ALGEBRAIC DERIVATION OF GENERAL RADIX COOLEY-TUKEY ALGORITHMS FOR THE REAL DISCRETE FOURIER TRANSFORM

Yevgen Voronenko and Markus Püschel

Electrical and Computer Engineering
Carnegie Mellon University

ABSTRACT

We first show that the real version of the discrete Fourier transform (called RDFT) can be characterized in the framework of polynomial algebras just as the DFT and the discrete cosine and sine transforms. Then, we use this connection to algebraically derive a general radix Cooley-Tukey type algorithm for the RDFT. The algorithm has a similar structure as its complex counterpart, but there are also important differences, which are exhibited by our Kronecker product style presentation. In particular, the RDFT is decomposed into smaller RDFTs but also other auxiliary transforms, which we then decompose by their own Cooley-Tukey type algorithms to obtain a full recursive algorithm for the RDFT.

1. INTRODUCTION

The discrete Fourier transform (DFT) is the ubiquitous tool in signal processing. By definition, the DFT processes complex inputs to produce complex outputs. In many cases, the input signal is real, and it is well-known that in this case the computational cost can be reduced by a factor of roughly 2 [1]. In fact, this "real" DFT (RDFT) can be viewed again as a transform that is related to the DFT, but maps real inputs to real outputs. So it is not surprising that many DFT algorithms have RDFT counterparts, but the derivation turns out to be a nontrivial task. In fact, while there is a considerable body of literature on complex fast Fourier transforms (FFTs), publications on real FFTs are comparatively few and far. This becomes particularly apparent in standard books on the topic: the use of polynomial algebra in [2] to derive DFT algorithms is restricted to the complex case; similarly [3, 4] develop the underpinning of FFTs and propose the very concise Kronecker product formalism for their representation, but again focus only on the complex case. Only a radix-2 Cooley-Tukey real FFT is discussed in [3].

The contribution of this paper is two-fold. First, we show that the RDFT, just as the DFT, can be characterized in the framework of polynomial algebras. Then, we derive a general-radix Cooley-Tukey type algorithm for the RDFT using the same general principle that accounts for the (complex) Cooley-Tukey FFT. Thus, in contrast to previous work, we do not derive the algorithms by manipulating the actual transform, nor by specializing the corresponding complex FFT. This is theoretically satisfying, since we already showed that also many discrete cosine and sine transform algorithms can be derived this way [5, 6]. Second, the derivation naturally leads to a presentation using Kronecker products in the spirit of [3, 4]. This way, it also becomes apparent that the RDFT decomposes into smaller transforms, which however are of different types. These "auxiliary" transforms are then decomposed using their own Cooley-Tukey type algorithms, again derived using polynomial algebras. Note that the complex FFT decomposes the DFT into smaller DFTs of the same type and is thus of simpler structure.

Cooley-Tukey type algorithms for the RDFT were derived in [1] for radix 2 and split radix 2/4, and for a general radix in [7]. FFTW [8] also uses a general radix RDFT algorithm, which is very close to the one we derive this paper. An algebraic method for projecting complex FFTs was presented in [9], but it appears that the method, as presented based on normal bases, works for the discrete Hartley transform, but not for the RDFT.

In Section 2 we provide the background on polynomial algebras and explain their relationship to signal transforms. Then we present a general method that, as we illustrate, allows us to concisely derive the Cooley-Tukey FFT through a stepwise decomposition of its associated polynomial algebra. The same method is then used in Section 3 to derive the corresponding RDFT algorithms, after establishing the algebraic interpretation of the RDFT.

2. POLYNOMIAL ALGEBRAS AND TRANSFORMS

In this section we introduce polynomial algebras and explain how they are associated to transforms. Then we present a general decomposition theorem for polynomial algebras and show that it can be used to derive the Cooley-Tukey FFT. Later, we will use the same theorem to derive Cooley-Tukey type algorithms for the RDFT.

Polynomial algebra. An algebra is a vector space that permits also multiplication and that is closed under multiplication. Examples include the set of complex or real numbers $\mathbb{C}$ or $\mathbb{R}$, and the set of polynomials $\mathbb{C}[x]$ or $\mathbb{R}[x]$.

