Contemporary Mathematics

A Comrade-Matrix-Based Derivation of the Eight Versions of Fast Cosine and Sine Transforms

Alexander Olshevsky, Vadim Olshevsky, and Jun Wang

"Life as we know it would considerably different if, from 1965 Cooley-Turkey paper onwards the FFT community has made systematic and heavy use of matrix-vector notation. Indeed, couching results and algorithms in matrix-vector notation the FFT literature can be unified and made more understandable to the outsider."

Charles Van Loan, *"Computational frameworks for the FFT"* [**VL92**]

Abstract. The paper provides a full self-contained derivation of fast algorithms to compute discrete Cosine and Sine transforms I - IV. For the Sine I/II and Cosine I/II transforms a unified derivation based on the concept of the comrade matrix is presented. The comrade matrices associated with different versions of the transforms differ in only a few boundary elements; hence, in each case algorithms can be derived in a unified manner. The algorithm is then modified to compute Sine III/IV and Cosine III/IV transforms as well. The resulting algorithms for the versions III/IV are direct and recursive, such algorithms were missing in the existing literature. Finally, formulas reducing Cosine and Sine transforms of the types III and IV to each other are presented.

Part I: Versions I and II Of The Transforms

1. Introduction

1.1. Preliminaries. The FFT is, without doubt, the most important algorithm in applied mathematics and engineering. As Charles Van Loan writes in his book [**VL92**]: "The fast Fourier transform (FFT) is one of the truly great computational developments of this century. It has changed the face of science and engineering so that it is not an exaggeration to say that life as we know it would be very different without FFT." The importance of the FFT stems from two distinct factors: its presence in a plethora of applications, spanning almost every area of

1991 *Mathematics Subject Classification.* Primary 65T50, 94A12; Secondary 65F30.

Key words and phrases. Discrete Cosine Transform, Discrete Sine Transform, Fast Fourier Transform, Comrade Matrix, Superfast Algorithms.

This work was supported in part by NSF contracts CCR 0098222 and 0242518.

computational engineering, and the availability of fast and accurate algorithms for its computation.

The FFT uses complex arithmetic. There exist real-arithmetic analogues of the FFT, namely the Discrete Cosine Transform(DCT) and the Discrete Sine Transform(DST). Some references for these these transforms can be found in [**RY90**] and [**S99**]. In this paper, the eight versions of the cosine and sine transform are considered. For these real arithmetic transforms there also exist many applications – ranging from adaptive filtering to speech and image compression. The availability of fast algorithms for their computation has led to the increasing popularity of these transforms in the last decade.

There are several different mathematical techniques which can be used to derive fast algorithms for these transforms: polynomial, matrix factorization, and others. However, none of them seem to suggest a transparent approach: algorithms for each transform are typically derived differently, with somewhat painstaking computation. In fact, there is no monograph where one could find all these transforms derived at once. Secondly, the transforms III and IV are a bit more involved than I and II (see, e.g., sec. 4 for an explanation), and for them direct recursive algorithms are missing. In this paper the latter two disadvantages are removed. In part I fast algorithms for the transforms I and II are derived via operations on companion matrices and their certain generalizations called comrade matrices. The divide-and-conquer nature of the approach causes all of the algorithms to run in $O(n \log n)$ time. Secondly, in parts II and III new recursive algorithms are developed for the versions III and IV.

1.2. The Fast Fourier Transform.

1.2.1. *Definitions and the divide-and-conquer algorithm.* The Discrete Fourier Transform of a sequence $\{u_n\}_{n=0}^{N-1}$ is the sequence $\{U_n\}_{n=0}^{N-1}$ defined by

$$\begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ U_{N-1} \end{bmatrix} = \underbrace{\frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N & W_N^2 & \cdots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & & & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix}}_{F_N} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \tag{1.1}$$

where $W_N = e^{\frac{-2\pi i}{N}}$. The sine and cosine transforms of the sequence $\{u_n\}$ are defined in a similar manner, with a different matrix in place of F_n. The matrices for each transform are listed in Table 1.

	Discrete transform	Inverse transform
DCT-I	$C_N^I = \sqrt{\frac{2}{N-1}} \left[\alpha_{kj} \cos \frac{kj\pi}{N-1} \right]_{k,j=0}^{N-1}$ where $\alpha_{kj} = \eta_k \eta_{N-1-k} \eta_j \eta_{N-1-j}$	$[C_N^I]^{-1} = [C_N^I]^T = C_N^I$
DCT-II	$C_N^{II} = \sqrt{\frac{2}{N}} \left[\eta_k \cos \frac{k(2j+1)\pi}{2N} \right]_{k,j=0}^{N-1}$	$[C_N^{II}]^{-1} = [C_N^{II}]^T = C_N^{III}$
DST-I	$S_N^I = \sqrt{\frac{2}{N+1}} \left[\sin \frac{kj}{N+1}\pi \right]_{k,j=1}^{N}$	$[S_N^I]^{-1} = [S_N^I]^T = S_N^I$
DST-II	$S_N^{II} = \sqrt{\frac{2}{N}} \left[\eta_k \sin \frac{k(2j-1)}{2N}\pi \right]_{k,j=1}^{N}$	$[S_N^{II}]^{-1} = [S_N^{II}]^T = S_N^{III}$

Table 1. Discrete trigonometric transforms. Here, $\eta_k = \begin{cases} \frac{1}{\sqrt{2}} & k = 0, N \\ 1 & \text{otherwise} \end{cases}$

Computation of the DFT by standard matrix-vector multiplication would take order $O(n^2)$ operations. The Fast Fourier Transform speeds up this computation to $O(n \log n)$ by using a divide-and-conquer approach. Namely, the FFT reduces solving the problem of size N to two problems of size $\frac{N}{2}$ at the cost of only $O(N)$. Since the recursive application of this method will result in approximately $\log N$ halving steps, the result is $O(N \log N)$ running time. The idea behind the FFT is highlighted by the following formula (see, e.g., [**S86**]):

$$F_N = \begin{bmatrix} odd - even \\ permutation \end{bmatrix} \begin{bmatrix} F_{\frac{N}{2}} & 0 \\ 0 & F_{\frac{N}{2}} \end{bmatrix} \begin{bmatrix} I & I \\ \varepsilon_{\frac{N}{2}} & -\varepsilon_{\frac{N}{2}} \end{bmatrix} \tag{1.2}$$

where $\varepsilon_{\frac{N}{2}} = diag(1, W_N, \cdots, W_N^{\frac{N}{2}-1})$. Formula (1.2) can be applied repeatedly to reduce a size-N DFT's to two size-$\frac{N}{2}$ DFT, resulting in the Fast Fourier Transform.

1.2.2. *Polynomial Division Interpretation for the FFT.* Though deriving formula (1.2) for the DFT can be easily done, extending it by brute force to DCT/DST is cumbersome. Here we propose a different matrix approach which provides more insight. We first interpret the DFT as polynomial division and then express it in matrix form. In the introduction we show the result of this technique for the simplest DFT case. Then, in the main text, we spell out the details for the DCT/DST transforms.

It is possible to look at the FFT as an algorithm which computes polynomial evaluations. Indeed, if $u(x) = x^n + u_{N-1}x^{N-1} + u_{N-2}x^{N-2} + \cdots + u_1 x + u_0$ then the output sequence $\{U_k\}$ is composed of the evaluations of the polynomial $a(x)$ at the N'th roots of unity W_N^k:

$$\begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ U_{N-1} \end{bmatrix} = F_N \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} = \begin{bmatrix} u(1) \\ u(W_N) \\ \vdots \\ u(W_N^{N-1}) \end{bmatrix} \tag{1.3}$$

A straightforward evaluation of the polynomial $u(x)$ takes $O(n^2)$ operations, much like matrix-vector multiplication. However, a divide and conquer approach can be applied in order to get a faster $O(n \log n)$ algorithm.

Let us assume that N is even and denote by S_0 the N'th roots of unity, i.e.

$$S_0 = \{1, W_N, W_N^2, \ldots, W_N^{N-1}\} \tag{1.4}$$

S_0 can be split up into "even" and "odd" parts:

(1.5)
$$S_1 = \{1, W_N^2, W_N^4, \ldots, W_N^{N-2}\} \qquad \text{and} \qquad S_2 = \{W_N, W_N^3, W_N^5, \ldots, W_N^{N-1}\}$$

and associate a polynomial with each of them:

(1.6)
$$b_1(x) = \prod_{x_i \in S_1} (x - x_i) = x^{\frac{N}{2}} - 1 \qquad \text{and} \qquad b_2(x) = \prod_{x_i \in S_2} (x - x_i) = x^{\frac{N}{2}} + 1$$

Now polynomial division can be employed

$$u(x) = b_k(x) q_k(x) + r_k(x) \tag{1.7}$$

to reduce the problem of evaluating $u(x)$ at S_0 to two problems involving the evaluation of $r_1(x)$ at S_1 and $r_2(x)$ at S_2. Indeed, (1.6) and (1.7) imply that $u(x_k) = r_1(x_k)$ if $x_k \in S_1$. Similarly, $u(x_k) = r_2(x_k)$ if $x_k \in S_2$.

However, in order to accomplish $O(N \log N)$ time, the problem needs to be reduced to the evaluation of $\frac{n}{2} - 1$ degree polynomials *at exactly the same set* S_1. Hence one more step is needed: one of these lower degree evaluations must be obtained – in no more than $O(N)$ operations – from the other.

The following picture illustrates this:

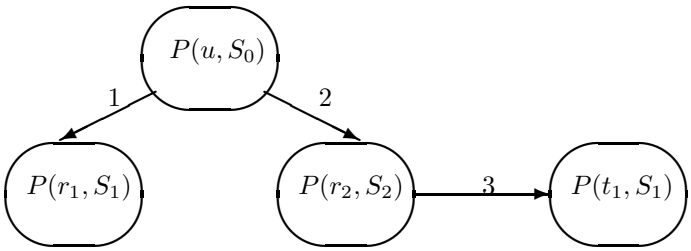

Figure 1. Halving the size of the problem. $P(u, S_0)$ denotes the evaluation of $u(x)$ at the set S_0; other terms are defined similarly.

To derive the FFT, it remains to show how to implement branch 3 in the above illustration, i.e. how to reduce the evaluations of $r_2(x)$ at S_2 to evaluations at S_1. Since

$$\underbrace{W_N^{2k+1}}_{\in S_2} = W_N \cdot \underbrace{W_N^{2k}}_{\in S_1}$$

it follows that

$$r_2(W_N^{2k+1}) = t_1(W_N^{2k})$$

where the coefficients of $t_1(x)$ can be obtained from the ones of $r_2(x)$ as follows:

$$r_2(x) = a_0 + a_1 x + a_2 x^2 + \ldots, \quad \text{and} \quad t_1(x) = a_0 + (a_1 W_N)x + (a_2 W_N^2)x^2 + \ldots \tag{1.8}$$

We next provide a purely matrix formulation of the latter derivation. Though it might look a bit artificial at first glance, it will provide the most transparent approach to deriving algorithms for the different versions of DCT and the DST at once.

