag{47}$$

This is an expansion of $T_{m+k,\ell}$ into polynomials which are not all in the basis b'_n. Thus, we rewrite the second and third polynomial on the right hand side of (47) using (58). We obtain

$$T_{k,\ell-m} = T_{k+\ell-m,m-\ell} = \begin{cases} T_{k+\ell-m,m-\ell} : k+\ell \geq m, \\ T_{m-\ell-k,k} : k+\ell < m. \end{cases}$$

Note that in both cases the given polynomial shown on the right hand side is an element of b'_n since it lies in region I.

Next, we modify the term $T_{k-m,\ell+m}$ in (47). Again, we distinguish the two cases $k+\ell < m$ and $k+\ell \geq m$. If $k+\ell < m$, we can use the rule $T_{k-m,\ell+m} = T_{m-k,k+\ell} \in b'_n$ and get

$$T_{m+k,\ell} = 3T_{m,0}T_{k,\ell} - T_{m-k-\ell,k} - T_{m-k,k+\ell}, \tag{48}$$

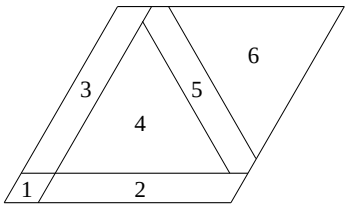


Fig. 7 Subdivision of each region in Fig. 6(b) into six subregions to compute the arithmetic cost of the base change.

provided that $0 \leq k + \ell < m$. Hence, we can assume that $k + \ell \geq m$. We define $\mu = k + \ell - m$ and $\nu = m - k$. Note that $0 \leq \mu, \nu \leq m$ and that $0 \leq \mu + \nu < m$. Hence, we can use (48) as follows:

$$\begin{aligned} T_{k-m,\ell+m} &= \overline{T}_{k+\ell,m-k} \\ &= \overline{T}_{m+\mu,\nu} \\ &= 3\overline{T}_{m,0}T_{\mu,\nu} - \overline{T}_{m-\mu-\nu,\mu} - \overline{T}_{m-\mu,\mu+\nu} \\ &= 3T_{0,m}T_{\nu,\mu} - T_{\mu,m-\mu-\nu} - T_{\mu+\nu,m-\mu}, \end{aligned}$$

where the last line is a decomposition into elements of b'_n. Substituting back the values for μ and ν yields the formula

$$T_{k-m,\ell+m} = 3T_{0,m}T_{m-k,k+\ell-m} - T_{k+\ell-m,m-\ell} - T_{\ell,2m-k-\ell}.$$

Hence, we have found expressions for all terms on the right hand side of (47) and can compute the resulting expansion as

$$\begin{aligned} T_{m+k,\ell} &= 3T_{m,0}T_{k,\ell} - T_{k,\ell-m} - T_{k-m,\ell+m} \\ &= 3T_{m,0}T_{k,\ell} - T_{k+\ell-m,m-\ell} - 3T_{0,m}T_{m-k,k+\ell-m} \\ &\quad + T_{k+\ell-m,m-\ell} + T_{\ell,2m-k-\ell} \\ &= 3T_{m,0}T_{k,\ell} - 3T_{0,m}T_{m-k,k+\ell-m} + T_{\ell,2m-k-\ell} \end{aligned}$$

as claimed for the case $k + \ell > m$ in the theorem. The other cases $k = 0$, $\ell = 0$, and $k + \ell = m$ for polynomials in region II arise as special cases in which the linear combinations can be further simplified. □

4.3 Arithmetic Cost

In this subsection we determine the arithmetic cost of the recursive DTT algorithm in Theorem 4. Our cost measure is *complex* additions and multiplications, were multiplications by ± 1 are not counted. We assume a $\mathrm{DTT}_{n \times n}$, where n is a two-power. The main task is to determine the number of operations incurred by the base change matrix B in Theorem 6.

Cost for the base change. To count the number of arithmetic operations necessary for the base change B we have to analyze the number and the values of the non-zero entries in each row of B. This information is obtained in a straightforward way from the description of B in Theorem 6.