The key player in this paper is the *polynomial algebra*. Given a fixed polynomial $p(x)$ of degree $\deg(p) = n$, we define a polynomial algebra as the set

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

of polynomials of degree smaller than p with addition and multiplication modulo p. As a vector space, $\mathbb{C}[x]/p(x)$ has dimension n. Intuitively, $\mathbb{C}[x]/p(x)$ is the set of polynomials $\mathbb{C}[x]$ with the condition $p(x) = 0$ imposed.

Chinese remainder theorem (CRT). Assume $p(x) = q(x)r(x)$ factorizes into two coprime (no common factors) polynomials q and r. Then the Chinese remainder theorem (CRT) for polynomials gives the linear mapping[1]

$$\begin{array}{rccl} \Delta: & C[x]/p(x) & \to & \mathbb{C}[x]/q(x) \oplus \mathbb{C}[x]/r(x), \\ & s(x) & \mapsto & (s(x) \bmod q(x), s(x) \bmod r(x)). \end{array}$$

Here, $\oplus$ is the direct sum of vector spaces with elementwise operation. If we choose bases b, c, d in the three polynomial algebras, then Δ can be expressed as a matrix. This matrix is obtained by mapping every element of b with Δ, expressing it in the concatenation $c \cup d$ of the bases c and d, and writing the results into the columns of the matrix. This is best explained using an example.

We consider $p(x) = x^2 - 1 = (x-1)(x+1)$ and

$$\Delta: \mathbb{C}[x]/(x^2-1) \to \mathbb{C}[x]/(x-1) \oplus \mathbb{C}[x]/(x+1).$$

As bases, we choose $b = (1, x),\ c = (1),\ d = (1)$. $\Delta(1) = (1, 1)$ with the same coordinate vector in $c \cup d = (1, 1)$. Further, because

This work was supported by NSF through award 0310941.

[1] More precisely, isomorphism of algebras.

of $x \equiv 1 \bmod (x-1)$ and $x \equiv -1 \bmod (x+1)$, $\Delta(x) = (x, x) \equiv (1, -1)$ with the same coordinate vector. Thus the desired matrix is the so-called butterfly $\left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right]$.

Polynomial transforms. Assume $p(x) \in \mathbb{C}[x]$ is separable, i.e., it has pairwise distinct zeros $\alpha = (\alpha_0, \ldots, \alpha_{n-1})$. Then the CRT decomposes $\mathbb{C}[x]/p(x)$ completely into its *spectrum*:

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

If we choose a basis $b = (p_0, \ldots, p_{n-1})$ in $\mathbb{C}[x]/p(x)$ and bases $b_i = (1)$ for the $\mathbb{C}[x]/(x-\alpha_i)$, then the corresponding matrix is given by

$$\mathcal{P}_{b,\alpha} = [p_j(\alpha_i)]_{0 \le i,j < n}$$

and is called a *polynomial transform* for $\mathbb{C}[x]/p(x)$ with basis b.

For example, the DFT of size n (viewed as a matrix) is a polynomial transform for $\mathbb{C}[x]/(x^n-1)$ with basis $b = (1, x, \ldots, x^{n-1})$. Namely, $x^n - 1 = \prod_{0 \le i < n}(x - w_{i/n})$, $w_u = \exp(-2\pi\sqrt{-1}u)$, and thus

$$\mathcal{P}_{b,\alpha} = [w_{i/n}^j]_{0 \le i,j < n} = [w_{ij/n}]_{0 \le i,jn} = \mathrm{DFT}_n\,.$$

Choosing $\mathbb{C}[x]/(x^n+1)$ instead with the same basis yields

$$\mathcal{P}_{b,\alpha} = [w_{(i+1/2)/n}^j]_{0 \le i,jn} = \text{DFT-3}_n,$$

which has been called the DFT of type 3 [10].

We have shown that all 16 discrete cosine and sine transform are polynomial transforms (if scalars $\neq 1$ are allowed in the b_i) [5].