1.2.3. *Companion Matrix Derivation of the FFT.* Let

$$A = \begin{bmatrix} 0 & I_{n-1} \\ -u_0 & -u_1, \ldots, -u_{N-1} \end{bmatrix} \tag{1.9}$$

be the companion matrix of the polynomial $u(x) = x^n + u_{N-1}x^{N-1} + u_{N-2}x^{N-2} + \cdots + u_1 x + u_0$. The following well-known result shows how to divide polynomials using matrix manipulations:

Proposition 1.1. Let $u(x)$ and $b(x)$ be given, and let the coefficients of $q(x)$ and $r(x)$ be determined by $u(x) = q(x)b(x) + r(x)$. Form the matrix $[D \;\; E] = \begin{bmatrix} I & 0 \end{bmatrix}_{(t+1,n-1)} \cdot b(A)$, where $t = deg\ q(x)$.

Then, the coefficients of $q(x)$ can be obtained from:

$$\begin{bmatrix} q_0 & q_1 & \cdots & q_{t-1} & 1 \end{bmatrix} E = 0 \tag{1.10}$$

and then the coefficients of $r(x)$ can be obtained from:

$$-\begin{bmatrix} q_0 & q_1 & \cdots & q_1 & 1 \end{pmatrix} D = \begin{bmatrix} r_0 & r_1 & \cdots & r_{m-1} \end{bmatrix} \tag{1.11}$$

where $m = deg\ b(x)$

Our goal here is to use this technique to accomplish the divisions by $b_1(x)$ and $b_2(x)$ in Figure 1. Since $b_1(x)$ and $b_2(x)$ are simple, the corresponding matrices $X_1 = [D_1 \ \ E_1]$ and $X_2 = [D_2 \ \ E_2]$ are simple as well:

$$X_1 = \left[\begin{array}{cccc|ccc} -1 & 0 & \cdots & & 1 & 0 & \dots \\ 0 & -1 & \cdots & & 0 & 1 & \cdots \\ \vdots & \cdots & \ddots & & \vdots & \ddots & \cdots \\ \vdots & & & -1 & \vdots & & 1 \\ -u_0 & -u_1 & \cdots & -u_{\frac{N}{2}-1} - 1 & -u_{\frac{N}{2}} & \dots & -u_{N-1} \end{array}\right] \tag{1.12}$$

$$X_2 = \left[\begin{array}{cccc|ccc} 1 & 0 & \cdots & & 1 & 0 & \dots \\ 0 & 1 & \cdots & & 0 & 1 & \cdots \\ \vdots & \cdots & \ddots & & \vdots & \ddots & \cdots \\ \vdots & & & 1 & \vdots & & 1 \\ -u_0 & -u_1 & \cdots & -u_{\frac{N}{2}-1} + 1 & -u_{\frac{N}{2}} & \dots & -u_{N-1} \end{array}\right] \tag{1.13}$$

Now that explicit expressions for X_k are available, it is possible to explicitly compute the remainders. Due to the sparsity of X_k, simple expressions are available. Indeed, from (1.12) it follows that $r_1(x) = (u_{\frac{N}{2}-1} + u_{N-1})x^{\frac{N}{2}-1} + (u_{\frac{N}{2}-2} + u_{N-2})x^{\frac{N}{2}-2} + \cdots + (u_0 + u_{\frac{N}{2}})$ while (1.13) implies that $r_2(x) = (u_{\frac{N}{2}-1} - u_{N-1})x^{\frac{N}{2}-1} + (u_{\frac{N}{2}-2} - u_{N-2})x^{\frac{N}{2}-2} + \cdots + (u_0 - u_{\frac{N}{2}})$. From these expressions for the remainders, the decomposition (1.2) immediately follows, which is listed again for convenience:

$$F_N = \begin{bmatrix} odd - even \\ permutation \end{bmatrix} \begin{bmatrix} F_{\frac{N}{2}} & 0 \\ 0 & F_{\frac{N}{2}} \end{bmatrix} \begin{bmatrix} I & I \\ \varepsilon_{\frac{N}{2}} & -\varepsilon_{\frac{N}{2}} \end{bmatrix} \tag{1.14}$$

1.3. Main Results. In the rest of this paper we use the above matrix techniques to derive in a unified way four fast algorithms to compute DCT and DST of the types I and II. In each of these cases computing the fast transform means evaluation of a polynomial in the basis of Chebyshev-like polynomials. This change of basis means that instead of the companion matrix (1.9) its analogue called the comrade matrix must be used. We next highlight a modification(of the above derivation for the FFT) for the DCT-I case.

In the case of DCT-I, the transform of the sequence $\{a_0, a_1, a_2, \dots, a_{N-1}\}$ can be reduced to the polynomial evaluations of of $a(x) = a_0\frac{T_0}{\sqrt{2}} + a_1T_1 + \ldots + a_{N-1}\frac{T_{N-1}}{\sqrt{2}} + (xT_{N-1} - T_{N-2})$, where $\{T_k(x)\}$ are the Chebyshev polynomials of the first kind. For this, the comrade matrix of this polynomial can be used. This comrade matrix is as follows:

$$A = \begin{bmatrix} & \frac{1}{\sqrt{2}} & & & & & & \\ \frac{1}{\sqrt{2}} & & \frac{1}{2} & & & & & \\ & \frac{1}{2} & & \frac{1}{2} & & & & \\ & & \ddots & & \ddots & & & \\ & & & \frac{1}{2} & & \frac{1}{2} & & \\ & & & & \frac{1}{2} & & \frac{1}{\sqrt{2}} & \\ -\frac{a_0}{2} & -\frac{a_1}{2} & -\frac{a_2}{2} & \cdots & & -\frac{a_{N-2}}{2}+\frac{1}{\sqrt{2}} & -\frac{a_{N-1}}{2} & \end{bmatrix} \tag{1.15}$$

The techniques described above for the FFT can now be applied to the DCT-I. The corresponding (new) set of nodes S_0 can be divided into two sets, an "even" set S_1 and an "odd" set S_2. The polynomial $b_1(x)$ is chosen such that the set of roots of $b_1(x)$ is S_1. Then, $a(x)$ is divided by $b_1(x)$ and the problem of evaluating $a(x)$ at S_1 is reduced to the problem of evaluating the remainder of this division at S_1. The matrices D_1 and E_1 (DCT-I counterparts of the ones in (1.12))that used here to divide $a(x)$ by $b_1(x)$ are:

$$D_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & -\frac{\sqrt{2}}{4} & 0 \\ 0 & 0 & 0 & -\frac{1}{4} & 0 & \frac{1}{4} \\ 0 & 0 & \iddots & 0 & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & 0 & \iddots & 0 & 0 \\ -\frac{\sqrt{2}}{4} & 0 & \frac{1}{4} & 0 & 0 & 0 \\ -a_N\frac{\sqrt{2}}{4} & -a_{N-1}\frac{\sqrt{2}}{4} & \cdots & & & -a_{\frac{N+1}{2}}\frac{\sqrt{2}}{4} \end{bmatrix}$$

$$E_1 = \begin{bmatrix} \frac{\sqrt{2}}{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 & 0 & 0 \\ -\frac{1}{4} & 0 & -\frac{1}{4} & 0 & 0 & 0 \\ 0 & -\frac{1}{4} & 0 & \ddots & 0 & 0 \\ 0 & 0 & \ddots & 0 & -\frac{1}{4} & 0 \\ 0 & 0 & 0 & -\frac{1}{4} & 0 & \frac{\sqrt{2}}{4} \\ -a_{\frac{N-1}{2}}\frac{\sqrt{2}}{4} & & \cdots & -a_3\frac{\sqrt{2}}{4} & -a_2\frac{\sqrt{2}}{4} & -a_1\frac{\sqrt{2}}{4} \end{bmatrix}$$

To compute the remainder the equations $q \cdot E_1 = 0$ and $r = -qD_1$ must be solved. But note that E_1 is sparse! This means the equation $q \cdot E_1 = 0$ is very easy to solve. In fact explicit formulas for the remainder – which do not require any multiplications and only $O(N)$ additions – can be derived from these expressions for D_1 and E_1.

The structure of D_1 and E_1 also provides a way to put the result back together once the two smaller polynomial evaluations are complete:

$$F_N = \begin{bmatrix} odd-even \\ permutation \end{bmatrix} \begin{bmatrix} F_{\frac{N+1}{2}} & 0 \\ 0 & GF_{\frac{N+1}{2}}\widehat{G} \end{bmatrix} \begin{bmatrix} I_{\frac{N+1}{2},\frac{N+1}{2}} & \widehat{I}_{\frac{N+1}{2},\frac{N-1}{2}} \\ \check{I}_{\frac{N-1}{2},\frac{N+1}{2}} & -\check{I}_{\frac{N-1}{2},\frac{N-1}{2}} \end{bmatrix} \tag{1.16}$$

where G, $\widehat{G}$, $\widehat{I}$,$\check{I}$, and $\widetilde{I}$ are some simple sparse matrices with $O(N)$ complexity of multiplication. Since (1.16) reduces the problem of computing F_N to the problem of computing $F_{\frac{N+1}{2}}$, it can be applied repeatedly as the basis for a divide-and-conquer algorithm with the cost of $O(N \log N)$.

2. Discrete Transforms via Barnett's Comrade Matrices

The four transforms in table 1 can be reformulated in terms of the classical Chebysehev polynomials $T_k = \cos(k \arccos x)$ and $U_k = \frac{\sin((k+1)\arccos x)}{\sin \arccos x}$, or their minor modifications. In this chapter, it is shown how to use this fact to derive the doubling algorithms described in the introduction.

2.1. Polynomial Evaluation. Let F_N be the transform matrix. i.e. the matrix C in case of the cosine transforms and S in case of the sine transforms (these F_N's are listed in Table 1). Following [**KO96**] we represented them as follows:

$$F_N = W_Q \cdot V_Q \tag{2.1}$$

where

$$V_Q = \begin{bmatrix} Q_0(x_1) & Q_1(x_1) & \cdots & Q_{N-1}(x_1) \\ Q_0(x_2) & Q_1(x_2) & \cdots & Q_{N-1}(x_2) \\ \vdots & \vdots & & \vdots \\ Q_0(x_N) & Q_1(x_N) & \cdots & Q_{N-1}(x_N) \end{bmatrix} \tag{2.2}$$

and where the orthogonal(Chebyshev-like) polynomials $Q_k(x)_{k=0}^{n-1}$ are given in Table 2, the nodes x_k are the roots of the polynomial Q_n and are given in Table 3, and finally W_Q is the weight matrix specified in Table 4 [**KO96**].

	$\{Q_0,$	Q_1	$\ldots,$	Q_{N-2}	$Q_{N-1}\}$
DCT-I	$\{\frac{1}{\sqrt{2}}T_0,$	$T_1,$	$\ldots,$	$T_{N-2},$	$\frac{1}{\sqrt{2}}T_{N-1}\}$
DCT-II	$\{U_0,$	$U_1 - U_0,$	$\ldots,$		$U_{N-1} - U_{N-2}\}$
DST-I	$\{U_0,$	$U_1,$	$\ldots,$		$U_{N-1}\}$
DST-II	$\{U_0,$	$U_1 + U_0,$	$\ldots,$		$U_{N-1} + U_{N-2}\}$

Table 2. First n polynomials.

For each of the eight systems $\{Q_k(x)\}_{k=0}^{N-1}$ the above Table 2 lists the first n polynomials.