We start by counting for each of the regions I–IV in Fig. 6 the number of additions (adds) and multiplications (mults) necessary for the base change $B_{n\times n}(r,s)$ in Theorem 4. Each region is further subdivided into the six regions shown in Fig. 7. Fig. 8 summarizes the coefficients which occur whenever a particular row of $B_{n\times n}(r,s)$ corresponds to a point in one of these regions. Here $R_{\mathrm{I},1},\ldots,R_{\mathrm{I},6}$ denote the 6 subregions of region I, and so on.

Next, we consider how many additions and multiplications are needed for the matrix-vector product. In computing this we also perform obvious simplifications. For example, for computing the product $(3/2,3/2,1,-1)\cdot(u,v,w,x)^t$ we obtain a count of three additions, and one multiplication (since the two multiplications with the same scalar $3/2$ can be simplified).

For the polynomials in region I we obtain the following table, where each entry denotes the number of complex additions and multiplications.

Region	Occurrences	Adds	Mults
$R_{\mathrm{I},1}$	1	0	0
$R_{\mathrm{I},2}$	$m-1$	$4\ (3)$	$2\ (1)$
$R_{\mathrm{I},3}$	$m-1$	$4\ (3)$	$2\ (1)$
$R_{\mathrm{I},4}$	$(m-1)(m-2)/2$	$6\ (4)$	$3\ (1)$
$R_{\mathrm{I},5}$	$m-1$	3	1
$R_{\mathrm{I},6}$	$(m-1)(m-2)/2$	6	1

The numbers in parentheses are the corresponding count for the non-skew DTT, i.e., for the special case $B_{n\times n}(0,\frac{1}{3})$. Similarly, we compute the number of additions and multiplications for regions II and III which are summarized in the following table. Note that the numbers of region II and region III are identical. Hence, only the values for region II are given.

Region	Occurrences	Adds	Mults
$R_{\mathrm{II},1}$	1	0	0
$R_{\mathrm{II},2}$	$m-1$	1	2
$R_{\mathrm{II},3}$	$m-1$	1	2
$R_{\mathrm{II},4}$	$(m-1)(m-2)/2$	2	2
$R_{\mathrm{II},5}$	$m-1$	1	2
$R_{\mathrm{II},6}$	$(m-1)(m-2)/2$	2	2

Finally, the coefficients for region IV turn out to be very simple; no additions and only one multiplication for each entry has to be performed.

Region	Occurrences	Adds	Mults
$R_{\mathrm{IV},1}$	1	0	0
$R_{\mathrm{IV},2}$	$m-1$	0	1
$R_{\mathrm{IV},3}$	$m-1$	0	1
$R_{\mathrm{IV},4}$	$(m-1)(m-2)/2$	0	1
$R_{\mathrm{IV},5}$	$m-1$	0	1
$R_{\mathrm{IV},6}$	$(m-1)(m-2)/2$	0	1

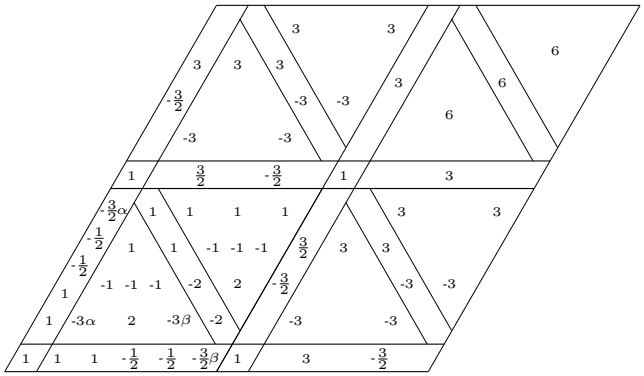


Fig. 8 A visualization of the coefficients of the base change $B_{n\times n}(r,s)$ that is useful for determining its arithmetic cost. The numbers indicate the coefficients of the rows of $B_{n\times n}(r,s)$, i.e., they indicate with which coefficient an element of the basis b'_n occurs when this element is used to express the basis elements in b_n. The division of the grid into four regions is as in Fig. 6(b). Those regions are then further subdivided into six subregions each as in Fig. 7. All basis elements of b'_n which appear in the same subregion occur with the same coefficients, namely the numbers written into the respective subregion. For instance, the figure shows that the rows of $B_{n\times n}(r,s)$ corresponding to any basis elements in region $R_{\mathrm{I},6}$ have precisely 7 non-zero elements $1,1,1,-1,-1,-1,2$. To implement the vector product with this row, 1 multiplication and 6 additions are required.

