Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Linear Algebra and its Applications 394 (2005) 309–345
LINEAR ALGEBRA AND ITS APPLICATIONS
www.elsevier.com/locate/laa

Fast and numerically stable algorithms for discrete cosine transforms

Gerlind Plonka [a,*], Manfred Tasche [b]

[a] *Institute of Mathematics, University of Duisburg-Essen, D-47048 Duisburg, Germany*
[b] *Institute of Mathematics, University of Rostock, D-18051 Rostock, Germany*

Received 1 August 2003; accepted 31 July 2004

Submitted by G. Heinig

Abstract

In this paper, we derive fast and numerically stable algorithms for discrete cosine transforms (DCT) of radix-2 length which are based on real factorizations of the corresponding cosine matrices into products of sparse, (almost) orthogonal matrices of simple structure. These algorithms are completely recursive, are simple to implement and use only permutations, scaling with $\sqrt{2}$, butterfly operations, and plane rotations/rotation–reflections. Our algorithms have low arithmetic costs which compare with known fast DCT algorithms. Further, a detailed analysis of the roundoff errors for the presented DCT algorithms shows their excellent numerical stability which outperforms a real fast DCT algorithm based on polynomial arithmetic.

AMS classification: 65T50; 65G50; 15A23

Keywords: Discrete cosine transform; Fast algorithm; Stable algorithm; Fast cosine transform; Arithmetic cost; Numerical stability; Factorization of cosine matrix; Direct sum decomposition; Sparse orthogonal matrix factors

* Corresponding author.
E-mail addresses: plonka@math.uni-duisburg.de (G. Plonka), manfred.tasche@mathematik.uni-rostock.de (M. Tasche).

0024-3795/$ - see front matter
doi:10.1016/j.laa.2004.07.015

1. Introduction

Discrete trigonometric transforms are widely used in processing and compression of signals and images (see [15]). Examples of such transforms are discrete cosine transforms (DCT) and discrete sine transforms (DST) of types I–IV. These transforms are represented by orthogonal cosine and sine matrices, respectively. Especially, the DCT-II and its inverse DCT-III have been shown to be well applicable for image compression. The roots of these transforms go back to trigonometric approximation in the eighteenth century (see [9]). Discrete trigonometric transforms have also found important applications in numerical Fourier methods, approximation via Chebyshev polynomials, quadrature methods of Clenshaw–Curtis type, numerical solution of partial differential equations (fast Poisson solvers), singular integral equations, and Toeplitz-plus-Hankel systems. In this paper, we shall concentrate on the construction of real, fast, and recursive DCT algorithms having excellent numerical stability in floating point arithmetic.

There is a close connection between fast DCT algorithms and factorizations of the corresponding transform matrix. Let $C_n \in \mathbb{R}^{n\times n}$ be an orthogonal cosine matrix of large radix-2 order n where radix-2 means $n = 2^t$ with some positive integer t. Assume that we know a factorization of C_n into a product of sparse matrices

$$C_n = M_n^{(m-1)} \cdots M_n^{(0)}, \quad 1 < m \ll n. \tag{1.1}$$

Then the transformed vector $C_n\mathbf{x}$ with arbitrary $\mathbf{x} \in \mathbb{R}^n$ can be computed iteratively by

$$\mathbf{x}^{(s+1)} := M_n^{(s)}\mathbf{x}^{(s)} \quad (\mathbf{x}^{(0)} := \mathbf{x})$$

for $s = 0, \ldots, m-1$ such that $\mathbf{x}^{(m)} = C_n\mathbf{x}$. Since all matrix factors in (1.1) are sparse, the arithmetical cost of this method will be low such that the factorization (1.1) of C_n generates a fast DCT algorithm. For an algebraic approach to fast algorithms for discrete trigonometric transforms we refer to [14].

An important result in [17] (see also [23]) says that a fast DCT algorithm possesses an excellent numerical stability, if the algorithm is based on a factorization of C_n into sparse, (almost) orthogonal matrices. Here a matrix is called *almost orthogonal*, if it is orthogonal up to a positive factor. Therefore, in order to get real, fast, and numerically stable DCT algorithms, one should be especially interested in a factorization (1.1) with sparse, (almost) orthogonal matrix factors of simple structure.

We may distinguish the following two methods to obtain real, fast DCT algorithms:

1. *Fast* DCT *algorithms via polynomial arithmetic*. All components of $C_n\mathbf{x}$ can be interpreted as values of one polynomial at n nodes. Reducing the degree of this polynomial by divide-and-conquer technique, one can get a real fast DCT algorithm with low arithmetical cost (see [7,8,19,20,14]). Efficient algorithms for DCT of radix-2 length n require about $2n \log_2 n$ flops. Such a DCT algorithm generates a factorization (1.1) of C_n with sparse, *non-orthogonal* matrix factors $M_n^{(s)}$, i.e., the factoriza-

tion (1.1) does not preserve the orthogonality of C_n (see e.g. [17,2,23]). This fact leads to an inferior numerical stability of these DCT algorithms [17,2,23].

2. *Fast* DCT *algorithms via direct matrix factorization*. Using simple properties of trigonometric functions, one may find direct factorizations of the transform matrix C_n into a product of real, sparse matrices. The trigonometric approximation algorithm of Runge (see [16, pp. 211–218]) can be considered as a first example of this approach. Results on direct matrix factorizations of C_n into orthogonal, sparse matrices are due to Chen et al. [4] and Wang [25,26]. An "orthogonal" factorization of the cosine matrix of type II and of order 8 was given by Loeffler et al. [13] and is used even today in JPEG standard (cf. Example 2.9). Improving the earlier results in [4,25], Schreiber has given a constructive proof of a factorization of some cosine matrices of order $n = 2^t$ into a product of sparse, orthogonal matrices in [17]. The orthogonal matrix factorizations in [4,25,17] are not completely recursive and hence lead not to simple recursive algorithms. However taking the results of Wang [25,26] together, one finds an orthogonal factorization of the cosine matrix of type II and order $n = 2^t$ yielding a recursive DCT-II algorithm (see Section 2). Note that various direct factorizations of C_n use non-orthogonal matrix factors (see [15, pp. 53–62], [3,11,12,18]). Many results were published without proofs.

In this paper, we shall derive fast, completely recursive DCT algorithms of radix-2 length. As usual, all algorithms use divide-and-conquer techniques. Further, we shall present the complete real factorizations of the cosine/sine matrices into products of sparse, (almost) orthogonal matrices. Here a matrix factor is said to be sparse if each row and column contains at most 2 non-zero entries. The DCT algorithms require only permutations, scaling (with $\sqrt{2}$), butterfly operations, and plane rotations/rotation–reflections with small rotation angles. These matrix factorizations can also be used for an iterative (instead of recursive) implementation of the algorithms. Finally, a comprehensive analysis of the matrix factorization will show that the presented algorithms possess an excellent numerical stability. Using the Wilkinson model for binary floating point arithmetic which is true for the IEEE standard, we shall give new, explicit worst case estimates for the error caused by the application of our algorithms in floating point arithmetic. The arithmetical costs of the presented algorithms are only unessentially higher than those for the known DCT algorithms based on polynomial arithmetic. However, regarding the numerical stability, polynomial algorithms are clearly outperformed.

Let us remark that the split-radix DCT-II introduced in [5] seems to be very close to our approach to the DCT-II in spirit, but unfortunately the matrix factorization for the cosine matrix of type II is not correct (see Remark 2.6). In [22], a split-radix approach for the computation of the unnormalized DCT-I and DST-I is presented. However, the corresponding matrix factorizations, given explicitly for $n = 16$, are not correct (see Remark 2.6). But the idea of Sun and Yip [22] to split a DCT-I of length $n + 1$ into a DCT-I of length $n/2 + 1$ as well as a DCT-I of length $n/4 + 1$ and a DST-I of length $n/4 - 1$ leads us to a new approach to factorize the cosine

matrix of type III (see Lemma 2.5). In this way we can give also a new fast and stable DCT-I algorithm in Section 3.

The paper is organized as follows: In Section 2 we introduce different types of cosine and sine matrices and derive factorizations of the cosine matrices of types I–IV. All proofs are based on divide-and-conquer technique applied *directly* to a matrix. That is, a given trigonometric matrix can be represented as a *direct sum* of trigonometric matrices of half order (and maybe of different type). While similar factorizations as in Lemmas 2.2 and 2.4 have already been used before (see e.g. [16,4,25,26]), the factorizations of the cosine and sine matrices of type III in Lemma 2.5 are new. They enable us to derive the corresponding fast DCT algorithms as recursive procedures in Section 3. We also compute the arithmetic costs of the new algorithms. Corresponding factorizations of cosine matrices of types II, IV, and I into sparse, (almost) orthogonal matrix factors of simple structure are given in Theorems 4.2, 4.4, and 4.6. Using this matrix factorizations, we give a comprehensive analysis of the numerical stability of these fast DCT algorithms in Section 5. Finally we show that the new algorithms have an excellent numerical stability.

2. Trigonometric matrices

2.1. Definitions and notations

Let $n \geqslant 2$ be a given integer. In the following, we consider *cosine* and *sine matrices of types* I–IV which are defined by

$$
\begin{aligned}
C_{n+1}^{\mathrm{I}} &:= \sqrt{\tfrac{2}{n}} \left(\epsilon_n(j)\epsilon_n(k) \cos \tfrac{jk\pi}{n} \right)_{j,k=0}^{n}, \\
C_n^{\mathrm{II}} &:= \sqrt{\tfrac{2}{n}} \left(\epsilon_n(j) \cos \tfrac{j(2k+1)\pi}{2n} \right)_{j,k=0}^{n-1}, \\
C_n^{\mathrm{III}} &:= \left(C_n^{\mathrm{II}} \right)^{\mathrm{T}}, \\
C_n^{\mathrm{IV}} &:= \sqrt{\tfrac{2}{n}} \left(\cos \tfrac{(2j+1)(2k+1)\pi}{4n} \right)_{j,k=0}^{n-1}, \\
S_{n-1}^{\mathrm{I}} &:= \sqrt{\tfrac{2}{n}} \left(\sin \tfrac{(j+1)(k+1)\pi}{n} \right)_{j,k=0}^{n-2}, \\
S_n^{\mathrm{II}} &:= \sqrt{\tfrac{2}{n}} \left(\epsilon_n(j+1) \sin \tfrac{(j+1)(2k+1)\pi}{2n} \right)_{j,k=0}^{n-1}, \\
S_n^{\mathrm{III}} &:= \left(S_n^{\mathrm{II}} \right)^{\mathrm{T}}, \\
S_n^{\mathrm{IV}} &:= \sqrt{\tfrac{2}{n}} \left(\sin \tfrac{(2j+1)(2k+1)\pi}{4n} \right)_{j,k=0}^{n-1}.
\end{aligned}
\tag{2.1}
$$

Here we set $\epsilon_n(0) = \epsilon_n(n) := \sqrt{2}/2$ and $\epsilon_n(j) := 1$ for $j \in \{1, \dots, n-1\}$. In our notation a subscript of a matrix denotes the corresponding order, while a superscript signifies the "type" of the matrix. First cosine and sine matrices appeared in connection with trigonometric approximation (see [9,16]). In signal processing, cosine matrices of types II and III were introduced in [1]. The above classification was given in [25] (cf. [15, pp. 12–21]).

The cosine and sine matrices of types I–IV are orthogonal (see e.g. [15, pp. 13–14], [21,17]). Strang [21] pointed out that the column vectors of each cosine matrix are eigenvectors of a symmetric second difference matrix and therefore orthogonal.

A *discrete trigonometric transform of length m* is a linear mapping which maps any vector $\mathbf{x} \in \mathbb{R}^m$ onto $M\mathbf{x}$, where $M \in \mathbb{R}^{m\times m}$ is a cosine or sine matrix of (2.1). Especially, the *discrete cosine transform of type* II (DCT-II) with length n is represented by $M = C_n^{\mathrm{II}}$. The *discrete sine transform of type* I (DST-I) with length $n-1$ is represented by $M = S_{n-1}^{\mathrm{I}}$, etc.

In the following, we give a collection of all auxiliary matrices frequently used in this paper for matrix factorizations for an easy lookup.

Let I_n denote the identity matrix of order n and J_n the counteridentity matrix which is defined by $J_n\mathbf{x} := (x_{n-1}, x_{n-2}, \dots, x_0)^{\mathrm{T}}$ for any $\mathbf{x} = (x_j)_{j=0}^{n-1}$. Blanks in a matrix indicate zeros or blocks of zeros. The direct sum of two matrices A, B is defined to be the block diagonal matrix $A \oplus B := \mathrm{diag}(A, B)$.

Let

$$D_n := \mathrm{diag}((-1)^k)_{k=0}^{n-1}$$

be the diagonal sign matrix.