To specify V_Q the nodes $\{x_k\}_{k=1}^{n}$ must also be specified, or, equivalently, the last polynomial $Q_n(x)$, which is done in Table 3.

	Q_N	zeros of Q_N
DCT-I	$xT_{N-1} - T_{N-2}$	$\{\cos(\frac{k\pi}{N-1})\}_0^{N-1}$
DCT-II	$U_N - 2U_{N-1} + U_{N-2}$	$\{\cos(\frac{k\pi}{N})\}_0^{N-1}$
DST-I	U_N	$\{\cos(\frac{k\pi}{N+1})\}_1^{N}$
DST-II	$U_N + 2U_{N-1} + U_{N-2}$	$\{\cos(\frac{k\pi}{N})\}_1^{N}$

Table 3. The last polynomial $Q_N(x)$ of DCT/DST I–IV.

DCT-I	$C_N^I = W_Q \cdot V_Q$ with
	$W_Q = \sqrt{\frac{2}{N-1}}\text{diag}(\frac{1}{\sqrt{2}}, 1, \ldots, 1, \frac{1}{\sqrt{2}})$
DCT-II	$C_N^{II} = W_Q \cdot V_Q$ with
	$W_Q = \sqrt{\frac{2}{N}}\text{diag}(\frac{1}{\sqrt{2}}, \cos(\frac{\pi}{2N}), \ldots, \cos(\frac{(N-1)\pi}{2N}))$
DST-I	$S_N^I = W_Q \cdot V_Q$ with
	$W_Q = \sqrt{\frac{2}{N+1}}\text{diag}(\sin(\frac{\pi}{N+1}), \ldots, \sin(\frac{N\pi}{N+1}))$
DST-II	$S_N^{II} = W_Q \cdot V_Q$ with
	$W_Q = \sqrt{\frac{2}{N}}\text{diag}(\sin(\frac{\pi}{2N}), \ldots, \sin(\frac{(N-1)\pi}{2N}), \frac{1}{\sqrt{2}}\sin(\frac{\pi}{2}))$

Table 4.The W_Q for DCT/DST I–IV

So the problem of computing sine and cosine transforms can be reduced to the problem of computing a matrix-vector product. Since the matrix in question is a Chebyshev-Vandermonde matrix, computing a sine or cosine transform is equivalent to evaluating a polynomial:

$$a(x) = \sum_{k=0}^{nN-1} a_k Q_k(x) \tag{2.3}$$

The only way in which this differs from the FFT is in that a different basis is used.

Now the steps described in Figure 1 should be implemented for the transforms considered in this paper. However, in order execute branches 1 and 2 in Figure 1 an appropriate way of dividing polynomials expressed in Chebyshev like bases is needed. Next, a nice algorithm for doing so is recalled.

2.2. Barnett's Polynomial Division. In [**B84**] Barnett gave a good algorithm for dividing polynomials using the comrade matrix, which will be summarized here. The advantages of this approach in are multiple: it is fast, requiring very few operations; it is convenient, as all the quantities the algorithm requires are readily available; and it is general, applying to all the transforms considered in this paper. Barnett's theorem is a direct generalization of Proposition 1.1, which was used (in the introduction) in the case of the DFT.

Before introducing Barnett's theorem, however, a few definitions must be given. Let us say a polynomial basis $P_i(x)$ is defined by:

$$P_0(x) = 1, \quad P_1(x) = \alpha_1 x + \beta_1, \quad P_i(x) = (\alpha_i x + \beta_i)P_{i-1}(x) - \gamma_i P_{i-2}(x), \tag{2.4}$$

$(i = 2, 3, \ldots, n)$, and a polynomial $a(x)$ monic in this basis:

$$a(x) = P_n(x) + a_1 P_{n-1}(x) + \ldots + a_n P_0(x) \tag{2.5}$$

Then, the comrade matrix (a generalization of the companion matrix) of $a(x)$ is defined [**B84**] to be :

$$A = \begin{bmatrix} \frac{-\beta_1}{\alpha_1} & \frac{1}{\alpha_1} & 0 & 0 & & \\ \frac{\gamma_2}{\alpha_2} & \frac{-\beta_2}{\alpha_2} & \frac{1}{\alpha_2} & 0 & 0 & \\ 0 & \frac{\gamma_3}{\alpha_3} & \frac{-\beta_3}{\alpha_3} & \frac{1}{\alpha_3} & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \frac{\gamma_{n-1}}{\alpha_{n-1}} & \frac{-\beta_{n-1}}{\alpha_{n-1}} & \frac{1}{\alpha_{n-1}} \\ \frac{-a_n}{\alpha_n} & \frac{-a_{n-1}}{\alpha_n} & \ldots & \frac{-a_3}{\alpha_n} & \frac{-a_2+\gamma_n}{\alpha_n} & \frac{-a_1-\beta_n}{\alpha_n} \end{bmatrix} \tag{2.6}$$

It is now possible to state Barnett's theorem on polynomial division. For convenience, henceforth a polynomial monic with respect to P_i will be referred to as just monic.

THEOREM 2.1. *[Barnett] Let $a(x)$ be as defined above, and let $b(x)$ be another monic polynomial of the m'th degree. The choice $\mu(n,m) = \frac{\alpha_{t+1}\alpha_{t+2}\cdots\alpha_n}{\alpha_1\alpha_2\cdots\alpha_m}$ makes the polynomial $q(x)$ in*

$$a(x) = \mu(n,m)[q(x)b(x) + r(x)] \tag{2.7}$$

be monic. Moreover, it is true that

$$\begin{bmatrix} q_t & q_{t-1} & \cdots & q_1 & 1 \end{bmatrix} \cdot X + [r_{m-1}, \cdots, r_1, r_0, 0, \cdots, 0] = 0 \tag{2.8}$$

where

$$X = \begin{bmatrix} I & 0 \end{bmatrix}_{(t+1,n-1)} \cdot b(A) \tag{2.9}$$

Further, the rows of X satisfy the following recurrence relationship:

(2.10)

$$\rho_1 = \begin{bmatrix} b_m & b_{m-1} & \cdots & b_0 & 0 & \cdots & 0 \end{bmatrix}, \quad \rho_i = \rho_{i-1}(\alpha_{i-1}A + \beta_{i-1}I_n) - \gamma_{i-1}\rho_{i-2}$$

where

$$\rho_i = \textit{the i'th row of } X, \;\; \textit{deg } q(x) = t \tag{2.11}$$

A further simplification arises if it is noted that X can be split into two multiplier matrices: $X = \begin{bmatrix} D & E \end{bmatrix}$, so that the above theorem can be adapted to compute the remainder as follows:

(1) Solve $q \cdot E = 0$. Since E will be sparse, this will take very few operations.
(2) Compute the remainder using $-q \cdot D = r$

2.3. Branches 1 and 2 in Figure 1. We conclude this section with explicit formulas which can be used for the computations of remainders. In other words, numerical expressions for D_k and E_k mentioned in the previous section are provided which can be used to practically implement the algorithms described in this paper.

2.3.1. *Choice of Polynomials For DCT/DST I and II.* Given the sequence $\{y_0, y_1, \ldots, y_{N-1}\}$ it is needed to compute the polynomial evaluations of $\Sigma_{k=0}^{M-1} y_k Q_k$ at the roots of Q_N. Since Q_N is obviously zero at its own roots, instead it is better to consider the slightly more convenient polynomial $Y(x) = \Sigma_{k=0}^{N-1} y_k Q_k(x) + \tau Q_N$, where $\tau = \begin{cases} \frac{1}{\sqrt{2}} & \text{DCT-I or DST-I} \\ 1 & \text{DCT-II or DST-II} \end{cases}$ This addition of a scaled Q_n makes it possible to construct the comrade matrix without scaling the original polynomial. A full list of these comrade matrices is given in following table. (It can be obtained from the fact that all families of polynomials $\{Q_k\}$ in Tables 2 and 3 satisfy (almost, i.e., for $k = 3, 4, \ldots, n-2$ only) the same recurrence relations $Q_k(x) = 2xQ_{k-1}(x) - Q_{k-2}(x)$ used to define the Chebyshev polynomials $\{T_k(x)\}$. The difference between $\{T_k(x)\}$ and $\{Q_k(x)\}$ is only in recurrences for the "bordering" polynomials $Q_0(x), Q_1(x)$ and $Q_{n-1}(x), Q_n(x)$.

DCT-I	$\begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \ddots & 0 & 0 \\ 0 & 0 & \ddots & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \\ -\frac{y_0}{2} & -\frac{y_1}{2} & \cdots & -\frac{y_{N-3}}{2} & -\frac{y_{N-2}}{2}+\frac{1}{\sqrt{2}} & -\frac{y_{N-1}}{2} \end{bmatrix}$
DCT-II	$\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \ddots & 0 & 0 \\ 0 & 0 & \ddots & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \\ -\frac{y_0}{2} & -\frac{y_1}{2} & \cdots & -\frac{y_{N-3}}{2} & -\frac{y_{N-2}}{2}+\frac{1}{2} & -\frac{y_{N-1}}{2}+\frac{1}{2} \end{bmatrix}$
DST-I	$\begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \ddots & 0 & 0 \\ 0 & 0 & \ddots & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ -\frac{y_0}{2} & -\frac{y_1}{2} & \cdots & -\frac{y_{N-3}}{2} & -\frac{y_{N-2}}{2}+\frac{1}{2} & -\frac{y_{N-1}}{2} \end{bmatrix}$
DST-II	$\begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \ddots & 0 & 0 \\ 0 & 0 & \ddots & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \\ -\frac{y_0}{2} & -\frac{y_1}{2} & \cdots & -\frac{y_{N-3}}{2} & -\frac{y_{N-2}}{2}+\frac{1}{2} & -\frac{y_{N-1}}{2}-\frac{1}{2} \end{bmatrix}$

Table 5. Comrade Matrices

Next, the polynomials $b_1(x)$ and $b_2(x)$ for the divisions in Figure 1 must be calculated. Recall that these are obtained by dividing the set of nodes into an two parts, and associating a polynomial with each part. Some elementary calculation will yield the following table for $b_1(x)$ and $b_2(x)$:

	$b_1(x)$	$b_2(x)$
DCT-I	$\frac{1}{2}(T_{\frac{N+1}{2}} - T_{\frac{N-1}{2}})$	$2T_{\frac{N-1}{2}}$
DCT-II	$U_{\frac{N}{2}} - 2U_{\frac{N}{2}-1} + U_{\frac{N}{2}-2}$	$U_{\frac{N}{2}} - U_{\frac{N}{2}-2}$
DST-I	$U_{\frac{N-1}{2}}$	$U_{\frac{N+1}{2}} - U_{\frac{N-3}{2}}$
DST-II	$U_{\frac{N}{2}} + 2U_{\frac{N}{2}-1} + U_{\frac{N}{2}-2}$	$U_{\frac{N}{2}} - U_{\frac{N}{2}-2}$

Table 6. $b_1(x)$ and $b_2(x)$ for DCT/DST I,II.

Note that in the case of DCT-I and DST-I, it is assumed that N is odd, while in the case of DCT-II and DST-II it is assumed that N is even.