From the above we determine the total number of additions for the matrix-vector multiplication $y = B_{n\times n}(\alpha,\beta)x$ as

$$17(m-1) + 20(m-1)(m-2)/2 = 10m^2 - 13m + 3,$$

and the number of multiplications as

$$7m^2 - m - 6.$$

For the special (non-skew) case $B_{n\times n}(0,\frac{1}{3})$ the count is slightly less, namely $9m^2 - 12m + 3$ additions and $6m^2 - 6$ multiplications.

Cost for the base cases. Next, we consider the base case for the recursion, i. e., the operations needed for a skew-DTT of size 4×4. Since $T_{0,0} = 1$, each skew $\mathrm{DTT}_{2\times 2}(r,s)$ has in the first column only 1's. Thus, the arithmetic cost is at most 12 additions and 9 multiplications.

For the non-skew $\mathrm{DTT}_{2\times 2}$ in (13) the count is obviously less, but we can do even better by generating an algorithm using the algorithm discovery tool provided by AREP [35,36]. Specifically, applying the AREP function MatrixDecompositionByMonMonSymmetry to $\mathrm{DTT}_{2\times 2}$ produces the following factorization (the dots represent zero entries):

$$\begin{bmatrix} 1 & \frac{2}{3} & \frac{2}{3} & \frac{1}{6} \\ 1 & 0 & 0 & -\frac{1}{2} \\ 1 & \frac{2}{3}\omega_3^2 & \frac{2}{3}\omega_3 & \frac{1}{6} \\ 1 & \frac{2}{3}\omega_3 & \frac{2}{3}\omega_3^2 & \frac{1}{6} \end{bmatrix} = \begin{bmatrix} \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 \\ 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \end{bmatrix} \cdot \begin{bmatrix} 1 & \frac{2}{3}\omega_3^2 & \frac{2}{3}\omega_3 & \cdot \\ 1 & \frac{2}{3} & \frac{2}{3} & \cdot \\ 1 & \frac{2}{3}\omega_3 & \frac{2}{3}\omega_3^2 & \cdot \\ \cdot & \cdot & \cdot & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & \cdot & \cdot & \frac{1}{6} \\ \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \\ 1 & \cdot & \cdot & -\frac{1}{2} \end{bmatrix}.$$

This shows that a matrix-vector multiplication with $\mathrm{DTT}_{2\times 2}$ can be perfomed using 8 additions and 7 multiplications.

Cost of the DTT algorithm. Now we area ready to determine the overall cost of the algorithm in Theorem 4.

Theorem 7 *Let $n = 2^k$, where $k \geq 1$. Then, using Theorem 4, the discrete triangle transform $\mathrm{DTT}_{n\times n}$ can be computed using at most*

$$A(n) = \frac{11}{2}n^2 \log n - \frac{43}{6}n^2 + \frac{15}{2}n - \frac{1}{3},$$
$$M(n) = 4n^2 \log n - \frac{7}{2}n^2 + \frac{3}{2}n + 2$$

complex additions and multiplications, respectively. In particular, the complexity of the DTT is $O(n^2 \log(n))$.

Proof: We first determine an upper bound on the number of operations for an arbitrary skew $\mathrm{DTT}_{n\times n}(r, s)$ computed using the recursion (44) given in Theorem 4, from left to right. The cost for the base change $B_{n\times n}(r, s)$ was determined above. The matrix $(\mathrm{DTT}_{2\times 2}(r, s) \otimes I_{m^2})$ incurs $12m^2$ additions and $9m^2$ multiplications. Next, the four skew DTTs are computed recursively and P does not incur operations. We obtain the following recurrences for additions and multiplications ($m = n/2$):

$$A_s(n) = 4A_s(m) + 22m^2 - 13m + 3,$$
$$M_s(n) = 4M_s(m) + 16m^2 - m - 6.$$

The base cases are $A_s(2) = 12$ and $M_s(2) = 9$ as explained above. The solutions are the cost of a skew $\mathrm{DTT}_{n\times n}$:

$$A_s(n) = \frac{11}{2}n^2 \log n - \frac{11}{2}n^2 + \frac{13}{2}n - 1,$$
$$M_s(n) = 4n^2 \log n - \frac{5}{2}n^2 + \frac{1}{2}n + 2.$$