Cooley-Tukey type algorithms. The connection to polynomial algebras can be used to derive fast algorithms for the associated transforms using a general method that is related to, but somewhat different from the early work [2]. In short, we derive Cooley-Tukey algorithms by performing the decomposition (1) *in steps* based on a *decomposition* of p (if one exists). To state the general method we first introduce the product of bases of polynomials. Let $b = (p_0, \ldots, p_{k-1})$ and $c = (q_0, \ldots, q_{m-1})$ be two lists of polynomials. Then their *product* is the list of length km

$$b \star c = (p_0 q_0, \ldots, p_0 q_{m-1}, \ldots\ \ldots\ \ldots, p_{k-1} q_0, \ldots, p_{k-1} q_{m-1}).$$

Further, if b is as above, and $r(x)$ is any polynomial, then we denote with

$$b(r(x)) = (p_0(r(x)), \ldots, p_{k-1}(r(x)))$$

the same list but with $r(x)$ inserted for x.

Further, recall that if $A = [a_{i,j}]$ and B are matrices, then the direct sum and the tensor or Kronecker product are respectively defined as

$$A \oplus B = \begin{bmatrix} A & \\ & B \end{bmatrix}, \quad A \otimes B = [a_{i,j}B].$$

The $n \times n$ identity matrix is denoted with I_n.

Now we can state our lemma, a variation of a theorem we already used in [5] to derive DCT algorithms. Thus we also omit the proof. Note that the lemma also holds if $\mathbb{C}$ is replaced by $\mathbb{R}$.

Lemma 1 Let $q(x)$ be separable and $q(x) = \prod_{0 \le i < k} q_i(x)$. Further, let c and c_i, $0 \le i < k$, be bases for $\mathbb{C}[x]/q(x)$ and $\mathbb{C}[x]/q_i(x)$, respectively, and let, with these bases, M be the matrix associated with the decomposition

$$\mathbb{C}[x]/q(x) \to \bigoplus_{0 \le i < k} \mathbb{C}[x]/q_i(x).$$

If $r(x)$ is an arbitrary polynomial of degree m and d a basis for $\mathbb{C}[x]/r(x)$, then $M \otimes I_m$ is the matrix associated with

$$\mathbb{C}[x]/q(r(x)) \to \bigoplus_{0 \le i < k} \mathbb{C}[x]/q_i(r(x)),$$

w.r.t. the bases $b = c(r(x)) \star d$ and $b_i = c_i(r(x)) \star d$.

We use Lemma 1 to derive the Cooley-Tukey FFT. The polynomial algebra for DFT_n is $\mathbb{C}[x]/(x^n-1)$ with basis $b = (1, x, \ldots, x^{n-1})$. Assuming $n = km$, then $x^n - 1 = (x^m)^k - 1$ decomposes. Applying the CRT in steps yields

$$\begin{aligned} & \mathbb{C}[x]/(x^n-1) = \mathbb{C}[x]/((x^m)^k - 1) \\ \to\ & \bigoplus_{0 \le i < k} \mathbb{C}[x]/(x^m - w_{i/k}) && (2) \\ \to\ & \bigoplus_{0 \le i < k} \bigoplus_{0 \le j < m} \mathbb{C}[x]/(x - w_{(jk+i)/n}) && (3) \\ \to\ & \bigoplus_{0 \le i < n} \mathbb{C}[x]/(x - w_{i/n}). && (4) \end{aligned}$$

We read off the matrices for each decomposition step. First, we observe that $b = c(x^m) \star d$, with $c = (1, x, \ldots, x^{k-1})$ and $d = (1, x, \ldots, x^{m-1})$. Thus, Lemma 1 is applicable: step (2) corresponds to $\mathrm{DFT}_k \otimes I_m$ and d is the basis in each $\mathbb{C}[x]/(x^m - w_{i/k})$. Each of the latter polynomial algebras is completely decomposed in step (3) by the polynomial transform

$$\begin{aligned} \mathrm{DFT}_m(i/k) &= [w_{(jk+i)/n}^{\ell}]_{0 \le j,\ell < n} \\ &= \mathrm{DFT}_n \cdot \mathrm{diag}_{0 \le j < m}(w_{ij/n}). \end{aligned} \tag{5}$$