For $n \geqslant 4$, P_n denotes the *even–odd permutation matrix* (or 2-*stride permutation matrix*) defined by

$$P_n\mathbf{x} := \begin{cases} (x_0, x_2, \dots, x_{n-2}, x_1, x_3, \dots, x_{n-1})^{\mathrm{T}} & \text{for even } n, \\ (x_0, x_2, \dots, x_{n-1}, x_1, x_3, \dots, x_{n-2})^{\mathrm{T}} & \text{for odd } n \end{cases}$$

with $\mathbf{x} = (x_j)_{j=0}^{n-1}$. Note that $P_n^{-1} = P_n^{\mathrm{T}}$ is the $\lceil n/2 \rceil$-stride permutation matrix.

Further for even $n \geqslant 4$, we introduce

$$n_1 := \tfrac{n}{2}$$

and the orthogonal matrices

$$A_n(1) := \left(1 \oplus \tfrac{1}{\sqrt{2}} \begin{pmatrix} I_{n_1-1} & I_{n_1-1} \\ I_{n_1-1} & -I_{n_1-1} \end{pmatrix} \oplus (-1)\right)(I_{n_1} \oplus D_{n_1} J_{n_1}),$$

$$\tilde{A}_n(0) := (I_{n_1} \oplus J_{n_1}) \left(1 \oplus \tfrac{1}{\sqrt{2}} \begin{pmatrix} I_{n_1-1} & & J_{n_1-1} \\ & \sqrt{2} & \\ J_{n_1-1} & & -I_{n_1-1} \end{pmatrix}\right) \times (I_{n_1+1} \oplus (-1)^{n_1} D_{n_1-1}),$$

$$\tilde{A}_{n-1}(-1) := D_{n_1} \oplus I_{n_1-1},$$

$$T_n(0) := \tfrac{1}{\sqrt{2}} \begin{pmatrix} I_{n_1} & J_{n_1} \\ I_{n_1} & -J_{n_1} \end{pmatrix},$$

$$\tilde{T}_{n+1}(1) := \tfrac{1}{\sqrt{2}} \begin{pmatrix} I_{n_1} & & J_{n_1} \\ & \sqrt{2} & \\ I_{n_1} & & -J_{n_1} \end{pmatrix},$$

$$\tilde{T}_{n-1}(-1) := (J_{n_1} \oplus I_{n_1-1}) \tilde{T}_{n-1}(1),$$

$$T_n(1) := (I_{n_1} \oplus D_{n_1}) \begin{pmatrix} \operatorname{diag} \mathbf{c}_{n_1} & (\operatorname{diag} \mathbf{s}_{n_1}) J_{n_1} \\ -J_{n_1} \operatorname{diag} \mathbf{s}_{n_1} & \operatorname{diag}(J_{n_1} \mathbf{c}_{n_1}) \end{pmatrix},$$

$$\tilde{T}_n(0) := (I_{n_1+1} \oplus D_{n_1-1}) \times \left(1 \oplus \begin{pmatrix} \operatorname{diag} \tilde{\mathbf{c}}_{n_1-1} & & (\operatorname{diag} \tilde{\mathbf{s}}_{n_1-1}) J_{n_1-1} \\ & 1 & \\ -J_{n_1-1} \operatorname{diag} \tilde{\mathbf{s}}_{n_1-1} & & \operatorname{diag}(J_{n_1-1} \tilde{\mathbf{c}}_{n_1-1}) \end{pmatrix} \right),$$

where

$$\mathbf{c}_{n_1} := \left(\cos \tfrac{(2k+1)\pi}{4n} \right)_{k=0}^{n_1-1}, \quad \mathbf{s}_{n_1} := \left(\sin \tfrac{(2k+1)\pi}{4n} \right)_{k=0}^{n_1-1},$$

and

$$\tilde{\mathbf{c}}_{n_1-1} := \left(\cos \tfrac{k\pi}{2n} \right)_{k=1}^{n_1-1}, \quad \tilde{\mathbf{s}}_{n_1-1} := \left(\sin \tfrac{k\pi}{2n} \right)_{k=1}^{n_1-1}.$$

The modified identity matrices are denoted by $I_n' := \sqrt{2} \oplus I_{n-1}$ and $I_n'' := I_{n-1} \oplus \sqrt{2}$. Finally, let V_n be the forward shift matrix. Note that for arbitrary $\mathbf{x} = (x_j)_{j=0}^{n-1} \in \mathbb{R}^n$ we have

$$V_n \mathbf{x} = (0, x_0, x_1, \ldots, x_{n-2})^{\mathrm{T}}, \quad V_n^{\mathrm{T}} \mathbf{x} = (x_1, x_2, \ldots, x_{n-1}, 0)^{\mathrm{T}}.$$

In the following lemma we recall the intertwining relations of above cosine and sine matrices.

Lemma 2.1. *Let $n \geqslant 2$ be an integer. The cosine and sine matrices in (2.1) satisfy the intertwining relations*

$$\begin{aligned} &C_{n+1}^{\mathrm{I}} J_{n+1} = D_{n+1} C_{n+1}^{\mathrm{I}}, && S_{n-1}^{\mathrm{I}} J_{n-1} = D_{n-1} S_{n-1}^{\mathrm{I}}, \\ &C_n^{\mathrm{II}} J_n = D_n C_n^{\mathrm{II}}, && S_n^{\mathrm{II}} J_n = D_n S_n^{\mathrm{II}}, \\ &(-1)^{n-1} C_n^{\mathrm{IV}} J_n D_n = J_n D_n C_n^{\mathrm{IV}}, && (-1)^{n-1} S_n^{\mathrm{IV}} J_n D_n = J_n D_n S_n^{\mathrm{IV}} \end{aligned} \tag{2.2}$$

and

$$J_n C_n^{\mathrm{II}} = S_n^{\mathrm{II}} D_n, \quad C_n^{\mathrm{IV}} J_n = D_n S_n^{\mathrm{IV}}. \tag{2.3}$$

The proof is straightforward and is omitted here (see [17]). The corresponding properties of C_n^{III} and S_n^{III} follow from (2.2)–(2.3) by transposing.

2.2. Recursions for trigonometric matrices

In this paper, we are interested in fast and numerically stable algorithms for discrete trigonometric transforms. As an immediate consequence of Lemma 2.1, we only need to construct algorithms for DCT-I, DCT-II, DCT-IV, and DST-I.

Let us now recall the following orthogonal factorizations for C_n^{II}, C_{n+1}^{I}, and S_{n-1}^{I} which are similar to those presented in [25].

Lemma 2.2. *Let $n \geqslant 4$ be an even integer and $n_1 = n/2$.*

(i) *The matrix C_n^{II} can be factorized in the form*

$$C_n^{\mathrm{II}} = P_n^{\mathrm{T}} \big(C_{n_1}^{\mathrm{II}} \oplus C_{n_1}^{\mathrm{IV}}\big) T_n(0). \tag{2.4}$$

(ii) *The matrix C_{n+1}^{I} can be factorized in the form*

$$C_{n+1}^{\mathrm{I}} = P_{n+1}^{\mathrm{T}} \big(C_{n_1+1}^{\mathrm{I}} \oplus C_{n_1}^{\mathrm{III}}\big) \tilde{T}_{n+1}(1). \tag{2.5}$$

(iii) *The matrix S_{n-1}^{I} can be factorized in the form*

$$\begin{aligned} S_{n-1}^{\mathrm{I}} &= P_{n-1}^{\mathrm{T}} \big(S_{n_1}^{\mathrm{III}} \oplus S_{n_1-1}^{\mathrm{I}}\big) \tilde{T}_{n-1}(1) \\ &= P_{n-1}^{\mathrm{T}} \tilde{A}_{n-1}(-1) \big(C_{n_1}^{\mathrm{III}} \oplus S_{n_1-1}^{\mathrm{I}}\big) \tilde{T}_{n-1}(-1). \end{aligned} \tag{2.6}$$

Proof. We show (2.4) by divide-and-conquer technique. First we permute the rows of C_n^{II} by multiplying with P_n and write the result as a block matrix:

$$P_n C_n^{\mathrm{II}} = \frac{1}{\sqrt{n_1}} \begin{pmatrix} \Big(\epsilon_n(2j) \cos \frac{2j(2k+1)\pi}{2n}\Big)_{j,k=0}^{n_1-1} & \Big(\epsilon_n(2j) \cos \frac{2j(n+2k+1)\pi}{2n}\Big)_{j,k=0}^{n_1-1} \\ \Big(\epsilon_n(2j+1) \cos \frac{(2j+1)(2k+1)\pi}{2n}\Big)_{j,k=0}^{n_1-1} & \Big(\epsilon_n(2j+1) \cos \frac{(2j+1)(n+2k+1)\pi}{2n}\Big)_{j,k=0}^{n_1-1} \end{pmatrix}.$$

By (2.1) and

$$\cos \frac{j(n+2k+1)\pi}{n} = \cos \frac{j(n-2k-1)\pi}{n},$$

$$\cos \frac{(2j+1)(n+2k+1)\pi}{2n} = -\cos \frac{(2j+1)(n-2k-1)\pi}{2n},$$

it follows immediately that the four blocks of $P_n C_n^{\mathrm{II}}$ can be represented by $C_{n_1}^{\mathrm{II}}$ and $C_{n_1}^{\mathrm{IV}}$:

$$\begin{aligned} P_n C_n^{\mathrm{II}} &= \frac{1}{\sqrt{2}} \begin{pmatrix} C_{n_1}^{\mathrm{II}} & C_{n_1}^{\mathrm{II}} J_{n_1} \\ C_{n_1}^{\mathrm{IV}} & -C_{n_1}^{\mathrm{IV}} J_{n_1} \end{pmatrix} = \big(C_{n_1}^{\mathrm{II}} \oplus C_{n_1}^{\mathrm{IV}}\big) \frac{1}{\sqrt{2}} \begin{pmatrix} I_{n_1} & J_{n_1} \\ I_{n_1} & -J_{n_1} \end{pmatrix} \\ &= \big(C_{n_1}^{\mathrm{II}} \oplus C_{n_1}^{\mathrm{IV}}\big)\, T_n(0). \end{aligned}$$

Since $P_n^{-1} = P_n^{\mathrm{T}}$ and $T_n(0) T_n(0)^{\mathrm{T}} = I_n$, the matrices P_n and $T_n(0)$ are orthogonal.

The proofs of (2.5) and (2.6) follow similar lines. □

Remark 2.3. The trigonometric approximation algorithm of Runge (see [16, pp. 211–218]) is equivalent to (2.5) and (2.6). Note that similar factorizations of C_n^{II},

C^{I}_{n+1}, and S^{I}_{n-1} in [25] use modified even–odd permutation matrices $Q_n := (I_{n_1} \oplus J_{n_1})P_n$ and $Q_{n+1} := (I_{n_1+1} \oplus J_{n_1})P_{n+1}$. From (2.4) we obtain the following factorization of C^{III}_n:

$$C^{\mathrm{III}}_n = T_n(0)^{\mathrm{T}} \left(C^{\mathrm{III}}_{n_1} \oplus C^{\mathrm{IV}}_{n_1}\right) P_n. \tag{2.7}$$

The next lemma provides an orthogonal factorization of C^{IV}_n for even $n \geqslant 4$ and will be the key for our new fast algorithms for the DCT-II and DCT-IV in Section 3. In [26], one can find the following factorization of C^{IV}_n, if $n \geqslant 4$ is a power of 2.

Lemma 2.4. *For even $n \geqslant 4$, the matrix C^{IV}_n can be factorized in the form*

$$C^{\mathrm{IV}}_n = P^{\mathrm{T}}_n A_n(1)\big(C^{\mathrm{II}}_{n_1} \oplus C^{\mathrm{II}}_{n_1}\big)T_n(1). \tag{2.8}$$

Proof. We show (2.8) again by divide-and-conquer technique. Therefore we permute the rows of C^{IV}_n by multiplying with P_n and write the result as block matrix:

$$P_n C^{\mathrm{IV}}_n = \frac{1}{\sqrt{n_1}} \begin{pmatrix} \left(\cos \frac{(4j+1)(2k+1)\pi}{4n}\right)^{n_1-1}_{j,k=0} & \left(\cos \frac{(4j+1)(n+2k+1)\pi}{4n}\right)^{n_1-1}_{j,k=0} \\ \left(\cos \frac{(4j+3)(2k+1)\pi}{4n}\right)^{n_1-1}_{j,k=0} & \left(\cos \frac{(4j+3)(n+2k+1)\pi}{4n}\right)^{n_1-1}_{j,k=0} \end{pmatrix}.$$

Now we consider the single blocks of $P_n C^{\mathrm{IV}}_n$ and represent every block by $C^{\mathrm{II}}_{n_1}$ and $S^{\mathrm{II}}_{n_1}$.