2.4. Division Matrices. Having obtained the polynomials $b_1(x)$ and $b_2(x)$ associated with a step of reduction and the comrade matrices A, we now have everything needed in Theorem 2.1. Therefore, the matrices X_k associated with division using (2.11) can now be computed. Each matrix is split up into $X = [D \quad E]$. The results are given in the tables that follow.

$$D_1 = \left[\begin{array}{cccccc} 0 & 0 & 0 & 0 & -\frac{\sqrt{2}}{4} & 0 \\ 0 & 0 & 0 & -\frac{1}{4} & 0 & \frac{1}{4} \\ 0 & 0 & \iddots & 0 & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & 0 & \iddots & 0 & 0 \\ -\frac{\sqrt{2}}{4} & 0 & \frac{1}{4} & 0 & 0 & 0 \\ \hline -a_N\frac{\sqrt{2}}{4} & -a_{N-1}\frac{\sqrt{2}}{4} & \cdots & & & -a_{\frac{N+1}{2}}\frac{\sqrt{2}}{4} \end{array}\right]$$

$$E_1 = \left[\begin{array}{cccccc} \frac{\sqrt{2}}{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 & 0 & 0 \\ -\frac{1}{4} & 0 & -\frac{1}{4} & 0 & 0 & 0 \\ 0 & -\frac{1}{4} & 0 & \ddots & 0 & 0 \\ 0 & 0 & \ddots & 0 & -\frac{1}{4} & 0 \\ 0 & 0 & 0 & -\frac{1}{4} & 0 & \frac{\sqrt{2}}{4} \\ \hline -a_{\frac{N-1}{2}}\frac{\sqrt{2}}{4} & & \cdots & -a_3\frac{\sqrt{2}}{4} & -a_2\frac{\sqrt{2}}{4} & -a_1\frac{\sqrt{2}}{4} \end{array}\right]$$

$$D_2 = \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \iddots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \sqrt{2} & 0 & 0 & 0 & \\ \hline -\sqrt{2}a_N & -\sqrt{2}a_{N-1}+1 & -\sqrt{2}a_{n-2} & \cdots & \end{array}\right]$$

$$E_2 = \left[\begin{array}{cccccc} \sqrt{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & \sqrt{2} \\ \hline -a_{\frac{N+1}{2}}\sqrt{2} & -\sqrt{2}a_{\frac{N-1}{2}} & \cdots & -\sqrt{2}a_2+1 & -\sqrt{2}a_1 & \end{array}\right] xd$$

Table 7. Division matrices $X_k = [D_k \;\; E_k]$ for DCT-I.

$$D_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 \\ 0 & 0 & 0 & -1 & 2 & -2 \\ 0 & 0 & -1 & 2 & -2 & 2 \\ & \ddots & \ddots & \ddots & \ddots & \\ -1 & 2 & -2 & 2 & -2 & \cdots \\ \hline -a_N+1 & -a_{N-1}-2 & -a_{N-2}+2 & -a_{N-3}-2 & \cdots & \end{bmatrix}$$

$$E_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ -2 & 2 & -2 & 1 & 0 & 0 & 0 & 0 \\ 2 & -2 & 2 & -2 & 1 & 0 & 0 & 0 \\ -2 & 2 & -2 & 2 & -2 & 1 & 0 & 0 \\ & & & \ddots & \ddots & \ddots & \ddots & \\ & & & & \ddots & \ddots & \ddots & 0 \\ & & & & -2 & 2 & -2 & 1 \\ \hline \cdots & -a_6+2 & -a_5-2 & -a_4+2 & -a_3-2 & -a_2+2 & -a_1-1 & \end{bmatrix}$$

$$D_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ & & \ddots & & & \\ 0 & 0 & & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ \hline -a_N+1 & -a_{N-1} & -a_{N-2} & \cdots & & -a_{\frac{N}{2}+1} \end{bmatrix}$$

$$E_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ \hline -a_{\frac{N1}{2}} & & \cdots & -a_2 & -a_1+1 \end{bmatrix}$$

Table 8. Division matrices $X_k = [D_k \;\; E_k]$ for DCT-II.

$$D_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ & \ddots & \ddots & \ddots & & & \\ 0 & & & & & & \\ 1 & 0 & 1 & 0 & \cdots & & \\ \hline -a_N & -a_{N-1}+1 & -a_{N-2} & -a_{N-3}+1 & \cdots & & \end{bmatrix}$$

$$E_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ & & & \ddots & \ddots & \ddots & \ddots & \\ & & & & \ddots & \ddots & \ddots & 0 \\ & & & & 0 & 1 & 0 & 1 \\ \hline & & \cdots & -a_4+1 & -a_3 & -a_2+1 & -a_1 & \end{bmatrix}$$

$$D_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ & & \ddots & & & & \\ 0 & 0 & & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline -a_N & -a_{N-1} & -a_{N-2} & \cdots & & & -a_{\frac{N+1}{2}} \end{bmatrix}$$

$$E_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \hline -a_{\frac{N-1}{2}} & & \cdots & -a_2 & -a_1 \end{bmatrix}$$

Table 9. Division matrices $X_k = [D_k \;\; E_k]$ for DST-I.

$$D_1 = \left[\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \\
0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 \\
0 & 0 & 0 & 0 & 1 & 2 & 2 & 2 \\
0 & 0 & 0 & 1 & 2 & 2 & 2 & 2 \\
 & \ddots & \ddots & \ddots & \ddots & \ddots & & \\
0 & \ddots & \ddots & \ddots & \ddots & & & \\
1 & 2 & 2 & 2 & & & & \\
\hline
-a_N+1 & -a_{N-1}+2 & -a_{N-2}+2 & -a_{N-3}+2 & \cdots & & &
\end{array}\right]$$

$$E_1 = \left[\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\
2 & 2 & 2 & 1 & 0 & 0 & 0 & 0 \\
2 & 2 & 2 & 2 & 1 & 0 & 0 & 0 \\
2 & 2 & 2 & 2 & 2 & 1 & 0 & 0 \\
 & & & \ddots & \ddots & \ddots & \ddots & 0 \\
 & & & & 2 & 2 & 2 & 1 \\
\hline
 & & \cdots & -a_4+2 & -a_3+2 & -a_2+2 & -a_1+2 & -a_1+1
\end{array}\right]$$

$$D_2 = \left[\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & \ddots & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 \\
\hline
-a_N+1 & -a_{N-1} & -a_{N-2} & \cdots & & -a_{\frac{N}{2}+1}
\end{array}\right]$$

$$E_2 = \left[\begin{array}{ccccc}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
 & & \ddots & & \\
0 & 0 & \ddots & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 \\
\hline
-a_{\frac{N}{2}} & & \cdots & -a_2 & -a_1-1
\end{array}\right]$$

Table 10. Division matrices $X_k = [D_k \;\; E_k]$ for DST-II.

2.5. Matrix Decompositions.

2.5.1. *Efficient Implementation of Division Formulas.* Having obtained the division formulas, it is now possible to derive the matrix decompositions of V_Q for all the transforms considered here. First, however, the following definitions are necessery:

Let V_{even} be the following polynomial-Chebyshev matrix: $V_{even} = [Q_{2i}(x_{2j})]$ where

$$i, j = \begin{cases} 0, 1, 2, \ldots, \frac{N}{2} - 1 & \text{in the case of DCT I /II} \\ 1, 2, \ldots, \frac{N}{2} & \text{in the case of DST I/II} \end{cases}$$

Let V_{odd} is defined similarly, with the indexes now taking odd values. With these definitions, the following formula holds:

$$V_Q = \begin{bmatrix} odd - even \\ permutation \end{bmatrix} \begin{bmatrix} V_{even} & 0 \\ 0 & V_{odd} \end{bmatrix} H \tag{2.12}$$

where H can be shown to be:

	H
DCT-I	$\begin{bmatrix} I_{\frac{N+1}{2},\frac{N+1}{2}} & \widehat{I}_{\frac{N+1}{2},\frac{N-1}{2}} \\ \check{I}_{\frac{N-1}{2},\frac{N+1}{2}} & -\widetilde{I}_{\frac{N-1}{2},\frac{N-1}{2}} \end{bmatrix}$
DCT-II	$\begin{bmatrix} I_{\frac{N}{2},\frac{N}{2}} & \widetilde{I}_{\frac{N}{2},\frac{N}{2}} \\ I_{\frac{N}{2},\frac{N}{2}} & -\widetilde{I}_{\frac{N}{2},\frac{N}{2}} \end{bmatrix}$
DST-I	$\begin{bmatrix} \check{I}_{\frac{N-1}{2},\frac{N+1}{2}} & -I_{\frac{N-1}{2},\frac{N-1}{2}} \\ I_{\frac{N+1}{2},\frac{N+1}{2}} & -\widehat{I}_{\frac{N+1}{2},\frac{N-1}{2}} \end{bmatrix}$
DST-II	$\begin{bmatrix} I_{\frac{N}{2},\frac{N}{2}} & -\widetilde{I}_{\frac{N}{2},\frac{N}{2}} \\ I_{\frac{N}{2},\frac{N}{2}} & \widetilde{I}_{\frac{N}{2},\frac{N}{2}} \end{bmatrix}$

Table 11. The matrix H for DCT/DST I,II.

where $\widetilde{I}$ is the involution matrix, $\widehat{I}_{\frac{N+1}{2},\frac{N-1}{2}} = \begin{bmatrix} I_{\frac{N-1}{2},\frac{N-1}{2}} \\ 0 \end{bmatrix}$, and $\check{I}_{\frac{N-1}{2},\frac{N+1}{2}} = \begin{bmatrix} I_{(\frac{N-1}{2},\frac{N-1}{2})} & 0 \end{bmatrix}$.

These formulas are the direct analogues of (1.2). They make it possible to decompose the problem at hand into two problems of half the size. In the next chapter, a method for obtaining V_{odd} from V_{even} will be shown – thereby producing an algorithm which achieves $O(N \log N)$ complexity. Meanwhile, a proof of the formulas for H is provided based on the properties of the division matrices X_k.

2.5.2. *Proof.* The formulas for H are derived very similarly in all four cases. Therefore, the proof for the DCT-I case is provided here with the caveat that all of the other proofs are (almost) exactly the same.

In the case of DCT-I: let us partition X as $X = \begin{bmatrix} \bar{X} \\ \bar{a} \end{bmatrix}$ where $\bar{a}$ is the last row of X. This partition makes sense since, as can be seen in tables 7-10, the last row is the only one that depends on the coefficients of the polynomial being evaluated. Let us do the same thing with D and E; D can be partitioned as $D = \begin{bmatrix} \bar{D} \\ \bar{a_1} \end{bmatrix}$ and E can correspondingly be partitioned as $E = \begin{bmatrix} \bar{E} \\ \bar{a_2} \end{bmatrix}$.