The final step (4) is just a permutation (the so-called *stride permutation*) of the one-dimensional algebras and is given by

$$\mathrm{L}_m^n :\ jk + i \mapsto im + j, \quad 0 \le i, j < n. \tag{6}$$

In summary we get

$$\begin{aligned} \mathrm{DFT}_n &= \mathrm{L}_m^n \big(\bigoplus_{0 \le i < k} \mathrm{DFT}_m(i/k) \big) (\mathrm{DFT}_k \otimes I_m) \\ &= \mathrm{L}_m^n (I_k \otimes \mathrm{DFT}_m) D_m^n (\mathrm{DFT}_k \otimes I_m), \end{aligned}$$

where D_m^n is diagonal, namely the direct sum of the diagonal matrices in (5). The algorithm is the decimation-in-frequency FFT, its transpose is the decimation-in-time version.

3. REAL DFT

We associate a polynomial algebra with the real DFT (RDFT), then we derive its Cooley-Tukey type algorithms using again Lemma 1. This shows that the mathematical underpinning is the same in the real and complex case. The difference is in some of the details: the RDFT algorithms have a somewhat more complicated structure and require auxiliary transforms for a complete recursion.

RDFT. To obtain the RDFT, we decompose $\mathbb{R}[x]/(x^n-1)$ over the real numbers rather than $\mathbb{C}[x]/(x^n-1)$ over the complex numbers (as for the DFT). To do so, we set $c_u = \cos(2\pi u)$, $s_u = \sin(2\pi u)$, and introduce the polynomials (for $0 < u < 1/2$)

$$p_{2n,u}(x) = (x^n - w_u)(x^n - w_{-u}) = x^{2n} - 2c_u x^n + 1. \tag{7}$$

Over the real numbers, we have the following complete factorizations into polynomials of degree one and two:

$$\begin{aligned} x^n - 1 &= (x-1)\Big(\prod_{0<i<n/2} p_{2,i/n}(x)\Big)(x+1), \quad n \text{ even}, \\ x^n - 1 &= (x-1)\prod_{0<i\le(n-1)/2} p_{2,i/n}(x), \quad n \text{ odd}, \qquad (8) \\ p_{2n,u}(x) &= \prod_{0\le i<n} p_{2,(i+u)/n}(x). \qquad (9) \end{aligned}$$

In (9) some of the factors have $(i+u)/n > 1/2$. Thus we normalize so that the angles are again $< 1/2$ (using $p_{2,u} = p_{2,1-u}$) and reorder the factors according to the new angles. The result is

$$p_{2n,u}(x) = \prod_{0\le i<n} p_{2,U(n,i,u)}(x),\ U(n,i,u) = \begin{cases} \frac{u+\lfloor i/2\rfloor}{n}, & 2|i, \\ \frac{1-u+\lfloor i/2\rfloor}{n}, & \text{else.} \end{cases}$$

The RDFT corresponds to the following decomposition. We show it only for even size n; the odd case is obtained analogously from (8).

$$\begin{aligned} &\mathbb{R}[x]/(x^n-1) \\ \to\ &\mathbb{R}[x]/(x-1) \oplus \bigoplus_{0<i<n/2} \mathbb{R}[x]/p_{2,i/n} \oplus \mathbb{R}[x]/(x+1) \qquad (10) \end{aligned}$$

with bases $b = (1, x, \ldots, x^{2n-1})$ in $\mathbb{R}[x]/(x^n-1)$ and respectively $(1),\ e_{i/n},\ (1)$ in the summands of (10); $e_{i/n}$ has length 2 and is defined through[2]

$$e_u = (1, -c_u/s_u + 1/s_u \cdot x).$$

With these bases we can compute the matrix, which is, as desired[3]

$$\mathrm{RDFT}_{2n} = \begin{bmatrix} 1 & 1 & \ldots & 1 & 1 \\ \begin{bmatrix} c_{ij/n} \\ s_{ij/n} \end{bmatrix}_{0<i<n/2, 0\le j<n} & & & & \\ 1 & -1 & \ldots & 1 & -1 \end{bmatrix}. \qquad (11)$$