1. By (2.2) and
$$\cos \tfrac{(4j+1)(2k+1)\pi}{4n} = \cos \tfrac{j(2k+1)\pi}{n} \cos \tfrac{(2k+1)\pi}{4n} - \sin \tfrac{j(2k+1)\pi}{n} \sin \tfrac{(2k+1)\pi}{4n},$$
it follows that
$$\begin{aligned} &\tfrac{1}{\sqrt{n_1}} \left(\cos \tfrac{(4j+1)(2k+1)\pi}{4n}\right)^{n_1-1}_{j,k=0} \\ &\quad = \tfrac{1}{\sqrt{2}} \left(I'_{n_1} C^{\mathrm{II}}_{n_1} \operatorname{diag} \mathbf{c}_{n_1} - V_{n_1} S^{\mathrm{II}}_{n_1} \operatorname{diag} \mathbf{s}_{n_1}\right) \\ &\quad = \tfrac{1}{\sqrt{2}} \left(I'_{n_1} C^{\mathrm{II}}_{n_1} \operatorname{diag} \mathbf{c}_{n_1} - V_{n_1} D_{n_1} S^{\mathrm{II}}_{n_1} J_{n_1} \operatorname{diag} \mathbf{s}_{n_1}\right). \end{aligned} \tag{2.9}$$

2. By (2.2) and
$$\cos \tfrac{(4j+3)(2k+1)\pi}{4n} = \cos \tfrac{(j+1)(2k+1)\pi}{n} \cos \tfrac{(2k+1)\pi}{4n} + \sin \tfrac{(j+1)(2k+1)\pi}{n} \sin \tfrac{(2k+1)\pi}{4n},$$
we obtain that
$$\begin{aligned} &\tfrac{1}{\sqrt{n_1}} \left(\cos \tfrac{(4j+3)(2k+1)\pi}{4n}\right)^{n_1-1}_{j,k=0} \\ &\quad = \tfrac{1}{\sqrt{2}} \left(V^{\mathrm{T}}_{n_1} C^{\mathrm{II}}_{n_1} \operatorname{diag} \mathbf{c}_{n_1} + I''_{n_1} S^{\mathrm{II}}_{n_1} \operatorname{diag} \mathbf{s}_{n_1}\right) \\ &\quad = \tfrac{1}{\sqrt{2}} \left(V^{\mathrm{T}}_{n_1} C^{\mathrm{II}}_{n_1} \operatorname{diag} \mathbf{c}_{n_1} + I''_{n_1} D_{n_1} S^{\mathrm{II}}_{n_1} J_{n_1} \operatorname{diag} \mathbf{s}_{n_1}\right). \end{aligned} \tag{2.10}$$

3. By (2.2) and

$$\begin{aligned}&\cos\tfrac{(4j+1)(n+2k+1)\pi}{4n}\\&\quad=(-1)^j\cos\left(\tfrac{j(2k+1)\pi}{n}+\tfrac{(n+2k+1)\pi}{4n}\right)\\&\quad=(-1)^j\cos\tfrac{j(2k+1)\pi}{n}\sin\tfrac{(n-2k-1)\pi}{4n}\\&\qquad-(-1)^j\sin\tfrac{j(2k+1)\pi}{n}\cos\tfrac{(n-2k-1)\pi}{4n},\end{aligned}$$

it follows that

$$\begin{aligned}&\tfrac{1}{\sqrt{n_1}}\left(\cos\tfrac{(4j+1)(n+2k+1)\pi}{4n}\right)_{j,k=0}^{n_1-1}\\&\quad=\tfrac{1}{\sqrt{2}}\left(D_{n_1}I'_{n_1}C^{\mathrm{II}}_{n_1}\operatorname{diag}(J_{n_1}\mathbf{s}_{n_1})-D_{n_1}V_{n_1}S^{\mathrm{II}}_{n_1}\operatorname{diag}(J_{n_1}\mathbf{c}_{n_1})\right)\\&\quad=\tfrac{1}{\sqrt{2}}\left(I'_{n_1}C^{\mathrm{II}}_{n_1}J_{n_1}\operatorname{diag}(J_{n_1}\mathbf{s}_{n_1})+V_{n_1}D_{n_1}S^{\mathrm{II}}_{n_1}\operatorname{diag}(J_{n_1}\mathbf{c}_{n_1})\right).\end{aligned}\tag{2.11}$$

Here we have used that $D_{n_1}I'_{n_1}=I'_{n_1}D_{n_1}$ and $-D_{n_1}V_{n_1}=V_{n_1}D_{n_1}$.

4. By (2.2) and

$$\begin{aligned}&\cos\tfrac{(4j+3)(n+2k+1)\pi}{4n}\\&\quad=(-1)^{j+1}\cos\left(\tfrac{(j+1)(2k+1)\pi}{n}-\tfrac{(n+2k+1)\pi}{4n}\right)\\&\quad=(-1)^{j+1}\cos\tfrac{(j+1)(2k+1)\pi}{n}\sin\tfrac{(n-2k-1)\pi}{4n}\\&\qquad+(-1)^{j+1}\sin\tfrac{(j+1)(2k+1)\pi}{n}\cos\tfrac{(n-2k-1)\pi}{4n},\end{aligned}$$

we obtain that

$$\begin{aligned}&\tfrac{1}{\sqrt{n_1}}\left(\cos\tfrac{(4j+3)(n+2k+1)\pi}{4n}\right)_{j,k=0}^{n_1-1}\\&\quad=-\tfrac{1}{\sqrt{2}}\left(D_{n_1}V^{\mathrm{T}}_{n_1}C^{\mathrm{II}}_{n_1}\operatorname{diag}(J_{n_1}\mathbf{s}_{n_1})+D_{n_1}I''_{n_1}S^{\mathrm{II}}_{n_1}\operatorname{diag}(J_{n_1}\mathbf{c}_{n_1})\right)\\&\quad=\tfrac{1}{\sqrt{2}}\left(V^{\mathrm{T}}_{n_1}D_{n_1}C^{\mathrm{II}}_{n_1}\operatorname{diag}(J_{n_1}\mathbf{s}_{n_1})-I''_{n_1}D_{n_1}S^{\mathrm{II}}_{n_1}\operatorname{diag}(J_{n_1}\mathbf{c}_{n_1})\right)\\&\quad=\tfrac{1}{\sqrt{2}}\left(V^{\mathrm{T}}_{n_1}C^{\mathrm{II}}_{n_1}J_{n_1}\operatorname{diag}(J_{n_1}\mathbf{s}_{n_1})-I''_{n_1}D_{n_1}S^{\mathrm{II}}_{n_1}\operatorname{diag}(J_{n_1}\mathbf{c}_{n_1})\right).\end{aligned}\tag{2.12}$$

Using the relations (2.9)–(2.12), we get the following factorization:

$$P_nC^{\mathrm{IV}}_n=\tfrac{1}{\sqrt{2}}\begin{pmatrix}I'_{n_1}&V_{n_1}D_{n_1}\\V^{\mathrm{T}}_{n_1}&-I''_{n_1}D_{n_1}\end{pmatrix}\left(C^{\mathrm{II}}_{n_1}\oplus S^{\mathrm{II}}_{n_1}\right)\begin{pmatrix}\operatorname{diag}\mathbf{c}_{n_1}&(\operatorname{diag}\mathbf{s}_{n_1})J_{n_1}\\-J_{n_1}\operatorname{diag}\mathbf{s}_{n_1}&\operatorname{diag}(J_{n_1}\mathbf{c}_{n_1})\end{pmatrix},$$

where

$$\begin{pmatrix}I'_{n_1}&V_{n_1}D_{n_1}\\V^{\mathrm{T}}_{n_1}&-I''_{n_1}D_{n_1}\end{pmatrix}=\left(\sqrt{2}\oplus\begin{pmatrix}I_{n_1-1}&I_{n_1-1}\\I_{n_1-1}&-I_{n_1-1}\end{pmatrix}\oplus\left(-\sqrt{2}\right)\right)(I_{n_1}\oplus D_{n_1}).$$

Thus (2.8) follows by the intertwining relation (2.3), namely $S^{\mathrm{II}}_{n_1} = J_{n_1} C^{\mathrm{II}}_{n_1} D_{n_1}$. The orthogonality of $A_n(1)$ and $T_n(1)$ can be shown by simple calculation. Note that $T_n(1)$ consists only of n_1 plane rotations/rotation–reflections. □

Next, we give new orthogonal factorizations for C^{III}_n and S^{III}_n. These factorizations together with those of Lemma 2.2 will give rise to a fast DCT-I algorithm (see Section 3).

Lemma 2.5. *Let $n \geqslant 4$ be an even integer.*

(i) *The matrix C^{III}_n can be factorized in the form*

$$C^{\mathrm{III}}_n = P^{\mathrm{T}}_n \tilde{A}_n(0)\big(C^{\mathrm{I}}_{n_1+1} \oplus S^{\mathrm{I}}_{n_1-1}\big)\tilde{T}_n(0). \tag{2.13}$$

(ii) *The matrix S^{III}_n can be factorized in the form*

$$S^{\mathrm{III}}_n = P^{\mathrm{T}}_n (I_{n_1} \oplus (-I_{n_1}))\tilde{A}_n(0)\big(C^{\mathrm{I}}_{n_1+1} \oplus S^{\mathrm{I}}_{n_1-1}\big)\tilde{T}_n(0) J_n. \tag{2.14}$$

Proof. The proof of (2.13) follows similar lines as the proof of Lemma 2.4 and is therefore omitted. Using (2.3), we obtain $C^{\mathrm{III}}_n J_n = D_n S^{\mathrm{III}}_n$ and hence $S^{\mathrm{III}}_n = D_n C^{\mathrm{III}}_n J_n$. By (2.13) and $P_n D_n = (I_{n_1} \oplus (-I_{n_1}))P_n$, it follows the factorization (2.14) of S^{III}_n. □

Remark 2.6. Lemmas 2.2 and 2.4 imply that

$$C^{\mathrm{II}}_n = P^{\mathrm{T}}_n\big(I_{n_1} \oplus P^{\mathrm{T}}_{n_1} A_{n_1}(1)\big)\big(C^{\mathrm{II}}_{n_1} \oplus C^{\mathrm{II}}_{n_2} \oplus C^{\mathrm{II}}_{n_2}\big)(I_{n_1} \oplus T_{n_1}(1))T_n(0)$$

which is similar to the formula (2) given in [5]. In fact, that formula also contains the matrices $T_n(0)$ and $C^{\mathrm{II}}_{n_1} \oplus C^{\mathrm{II}}_{n_2} \oplus C^{\mathrm{II}}_{n_2}$. However, the factorization (2) given in [5] is not correct.

The formula

$$\begin{aligned} C^{\mathrm{I}}_{n+1} = {} & P^{\mathrm{T}}_{n+1}\big(I_{n_1+1} \oplus P^{\mathrm{T}}_{n_1} \tilde{A}_{n_1}(0)\big)\big(C^{\mathrm{I}}_{n_1+1} \oplus C^{\mathrm{I}}_{n_2+1} \oplus S^{\mathrm{I}}_{n_2-1}\big) \\ & \times (I_{n_1+1} \oplus \tilde{T}_{n_1}(0))\tilde{T}_{n+1}(1) \end{aligned}$$

following from (2.5) and (2.13) is the corrected and general version of the factorization for C^{I}_{n+1} given for $n = 16$ in [22]. Observe that the orthogonal matrix C^{I}_{n+1} considered here differs from the non-orthogonal matrix in [22]. However, the splitting formulas (2.5) and (2.13) and the connection between DCT-I and DCT-III have not been recognized in [22].

2.3. *Examples for $n = 4$ and $n = 8$*

Now we give a detailed description of the structures of C^{II}_4, C^{IV}_4, C^{II}_8, and C^{IV}_8. Here, we want to assume that a matrix–vector product of the form

$$\begin{pmatrix} a_0 & a_1 \\ -a_1 & a_0 \end{pmatrix}\begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$

is realized by 2 additions and 4 multiplications (see Remark 2.11). As we shall see, the factorization of C_n^{II} in Remark 2.6 for $n = 8$ can be used to construct a fast algorithm which needs only 10 butterfly operations and 3 rotations/rotation-reflections.

Example 2.7. For $n = 4$ we obtain by Lemma 2.2 that

$$C_4^{\mathrm{II}} = P_4^{\mathrm{T}} \left(C_2^{\mathrm{II}} \oplus C_2^{\mathrm{IV}}\right) T_4(0)$$

with

$$C_2^{\mathrm{II}} = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad C_2^{\mathrm{IV}} = \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} \end{pmatrix},$$

$$T_4(0) = \tfrac{1}{\sqrt{2}} \begin{pmatrix} I_2 & J_2 \\ I_2 & -J_2 \end{pmatrix}.$$

Note that $P_4^{\mathrm{T}} = P_4$. This yields

$$C_4^{\mathrm{II}} = \tfrac{1}{2} P_4 \left(\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \oplus \sqrt{2} \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} \end{pmatrix} \right) \begin{pmatrix} I_2 & J_2 \\ I_2 & -J_2 \end{pmatrix}$$

such that $C_4^{\mathrm{II}}\mathbf{x}$ with $\mathbf{x} \in \mathbb{R}^4$ can be computed with 8 additions and 4 multiplications. The final scaling with $1/2$ is not counted. We see that only 3 butterfly operations and 1 scaled rotation–reflection are required.