Finally, we compute the cost of the non-skew $\mathrm{DTT}_{n\times n}$ and the cost of the skew DTTs computed above. Note that we take into account that in (44) one of the smaller (size $m \times m$) DTTs is not skew. We obtain the recurrences ($m = n/2$)

$$A(n) = A(m) + 3A_s(m) + 17m^2 - 12m + 3,$$
$$M(n) = M(m) + 3M_s(m) + 13m^2 - 6,$$

with initial conditions $A(2) = 8$ and $M(2) = 7$. Solving these recurrences yields the desired result. □

5 Conclusions

We presented a fast, $O(n^2 \log(n))$ algorithm for the discrete triangle transform (DTT) for input size $n \times n$. This shows that the DTT has the same arithmetic cost as other, separable, two-dimensional transforms. Similar to our previous work on

trigonometric 1-D transforms, we derived the algorithm not by lengthy matrix entry manipulations, but by a stepwise decomposition of the polynomial algebra associated to the transform.

In algebraic terms, as we briefly explained, this technique is the stepwise decomposition of the induction of modules (where the polynomial algebra is viewed as regular module) into irreducible modules. This technique produces recursive algorithms for many 1-D trigonometric transforms (associated with polynomial algebras in one variable), the 2-D DTT (associated with a polynomial algebra in two variables), and Fourier transforms for at least solvable groups (associated with group algebras). The DFT is contained in two of these classes since $\mathbb{C}[x]/(x^n - 1) = \mathbb{C}[Z_n]$ is equivalently a polynomial algebra or the group algebra for the cyclic group. The stepwise decomposition yields the Cooley-Tukey FFT (in recursive form) in this case. For this reason we term the entire class of algorithms based on the stepwise decomposition of inductions "Cooley-Tukey type," including the algorithm derived in this paper.

References

1. 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.
2. M. Püschel and J. M. F. Moura, "Algebraic signal processing theory: Foundation and 1-D time," submitted for publication, part of [3].
3. M. Püschel and J. M. F. Moura, "Algebraic signal processing theory," available at http://arxiv.org/abs/cs.IT/0612077, parts of this manuscript are submitted as [2] and [4].
4. M. Püschel and J. M. F. Moura, "Algebraic signal processing theory: 1-D space," submitted for publication, part of [3].
5. T. J. Rivlin, *The Chebyshev Polynomials*, Wiley Interscience, 1974.
6. T. Koornwinder, "Orthogonal polynomials in two variables which are eigenfunctions of two algebraically independent partial differential operators (part III)," *Indag. Math.*, vol. 36, pp. 357–369, 1974.
7. M. Püschel and J. M. F. Moura, "Algebraic signal processing theory: Cooley-Tukey type algorithms for DCTs and DSTs," *IEEE Transactions on Signal Processing*, to appear; a longer version is available at http://arxiv.org/abs/cs.IT/0702025.
8. 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.
9. M. Püschel and M. Rötteler, "Cooley-Tukey FFT like algorithm for the discrete triangle transform," in *Proc. 11th IEEE DSP Workshop*, 2004, pp. 158–162.
10. R. M. Mersereau, "The processing of hexagonally sampled two-dimensional signals," *Proceedings of the IEEE*, vol. 67, no. 6, pp. 930–949, 1979.
11. D. E. Dudgeon and R. M. Mersereau, *Multidimensional Digital Signal Processing*, Prentice-Hall, 1984.
12. L. Middleton and J. Sivaswamy, *Hexagonal Image Processing*, Springer, 2005.
13. A. M. Grigoryan, "Hexagonal discrete cosine transform for image coding," *IEEE Transactions on Signal Processing*, vol. 50, no. 6, pp. 1438–1448, 2002.
14. H. J. Nussbaumer, *Fast Fourier Transformation and Convolution Algorithms*, Springer, 2nd edition, 1982.
15. 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.
16. J. R. Driscoll, D. M. Healy Jr., and D. Rockmore, "Fast discrete polynomial transforms with applications to data analysis for distance transitive graphs," *SIAM Journal Computation*, vol. 26, pp. 1066–1099, 1997.
17. D. Potts, G. Steidl, and M. Tasche, "Fast algorithms for discrete polynomial transforms," *Mathematics of Computation*, vol. 67, no. 224, pp. 1577–1590, 1998.