Further, we will need two auxiliary transforms. First, the so-called RDFT-3, which for even n corresponds to the decomposition

$$\mathbb{R}[x]/(x^n+1) \to \bigoplus_{0\le i<n/2} \mathbb{R}[x]/p_{2,(i+1/2)/n}$$

with bases b on the left side and $e_{(i+1/2)/n}$ in the summands on the right side. For odd n, the decomposition is analogous but with one one-dimensional summand $\mathbb{R}[x]/(x+1)$. As a matrix

$$\text{RDFT-3}_n = \begin{bmatrix} c_{(i+1/2)j/n} \\ s_{(i+1/2)j/n} \end{bmatrix}_{0\le i<n/2, 0\le j<n}.$$

Second, we need $\mathrm{rDFT}_{2n}(u)$, decomposing

$$\mathbb{R}[x]/p_{2n,u} \to \bigoplus_{0\le i<n} \mathbb{R}[x]/p_{2,U(n,i,u)}$$

with basis $e_u(x^n) \star (1, x, \ldots, x^{n-1})$ on the left side, and $e_{U(n,i,u)}$ in the summands on the right side.[4]

[2]This seemingly unmotivated choice of basis is such that the operation of x on this basis becomes a rotation by u; and, of course, the one that leads to the RDFT as we show.

[3]Note that the definitions of the RDFT in the literature may differ by a permutation of the rows compared to (11).

[4]We deliberately use a lower case r in rDFT since the basis in the polynomial algebra is not the standard basis $(1, x, \ldots, x^{2n-1})$.

Table 1 summarizes all complex and real types of DFTs mentioned and needed in this paper.

Real Cooley-Tukey FFT. Analogous to (2)–(4) we derive the Cooley-Tukey FFT for the RDFT. We assume the size $n = km$ factors and consider the case that both k and m are even. The other three cases (even-odd, odd-even, and odd-odd) are analogous. Again, the algorithm is based on the decomposition property $x^n - 1 = (x^m)^k - 1$. We get

$$\begin{aligned} &\mathbb{R}[x]/(x^n-1) = \mathbb{R}[x]/((x^m)^k-1) \\ \to &\mathbb{R}[x]/(x^m-1) \oplus \bigoplus_{0<i<k/2} \mathbb{R}[x]/p_{2m,i/k} \oplus \mathbb{R}[x]/(x^m+1) \qquad (18) \\ \to &\Big(\mathbb{R}[x]/(x-1) \oplus \bigoplus_{0<i<m/2} \mathbb{R}[x]/p_{2,i/m} \oplus \mathbb{R}[x]/(x+1)\Big) \\ \oplus &\bigoplus_{\substack{0<i<k/2 \\ 0<j<m}} \bigoplus \mathbb{R}[x]/p_{2,U(m,j,i/k)} \oplus \Big(\bigoplus_{0\le i<m/2} \mathbb{R}[x]/p_{2,(i+1/2)/m}\Big) \qquad (19) \\ \to &\mathbb{R}[x]/(x-1) \oplus \bigoplus_{0<i<km/2} \mathbb{R}[x]/p_{2,i/km} \oplus \mathbb{R}[x]/(x+1). \qquad (20) \end{aligned}$$

As for the DFT, we apply Lemma 1 to establish that the matrix corresponding to (18) is $\mathrm{RDFT}_{2k} \otimes I_m$ if we choose in the bases $b = (1, x, \ldots, x^{m-1})$, $e_{i/k}(x^m) \star b$, and b in the summands.

The polynomial algebras of dimensions m and $2m$ in (18) are next completely decomposed over $\mathbb{R}$ in step (19) using RDFT_m, $\mathrm{rDFT}_{2m}(i/k)$, and RDFT-3_m, respectively.

A suitable permutation in (20) then reorders the one- and two-dimensional polynomial algebras into the required order in (10).

The final algorithm is shown in (12) and (13) in Table 2. In contrast to the complex DFT (see (5)), it is not efficient to convert the occurring RDFT-3 and rDFTs into ordinary RDFTs. Rather, we derive for these transforms their own Cooley-Tukey algorithms, using a derivation completely analogous to the RDFT. For RDFT-3 we use the decomposition $x^n + 1 = (x^k)^m + 1$ for $n = km$; for rDFT_{2km} we use the decomposition $p_{2km,u}(x) = p_{2k,u}(x^m)$ and the fact that the basis decomposes as $b = e_u(x^{km}) \star (1, \ldots, x^{k-1}) = [e_u(x^k) \star (1, \ldots, x^{k-1})](x^m) \star (1, \ldots, x^{m-1})$. Due to lack of space, we only state the final results in Table 2. The permutation P is given as a function permuting the output indices; the permutation K_m^{km} is given by the matrix

$$K_m^{km} = (I_k \oplus J_k \oplus I_k \oplus J_m \ldots) L_m^{km},$$

where J_k is I_k with the column order reversed. The permutations $\hat{P}$ and $\hat{Q}$ are not given due to lack of space. As P, they would be best visualized as matrices to assert their structure.