Example 2.8. For $n = 4$ we obtain by Lemma 2.4 that

$$\begin{aligned} C_4^{\mathrm{IV}} &= P_4^{\mathrm{T}} A_4(1) \big(C_2^{\mathrm{II}} \oplus C_2^{\mathrm{II}}\big) T_4(1) \\ &= \frac{\sqrt{2}}{2} P_4^{\mathrm{T}} A_4(1) \big(\sqrt{2} C_2^{\mathrm{II}} \oplus \sqrt{2} C_2^{\mathrm{II}}\big) T_4(1) \end{aligned}$$

with

$$A_4(1) = \tfrac{1}{\sqrt{2}} \begin{pmatrix} \sqrt{2} & & & \\ & 1 & & 1 \\ & 1 & & -1 \\ & & \sqrt{2} & \end{pmatrix},$$

$$T_4(1) = \begin{pmatrix} \cos\frac{\pi}{16} & & & \sin\frac{\pi}{16} \\ & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \\ & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & \\ \sin\frac{\pi}{16} & & & -\cos\frac{\pi}{16} \end{pmatrix}.$$

Hence, we can compute $C_4^{\mathrm{IV}}\mathbf{x}$ with $\mathbf{x} \in \mathbb{R}^4$ with 10 additions and 10 multiplications. We see that only 3 butterfly operations, 1 rotation, and 1 rotation–reflection are required.

Example 2.9. For $n = 8$ we obtain by Lemmas 2.2 and 2.4 that

$$C_8^{\mathrm{II}} = P_8^{\mathrm{T}}(P_4 \oplus P_4)(I_4 \oplus A_4(1)) \left(C_2^{\mathrm{II}} \oplus C_2^{\mathrm{IV}} \oplus C_2^{\mathrm{II}} \oplus C_2^{\mathrm{II}}\right) \times (T_4(0) \oplus T_4(1))T_8(0)$$

with

$$T_8(0) = \tfrac{1}{\sqrt{2}} \begin{pmatrix} I_4 & J_4 \\ I_4 & -J_4 \end{pmatrix}.$$

Note that $B_8 := P_8^{\mathrm{T}}(P_4 \oplus P_4)$ coincides with the bit reversal matrix (see [24, pp. 36–43]). This yields the factorization

$$C_8^{\mathrm{II}} = \tfrac{\sqrt{2}}{4} B_8(I_4 \oplus A_4(1))\left(\sqrt{2}C_2^{\mathrm{II}} \oplus \sqrt{2}C_2^{\mathrm{IV}} \oplus \sqrt{2}C_2^{\mathrm{II}} \oplus \sqrt{2}C_2^{\mathrm{II}}\right) \times \left(\sqrt{2}T_4(0) \oplus \sqrt{2}T_4(1)\right) \begin{pmatrix} I_4 & J_4 \\ I_4 & -J_4 \end{pmatrix}$$

which coincides with that of Loeffler et al. [13]. Note that

$$\sqrt{2}C_2^{\mathrm{II}} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

is a butterfly matrix. If we compute $C_8^{\mathrm{II}}\mathbf{x}$ for arbitrary $\mathbf{x} \in \mathbb{R}^8$, then the algorithm based on the above factorization requires only 26 additions and 14 multiplications (not including the final scaling by $\sqrt{2}/4$). Further we see that only 10 (scaled) butterfly operations, 1 scaled rotation, and 2 scaled rotation–reflections are used.

Example 2.10. For $n = 8$ we obtain by Lemmas 2.2 and 2.4 that

$$\begin{aligned} C_8^{\mathrm{IV}} &= P_8^{\mathrm{T}} A_8(1)(P_4 \oplus P_4)\left(C_2^{\mathrm{II}} \oplus C_2^{\mathrm{IV}} \oplus C_2^{\mathrm{II}} \oplus C_2^{\mathrm{IV}}\right)(T_4(0) \oplus T_4(0))T_8(1) \\ &= \tfrac{\sqrt{2}}{4} P_8^{\mathrm{T}} \sqrt{2} A_8(1)(P_4 \oplus P_4)\left(\sqrt{2}C_2^{\mathrm{II}} \oplus \sqrt{2}C_2^{\mathrm{IV}} \oplus \sqrt{2}C_2^{\mathrm{II}} \oplus \sqrt{2}C_2^{\mathrm{IV}}\right) \\ &\quad \times \left(\sqrt{2}T_4(0) \oplus \sqrt{2}T_4(0)\right) T_8(1) \end{aligned}$$

with the cross-shaped twiddle matrix

$$T_8(1) = \begin{pmatrix} \cos\frac{\pi}{32} & & & & & & & \sin\frac{\pi}{32} \\ & \cos\frac{3\pi}{32} & & & & & \sin\frac{3\pi}{32} & \\ & & \cos\frac{5\pi}{32} & & & \sin\frac{5\pi}{32} & & \\ & & & \cos\frac{7\pi}{32} & \sin\frac{7\pi}{32} & & & \\ & & & -\sin\frac{7\pi}{32} & \cos\frac{7\pi}{32} & & & \\ & & \sin\frac{5\pi}{32} & & & -\cos\frac{5\pi}{32} & & \\ & -\sin\frac{3\pi}{32} & & & & & \cos\frac{3\pi}{32} & \\ \sin\frac{\pi}{32} & & & & & & & -\cos\frac{\pi}{32} \end{pmatrix}$$

and the modified addition matrix

$$A_8(1) = \tfrac{1}{\sqrt{2}} \left(\sqrt{2} \oplus \begin{pmatrix} I_3 & D_3 \\ I_3 & -D_3 \end{pmatrix} \oplus \sqrt{2} \right) (I_4 \oplus J_4).$$

Remark 2.11. In [8,13] one can find the identity

$$\begin{pmatrix} a_0 & a_1 \\ -a_1 & a_0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} a_0 + a_1 & & \\ & a_1 & \\ & & a_0 - a_1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 1 & -1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}.$$

If the terms $a_0 + a_1$ and $a_0 - a_1$ are precomputed, then this formula suggests an algorithm with 3 additions and 3 multiplications. Using this method, the scaled DCT-II of length 8 in Example 2.9 needs only 11 multiplications and 29 additions. However, the numerical stability of this method is worse than the usual algorithm with 2 additions and 4 multiplications (see [27]).

For a rotation matrix

$$\begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix} \quad (\varphi \in (-\pi, \pi)),$$

there is a second way to realize the matrix vector multiplication with only 3 multiplications and 3 additions. Namely, using 3 *lifting steps* [6], one finds the factorization

$$\begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix} = \begin{pmatrix} 1 & \tan(\varphi/2) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\sin\varphi & 1 \end{pmatrix} \begin{pmatrix} 1 & \tan(\varphi/2) \\ 0 & 1 \end{pmatrix}.$$

The above matrix factors are not orthogonal. However, it has been shown (see [27]) that for small rotation angle φ the roundoff error is less than for classical rotation. The same method is also applicable to the rotation–reflection matrix

$$\begin{pmatrix} \cos\varphi & \sin\varphi \\ \sin\varphi & -\cos\varphi \end{pmatrix} \quad (\varphi \in (-\pi, \pi)).$$

3. Fast DCT algorithms

In this section we present some new fast DCT algorithms. Using Lemmas 2.2 and 2.4, we are able to present fast DCT-II and DCT-IV algorithms in the form of recursive procedures. In order to reduce the number of multiplications, we move the factor $1/\sqrt{2}$ in the matrices $T_n(0)$ and $T_n(1)$ to the end of the calculation such

that for given $\mathbf{x} \in \mathbb{R}^n$ we compute $\mathbf{y} = \sqrt{n} C_n^{\mathrm{II}} \mathbf{x}$ and $\mathbf{y} = \sqrt{n} C_n^{\mathrm{IV}} \mathbf{x}$, respectively. The corresponding recursive procedures are called **cos-II**$(\mathbf{x}, n)$ and **cos-IV**$(\mathbf{x}, n)$.

Algorithm 3.1 (**cos-II**$(\mathbf{x}, n)$).
Input: $n = 2^t$ $(t \geqslant 1)$, $n_1 = n/2$, $\mathbf{x} \in \mathbb{R}^n$.

1. If $n = 2$, then

$$\mathbf{y} := \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \mathbf{x}.$$

2. If $n \geqslant 4$, then

$$\begin{aligned} (u_j)_{j=0}^{n-1} &:= \sqrt{2} T_n(0) \mathbf{x}, \\ \mathbf{v}' &:= \textbf{cos-II}\big((u_j)_{j=0}^{n_1-1}, n_1\big), \\ \mathbf{v}'' &:= \textbf{cos-IV}\big((u_j)_{j=n_1}^{n-1}, n_1\big), \\ \mathbf{y} &:= P_n^{\mathrm{T}} \big((\mathbf{v}')^{\mathrm{T}}, (\mathbf{v}'')^{\mathrm{T}}\big)^{\mathrm{T}}. \end{aligned}$$

Output: $\mathbf{y} = \sqrt{n} C_n^{\mathrm{II}} \mathbf{x}$.

Algorithm 3.2 (**cos-IV**$(\mathbf{x}, n)$).
Input: $n = 2^t$ $(t \geqslant 1)$, $n_1 = n/2$, $\mathbf{x} \in \mathbb{R}^n$.

1. If $n = 2$, then

$$\mathbf{y} := \sqrt{2} C_2^{\mathrm{IV}} \mathbf{x}.$$

2. If $n \geqslant 4$, then

$$\begin{aligned} (u_j)_{j=0}^{n-1} &:= \sqrt{2} T_n(1) \mathbf{x}, \\ \mathbf{v}' &:= \textbf{cos-II}\big((u_j)_{j=0}^{n_1-1}, n_1\big), \\ \mathbf{v}'' &:= \textbf{cos-II}\big((u_j)_{j=n_1}^{n-1}, n_1\big), \\ \mathbf{w} &:= A_n(1) \big((\mathbf{v}')^{\mathrm{T}}, (\mathbf{v}'')^{\mathrm{T}}\big)^{\mathrm{T}}, \\ \mathbf{y} &:= P_n^{\mathrm{T}} \mathbf{w}. \end{aligned}$$

Output: $\mathbf{y} = \sqrt{n} C_n^{\mathrm{IV}} \mathbf{x}$.

Observe that the two algorithms depend on each other. For the DCT-II algorithm this can be easily overcome by inserting Algorithm 3.2 into Algorithm 3.1. The corresponding recursive algorithm is then based on the matrix factorization

$$\begin{aligned} \sqrt{n} C_n^{\mathrm{II}} = {} & P_n^{\mathrm{T}} \big(I_{n_1} \oplus P_{n_1}^{\mathrm{T}} A_{n_1}(1)\big) \\ & \times \Big(\sqrt{n_1} C_{n_1}^{\mathrm{II}} \oplus \big(\sqrt{n_2} C_{n_2}^{\mathrm{II}} \oplus \sqrt{n_2} C_{n_2}^{\mathrm{II}}\big) \sqrt{2} T_{n_1}(1)\Big) \sqrt{2} T_n(0), \end{aligned} \tag{3.1}$$

which is directly derived from (2.4) and (2.8). This factorization is almost orthogonal.

The number of arithmetic operations required to carry out a computation is called the *arithmetic cost*. Note that multiplications with ± 1 or 2^k for some $k \in \mathbb{Z}$ and permutations are not counted. Now we determine the arithmetic costs of these fast DCT-II and DCT-IV algorithms. For an arbitrary real matrix M_n of order n, let $\alpha(M_n)$ and $\mu(M_n)$ denote the number of additions and multiplications for computing $M_n \mathbf{x}$ with arbitrary $\mathbf{x} \in \mathbb{R}^n$. Analogously, the number of additions and multiplications of a fast DCT-II algorithm of length n is denoted by $\alpha(\text{DCT-II}, n)$ and $\mu(\text{DCT-II}, n)$, respectively.