18. W. C. Curtis and I. Reiner, *Representation Theory of Finite Groups*, Interscience, 1962.
19. Th. Beth, *Verfahren der Schnellen Fouriertransformation [Fast Fourier Transform Methods]*, Teubner, 1984.
20. M. Clausen, *Beiträge zum Entwurf schneller Spektraltransformationen (Habilitationsschrift)*, Univ. Karlsruhe, 1988.
21. M. Clausen, "Fast generalized Fourier transforms," *Theoretical Computer Science*, vol. 67, pp. 55–63, 1989.
22. M. Clausen and U. Baum, *Fast Fourier Transforms*, BI-Wiss.-Verl., 1993.
23. 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.
24. D. Rockmore, "Fast Fourier analysis for abelian group extensions," *Advances in Applied Mathematics*, vol. 11, pp. 164–204, 1990.
25. D. Maslen and D. Rockmore, "Double coset decompositions and computational harmonic analysis on groups," *Journal of Fourier Analysis and Applications*, vol. 6, no. 4, 2000.
26. N. Jacobson, *Basic Algebra I*, W. H. Freeman and Co., 1974.
27. D. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms*, Springer, 1997.
28. Th. Becker and V. Weispfenning, *Gröbner Bases*, Springer, 1993.
29. Paul A. Fuhrman, *A Polynomial Approach to Linear Algebra*, Springer Verlag, New York, 1996.
30. N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. on Computers*, vol. C-23, pp. 90–93, 1974.
31. M. Püschel and M. Rötteler, "The discrete triangle transform," in *Proc. ICASSP*, 2004, vol. 3, pp. 45–48.
32. 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.
33. 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.
34. M. Püschel, "Decomposing monomial representations of solvable groups," *Journal of Symbolic Computation*, vol. 34, no. 6, pp. 561–596, 2002.
35. 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.
36. S. Egner and M. Püschel, "Symmetry-based matrix factorization," *Journal of Symbolic Computation*, vol. 37, no. 2, pp. 157–186, 2004.
37. T. S. Chihara, *An Introduction to Orthogonal Polynomials*, Gordon and Breach, 1978.
38. R. Eier and R. Lidl, "A class of orthogonal polynomials in k variables," *Math. Ann.*, vol. 260, pp. 93–99, 1982.
39. P. E. Ricci, "An iterative property of Chebyshev polynomials of the first kind in several variables," *Rendiconti di Matematica e delle sue Applicazioni*, vol. 6, no. 4, pp. 555–563, 1986.

A Chebyshev Polynomials in One Variable

We collect some properties of Chebyshev polynomials which are used in the text to define the DCT, type 3. See [5] for more details on Chebyshev polynomials. For a general introduction to the theory of orthogonal polynomials we refer to [37].

Definition through recurrence. The Chebyshev polynomials of the first kind (in one variable) are denoted with $(T_n(x) \mid n \in \mathbb{Z})$ and defined by the three-term recurrence

$$T_{n+1} = 2xT_n - T_{n-1}, \tag{49}$$

with initial values $T_0 = 1, T_1 = x$. The recurrence can be run in both directions to compute T_n for $n < 0$.

A few examples are $T_{-3} = 4x^3 - 3x$, $T_{-2} = 2x^2 - 1$, $T_{-1} = x$, $T_0 = 1$, $T_1 = x$, $T_2 = 2x^2 - 1$, $T_3 = 4x^3 - 3x$.

Parameterization. The T_n can be written in a parameterized form, called *power form*, as

$$T_n = \frac{1}{2}(u^n + u^{-n}), \quad x = \frac{1}{2}(u + u^{-1}). \tag{50}$$

Further, the *trigonometric form* of T_n is obtained by substituting $u = e^{\mathrm{i}\theta}$ into (50):

$$T_n = \cos n\theta, \quad \cos\theta = x, \tag{51}$$

which is valid for $x \in [-1, 1]$.