Next, consider the equation $\begin{bmatrix} q_t & q_{t-1} & \cdots & q_1 & 1 \end{bmatrix} \cdot E = 0$. Labelling $\bar{q} = \begin{bmatrix} q_t & q_{t-1} & \cdots & q_1 \end{bmatrix}$, it can be rewritten as $\bar{q} = -\bar{a_2}\bar{E}^{-1}$. Further, the equation $q \cdot D = -r$ can be rewritten as $\bar{q}\bar{D} + \bar{a_1} = -r$. Using the expression for $\bar{q}$, this last equation reduces to:

$$r = -(\bar{a_1} - \bar{a_2}\bar{E}^{-1}\bar{D}) \tag{2.13}$$

However, due to the symmetry of X, the expression $\bar{E}^{-1}\bar{D}$ reduces nicely. Indeed, using the matrices E_1, D_1, E_2, D_2 from table 6, it can be seen that in the case of dividing by $b_1(x)$, $\bar{E}^{-1}\bar{D} = \begin{bmatrix} \widetilde{I} & 0 \end{bmatrix}$ and in the case of dividing by $b_2(x)$, $\bar{E}^{-1}\bar{D} = \widetilde{I}$. Using Barnett's theorem, it is clear that $\mu = \frac{4}{\sqrt{2}}$ in this case. From

this, two obvious expressions for the remainder coefficients follow:

$$r_1 = \begin{bmatrix} a_1 + a_N \\ a_2 + a_{N-1} \\ \vdots \\ a_{\frac{N-1}{2}} + a_{\frac{N+3}{2}} \\ a_{\frac{N+1}{2}} \end{bmatrix}, \quad r_1 = \begin{bmatrix} a_1 - a_N \\ a_2 - a_{N-1} \\ a_3 - a_{N-2} \\ \vdots \\ a_{\frac{N-1}{2}} - a_{\frac{N+3}{2}} \end{bmatrix} \tag{2.14}$$

However, a careful look at the matrix H for DCT-I in table 7 shows that it satisfies $H \cdot \begin{bmatrix} a_N \\ a_{N-1} \\ \vdots \\ a_1 \end{bmatrix} = \begin{bmatrix} r_1 \\ \hline r_2 \end{bmatrix}$.

However, this is exactly the equality H needs to satisfy in order to be correct. Indeed, since the evaluations of $a(x)$ at the "even" points are equal to the evaluations of r_1, the top half of the vector $H\vec{a}$ after the odd-even permutation is applied should be composed of r_1. Similarly, the bottom half should be r_2. Since H satisfies this function, it follows that the formula is correct.

3. Branch 3

In the previous section, it has been shown how to efficiently go through branches one and two in Figure 1. In other words, it was explained how to reduce the problem of evaluating V_Q to the problem of evaluating two *different* matrices half the size: V_{even} and V_{odd}.

However, in order for the algorithms presented here to have $O(N \log N)$ running time, it is necessery that the problem of evaluating V_Q be reduced to two exactly the same problems of half the size. This is reflected in branch three of Figure 1, where it is shown that one of these half-sized problems must be reduced to the other.

This task that is considered in this chapter. It is shown how to obtain V_{odd} from V_{even}. Namely, matrices G and $\widehat{G}$ are found with low complexity of multiplication such that $V_{odd} = GV_{even}\widehat{G}$.

The approach is based on some simple trigonometric formulas which are applied to determine formulas for odd nodes in terms of even nodes.

The following table summarizes four formulas which will used. They can be checked using direct computation.

DCT-I	$Q_k(x_{2j+1}) = \frac{Q_k(x_{2j+2}) + Q_k(x_{2j})}{2Q_k(x_1)}$
DCT-II	$Q_k(x_{2j+1}) = \frac{Q_k(x_{2j+2})\cos(\frac{1}{2}\arccos x_{2j+2}) + Q_k(x_{2j})\cos(\frac{1}{2}\arccos x_{2j})}{2\cos(\frac{(2j+1)\pi}{2N})\cos(\frac{(2k+1)\pi}{2N})}$
DST-I	$Q_k(x_{2j+1}) = \frac{Q_k(x_{2j+2})\sin(\arccos x_{2j+2}) + Q_k(x_{2j})\sin(\arccos x_{2j})}{2\cos(\frac{(2j+1)\pi}{N+1})\cos(\frac{(k+1)\pi}{N+1})}$
DST-II	$Q_k(x_{2j+1}) = \frac{Q_k(x_{2j+2})\sin(\frac{1}{2}\arccos x_{2j+2}) + Q_k(x_{2j})\sin(\frac{1}{2}\arccos x_{2j})}{2\cos(\frac{(2j+1)\pi}{2N})\cos(\frac{(2k+1)\pi}{2N})}$

Table 12. Odd nodes in terms of even nodes

Note again that $\{Q_k\}$ is the polynomial basis for each transform and can be found in Table 2; x_k are the nodes for each transform and can be found in Table 3.

From these formulas, it is easy to derive the needed matrices G and $\widehat{G}$. These matrices are given in Table 13. Note that in the case of DST-I, the number of odd nodes is larger than the number of even nodes. Therefore, in that case the matrices G and $\widehat{G}$ satisfy: $G \begin{bmatrix} V_{even} & 0 \\ 0 & 1 \end{bmatrix} \widehat{G} = V_{odd}$.

DCT-I	$G = \begin{bmatrix} 1 & 1 & & & \\ & 1 & 1 & & \\ & & \ddots & \ddots & \\ & & & 1 & 1 \end{bmatrix}$ $\widehat{G} = \begin{bmatrix} \frac{1}{T_0(x_1)} & & & & \\ & \frac{1}{T_1(x_1)} & & & \\ & & \ddots & & \\ & & & \frac{1}{T_{\frac{N-1}{2}-1}(x_1)} & \\ 0 & 0 & \cdots & 0 & \end{bmatrix}$
DCT-II	$G = \begin{bmatrix} \frac{\cos(\frac{acosx_0}{2})}{\cos(\frac{acosx_1}{2})} & \frac{\cos(\frac{acosx_2}{2})}{\cos(\frac{acosx_1}{2})} & & & \\ & \frac{\cos(\frac{acosx_2}{2})}{\cos(\frac{acosx_3}{2})} & \frac{\cos(\frac{acosx_4}{2})}{\cos(\frac{acosx_3}{2})} & & \\ & & \ddots & \ddots & \\ & & & \frac{\cos(\frac{acosx_{N-2}}{2})}{\cos(\frac{acosx_{N-3}}{2})} & \\ & & & \frac{\cos(\frac{acosx_{N-2}}{2})}{\cos(\frac{acosx_{N-1}}{2})} & \end{bmatrix}$ $\widehat{G} = \frac{1}{2} diag\{\frac{1}{\cos(\frac{acosx_1}{2})}, \frac{1}{\cos(\frac{acosx_3}{2})}, \cdots, \frac{1}{\cos(\frac{acosx_{N-1}}{2})}\}$
DST-I	$G = \begin{bmatrix} \frac{\sin(acosx_2)}{\sin(acosx_1)} & & & \frac{(-1)^0}{\sin acosx_1} \\ \frac{\sin(acosx_2)}{\sin(acosx_3)} & \frac{\sin(acosx_4)}{\sin(acosx_3)} & & \frac{(-1)^1}{\sin acosx_3} \\ \ddots & \ddots & & \vdots \\ & & \frac{\sin(acosx_{N-1})}{\sin(acosx_{N-2})} & \frac{(-1)^{\frac{N-3}{2}}}{\sin acosx_{N-2}} \\ & & \frac{\sin(acosx_{N-1})}{\sin(acosx_N)} & \frac{(-1)^{\frac{N-1}{2}}}{\sin acosx_N} \end{bmatrix}$ $\widehat{G} = \frac{1}{2} diag\{\frac{1}{T_1(x_1)}, \frac{1}{T_2(x_1)}, \ldots, \frac{1}{T_{\frac{N-1}{2}}(x_1)}, 1\}$

Table 13. The matrices G and $\widehat{G}$

DST-II	$G = \begin{bmatrix} \frac{\sin(\frac{a\cos x_2}{2})}{\sin(\frac{a\cos x_1}{2})} & & & \\ \frac{\sin(\frac{a\cos x_2}{2})}{\sin(\frac{a\cos x_3}{2})} & \frac{\sin(\frac{a\cos x_4}{2})}{\sin(\frac{a\cos x_3}{2})} & & \\ & \ddots & \ddots & \\ & & \frac{\sin(\frac{a\cos x_{N-2}}{2})}{\sin(\frac{a\cos x_{N-1}}{2})} & \frac{\sin(\frac{a\cos x_N}{2})}{\sin(\frac{a\cos x_{N-1}}{2})} \end{bmatrix}$ $\widehat{G} = \frac{1}{2} diag\{\frac{1}{cos(\frac{\pi}{2N})}, \frac{1}{cos(\frac{3\pi}{2N})}, \frac{1}{cos(\frac{5\pi}{2N})}, \ldots, \frac{1}{cos(\frac{(N-1)\pi}{2N})}\}$

Table 13 Continued. The matrices G and $\widehat{G}$

3.1. Flow Graphs. Traditional recursive algorithms reduce the computation of F_N to the computation of $F_{\frac{N}{2}}$. Using the formulas obtained above, it is more natural to reduce V_N to V_{even} and use the identity $F_N = W_N V_N$. We provide flow graphs implementing this approach.