Theorem 3.3. *Let $n = 2^t$ $(t \geqslant 2)$ be given. Using Algorithms 3.1 and 3.2, the arithmetic cost of the fast* DCT-II *algorithm of length n is given by*

$$\alpha(\text{DCT-II}, n) = \tfrac{4}{3} nt - \tfrac{8}{9} n - \tfrac{1}{9}(-1)^t + 1,$$
$$\mu(\text{DCT-II}, n) = nt - \tfrac{4}{3} n + \tfrac{1}{3}(-1)^t + 1.$$

Further, the arithmetic cost of the fast DCT-IV *algorithm of length n is determined by*

$$\alpha(\text{DCT-IV}, n) = \tfrac{4}{3} nt - \tfrac{2}{9} n + \tfrac{2}{9}(-1)^t,$$
$$\mu(\text{DCT-IV}, n) = nt + \tfrac{2}{3} n - \tfrac{2}{3}(-1)^t.$$

Proof. We compute only $\alpha(\text{DCT-II}, n)$ and $\alpha(\text{DCT-IV}, n)$. The results for $\mu(\text{DCT-II}, n)$ and $\mu(\text{DCT-IV}, n)$ can be derived analogously. From Examples 2.7 and 2.8 it follows that

$$\begin{aligned} &\alpha(\text{DCT-II}, 2) = 2, \quad \alpha(\text{DCT-II}, 4) = 8, \\ &\alpha(\text{DCT-IV}, 2) = 2, \quad \alpha(\text{DCT-IV}, 4) = 10. \end{aligned} \tag{3.2}$$

For $n = 2^t$ $(t \geqslant 3)$ we obtain by Algorithms 3.1 and 3.2 that

$$\alpha(\text{DCT-II}, n) = \alpha\big(\sqrt{2} T_n(0)\big) + \alpha(\text{DCT-II}, n_1) + \alpha(\text{DCT-IV}, n_1), \tag{3.3}$$
$$\alpha(\text{DCT-IV}, n) = \alpha\big(\sqrt{2} T_n(1)\big) + 2\alpha(\text{DCT-II}, n_1) + \alpha(A_n(1)). \tag{3.4}$$

Using the definitions of the matrices $T_n(0)$, $T_n(1)$ and $A_n(1)$ in beginning of Section 2, we see immediately that

$$\alpha\big(\sqrt{2} T_n(0)\big) = \alpha\big(\sqrt{2} T_n(1)\big) = n, \quad \alpha(A_n(1)) = n - 2.$$

Thus by (3.3) and (3.4) we obtain the linear difference equation of order 2 (with respect to $t \geqslant 3$)

$$\alpha(\text{DCT-II}, 2^t) = \alpha(\text{DCT-II}, 2^{t-1}) + 2\alpha(\text{DCT-II}, 2^{t-2}) + 2^{t+1} - 2.$$

Solving it under the initial conditions (3.2), we obtain that

$$\alpha(\text{DCT-II}, 2^t) = \tfrac{4}{3} nt - \tfrac{8}{9} n - \tfrac{1}{9}(-1)^t + 1$$

and hence by (3.4)

$$\alpha(\text{DCT-IV}, 2^t) = \tfrac{4}{3}nt - \tfrac{2}{9}n + \tfrac{2}{9}(-1)^t.$$

This completes the proof. □

Compared with the best known algorithms for DCT-II of length n, which need only $2n \log_2 n$ arithmetic operations (see e.g. [8,19,20,14]), the obtained algorithms are not the fastest. But, as we will see in Section 5, they possess an excellent numerical stability, which outperforms the faster algorithms.

In order to delight this fact, we shall give here a slightly modified recursive algorithm for the DCT-II having (almost) slowest arithmetic costs, but the orthogonality of the matrix factors in the underlying factorization is given up. Instead of (3.1) consider now the matrix factorization for $n \geqslant 8$,

$$\begin{aligned}\sqrt{n}C_n^{\mathrm{T}} = {} & P_n^{\mathrm{T}}\big(I_{n_1} \oplus \sqrt{2}P_{n_1}^{\mathrm{T}} A_{n_1}(1)\big) \\ & \times \big(\sqrt{n_1}C_{n_1}^{\mathrm{II}} \oplus \big(\sqrt{n_2}C_{n_2}^{\mathrm{II}} \oplus \sqrt{n_2}C_{n_2}\big)T_{n_1}(1)\big)\sqrt{2}T_n(0).\end{aligned}$$

Here the matrix factors $I_{n_1} \oplus \sqrt{2}P_{n_1}^{\mathrm{T}} A_{n_1}(1)$ and $\sqrt{n_1}C_{n_1}^{\mathrm{II}} \oplus (\sqrt{n_2}\,(C_{n_2}^{\mathrm{II}} \oplus \sqrt{n_2}C_{n_2}^{\mathrm{II}})$ $T_{n_1}(1)$ are not longer (almost) orthogonal. The corresponding recursive procedure of this *modified* DCT-II (MDCT-II) reads as follows:

Algorithm 3.4 (**mcos-II**($\mathbf{x}$, n)).
Input: $n = 2^t$ ($t \geqslant 1$), $n_1 = n/2$, $n_2 = n/4$, $\mathbf{x} \in \mathbb{R}^n$.

1. If $n = 2$, then

$$\mathbf{y} := \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\mathbf{x}.$$

2. If $n = 4$, then

$$\mathbf{y} := P_4\left(\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \oplus \begin{pmatrix} \sqrt{2}\cos\frac{\pi}{8} & \sqrt{2}\sin\frac{\pi}{8} \\ \sqrt{2}\sin\frac{\pi}{8} & -\sqrt{2}\cos\frac{\pi}{8} \end{pmatrix}\right)\begin{pmatrix} I_2 & J_2 \\ I_2 & -J_2 \end{pmatrix}\mathbf{x}.$$

3. If $n \geqslant 8$, then

$$\begin{aligned}(u_j)_{j=0}^{n-1} &:= \sqrt{2}T_n(0)\mathbf{x}, \\ \mathbf{v}' &:= \textbf{mcos-II}\big((u_j)_{j=0}^{n_1-1}, n_1\big), \\ \big(v_j^{(1)}\big)_{j=0}^{n_1-1} &:= T_{n_1}(1)(u_j)_{j=n_1}^{n-1}, \\ \mathbf{w}' &:= \textbf{mcos-II}\big(\big(v_j^{(1)}\big)_{j=0}^{n_2-1}, n_2\big), \\ \mathbf{w}'' &:= \textbf{mcos-II}\big(\big(v_j^{(1)}\big)_{j=n_2}^{n_1-1}, n_2\big), \\ \mathbf{v}'' &:= \sqrt{2}P_{n_1}^{\mathrm{T}} A_{n_1}(1)\big((\mathbf{w}')^{\mathrm{T}}, (\mathbf{w}'')^{\mathrm{T}}\big)^{\mathrm{T}}, \\ \mathbf{y} &:= P_n^{\mathrm{T}}\big((\mathbf{v}')^{\mathrm{T}}, (\mathbf{v}'')^{\mathrm{T}}\big)^{\mathrm{T}}.\end{aligned}$$

Output: $\mathbf{y} = \sqrt{n} C_n^{\mathrm{II}} \mathbf{x}$.

For this MDCT-II algorithm we obtain

$$\begin{aligned}\alpha(\text{MDCT-II}, n) &= \alpha\big(\sqrt{2}T_n(0)\big) + \alpha(\text{MDCT-II}, n_1) + \alpha(T_{n_1}(1)) \\ &\quad + 2\alpha(\text{MDCT-II}, n_2) + \alpha\big(\sqrt{2}A_{n_1}(1)\big),\end{aligned}$$

$$\begin{aligned}\mu(\text{MDCT-II}, n) &= \mu\big(\sqrt{2}T_n(0)\big) + \mu(\text{MDCT-II}, n_1) + \mu(T_{n_1}(1)) \\ &\quad + 2\mu(\text{MDCT-II}, n_2) + \mu\big(\sqrt{2}A_{n_1}(1)\big),\end{aligned}$$

and by

$$\begin{aligned}&\alpha\big(\sqrt{2}T_n(0)\big) = n, \quad \alpha(T_{n_1}(1)) = n_1, \quad \alpha\big(\sqrt{2}A_{n_1}(1)\big) = n_1 - 2, \\ &\mu\big(\sqrt{2}T_n(0)\big) = 0, \quad \mu(T_{n_1}(1)) = n, \quad \mu\big(\sqrt{2}A_{n_1}(1)\big) = 2,\end{aligned}$$

we find

$$\begin{aligned}\alpha(\text{MDCT-II}, n) &= \tfrac{4}{3}nt - \tfrac{8}{9}n - \tfrac{1}{9}(-1)^t + 1, \\ \mu(\text{MDCT-II}, n) &= \tfrac{2}{3}nt - \tfrac{1}{9}n + \tfrac{1}{9}(-1)^t - 1.\end{aligned}$$

Remark 3.5. For comparison, the algorithm presented by Wang [25] (which already outperforms the algorithm in [4]) for the DCT-II of length $n = 2^t$ needs $\frac{3}{4}nt - n + 3$ multiplications and $\frac{7}{4}nt - 2n + 3$ additions. The arithmetic cost of our MDCT-II algorithm is even comparable with fast DCT-II algorithms based on polynomial arithmetic which need $\frac{1}{2}nt$ multiplications and $\frac{3}{2}nt - n + 1$ additions (see e.g. [7,11,12, 19,20]). Namely, using the method of Remark 2.11 computing a (scaled) rotation matrix/rotation–reflection matrix with only 3 multiplications and 3 additions, we obtain for the MDCT-II Algorithm 3.4

$$\alpha(\text{MDCT-II}, n) = \tfrac{3}{2}nt - n + 1, \quad \mu(\text{MDCT-II}, n) = \tfrac{1}{2}nt - 1.$$

The idea of the MDCT-II Algorithm 3.4 can also be used to compute the DCT-IV of length n with only $2n \log_2 n + n + 1$ arithmetical operations. However, as we will see in Section 5, this MDCT-II algorithm does not have a very good numerical stability.

Now, using Lemmas 2.2 and 2.5, we obtain fast DCT-I, DCT-III and DST-I algorithms in recursive form:

Algorithm 3.6 (**cos-I**($\mathbf{x}$, $n + 1$)).
Input: $n = 2^t$ ($t \geqslant 1$), $n_1 = n/2$, $\mathbf{x} \in \mathbb{R}^{n+1}$.

1. If $n = 2$, then

$$\mathbf{y} := \tfrac{1}{2}\begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & \sqrt{2} \\ 1 & -1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 & 1 \\ 0 & \sqrt{2} & 0 \\ 1 & 0 & -1 \end{pmatrix}\mathbf{x}.$$

2. If $n \geqslant 4$, then

$$\begin{aligned}(u_j)_{j=0}^{n} &:= \sqrt{2}\tilde{T}_{n+1}(1)\mathbf{x},\\ \mathbf{v}' &:= \textbf{cos-I}\big((u_j)_{j=0}^{n_1}, n_1+1\big),\\ \mathbf{v}'' &:= \textbf{cos-III}\big((u_j)_{j=n_1+1}^{n}, n_1\big),\\ \mathbf{y} &:= P_{n+1}^{\mathrm{T}}\big((\mathbf{v}')^{\mathrm{T}}, (\mathbf{v}'')^{\mathrm{T}}\big)^{\mathrm{T}}.\end{aligned}$$

Output: $\mathbf{y} = \sqrt{n_1}C_{n+1}^{\mathrm{I}}\mathbf{x}$.

Algorithm 3.7 (**cos-III**$(\mathbf{x}, n)$).
Input: $n = 2^t$ $(t \geqslant 1)$, $n_1 = n/2$, $\mathbf{x} \in \mathbb{R}^n$.

1. If $n = 2$, then

$$\mathbf{y} := \tfrac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\mathbf{x}.$$

2. If $n \geqslant 4$, then

$$\begin{aligned}(u_j)_{j=0}^{n-1} &:= \sqrt{2}\tilde{T}_{n}(0)\mathbf{x},\\ \mathbf{v}' &:= \textbf{cos-I}\big((u_j)_{j=0}^{n_1}, n_1+1\big),\\ \mathbf{v}'' &:= \textbf{sin-I}\big((u_j)_{j=n_1+1}^{n-1}, n_1-1\big),\\ \mathbf{w} &:= \tilde{A}_n(0)\big((\mathbf{v}')^{\mathrm{T}}, (\mathbf{v}'')^{\mathrm{T}}\big)^{\mathrm{T}},\\ \mathbf{y} &:= P_n^{\mathrm{T}}\mathbf{w}.\end{aligned}$$

Output: $\mathbf{y} = \sqrt{n_1}C_n^{\mathrm{III}}\mathbf{x}$.

Algorithm 3.8 (**sin-I**$(\mathbf{x}, n-1)$).
Input: $n = 2^t$ $(t \geqslant 1)$, $n_1 = n/2$, $\mathbf{x} \in \mathbb{R}^{n-1}$.

1. If $n = 2$, then $\mathbf{y} := \mathbf{x}$.
2. If $n \geqslant 4$, then

$$\begin{aligned}(u_j)_{j=1}^{n-1} &:= \sqrt{2}\tilde{T}_{n-1}(-1)\mathbf{x},\\ \mathbf{v}' &:= \textbf{cos-III}\big((u_j)_{j=1}^{n_1}, n_1\big),\\ \mathbf{v}'' &:= \textbf{sin-I}\big((u_j)_{j=n_1+1}^{n-1}, n_1-1\big),\\ \mathbf{w} &:= \tilde{A}_{n-1}(-1)\big((\mathbf{v}')^{\mathrm{T}}, (\mathbf{v}'')^{\mathrm{T}}\big)^{\mathrm{T}},\\ \mathbf{y} &:= P_{n-1}^{\mathrm{T}}\mathbf{w}.\end{aligned}$$

Output: $\mathbf{y} = \sqrt{n_1} S^{\mathrm{I}}_{n-1} \mathbf{x}$.

Note that these three recursive algorithms can not be simply decoupled.