For two-power sizes, these algorithms specify the entire computation, together with the size-2 base cases also given in Table 2.

Discussion. In the literature we find an arbitrary radix algorithm for RDFT_n in [7], expressed solely in terms of RDFT's of smaller sizes, which leads to suboptimal arithmetic cost. The authors noticed that part of the computation is equivalent to a RDFT-3; however, they only show how to compute RDFT-3 for an even size, namely by decomposing it into a discrete cosine and sine transform. In the present paper, RDFT-3 naturally occurs as a polynomial transform. Further, completely analogously to the RDFT we derive the arbitrary radix RDFT-3 algorithm for both even and odd sizes.

Sorensen, et al. [1] show the radix-2 and split-radix variants of this general algorithm, without recognizing RDFT-3 and rDFT as auxiliary transforms. The real FFT used in the FFTW [8] source

Transform	Polynomial algebra	Basis b	Spectrum (polynomials only)	Bases in spectrum b_k
DFT_n	$\mathbb{C}[x]/(x^n-1)$	(x^j)	$\{x-w_{i/n}\}_{0\le i<n}$	$\{(1)\}$
$\mathrm{DFT}\text{-}3_n$	$\mathbb{C}[x]/(x^n+1)$	(x^j)	$\{x-w_{(i+1/2)/n}\}_{0\le i<n}$	$\{(1)\}$
$\mathrm{DFT}_n(u)$	$\mathbb{C}[x]/(x^n-w_u)$	(x^j)	$\{x-w_{(i+u)/n}\}_{0\le i<n}$	$\{(1)\}$
RDFT_n (n even)	$\mathbb{R}[x]/(x^n-1)$	(x^j)	$(x-1), \{p_{2,i/n}\}_{1\le i<n/2}, (x+1)$	$(1), \{e_{i/n}\}, (1)$
RDFT_n (n odd)	$\mathbb{R}[x]/(x^n-1)$	(x^j)	$(x-1), \{p_{2,i/n}\}_{1\le i\le (n-1)/2}$	$(1), \{e_{i/n}\}$
$\mathrm{RDFT}\text{-}3_n$ (n even)	$\mathbb{R}[x]/(x^n+1)$	(x^j)	$\{p_{2,(i+1/2)/n}\}_{0\le i<n/2}$	$\{e_{(i+1/2)/n}\}$
$\mathrm{RDFT}\text{-}3_n$ (n odd)	$\mathbb{R}[x]/(x^n+1)$	(x^j)	$(x+1), \{p_{2,(i+1/2)/n}\}_{1\le i\le (n-1)/2}$	$(1),\ \{e_{(i+1/2)/n}\}$
$\mathrm{rDFT}_{2n}(u)$	$\mathbb{R}[x]/p_{2n,u}$	$e_u(x^n)\star(x^j)$	$\{p_{2,U(n,i,u)}\}_{0\le i<n/2}$	$\{e_{U(n,i,u)}\}$

Table 1. Three complex and real types of DFTs that occur in the complex and real Cooley-Tukey FFT.