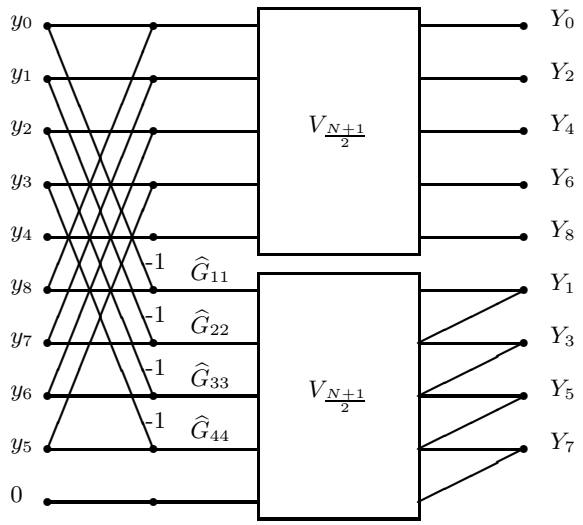

Figure 2. Flow Graph for V_N in the case of DCT-I with $N = 9$

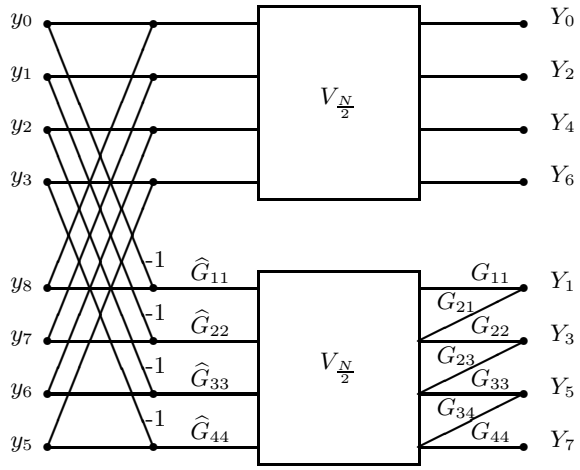

Figure 3 Flow Graph for V_N in the case of DCT-II with $N = 8$

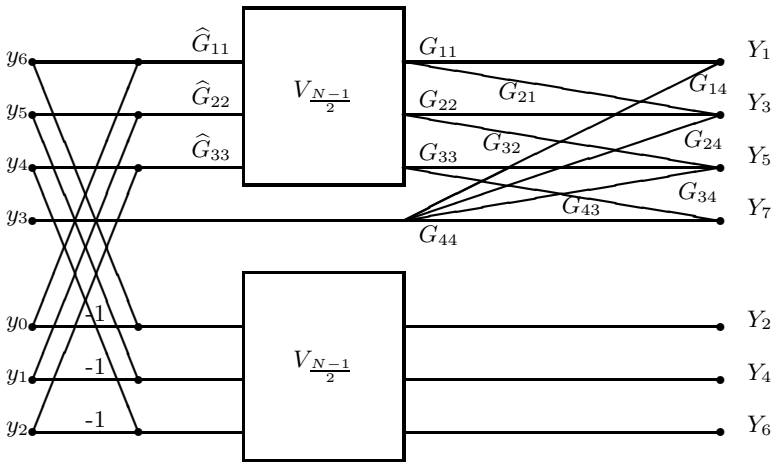

Figure 4. Flow Graph for V_N in the case of DST-I with $N = 7$

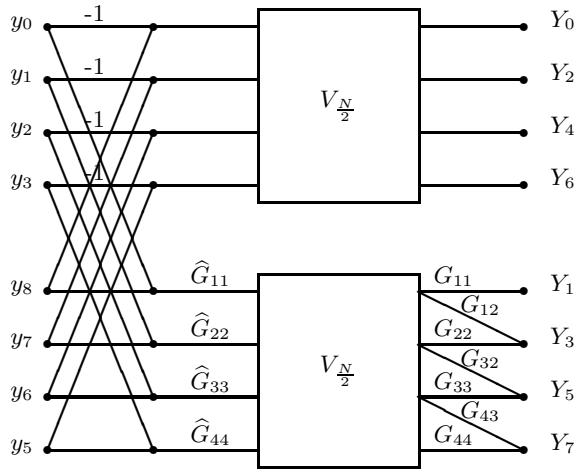

Figure 5. Flow Graph for V_N in the case of DST-II with $N = 8$

3.2. Passing to Recursions for the Transform. Thus far, the following formula has been established:

$$V_Q = \begin{bmatrix} odd - even \\ permutation \end{bmatrix} \begin{bmatrix} V_{even} & 0 \\ 0 & GV_{even}\widehat{G} \end{bmatrix} H \tag{3.1}$$

This is a recursion for the Chebyshev-Vandermonde matrix V_Q. While this recursion can be used to obtain an $O(N \log N)$ algorithm, one may wish to also see the direct recursions for F_N. As has been shown before, $F_N = W_Q V_Q$; therefore, the recursions obtained can be easily modified to compute F_N. Indeed, if $F_{even} = W_e \cdot V_{even}$ and $F_{odd} = W_o \cdot V_{odd}$, and if $P = W_Q \begin{bmatrix} odd - even \\ permutation \end{bmatrix}^{-1} \begin{bmatrix} W_e^{-1} & \\ & I \end{bmatrix}$, then it follows trivially from (3.1) that the following formula holds:

$$F_N = P \begin{bmatrix} F_{even} & 0 \\ 0 & GW_e^{-1}F_{even}\widehat{G} \end{bmatrix} H \tag{3.2}$$

where F_{even} is the transform of dimension equal to the number of even nodes in the set. In other words,

$$F_{even} = \begin{cases} F_{\frac{N+1}{2}} & \text{DCT-I} \\ F_{\frac{N-1}{2}} & \text{DST-I} \\ F_{\frac{N}{2}} & \text{DCT-II and DST-II} \end{cases} \tag{3.3}$$

Part II: DCT III and DST III.

In this part, we will consider the versions III and IV. Their relation to Vandermonde matrices will be used to derive recursion formulas for version III transforms.

4. Preliminaries

At the heart of the derivation presented in Part I is Figure 1 and the formula:

$$V_Q = \begin{bmatrix} odd-even \\ permutation \end{bmatrix} \begin{bmatrix} V_{even} & 0 \\ 0 & V_{odd} \end{bmatrix} H \tag{4.1}$$

The approach described in part I could in principle be applied to the III and IV versions of the cosine and sine transforms. These transformations are defined by the transformation matrix in table 14:

DCT-III	$C_N^{III} = \sqrt{\frac{2}{N}} \left[\eta_j \cos \frac{(2k+1)j\pi}{2N} \right]_{k,j=0}^{N-1}$	$[C_N^{III}]^{-1} = [C_N^{III}]^T = C_N^{II}$
DCT-IV	$C_N^{IV} = \sqrt{\frac{2}{N}} \left[\cos \frac{(2k+1)(2j+1)\pi}{4N} \right]_{k,j=0}^{N-1}$	$[C_N^{IV}]^{-1} = [C_N^{IV}]^T = C_N^{IV}$
DST-III	$S_N^{III} = \sqrt{\frac{2}{N}} \left[\eta_j \sin \frac{(2k-1)j}{2N}\pi \right]_{k,j=1}^{N}$	$[S_N^{III}]^{-1} = [S_N^{III}]^T = S_N^{II}$
DST-IV	$S_N^{IV} = \sqrt{\frac{2}{N}} \left[\sin \frac{(2k-1)(2j-1)}{4N}\pi \right]_{k,j=1}^{N}$	$[S_N^{IV}]^{-1} = [S_N^{IV}]^T = S_N^{IV}$

Table 14. The III and IV versions of the cosine and sine transforms

These transforms are also polynomial evaluations in Chebyshev-like bases, much like their counterparts considered earlier. Tables 15,16, and 17 are borrowed from [**KO96**], they contain the polynomials, the nodes, and the weight matrices required to express the DCT/DST III and IV transforms in terms of Chebyshev like bases:

	$\{Q_0,$	Q_1	$\ldots,$	Q_{N-2}	$Q_{N-1}\}$
DCT-III	$\{\frac{1}{\sqrt{2}}T_0,$	$T_1,$	$\ldots,$		$T_{N-1}\}$
DCT-IV	$\{U_0,$	$U_1 - U_0,$	$\ldots,$		$U_{N-1} - U_{N-2}\}$
DST-III	$\{U_0,$	$U_1,$	$\ldots,$	$U_{N-2},$	$\frac{1}{\sqrt{2}}U_{N-1}\}$
DST-IV	$\{U_0,$	$U_1 + U_0,$	$\ldots,$		$U_{N-1} + U_{N-2}\}$

Table 15. Chebyshev-like polynomials for DCT/DST III and IV

	Q_n	zeros of Q_n
DCT-III	T_n	$\{\cos(\frac{(2k+1)\pi}{2N})\}_0^{N-1}$
DCT-IV	$2T_n$	$\{\cos(\frac{(2k+1)\pi}{2N})\}_0^{N-1}$
DST-III	T_n	$\{\cos(\frac{(2k-1)\pi}{2N})\}_1^{N}$
DST-IV	$2T_n$	$\cos(\frac{(2k-1)\pi}{2N})\}_1^{N}$

Table 16. The last polynomial and the nodes of DCT/DST III and IV

DCT-III	$C_N^{III} = V_Q \cdot W_Q$ with $W_Q = \sqrt{\frac{2}{N}}\text{diag}(\frac{1}{\sqrt{2}}, \cos(\frac{\pi}{2N}), \ldots, \cos(\frac{(N-1)\pi}{2N}))$
DCT-IV	$C_N^{IV} = W_Q \cdot V_Q$ with $W_Q = \sqrt{\frac{2}{N}}\text{diag}(\cos(\frac{\pi}{4N}), \cos(\frac{3\pi}{4N}) \ldots, \cos(\frac{(2N-1)\pi}{4N}))$
DST-III	$S_N^{III} = W_Q \cdot V_Q$ with $W_Q = \sqrt{\frac{2}{N}}\text{diag}(\sin(\frac{\pi}{2N}), \ldots, \sin(\frac{(N-1)\pi}{2N}), \frac{1}{\sqrt{2}}\sin(\frac{\pi}{2}))$
DST-IV	$S_N^{IV} = W_Q \cdot V_Q$ with $W_Q = \sqrt{\frac{2}{N}}\text{diag}(\sin(\frac{\pi}{4N}), \sin(\frac{3\pi}{4N}), \ldots, \sin(\frac{(2N-1)\pi}{4N}))$

Table 17.The W_Q for DCT/DST III and IV

However, the approach presented in this paper becomes more involved when applied to the DCT/DST III and IV. One reason for this is as follows: the nodes of versions I and II are recursive whereas the nodes of versions III and IV are not. For example, in the case of DCT-II, the roots of Q_4 when $N = 4$ are a subset of the roots of Q_8 when $N = 8$. However, table 16 shows that this is not case for versions III and IV. Hence, Figure 1 should be modified for the III and IV versions; more branches should be added and the approach presented here becomes more involved. May be this is the reason why recursive algorithms for the versions III and IV are (to the best of our knowledge) missing in the literature. To derive them a different, simpler approach can be applied as shown next.

5. Algorithms for Versions III of the Transforms

In order to simultaneously derive formulas for both DCT-III and DST-III, we will simply label the Chebyshev-Vandermonde matrix involved in the computation of the transform as V_N^{III}. The arguments used will apply both to the DCT and DST. Our approach rests on the following well-known fact:

$$F_N^{(III)} = {F_N^{(II)}}^T \tag{5.1}$$

where V_N^{II} is the Chebyshev-Vandermonde matrix of the corresponding version two transform(i.e. DCT-II if we are considering the transform DCT-III and DST-II if we are considering DST-III).

Because of this, we can simply take transposes of (3.1) to obtain recursions for the DCT/DST III. The end result is:

(5.2)
$$V_N^{III} = {W_N^{III}}^{-1} H^T \begin{bmatrix} W_{\frac{N}{2}}^{III} V_{\frac{N}{2}}^{III} {W_{\frac{N}{2}}^{II}}^{-1} & 0 \\ 0 & \widehat{G}^T W_{\frac{N}{2}}^{III} V_{\frac{N}{2}}^{III} {W_{\frac{N}{2}}^{II}}^{-1} G^T \end{bmatrix} R^T W_N^{II}$$

where the quantities G,$\widehat{G}$, W_0, P, and H are taken from the corresponding II versions of the transforms, while R is the odd even permutation. We can rewrite this in more compact notation as:

$$V_N^{III} = L \begin{bmatrix} V_{\frac{N}{2}}^{III} & 0 \\ 0 & V_{\frac{N}{2}}^{III} \end{bmatrix} J, \qquad \text{where} \tag{5.3}$$

$$L = {W_N^{III}}^{-1} H^T \begin{bmatrix} W_{\frac{N}{2}}^{III} & 0 \\ 0 & \widehat{G}^T W_{\frac{N}{2}}^{III} \end{bmatrix}, \quad J = \begin{bmatrix} {W_{\frac{N}{2}}^{II}}^{-1} & 0 \\ 0 & {W_{\frac{N}{2}}^{II}}^{-1} G^T \end{bmatrix} R^T W_N^{II}.$$

It is possible to obtain a recursion for the transform matrix itself. This can be done by transposing (3.2). The result is the following recursion:

$$F_N = H^T \begin{bmatrix} F_{even} & 0 \\ 0 & \widehat{G}^T F_{even} W_e^{-T} G^T \end{bmatrix} P^T \tag{5.4}$$

6. Flow Graphs

The following flow graphs represent the algorithms derived in the previous section. Note that each line on the right-hand side of the picture entails a multiplication which has been left off the picture due to lack of space.