Theorem 3.9. *Let $n = 2^t$ $(t \geqslant 2)$ be given. Using Algorithms 3.6, 3.7 and 3.8, the arithmetic costs of the fast algorithms for* DCT-I, DCT-III *and* DST-I *are given by*

$$\begin{aligned}
&\alpha(\text{DCT-I}, n+1) = \tfrac{4}{3}nt - \tfrac{14}{9}n + t + \tfrac{7}{2} + \tfrac{1}{18}(-1)^t,\\
&\mu(\text{DCT-I}, n+1) = nt - \tfrac{4}{3}n + \tfrac{5}{2} - \tfrac{1}{6}(-1)^t,\\
&\alpha(\text{DCT-III}, n) = \tfrac{4}{3}nt - \tfrac{8}{9}n + 1 - \tfrac{1}{9}(-1)^t,\\
&\mu(\text{DCT-III}, n) = nt + \tfrac{2}{3}n - 1 + \tfrac{1}{3}(-1)^t,\\
&\alpha(\text{DST-I}, n-1) = \tfrac{4}{3}nt - \tfrac{14}{9}n - t + \tfrac{3}{2} + \tfrac{1}{18}(-1)^t,\\
&\mu(\text{DST-I}, n-1) = nt - \tfrac{4}{3}n + \tfrac{1}{2} - \tfrac{1}{6}(-1)^t.
\end{aligned}$$

Proof. We only compute the number of additions for the three algorithms. The number of multiplications can be derived analogously. We observe that by

$$\begin{aligned}
&\alpha\big(\sqrt{2}\tilde{T}_{n+1}(1)\big) = n, \quad \alpha\big(\sqrt{2}\tilde{T}_n(0)\big) = n-2, \quad \alpha\big(\sqrt{2}\tilde{T}_{n-1}(-1)\big) = n-2,\\
&\alpha\big(\tilde{A}_n(0)\big) = n-2, \quad \alpha\big(\tilde{A}_{n-1}(-1)\big) = 0,
\end{aligned}$$

it follows that

$$\begin{aligned}
&\alpha(\text{DCT-I}, n+1) = \alpha(\text{DCT-I}, n_1+1) + \alpha(\text{DCT-III}, n_1) + n,\\
&\alpha(\text{DCT-III}, n) = \alpha(\text{DCT-I}, n_1+1) + \alpha(\text{DST-I}, n_1-1) + 2n - 4,\\
&\alpha(\text{DST-I}, n-1) = \alpha(\text{DST-I}, n_1-1) + \alpha(\text{DCT-III}, n_1) + n - 2,
\end{aligned}$$

and hence

$$\begin{aligned}
&\alpha(\text{DCT-I}, n+1) + \alpha(\text{DST-I}, n-1)\\
&\quad = \alpha(\text{DCT-I}, n_1+1) + \alpha(\text{DST-I}, n_1-1)\\
&\qquad + 2(\alpha(\text{DCT-I}, n_2+1) + \alpha(\text{DST-I}, n_2-1)) + 4n - 10,
\end{aligned}$$

$$\begin{aligned}
&\alpha(\text{DCT-I}, n+1) - \alpha(\text{DST-I}, n-1)\\
&\quad = \alpha(\text{DCT-I}, n_1+1) - \alpha(\text{DST-I}, n_1-1) + 2\\
&\quad = \alpha(\text{DCT-I}, 3) - \alpha(\text{DST-I}, 1) + 2(t-1) = 2(t+1).
\end{aligned}$$

This leads to the linear difference equation (with respect to t)

$$\begin{aligned}
\alpha(\text{DCT-I}, 2^t+1) = {}&\alpha(\text{DCT-I}, 2^{t-1}+1) + 2\alpha(\text{DCT-I}, 2^{t-2}+1)\\
&+ 2^{t+1} - 2t - 2
\end{aligned}$$

which under the initial conditions $\alpha(\text{DCT-I}, 3) = 4$ and $\alpha(\text{DCT-I}, 5) = 10$ has the solution

$$\alpha(\text{DCT-I}, n+1) = \tfrac{4}{3}nt - \tfrac{14}{9}n + t + \tfrac{7}{2} + \tfrac{1}{18}(-1)^t.$$

Further, we now simply obtain

$$\begin{aligned}\alpha(\text{DST-I}, n-1) &= \alpha(\text{DCT-I}, n+1) - 2(t+1)\\ &= \tfrac{4}{3}nt - \tfrac{14}{9}n - t + \tfrac{3}{2} + \tfrac{1}{18}(-1)^t\end{aligned}$$

and

$$\begin{aligned}\alpha(\text{DCT-III}, n) &= \alpha(\text{DCT-I}, n+1) + \alpha(\text{DST-I}, n-1) + 2n - 4\\ &= \tfrac{4}{3}nt - \tfrac{8}{9}n + 1 - \tfrac{1}{9}(-1)^t.\end{aligned}$$

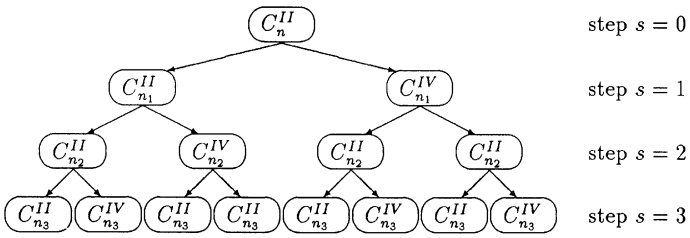
This completes the proof. □

Remark 3.10. As before, one can modify these algorithms slightly in order to reduce the arithmetic costs to $2n \log_2 n$ flops, but then giving up the orthogonality of the underlying matrix factorization and destroying the excellent numerical stability (see Section 5).

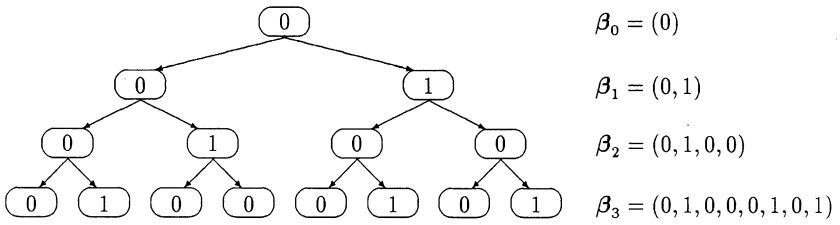
4. Factorizations of cosine matrices

A fast DCT algorithm is best understood by interpreting it as the application of a factorization of the corresponding cosine matrix. In this section, we present factorizations of the orthogonal cosine matrices of types I, II and IV into products of sparse and orthogonal matrices. These factorizations directly lead to iterative algorithms, which may be preferred on special platforms. Note that the following iterative DCT algorithms coincide with the ones of Section 3 and that they arise by resolving the recursions.

Let us start with the cosine matrix C_n^{II}. Recursive application of (2.4) and (2.8) provides the wanted factorization of C_n^{II}. Let $n = 2^t$ $(t \geqslant 2)$ be given. Further, let $n_s := 2^{t-s}$ $(s = 0, \ldots, t-1)$. In the first factorization step, C_n^{II} is splitted into $C_{n_1}^{\text{II}} \oplus C_{n_1}^{\text{IV}}$ by (2.4). Then in the second step, we use (2.4) and (2.8) in order to split $C_{n_1}^{\text{II}} \oplus C_{n_1}^{\text{IV}}$ into $C_{n_2}^{\text{II}} \oplus C_{n_2}^{\text{IV}} \oplus C_{n_2}^{\text{II}} \oplus C_{n_2}^{\text{II}}$. In the case $n_2 > 2$ we continue this procedure. Finally, we obtain a factorization of C_n^{II}. The first factorization steps are illustrated by the following diagram:



Note that the factorization tree of C_n^{II} contains a factorization tree of $C_{n_1}^{\mathrm{IV}}$. Now we use binary vectors $\boldsymbol{\beta}_s = (\beta_s(1), \ldots, \beta_s(2^s))$ for $s \in \{0, \ldots, t-1\}$ as introduced in [24, pp. 115–119]. We put $\beta_s(k) := 0$ if $C_{n_s}^{\mathrm{II}}$ stands at position $k \in \{1, \ldots, 2^s\}$ in step s, and $\beta_s(k) := 1$ if $C_{n_s}^{\mathrm{IV}}$ stands at position k in step s. By Lemma 2.2, from $\beta_s(k) = 0$ it follows immediately that $\beta_{s+1}(2k-1) = 0$ and $\beta_{s+1}(2k) = 1$. Further, from $\beta_s(k) = 1$ it follows that $\beta_{s+1}(2k-1) = \beta_{s+1}(2k) = 0$. Now, our diagram looks as follows:



For a simple computation of vectors $\boldsymbol{\beta}_s$ we need

Lemma 4.1. *Let $t \in \mathbb{N}$ with $t \geqslant 2$ be given and $\boldsymbol{\beta}_0 := (0)$. Then*

$$\boldsymbol{\beta}_{s+1} = (\boldsymbol{\beta}_s, \tilde{\boldsymbol{\beta}}_s), \quad s = 0, \ldots, t-2, \tag{4.1}$$

where $\tilde{\boldsymbol{\beta}}_s$ equals $\boldsymbol{\beta}_s$ with the exception that the last bit position is reversed. Further,

$$\|\boldsymbol{\beta}_s\|_1 = \sum_{k=1}^{2^s} \beta_s(k) = \tfrac{1}{3}(2^s - (-1)^s) \tag{4.2}$$

is the number of ones in the binary vector $\boldsymbol{\beta}_s$.

Proof. Formula (4.1) follows by induction. Formula (4.2) is a result of classical difference equation theory, taking into account that $\|\boldsymbol{\beta}_s\|_1 + \|\boldsymbol{\beta}_{s+1}\|_1 = 2^s$ and $\|\boldsymbol{\beta}_0\|_1 = 0$. □

For each pointer $\boldsymbol{\beta}_s$ we define the modified addition matrix

$$A_n(\boldsymbol{\beta}_s) := A_{n_s}(\beta_s(1)) \oplus \cdots \oplus A_{n_s}(\beta_s(2^s)), \quad s = 0, \ldots, t-2,$$

with $A_{n_s}(0) := I_{n_s}$ and $A_{n_s}(1)$ as in Lemma 2.4, further the modified twiddle matrix

$$T_n(\boldsymbol{\beta}_s) := T_{n_s}(\beta_s(1)) \oplus \cdots \oplus T_{n_s}(\beta_s(2^s)), \quad s = 0, \ldots, t-2,$$

with $T_{n_s}(0)$ and $T_{n_s}(1)$ as in Lemmas 2.2 and 2.4, and finally the block cosine matrix

$$C_n(\boldsymbol{\beta}_s) := C_{n_s}(\beta_s(1)) \oplus \cdots \oplus C_{n_s}(\beta_s(2^s)), \quad s = 0, \ldots, t-1,$$

with $C_{n_s}(0) := C_{n_s}^{\mathrm{II}}$ and $C_{n_s}(1) := C_{n_s}^{\mathrm{IV}}$. Additionally, we set $T_n(\boldsymbol{\beta}_{t-1}) := C_n(\boldsymbol{\beta}_{t-1})$. Note that $C_n(\boldsymbol{\beta}_0) = C_n^{\mathrm{II}}$. Further we introduce the permutation matrices

$$P_n(s) := P_{n_s}^{\mathrm{T}} \oplus \cdots \oplus P_{n_s}^{\mathrm{T}}, \quad s = 0, \ldots, t-2.$$

We shall see in the following that the block cosine matrix $C_n(\boldsymbol{\beta}_s)$ appears as intermediate result in our recursive factorization of C_n^{II}. By construction, all matrices $P_n(s)$, $A_n(\boldsymbol{\beta}_s)$ and $T_n(\boldsymbol{\beta}_s)$ are sparse and orthogonal.