Zeros. From (51), the zeros of T_n are obtained as

$$\cos\frac{2k+1}{2n}\pi, \quad 0 \le k < n.$$

Symmetry property. Both parameterizations exhibit the symmetry property

$$T_{-n} = T_n. \tag{52}$$

Shift property. The following property can be readily derived from the three-term recursion:

$$T_k \cdot T_n = \frac{1}{2}(T_{n+k} + T_{n-k}), \quad k, n \in \mathbb{Z}. \tag{53}$$

Decomposition property. Finally, we have the decomposition property

$$T_{km} = T_k(T_m),$$

which underlies the Cooley-Tukey type algorithms for the DCT, type 3.

B Chebyshev Polynomials in Two Variables

The Chebyshev polynomials in two variables are not as well known as their counterparts in one variable. We use the definitions given in [6,38] with minor modifications to parallel the case of the univariate Chebyshev polynomials of the first kind presented above in Appendix A. In contrast to the univariate case, the polynomials are now labeled by two integers $T_{m,n}(x, y)$, where $m, n \in \mathbb{Z}$.

Definition through recurrence. The Chebyshev polynomials of the first kind in two variables are denoted with $(T_{m,n}(x, y) \mid m, n \in \mathbb{Z})$ and are defined by the two four-term recurrences

$$\begin{aligned} T_{m+1,n} &= 3xT_{m,n} - T_{m,n-1} - T_{m-1,n+1}, \\ T_{m,n+1} &= 3yT_{m,n} - T_{m-1,n} - T_{m+1,n-1}. \end{aligned} \tag{54}$$

The initial conditions are

$$\begin{gathered} T_{0,0} = 1, \quad T_{1,0} = x, \quad T_{2,0} = 3x^2 - 2y, \\ T_{0,1} = y, \quad T_{1,1} = (3xy - 1)/2, \quad T_{0,2} = 3y^2 - 2x. \end{gathered}$$

The recurrences (54) can be run in all direction to obtain a full lattice of polynomials. The recurrence equations also show that each polynomial has six neighbors, which naturally arranges them into the hexagonal 2-D array shown in Fig. 9.

Recall that for a polynomial $p(x, y)$ the total degree is the largest $a + b$ over all nontrivial summands $cx^a y^b$ of p. Using the recurrences (54) it is easy to show that the following lemma holds for all $T_{m,n}$.

Lemma 2 *Every polynomial $T_{m,n}$ has total degree $m + n$. Furthermore, every $T_{m,n}$ has precisely one summand $cx^a y^b$ for which $a + b = m + n$ holds, and for this summand $a = m$ and $b = n$. Hence, $\{T_{m,n} \mid m, n \ge 0\}$ is a basis of $\mathbb{C}[x, y]$.*

This lemma implies that every polynomial $T_{m,n}$ with $m < 0$ or $n < 0$ is a linear combination of Chebyshev polynomials $T_{m,n}$ with $m, n \ge 0$.

Parameterization. Similar to the univariate case, there is a power form for the Chebyshev polynomials in two variables. It uses the u, v-parameterization

$$x = \frac{1}{3}(u + v + (uv)^{-1}), \; y = \frac{1}{3}(u^{-1} + v^{-1} + uv). \tag{55}$$

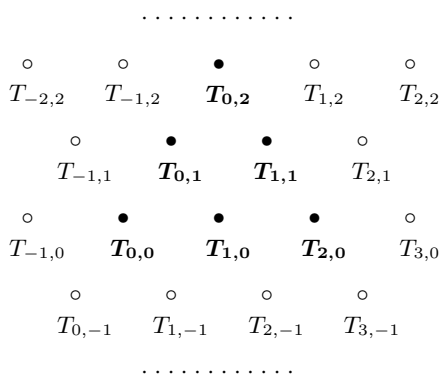


Fig. 9 The entire 2-D array of Chebyshev polynomials in two variables is uniquely determined by the initial conditions $T_{0,0}, T_{1,0}, T_{2,0}, T_{0,1}, T_{1,1}, T_{0,2}$ (solid bullets) via (54).