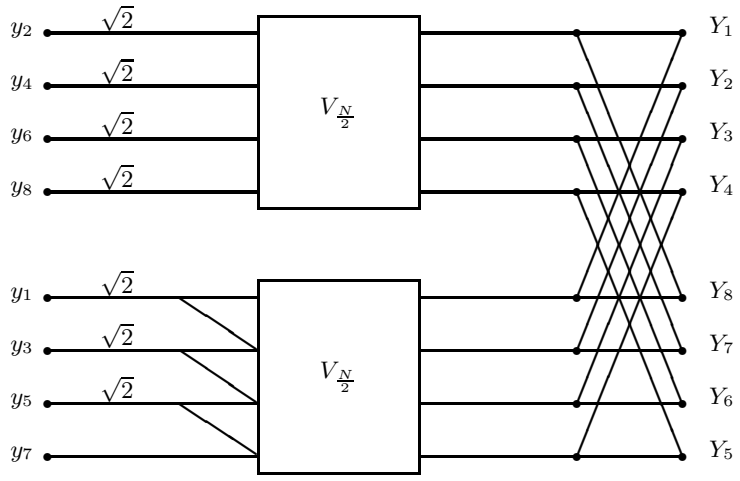

Figure 6 Flow Graph for V_N in the case of DCT-III with $N = 8$

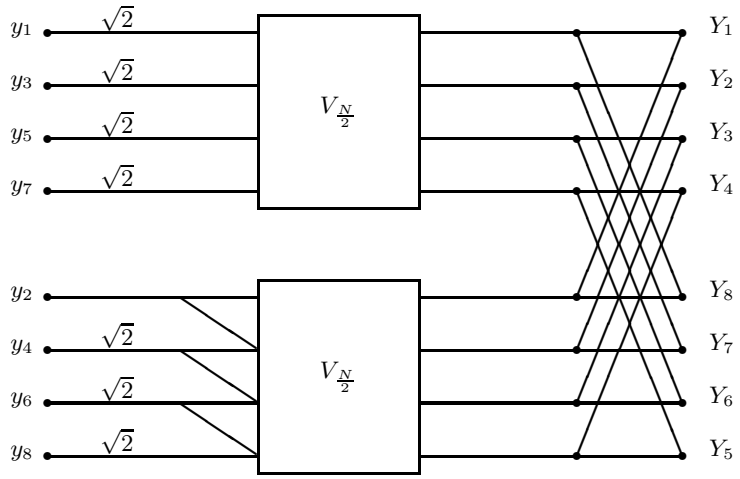

Figure 7 Flow Graph for V_N in the case of DST-III with $N = 8$

Part III: Reduction Formulas and Version IV

Instead of computing a particular version cosine or sine transform directly, it may be desirably to reduce it instead to the computation of a different version cosine or sine transform. In this section, formulas will be derived which do this. Furthermore, these reduction formulas will be used to easily adapt the formulas we derived earlier to the computation of version IV of the transforms.

7. Reduction Formulas

One approach to the reduction of one transform to another is based on the following observation: the III and IV versions of the transforms all have the same nodes, as a quick glance at table 16 reveals. Let us label these nodes $x_0, x_1, \ldots, x_{N-1}$.

This sharing of nodes implies that the transform matrix F_N can be expressed as follows:

$$F_N = W_N \cdot V \cdot M_N \tag{7.1}$$

where W_N and M_N depend on the transform while V does not. In all cases, V is the usual Vandermonde matrix: $V_{ij} = [x_{i-1}^{j-1}]$.

This factorization can easily be used to obtain the reduction formulas we seek. Indeed, let F_{P1} be the transform matrix of some transform P_1. Similarly, let F_{P2} be the transform matrix of some transform P_2. Of course, P_1 and P_2 can be any one of the transforms DCT/DST III and IV. We will further denote by W_{P1} and M_{P1} the matrices W_N and M_N associated with P_1; similar notation will be used for the matrices associated with P_2.

Then,

$$F_{P1} = W_{P1} W_{P2}^{-1} F_{P2} M_{P2}^{-1} M_{P1} \tag{7.2}$$

Using the notation $W_{P1}^{P2} = W_{P1}W_{P2}^{-1}$ and $M_{P1}^{P2} = M_{P2}^{-1}M_{P1}$, (7.2) can be rewritten as:

$$F_{P1} = W_{P1}^{P2} F_{P2} M_{P1}^{P2} \tag{7.3}$$

There are a total of 12 formulas compactly written in (7.3), making it possible to go from any version III or IV transform to any other version III or IV transform.

For convenience, we provide two tables of W_{P1}^{P2} and M_{P1}^{P2} for all the possibilities:

$W_{DCT\ III}^{DCT\ IV} = diag(\frac{1}{cos(\frac{\pi}{4N})}, \frac{1}{cos(\frac{3\pi}{4N})}, \cdots, \frac{1}{cos(\frac{(2N-1)\pi}{4N})})$
$W_{DCT\ III}^{DST\ III} = diag(\frac{1}{sin(\frac{\pi}{2N})}, \frac{1}{sin(\frac{3\pi}{2N})}, \cdots, \frac{1}{sin(\frac{(2N-1)\pi}{2N})})$
$W_{DCT\ III}^{DST\ IV} = diag(\frac{1}{sin(\frac{\pi}{4N})}, \frac{1}{sin(\frac{3\pi}{4N})}, \cdots, \frac{1}{sin(\frac{(2N-1)\pi}{4N})})$
$W_{DCT\ IV}^{DCT\ III} = diag(cos(\frac{\pi}{4N}), cos(\frac{3\pi}{4N}), \cdots, cos(\frac{(2N-1)\pi}{4N}))$
$W_{DCT\ IV}^{DST\ III} = diag(\frac{1}{2sin(\frac{\pi}{4N})}, \frac{1}{2sin(\frac{3\pi}{4N})}, \cdots, \frac{1}{2sin(\frac{(2N-1)\pi}{2N})}$
$W_{DCT\ IV}^{DST\ IV} = diag(cotan(\frac{\pi}{4N}), cotan(\frac{3\pi}{2N}), \cdots, cotan(\frac{(2N-1)\pi}{2N}))$
$W_{DST\ III}^{DCT\ III} = diag(sin(\frac{\pi}{2N}), sin(\frac{3\pi}{2N}), \cdots, sin(\frac{(2N-1)\pi}{2N}))$
$W_{DST\ III}^{DCT\ IV} = diag(2sin(\frac{\pi}{4N}), 2sin(\frac{3\pi}{4N}), \cdots, 2sin(\frac{(2N-1)\pi}{2N})$
$W_{DST\ III}^{DST\ IV} = diag(2cos(\frac{\pi}{4N}), 2cos(\frac{3\pi}{4N}), \cdots, 2cos(\frac{(2N-1)\pi}{4N}))$
$W_{DST\ IV}^{DCT\ III} = diag(sin(\frac{\pi}{4N}), sin(\frac{3\pi}{4N}), \cdots, sin(\frac{(2N-1)\pi}{4N}))$
$W_{DST\ IV}^{DCT\ IV} = diag(tan(\frac{\pi}{4N}), tan(\frac{3\pi}{4N}), \cdots, tan(\frac{(2N-1)\pi}{4N})$
$W_{DST\ IV}^{DST\ III} = diag(\frac{1}{2cos(\frac{\pi}{4N})}, \frac{1}{2cos(\frac{3\pi}{4N})}, \cdots, \frac{1}{2cos(\frac{(2N-1)\pi}{4N})})$

Table 18. The Matrices W Involved in the Reduction of Transforms

$$M_{DCT\ III}^{DCT\ IV} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

$$M_{DCT\ III}^{DST\ III} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \ddots & 0 \\ 0 & 0 & 0 & \ddots & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$M_{DCT\ III}^{DST\ IV} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

$$M_{DCT\ IV}^{DCT\ III} = \begin{bmatrix} \sqrt{2} & -\sqrt{2} & \sqrt{2} & -\sqrt{2} & \sqrt{2} & \dots & -\sqrt{2} \\ 0 & 2 & -2 & 2 & -2 & \cdots & 2 \\ 0 & 0 & 2 & -2 & 2 & \cdots & -2 \\ 0 & 0 & 0 & 2 & -2 & \cdots & 2 \\ 0 & 0 & 0 & 0 & 2 & \cdots & -2 \\ 0 & 0 & 0 & 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$M_{DCT\ IV}^{DST\ III} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & \sqrt{2} \end{bmatrix}$$

Table 19 The Matrices M Involved in the Reduction of Transforms

$$M_{DCT\ IV}^{DST\ IV} = \begin{bmatrix} 1 & -2 & 2 & -2 & 2 & \cdots & -2 \\ 0 & 1 & -2 & 2 & -2 & \cdots & 2 \\ 0 & 0 & 1 & -2 & 2 & \cdots & -2 \\ 0 & 0 & 0 & 1 & -2 & \cdots & 2 \\ 0 & 0 & 0 & 0 & 1 & & \vdots \\ 0 & 0 & 0 & 0 & 0 & \ddots & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_{DST\ III}^{DCT\ III} = \begin{bmatrix} \sqrt{2} & 0 & \sqrt{2} & 0 & \sqrt{2} & \cdots & \sqrt{2} & 0 \\ 0 & 2 & 0 & 2 & \cdots & 2 & 0 & \sqrt{2} \\ 0 & 0 & 2 & 0 & \cdots & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 & 0 & \cdots & & \sqrt{2} \\ 0 & 0 & 0 & 0 & \ddots & & & \\ 0 & 0 & 0 & 0 & 0 & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{2} \end{bmatrix}$$

$$M_{DST\ III}^{DCT\ IV} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & \frac{1}{\sqrt{2}} \\ 0 & 1 & 1 & \cdots & 1 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & \cdots & 1 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & \ddots & & \vdots \\ 0 & 0 & 0 & 0 & 1 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$M_{DST\ III}^{DST\ IV} = \begin{bmatrix} 1 & -1 & 1 & -1 & \cdots & 1 & -\frac{1}{\sqrt{2}} \\ 0 & 1 & -1 & 1 & \cdots & -1 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & -1 & \cdots & 1 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & \ddots & & & \vdots \\ 0 & 0 & 0 & 0 & 1 & -1 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & 0 & 0 & 1 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

Table 19 Continued The Matrices M Involved in the Reduction of Transforms

$$M_{DST\ IV}^{DCT\ III} = \begin{bmatrix} \sqrt{2} & \sqrt{2} & \sqrt{2} & \sqrt{2} & \sqrt{2} & \cdots & \sqrt{2} & \sqrt{2} \\ 0 & 2 & 2 & 2 & \cdots & 2 & 2 & 2 \\ 0 & 0 & 2 & 2 & \cdots & 2 & 2 & 2 \\ 0 & 0 & 0 & 2 & 2 & \cdots & 2 & 2 \\ 0 & 0 & 0 & 0 & \ddots & & & \\ 0 & 0 & 0 & 0 & 0 & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$M_{DST\ IV}^{DCT\ IV} = \begin{bmatrix} 1 & 2 & 2 & 2 & 2 & \cdots & 2 \\ 0 & 1 & 2 & 2 & 2 & \cdots & 2 \\ 0 & 0 & 1 & 2 & 2 & \cdots & 2 \\ 0 & 0 & 0 & 1 & 2 & \cdots & 2 \\ 0 & 0 & 0 & 0 & 1 & & \vdots \\ 0 & 0 & 0 & 0 & 0 & \ddots & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_{DST\ IV}^{DST\ III} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & \sqrt{2} \end{bmatrix}$$

Table 19 Continued The Matrices M Involved in the Reduction of Transforms

8. Recursive Algorithms for Versions IV

In this section, recursive algorithms are derived for DCT/DST IV. The technique is based on the relationships between these transforms and the DST-III via the reduction formulas obtained earlier.