Theorem 4.2. *Let $n = 2^t$ $(t \geqslant 2)$. Then C_n^{II} can be factorized into the following product of sparse orthogonal matrices*:

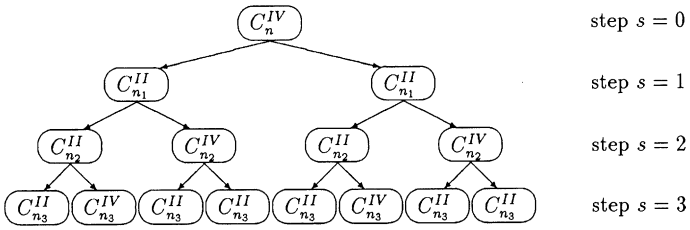
$$C_n^{\mathrm{II}} = (P_n(0)A_n(\boldsymbol{\beta}_0)) \cdots (P_n(t-2)A_n(\boldsymbol{\beta}_{t-2}))T_n(\boldsymbol{\beta}_{t-1}) \cdots T_n(\boldsymbol{\beta}_0). \tag{4.3}$$

Proof. The factorization (4.3) follows immediately from

$$C_n(\boldsymbol{\beta}_s) = P_n(s)A_n(\boldsymbol{\beta}_s)C_n(\boldsymbol{\beta}_{s+1})T_n(\boldsymbol{\beta}_s), \quad s = 0, \ldots, t-2. \tag{4.4}$$

By definition of $P_n(s)$, $A_n(\boldsymbol{\beta}_s)$, $C_n(\boldsymbol{\beta}_s)$ and $T_n(\boldsymbol{\beta}_s)$, these formulas (4.4) are direct consequences of Lemmas 2.2 and 2.4. □

The same technique can be applied to the cosine matrix of type IV with radix-2 order. Let $n = 2^t$ $(t \geqslant 2)$ be given. In the first step we can split C_n^{IV} into $C_{n_1}^{\mathrm{II}} \oplus C_{n_1}^{\mathrm{II}}$ by (2.8). Then in the second step, we use (2.4) in order to split $C_{n_1}^{\mathrm{II}} \oplus C_{n_1}^{\mathrm{II}}$ into $C_{n_2}^{\mathrm{II}} \oplus C_{n_2}^{\mathrm{IV}} \oplus C_{n_2}^{\mathrm{II}} \oplus C_{n_2}^{\mathrm{IV}}$. In the case $n_2 \geqslant 4$, we continue the procedure. This method is illustrated by the following diagram:



We introduce binary vectors $\boldsymbol{\gamma}_s := (\gamma_s(1), \ldots, \gamma_s(2^s))$, $s \in \{0, \ldots, t-1\}$. We put $\gamma_s(k) := 0$ if $C_{n_s}^{\mathrm{II}}$ stands at position $k \in \{1, \ldots, 2^s\}$ of step s, and $\gamma_s(k) := 1$ if $C_{n_s}^{\mathrm{IV}}$ stands at position k of step s. These pointers possess different properties as those in Lemma 4.1.

Lemma 4.3. *Let $t \in \mathbb{N}$ $(t \geqslant 2)$ and $\boldsymbol{\gamma}_0 := (1)$. Then*

$$\boldsymbol{\gamma}_{s+1} = (\tilde{\boldsymbol{\gamma}}_s, \boldsymbol{\gamma}_s), \quad s = 0, \ldots, t-2,$$

where $\tilde{\boldsymbol{\gamma}}_s$ equals $\boldsymbol{\gamma}_s$ with the exception that the last bit position is reversed. Further,

$$\|\boldsymbol{\gamma}_s\|_1 = \sum_{k=1}^{2^s} \gamma_s(k) = \tfrac{1}{3}(2^s + 2(-1)^s).$$

The proof is similar to that of Lemma 4.1 and is omitted here. Now, for each pointer $\boldsymbol{\gamma}_s$ we define $A_n(\boldsymbol{\gamma}_s)$ and $T_n(\boldsymbol{\gamma}_s)$ (or their modified versions) in the same way as $A_n(\boldsymbol{\beta}_s)$ and $T_n(\boldsymbol{\beta}_s)$.

Theorem 4.4. *Let $n = 2^t$ $(t \geqslant 2)$. Then the matrix C_n^{IV} can be factorized into the following product of sparse orthogonal matrices*:

$$C_n^{\mathrm{IV}} = (P_n(0)A_n(\boldsymbol{\gamma}_0)) \cdots (P_n(t-2)A_n(\boldsymbol{\gamma}_{t-2}))T_n(\boldsymbol{\gamma}_{t-1}) \cdots T_n(\boldsymbol{\gamma}_0).$$

The proof directly follows from Lemmas 2.2 and 2.4.

We now want to derive a matrix factorization for the MDCT-II Algorithm 3.4. We slightly change the derived orthogonal matrix product (4.3) in the following way. Instead of $A_n(\boldsymbol{\beta}_s)$, consider the modified addition matrices

$$A'_n(\boldsymbol{\beta}_s) = A'_{n_s}(\beta_s(1)) \oplus \cdots \oplus A'_{n_s}(\beta_s(2^s)), \quad s = 0, \ldots, t-2,$$

with $A'_{n_s}(0) := A_{n_s}(0) = I_{n_s}$ and $A'_{n_s}(1) := \sqrt{2}A_{n_s}(1)$. Hence $A_n(\boldsymbol{\beta}_s)$ and $A'_n(\boldsymbol{\beta}_s)$ are connected by

$$A'_n(\boldsymbol{\beta}_s) = D_n(\boldsymbol{\beta}_s)A_n(\boldsymbol{\beta}_s), \quad s = 0, \ldots, t-2,$$

with a diagonal matrix

$$D_n(\boldsymbol{\beta}_s) := \left(\sqrt{2}\right)^{\beta_s(1)} I_{n_s} \oplus \cdots \oplus \left(\sqrt{2}\right)^{\beta_s(2^s)} I_{n_s}, \quad s = 0, \ldots, t-2.$$

Further, consider the modified twiddle matrices

$$S'_n(\boldsymbol{\beta}_s) := S'_{n_s}(\beta_s(1)) \oplus \cdots \oplus S'_{n_s}(\beta_s(2^s)), \quad s = 0, \ldots, t-2,$$

with $S'_{n_s}(0) := \sqrt{2}T_{n_s}(0)$ and $S'_{n_s}(1) := T_{n_s}(1)$ such that

$$S'_n(\boldsymbol{\beta}_s) = (D_n(\boldsymbol{\beta}_s))^{-1} S_n(\boldsymbol{\beta}_s), \quad s = 0, \ldots, t-2.$$

As before, the matrices $A'_n(\boldsymbol{\beta}_s)$ and $S'_n(\boldsymbol{\beta}_s)$ are sparse matrices with at most 2 nonzero entries in each row. More precisely, after suitable permutations, $A'_n(\boldsymbol{\beta}_s)$, $s = 1, \ldots, t-2$, contain only butterfly matrices (with entries ± 1) and diagonal matrices (with entries 1 and $\pm\sqrt{2}$). The matrices $S'_n(\boldsymbol{\beta}_s)$, $s = 0, \ldots, t-2$, only contain butterfly matrices (with entries ± 1), rotation matrices, and rotation–reflection matrices. Note that $A'_n(\boldsymbol{\beta}_0) = A_n(0) = I_n$ and $S'_n(\boldsymbol{\beta}_0) = \sqrt{2}T_n(0)$.

With the changed matrices we find from (4.3) the factorization

$$\begin{aligned} C_n^{\mathrm{II}} = {} & \frac{1}{\left(\sqrt{2}\right)^{t-1}} P_n(0)A'_n(\boldsymbol{\beta}_0) \cdots P_n(t-2)A'_n(\boldsymbol{\beta}_{t-2}) \\ & \times T_n(\boldsymbol{\beta}_{t-1})S'_n(\boldsymbol{\beta}_{t-2}) \cdots S'_n(\boldsymbol{\beta}_0), \end{aligned}$$

since for each $s = 0, \ldots, t-2$ the product of diagonal matrices $D_n(\boldsymbol{\beta}_s)^{-1} \cdots D_n(\boldsymbol{\beta}_0)^{-1}$ commutes with $T_n(\boldsymbol{\beta}_{s+1})$ and with $A_n(\boldsymbol{\beta}_s)$. Observe that the inner matrix $T_n(\boldsymbol{\beta}_{t-1})$ is not changed. For the algorithm, we multiply this matrix with $\sqrt{2}$ and finally obtain

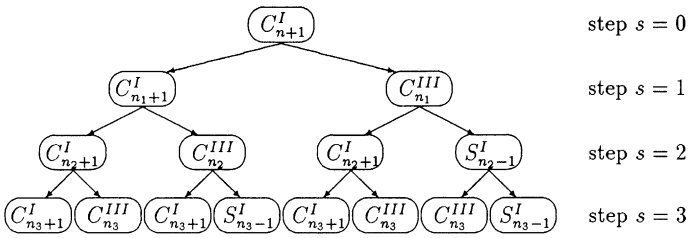
$$C_n^{\mathrm{II}} = \frac{1}{\sqrt{n}} P_n(0) A_n'(\boldsymbol{\beta}_0) \cdots P_n(t-2) A_n'(\boldsymbol{\beta}_{t-2}) S_n(\boldsymbol{\beta}_{t-1}) S_n'(\boldsymbol{\beta}_{t-2}) \cdots S_n'(\boldsymbol{\beta}_0). \tag{4.5}$$

This factorization leads to the MDCT-II Algorithm 3.4.

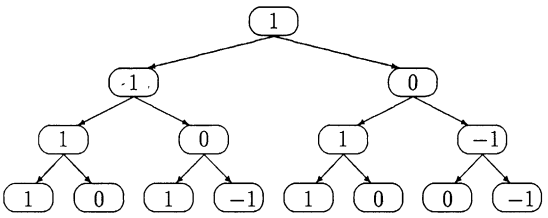
Our factorization of the cosine matrix of type I is based on the factorizations (2.5), (2.6), and (2.13). Let $n = 2^t (t \geqslant 2)$ be given. Further, let $n_s = 2^{t-s}$ $(s = 0, \ldots, t-1)$. In the first factorization step, C_{n+1}^{I} is split into $C_{n_1+1}^{\mathrm{I}} \oplus C_{n_1}^{\mathrm{III}}$ by (2.5). Then in the second step, we use (2.5) and (2.13) in order to split $C_{n_1+1}^{\mathrm{I}} \oplus C_{n_1}^{\mathrm{III}}$ into $C_{n_2+1}^{\mathrm{I}} \oplus C_{n_2}^{\mathrm{III}} \oplus C_{n_2+1}^{\mathrm{I}} \oplus S_{n_2-1}^{\mathrm{I}}$. In the case $t > 3$ we continue this procedure. For $S_{n_2-1}^{\mathrm{I}}$ we use the second factorization (2.6). Finally, we obtain a factorization of C_{n+1}^{I}. Note that (2.5) is in some sense also true for $n = 2$:

$$C_3^{\mathrm{I}} = P_3^{\mathrm{T}} \left(C_2^{\mathrm{II}} \oplus 1 \right) \tilde{T}_3(1).$$

The first factorization steps can be illustrated by the following diagram:



Note that the factorization tree of C_{n+1}^{I} contains factorization trees of $C_{n_1}^{\mathrm{III}}$ and $S_{n_2-1}^{\mathrm{I}}$. Now we have to indicate on which position $k \in \{1, \ldots, 2^k\}$ in step $s \in \{0, \ldots, t-1\}$ stands $C_{n_s+1}^{\mathrm{I}}$, $C_{n_s}^{\mathrm{III}}$, and $S_{n_s-1}^{\mathrm{I}}$, respectively. We introduce *triadic vectors* $\boldsymbol{\delta}_s = (\delta_s(1), \ldots, \delta_s(2^s))$ for $s \in \{0, \ldots, t-1\}$ as pointers, where $\delta_s(k) := 1$, if $C_{n_s+1}^{\mathrm{I}}$ stands at position k in step s, where $\delta_s(k) := 0$, if $C_{n_s}^{\mathrm{III}}$ stands at position k in step s, and where $\delta_s(k) := -1$, if $S_{n_s-1}^{\mathrm{I}}$ stands at position k in step s.



These pointers $\boldsymbol{\delta}_s$ have similar properties as $\boldsymbol{\beta}_s$ in Lemma 4.1.

Lemma 4.5. *Let $t \in \mathbb{N}$ with $t \geqslant 2$ be given and $\boldsymbol{\delta}_0 := (1)$. Then*

$$\boldsymbol{\delta}_{s+1} = (\boldsymbol{\delta}_s, \hat{\boldsymbol{\delta}}_s), \quad s = 0, \dots, t-2, \tag{4.6}$$

where $\hat{\boldsymbol{\delta}}_0 := (0)$ and for $s \geqslant 1$ the vector $\hat{\boldsymbol{\delta}}_s = (\hat{\delta}_s(1), \dots, \hat{\delta}_s(2^s))$ is defined by

$$\hat{\delta}_s(k) := \begin{cases} \delta_s(k) - 1 & \text{if } k = \frac{1}{3}2^{s+1} + \frac{1}{2} - \frac{1}{6}(-1)^s, \\ \delta_s(k) & \text{otherwise.} \end{cases}$$

For $s \geqslant 1$, the vectors $\boldsymbol{\delta}_s$ consists of $\frac{1}{3}(2^s - (-1)^s)$ zeros, $\frac{1}{3}2^s + \frac{1}{2} + \frac{1}{6}(-1)^s$ ones, and $\frac{1}{3}2^s - \frac{1}{2} + \frac{1}{6}(-1)^s$ minus ones.