$$\mathrm{RDFT}_{km} = P_m^{km}\Big(\mathrm{RDFT}_m \oplus \Big(\bigoplus_{1\le i<k/2} \mathrm{rDFT}_{2m}(i/k)\Big) \oplus \mathrm{RDFT}\text{-}3_m\Big)(\mathrm{RDFT}_k \otimes I_m), \quad k \text{ even}, \tag{12}$$

$$\mathrm{RDFT}_{km} = \hat{P}_m^{km}\Big(\mathrm{RDFT}_m \oplus \Big(\bigoplus_{1\le i\le (k-1)/2} \mathrm{rDFT}_{2m}(i/k)\Big)\Big)(\mathrm{RDFT}_k \otimes I_m), \quad k \text{ odd}, \tag{13}$$

$$\mathrm{RDFT}\text{-}3_{km} = (K_m^{km/2} \otimes I_2)\Big(\bigoplus_{0\le i<k/2} \mathrm{rDFT}_{2m}((1/2+i)/k)\Big)(\mathrm{RDFT}\text{-}3_k \otimes I_m), \quad k \text{ even}, \tag{14}$$

$$\mathrm{RDFT}\text{-}3_{km} = \hat{Q}_m^{km}\Big(\Big(\bigoplus_{1\le i\le (k-1)/2} \mathrm{rDFT}_{2m}((1/2+i)/k)\Big) \oplus \mathrm{RDFT}\text{-}3_m\Big)(\mathrm{RDFT}\text{-}3_k \otimes I_m), \quad k \text{ odd}, \tag{15}$$

$$\mathrm{rDFT}_{2km}(u) = (K_m^{km} \otimes I_2)\Big(\bigoplus_{0\le i<k} \mathrm{rDFT}_{2m}(U(k,i,u))\Big)(\mathrm{rDFT}_{2k}(u) \otimes I_m). \tag{16}$$

$$\mathrm{RDFT}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathrm{RDFT}\text{-}3_2 = I_2, \quad \mathrm{rDFT}_4(u) = \begin{bmatrix} 1 & & 1 & \\ & 1 & & 1 \\ 1 & & -1 & \\ & -1 & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & \cos(\pi u) & & -\sin(\pi u) \\ & & 1 & \\ & \sin(\pi u) & & \cos(\pi u) \end{bmatrix}.$$

$$P_m^{km} = i \to \begin{cases} 0, & i = 0, \\ -1 + 2k\lfloor (i+1)/2 \rfloor + (i-1) \mod 2, & 1 \le i < m, \\ k + 2k\lfloor (i-(km-m))/2 \rfloor - 1 + (i-m) \mod 2, & i \ge km-m, \\ 1 + k\lfloor (i-m)/2 \rfloor \mod m + 2\lfloor (i-m)/(2m) \rfloor + (i-m) \mod 2, & (\lfloor (i-m)/2 \rfloor \mod m) \mod 2 = 0, \\ 1 + k\lfloor (i-m)/2 \rfloor \mod m + 2\lfloor (km-m-i)/(2m) \rfloor + (i-m) \mod 2, & \text{else}. \end{cases} \tag{17}$$

Table 2. The derived arbitrary-radix real FFTs including the base cases for size 2. The permutations $\hat{P}$ and $\hat{Q}$ are not provided.

code is very similar to the one derived here. The occurring rDFTs are computed using complex DFTs of half the size.

The derivation of Bruun's real FFT [11] is similar to ours but considers only $k = 2$ and uses (in our language) the bases $(1, x)$ in the spectral components, thus leading to a slightly different algorithm. This turns the rDFTs into RDFTs and increases the arithmetic cost.

4. REFERENCES

[1] H. V. Sorensen, D. L. Jones, M. T. Heideman, and C. S. Burrus, "Real-valued fast Fourier transform algorithms," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 6, pp. 849–863, 1987.

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

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

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

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

[6] M. Püschel, "Cooley-Tukey FFT like algorithms for the DCT," in *Proc. ICASSP*, 2003, vol. 2, pp. 501–504.

[7] N.-C. Hu and O. K. Ersoy, "Fast computation of real discrete Fourier transform for any number of data points," *IEEE Trans. on Circ. and Sys.*, vol. 38, no. 11, pp. 1280–1292, 1991.

[8] Matteo Frigo and Steven G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005, special issue on "Program Generation, Optimization, and Adaptation".

[9] J. Hong, M. Vetterli, and P. Duhamel, "Basefield transforms with the convolution property," *Proceedings of the IEEE*, vol. 82, no. 3, pp. 400–412, 1994.

[10] 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.

[11] G. Bruun, "z-transform DFT filters and FFTs," *IEEE Trans. ASSP*, vol. 26, no. 1, pp. 56–63, 1978.