It has been shown previously that $V^{DCT-IV} = V^{DST-III} \cdot M_{DCT-IV}^{DST-III}$. We can use this to modify (5.3) to obtain:

$$V^{DCT-IV} = L \begin{bmatrix} V_{even}^{DCT-IV} & \\ & V_{even}^{DCT-IV} \end{bmatrix} \cdot \begin{bmatrix} {M_{DCT-IV}^{DST-III}}^{-1} & \\ & {M_{DCT-IV}^{DST-III}}^{-1} \end{bmatrix} \cdot J \cdot M_{DCT-IV}^{DST-III} \tag{8.1}$$

where L and J are as defined earlier. The previous equation can be modified to obtain a recursive expression for F_N:

$$F_N^{DCT-IV} = W_{DCT-IV}^{DST-III} H^T \begin{bmatrix} {W_{DCT-IV,\frac{N}{2}}^{DST-III}}^{-1} & \\ & \widehat{G}^T {W_{DCT-IV,\frac{N}{2}}^{DST-III}}^{-1} \end{bmatrix} \cdot$$

$$\begin{bmatrix} F_{\frac{N}{2}}^{DCT-IV} & \\ & F_{\frac{N}{2}}^{DCT-IV} \end{bmatrix} \cdot \tag{8.2}$$

$$\begin{bmatrix} {M_{DCT-IV,\frac{N}{2}}^{DST-III}}^{-1} & \\ & {M_{DCT-IV,\frac{N}{2}}^{DST-III}}^{-1} W_e^{-T} G^T \end{bmatrix} P^T M_{DCT-IV}^{DST-III} \tag{8.3}$$

By using the explicit expressions for the the W's and M's in this formula, it is possible to simplify it further. Indeed, if

(8.4)

$$U_{DCT-IV} = \begin{bmatrix} \frac{\sin\frac{\pi}{2N}}{\sin\frac{\pi}{4N}} & & & \frac{\tan\frac{\pi}{2N}}{2\sin\frac{\pi}{4N}} & & \\ & \ddots & & & \ddots & \\ & & \frac{\sin\frac{(N-1)\pi}{2N}}{\sin\frac{(N-1)\pi}{4N}} & & & \frac{\tan\frac{(N-1)\pi}{2N}}{2\sin\frac{(N-1)\pi}{4N}} \\ & & -\frac{\sin\frac{(N-1)\pi}{2N}}{\sin\frac{(N+1)\pi}{4N}} & & & \frac{\tan\frac{(N-1)\pi}{2N}}{2\sin\frac{(N+1)\pi}{4N}} \\ & \iddots & & & \iddots & \\ -\frac{\sin\frac{\pi}{2N}}{\sin\frac{(2N-1)\pi}{4N}} & & & \frac{\tan\frac{\pi}{2N}}{2\sin\frac{(2N-1)\pi}{4N}} & & \end{bmatrix}$$

and

$$N_{DCT-IV} = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \cdots & & \frac{1}{\sqrt{2}} \\ & & & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ & & & & & \ddots & & \\ & & & & & & & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \sqrt{2} & -\sqrt{2} & \sqrt{2} & -\sqrt{2} & \cdots & \sqrt{2} \\ & & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \sqrt{2} & -\sqrt{2} & & \\ & & & & & \ddots & & -\sqrt{2} \\ & & & & & & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \tag{8.5}$$

Then

$$F_N^{DCT-IV} = U_{DCT-IV} \begin{bmatrix} F_{\frac{N}{2}}^{DCT-IV} & \\ & F_{\frac{N}{2}}^{DCT-IV} \end{bmatrix} N_{DCT-IV} \tag{8.6}$$

By the same method, a corresponding formula can be derived for DST-IV. We have that:

$$V^{DST-IV} = L \begin{bmatrix} V_{even}^{DST-IV} & \\ & V_{even}^{DST-IV} \end{bmatrix} \cdot$$

$$\begin{bmatrix} {M_{DST-IV}^{DST-III}}^{-1} & \\ & {M_{DST-IV}^{DST-III}}^{-1} \end{bmatrix} \cdot J \cdot M_{DST-IV}^{DST-III} \tag{8.7}$$

which can be modified to get the following formula for the transform:

$$F_N^{DST-IV} = W_{DST-IV}^{DST-III} H^T \begin{bmatrix} {W_{DST-IV,\frac{N}{2}}^{DST-III}}^{-1} & \\ & \widehat{G}^T {W_{DST-IV,\frac{N}{2}}^{DST-III}}^{-1} \end{bmatrix} \cdot$$

$$\begin{bmatrix} F_{\frac{N}{2}}^{DST-IV} & \\ & F_{\frac{N}{2}}^{DST-IV} \end{bmatrix} \cdot \tag{8.8}$$

$$\begin{bmatrix} {M_{DST-IV,\frac{N}{2}}^{DST-III}}^{-1} & \\ & {M_{DST-IV,\frac{N}{2}}^{DST-III}}^{-1} W_e^{-T} G^T \end{bmatrix} P^T M_{DST-IV}^{DST-III} \tag{8.9}$$

This equation can also be modified to provide recursions for the transform. If

$$U_{DST-IV} = \begin{bmatrix} \frac{\cos\frac{\pi}{2N}}{\cos\frac{\pi}{4N}} & & & \frac{1}{2\cos\frac{\pi}{4N}} & & \\ & \ddots & & & \ddots & \\ & & \frac{\cos\frac{(N-1)\pi}{2N}}{\cos\frac{(N-1)\pi}{4N}} & & & \frac{1}{2\cos\frac{(N-1)\pi}{4N}} \\ & & -\frac{\sin\frac{(N-1)\pi}{2N}}{\cos\frac{(N+1)\pi}{4N}} & & & \frac{1}{2\cos\frac{(N+1)\pi}{4N}} \\ & \iddots & & & \iddots & \\ -\frac{\cos\frac{\pi}{2N}}{\cos\frac{(2N-1)\pi}{4N}} & & & \frac{1}{2\cos\frac{(2N-1)\pi}{4N}} & & \end{bmatrix} \tag{8.10}$$

and

$$N_{DCT-IV} = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \cdots & & \\ & & & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ & & & & & \ddots & & \\ & & & & & & & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & & & & & & \\ & & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & & & & \\ & & & & & \ddots & & \\ & & & & & & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \tag{8.11}$$

Then

$$F_N^{DST-IV} = U_{DST-IV} \begin{bmatrix} F_{\frac{N}{2}}^{DST-IV} & \\ & F_{\frac{N}{2}}^{DST-IV} \end{bmatrix} N_{DST-IV} \tag{8.12}$$

These formulas can be used as a basis for recursive $O(N \log N)$ algorithms. Indeed, there is a clear recursion for multiplication by U and N: the k'th row can be easily computed from the $k+1$'st one allowing multiplication in $O(N)$ operations.

9. Flow Graphs

Flow graph implementations of the algorithms for DCT/DST IV are provided here.

Note that in the following two graphs the multiplications have been left off due to lack of space. In general each arrow entails a multiplication.

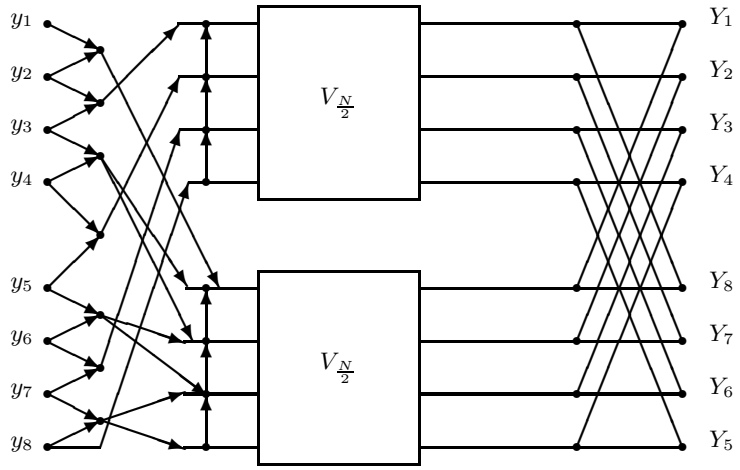

Figure 8 Flow Graph for V_N in the case of DCT-IV with $N = 8$

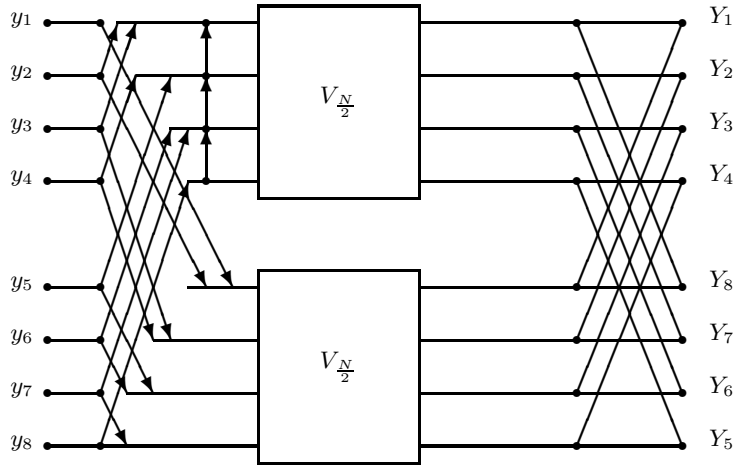

Figure 9 Flow Graph for V_N in the case of DST-IV with $N = 8$

References

[B84] S. Barnett, *Division of Generalized Polynomials Using the Comrade Matrix*, Linear Algebra and Its Applications **60**:159-175 (1984)

[KO96] T.Kailath and V.Olshevsky. *Displacement structure approach to discrete trigonometric transform based preconditioners of G.Strang and T.Chan types.* Calcolo, **33** 1996, 191-208.

[RY90] K.R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press, 1990.

[S99] G. Strang, *The Discrete Cosine Transform*, SIAM Review, **41**:135-147.

[S86] G. Strang, *Introduction to Applied Mathematics* , Wellesley-Cambridge Press, 1986

[VL92] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM Publications, 1992.

Alexander Olshevsky, Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332
E-mail address: alex.olshevsky@resnet.gatech.edu

Vadim Olshevsky, Department of Mathematics, University of Connecticut, Storrs, Connecticut, 06269. www.math.uconn.edu/~olshevsk
E-mail address: olshevsky@math.uconn.edu

Jun Wang, Department of Computer Science, Georgia State University, Atlanta, Georgia, 30303
E-mail address: jjuunn_wang@hotmail.com