and is given by (see [6])

$$T_{m,n}(x,y) = \tfrac{1}{6}(u^n v^{-m} + u^{-m} v^n + u^{n+m} v^m + u^m v^{n+m} + u^{-n-m} v^{-n} + u^{-n} v^{-n-m}). \quad (56)$$

In particular, the power forms of $T_{n,0}$ and $T_{0,n}$ have only three summands:

$$T_{n,0} = \frac{1}{3}(u^n + v^n + (uv)^{-n}), \quad T_{0,n} = \frac{1}{3}(u^{-n} + v^{-n} + (uv)^n). \quad (57)$$

Substituting $u = e^{\mathrm{i}\phi}$, $v = e^{\mathrm{i}\psi}$ yields the corresponding trigonometric form.

For $p(x,y) \in \mathbb{C}[x,y]$ we denote with $\overline{p}(x,y) = p(y,x)$ the same polynomial with exchanged variables. For $T_{m,n}(x,y)$, exchanging x and y is equivalent to replacing u, v by u^{-1}, v^{-1} in the power form. Evaluation shows that

$$\overline{T}_{m,n}(x,y) = T_{n,m}(x,y).$$

In the paper, we often set $T_{m,0} = T_m$ (not to be confused with the Chebyshev polynomials in one variable) for brevity and hence $T_{0,n} = \overline{T}_n$. Again from the power forms it follows that

$$T_{m,n} = \frac{1}{2}(3T_m \overline{T}_n - T_{m-n}).$$

Zeros. It can be shown that the equations $T_n(x,y) = \overline{T}_n(x,y) = 0$ have precisely n^2 pairwise distinct common complex zeros (x,y). It is convenient to represent these zeros in the u,v-parameterization (55), in which they are given by all pairs

$$(u_i, v_j) = (\omega_n^i, \omega_{3n}^{1+3j}), \quad 0 \le i,j < n, \quad \omega_n = e^{-2\pi \mathrm{i}/n}.$$

Symmetry property. The definition of $T_{m,n}$ exhibits two symmetry properties given by

$$T_{n,-m} = T_{n-m,m}, \quad T_{-n,m} = T_{n,m-n}. \quad (58)$$

Shift property. The following equation is the equivalent of (53):

$$T_{k,\ell} \cdot T_{m,n} = \tfrac{1}{6}(T_{m-k-\ell,n+k} + T_{m-k,n+k+\ell} + T_{m+k,n+\ell} + T_{m+k+\ell,n-\ell} + T_{m+\ell,n-k-\ell} + T_{m-\ell,n-k}). \quad (59)$$

Decomposition property. Finally, for the purpose of deriving fast algorithms the following decomposition property [39] is used:

$$T_{km} = T_k(T_m, \overline{T}_m),\ \overline{T}_{km} = \overline{T}_k(T_m, \overline{T}_m). \tag{60}$$

Gröbner basis property. As noted in the text following (1), the polynomials p and q in the definition of the algebra

$$\mathcal{A} = \mathbb{C}[x, y]/\langle p(x, y), q(x, y)\rangle$$

have to satisfy the property that they form a Gröbner basis to make the computation modulo p and q well-defined. Whereas in general a given set of polynomials has to be modified to form a Gröbner basis—sometimes involving the addition of an exponential number of generators—in special cases this is not necessary. We briefly argue why the polynomials $p(x, y) = T_{m,0}(x, y)$ and $q(x, y) = T_{0,m}(x, y)$, which are used to define the discrete triangle transform, already form a Gröbner basis for all $m \geq 1$.

Recall that the total degree of a polynomial $p(x, y)$ is the largest $a + b$ over all nontrivial summands $cx^a y^b$ of p. The leading term of $p(x, y)$ is the term of highest degree. Lemma 2 implies that the leading term of $T_{m,0}$ is given by x^m and that the leading term of $T_{0,m}$ is given by y^m.

Now, Buchberger's first criterion [28, Section 5.5] can be applied which says that two polynomials $p(x, y)$ and $q(x, y)$ with disjoint leading terms (with respect to the total degree term order) already constitute a Gröbner basis. This shows the correctness of our definition of reduced polynomials in $\mathcal{A}$.