The proof is omitted here for shortness. For each pointer $\boldsymbol{\delta}_s$ we define the following matrices:

$$\begin{aligned} P_{n+1}(\boldsymbol{\delta}_s) &:= P^{\mathrm{T}}_{n_s+\delta_s(1)} \oplus \cdots \oplus P^{\mathrm{T}}_{n_s+\delta_s(2^s)}, \\ \tilde{A}_{n+1}(\boldsymbol{\delta}_s) &:= \tilde{A}_{n_s+\delta_s(1)}(\delta_s(1)) \oplus \cdots \oplus \tilde{A}_{n_s+\delta_s(2^s)}(\delta_s(2^s)), \\ \tilde{T}_{n+1}(\boldsymbol{\delta}_s) &:= \tilde{T}_{n_s+\delta_s(1)}(\delta_s(1)) \oplus \cdots \oplus \tilde{T}_{n_s+\delta_s(2^s)}(\delta_s(2^s)), \end{aligned}$$

where $\tilde{A}_{n_s+1}(1) := I_{n_s+1}$, $\tilde{A}_{n_s-1}(-1)$, and $\tilde{T}_{n_s\pm1}(\pm1)$ are introduced in Lemma 2.2, and where $\tilde{A}_{n_s}(0)$, and $\tilde{T}_{n_s}(0)$ are defined in Lemma 2.5. The matrix $\tilde{T}_{n+1}(\boldsymbol{\delta}_{t-1})$ is a block matrix with blocks $\tilde{T}_3(1) := C^{\mathrm{I}}_3$, $\tilde{T}_2(0) := C^{\mathrm{III}}_2$ and $\tilde{T}_1(-1) := S^{\mathrm{I}}_1 = (1)$. By construction, all matrices $\tilde{A}_{n+1}(\boldsymbol{\delta}_s)$ and $\tilde{T}_{n+1}(\boldsymbol{\delta}_s)$ are sparse and orthogonal.

Theorem 4.6. *Let $n = 2^t$ $(t \geqslant 2)$. Then the matrix C^{I}_{n+1} can be factorized into the following product of sparse orthogonal matrices*:

$$\begin{aligned} C^{\mathrm{I}}_{n+1} = {} & (P_{n+1}(\boldsymbol{\delta}_0)\tilde{A}_{n+1}(\boldsymbol{\delta}_0)) \dots (P_{n+1}(\boldsymbol{\delta}_{t-2})\tilde{A}_{n+1}(\boldsymbol{\delta}_{t-2})) \\ & \times \tilde{T}_{n+1}(\boldsymbol{\delta}_{t-1}) \dots \tilde{T}_{n+1}(\boldsymbol{\delta}_0). \end{aligned} \tag{4.7}$$

The proof directly follows from Lemmas 2.2 and 2.5. The factorization of C^{I}_{n+1} in Theorem 4.6 implies a fast DCT-I algorithm which uses only permutations, scaled butterfly operations, and plane rotations/rotation–reflections and works without additional scaling. Factorizations for C^{III}_n and S^{I}_n can now be derived analogously.

5. Numerical stability of fast DCT algorithms

In the following we use Wilkinson's standard method for the binary floating point arithmetic for real numbers (see [10, p. 44]). If $x \in \mathbb{R}$ is represented by the floating point number $\hat{x} = \mathrm{fl}(x)$, then

$$\mathrm{fl}(x) = x(1+\delta) \quad (|\delta| \leqslant u),$$

where u denotes the *unit roundoff* or *machine precision* as long as we disregard underflow and overflow. For arbitrary floating point numbers x_0, x_1 and any arithmetical operation $\circ \in \{+, -, \times, /\}$, the exact value $y = x_0 \circ x_1$ and the computed value $\hat{y} = \mathrm{fl}(x_0 \circ x_1)$ are related by

$$\mathrm{fl}(x_0 \circ x_1) = (x_0 \circ x_1)(1 + \delta^\circ) \quad (|\delta^\circ| \leqslant u). \tag{5.1}$$

In the IEEE arithmetic of single precision (24 bits for the mantissa including 1 sign bit, 8 bits for the exponent), we have $u = 2^{-24} \approx 5.96 \times 10^{-8}$. For arithmetic double precision (53 bits for the mantissa including 1 sign bit, 11 bits for the exponent), we have $u = 2^{-53} \approx 1.11 \times 10^{-16}$ (see [10, p. 45]).

Usually the total roundoff error in the result of an algorithm is composed of a number of such errors. To make the origin of relative errors δ_k° clear in this notation, we use superscripts for the operation $\circ$ and subscripts for the operation step k.

In this section we show that, under weak assumptions, our fast DCT algorithms possess a remarkable good numerical stability.

Before we can start to analyze the numerical stability of DCT algorithms of length n, we need to consider the roundoff errors caused by multiplication of matrices of length 2, since all matrix factors in the factorizations of cosine matrices in Section 4 can be transformed into block-diagonal matrices with blocks of order $\leqslant 2$ by suitable permutations.

5.1. Two auxiliary lemmas

In this subsection we analyze the roundoff errors of simple matrix–vector products, where the matrix of order 2 is a (scaled) butterfly matrix or scaled rotation matrix. Before starting the detailed analysis, we show the following useful estimate.

Lemma 5.1. *For all $a, b, c, d \in \mathbb{R}$, we have*

$$\begin{aligned}&(|ac| + |bd| + |ac - bd|)^2 + (|ad| + |bc| + |ad + bc|)^2 \\ &\quad \leqslant \tfrac{16}{3}(a^2 + b^2)(c^2 + d^2),\end{aligned}$$

where the constant $16/3$ *is best possible*.

Proof. Without loss of generality, we can assume that $a, b, c, d \geqslant 0$. Further we can suppose that $ac \geqslant bd$, since otherwise we change the notation and replace (a, b, c, d) by (b, a, d, c). Then the above inequality reads as follows:

$$(ac)^2 + (ad + bc)^2 \leqslant \tfrac{4}{3}(a^2 + b^2)(c^2 + d^2).$$

This inequality is equivalent to

$$0 \leqslant (ad - bc)^2 + (ac - 2bd)^2$$

which is obvious. For $a = c = \sqrt{2}$ and $b = d = 1$ we have equality. □

In the following, the order term $\mathcal{O}(u^k)$ $(k = 1, 2, \ldots)$ has the usual meaning of a quantity bounded by a constant times u^k, where the constant does not depend on u.

Lemma 5.2. (i) *For the butterfly operation* $y_0 := x_0 + x_1$, $y_1 := x_0 - x_1$ *with* $\hat{y}_0 := \mathrm{fl}(x_0 + x_1)$ *and* $\hat{y}_1 := \mathrm{fl}(x_0 - x_1)$, *the roundoff error can be estimated by*

$$(\hat{y}_0 - y_0)^2 + (\hat{y}_1 - y_1)^2 \leqslant 2u^2\big(x_0^2 + x_1^2\big). \tag{5.2}$$

(ii) *If the scaling factor* $a \notin \{0, \pm 1\}$ *is precomputed by* $\hat{a} = a + \Delta a$ *with* $|\Delta a| \leqslant c_1 u$, *then for the scaled butterfly operation* $y_0 := a(x_0 + x_1)$, $y_1 := a(x_0 - x_1)$ *with* $\hat{y}_0 := \mathrm{fl}(\hat{a}(x_0 + x_1))$ *and* $\hat{y}_1 := \mathrm{fl}(\hat{a}(x_0 - x_1))$, *the roundoff error can be estimated by*

$$(\hat{y}_0 - y_0)^2 + (\hat{y}_1 - y_1)^2 \leqslant \big(2\sqrt{2}|a| + \sqrt{2}c_1 + \mathcal{O}(u)\big)^2 u^2\big(x_0^2 + x_1^2\big). \tag{5.3}$$

(iii) *If the different entries* $a_k \notin \{0, \pm 1\}$ *with* $b^2 := a_0^2 + a_1^2 > 0$ *are precomputed by* $\hat{a}_k = a_k + \Delta a_k$ *with* $|\Delta a_k| \leqslant c_2 u$ *for* $k = 0, 1$, *then for the scaled rotation*

$$y_0 := a_0 x_0 + a_1 x_1, \quad y_1 := -a_1 x_0 + a_0 x_1$$

with $\hat{y}_0 := \mathrm{fl}(\hat{a}_0 x_0 + \hat{a}_1 x_1)$, $\hat{y}_1 := \mathrm{fl}(-\hat{a}_1 x_0 + \hat{a}_0 x_1)$, *the roundoff error can be estimated by*

$$(\hat{y}_0 - y_0)^2 + (\hat{y}_1 - y_1)^2 \leqslant \big(\tfrac{4}{3}\sqrt{3}|b| + \sqrt{2}c_2 + \mathcal{O}(u)\big)^2 u^2\big(x_0^2 + x_1^2\big). \tag{5.4}$$

Proof. (i) By (5.1) we have

$$\begin{aligned}
\hat{y}_0 &= (x_0 + x_1)\big(1 + \delta_0^+\big) = y_0 + (x_0 + x_1)\delta_0^+, \\
\hat{y}_1 &= (x_0 - x_1)\big(1 + \delta_1^+\big) = y_1 + (x_0 - x_1)\delta_1^+
\end{aligned}$$

with $|\delta_k^+| \leqslant u$ for $k = 0, 1$ such that by

$$|\hat{y}_0 - y_0| \leqslant |x_0 + x_1|u, \quad |\hat{y}_1 - y_1| \leqslant |x_0 - x_1|u,$$

we obtain (5.2).

(ii) Putting $z_0 := \hat{a}(x_0 + x_1)$, $z_1 := \hat{a}(x_0 - x_1)$, it follows from (5.1) that

$$\begin{aligned}
\hat{y}_0 &= \hat{a}(x_0 + x_1)\big(1 + \delta_0^+\big)\big(1 + \delta_0^\times\big) = z_0 + \hat{a}(x_0 + x_1)\big(\delta_0^+ + \delta_0^\times + \delta_0^+\delta_0^\times\big), \\
\hat{y}_1 &= \hat{a}(x_0 - x_1)\big(1 + \delta_1^+\big)\big(1 + \delta_1^\times\big) = z_1 + \hat{a}(x_0 - x_1)\big(\delta_1^+ + \delta_1^\times + \delta_1^+\delta_1^\times\big)
\end{aligned}$$

with $|\delta_k^+| \leqslant u$, $|\delta_k^\times| \leqslant u$, $k = 0, 1$. Thus, by

$$|\hat{y}_0 - z_0| \leqslant |\hat{a}(x_0 + x_1)|(2u + u^2), \quad |\hat{y}_1 - z_1| \leqslant |\hat{a}(x_0 - x_1)|(2u + u^2),$$

we get the estimate

$$|\hat{y}_0 - z_0|^2 + |\hat{y}_1 - z_1|^2 \leqslant 2\hat{a}^2(x_0^2 + x_1^2)u^2(2 + u)^2$$

with $\hat{a}^2 = a^2 + \mathcal{O}(u)$ which yields

$$\left\| \begin{pmatrix} \hat{y}_0 - z_0 \\ \hat{y}_1 - z_1 \end{pmatrix} \right\|_2 \leqslant \big(2\sqrt{2}|a| + \mathcal{O}(u)\big)u \left\| \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \right\|_2.$$

By

$$\begin{pmatrix} z_0 - y_0 \\ z_1 - y_1 \end{pmatrix} = \Delta a \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix},$$

we obtain

$$\left\| \begin{pmatrix} z_0 - y_0 \\ z_1 - y_1 \end{pmatrix} \right\|_2 \leqslant \sqrt{2} c_1 u \left\| \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \right\|_2$$

and finally by triangle inequality

$$\left\| \begin{pmatrix} \hat{y}_0 - y_0 \\ \hat{y}_1 - y_1 \end{pmatrix} \right\|_2 \leqslant \big(2\sqrt{2}|a| + \sqrt{2} c_1 + \mathcal{O}(u)\big) u \left\| \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \right\|_2.$$

(iii) Introducing $z_0 := \hat{a}_0 x_0 + \hat{a}_1 x_1$, $z_1 := -\hat{a}_1 x_0 + \hat{a}_0 x_1$, it follows from (5.1) that

$$\begin{aligned} \hat{y}_0 &= \big[\hat{a}_0 x_0 (1 + \delta_0^{\times}) + \hat{a}_1 x_1 (1 + \delta_1^{\times})\big]\big(1 + \delta_0^{+}\big), \\ \hat{y}_1 &= \big[-\hat{a}_1 x_0 (1 + \delta_2^{\times}) + \hat{a}_0 x_1 (1 + \delta_3^{\times})\big]\big(1 + \delta_1^{+}\big) \end{aligned}$$

with $|\delta_j^{\times}| \leqslant u$ for $j = 0, \ldots, 3$ and $|\delta_k^{+}| \leqslant u$ for $k = 0, 1$. Hence we obtain

$$\begin{aligned} |\hat{y}_0 - z_0| &\leqslant (|\hat{a}_0 x_0| + |\hat{a}_1 x_1| + |\hat{a}_0 x_0 + \hat{a}_1 x_1|) u + (|\hat{a}_0 x_0| + |\hat{a}_1 x_1|) u^2, \\ |\hat{y}_1 - z_1| &\leqslant (|\hat{a}_1 x_0| + |\hat{a}_0 x_1| + |\hat{a}_1 x_0 - \hat{a}_0 x_1|) u + (|\hat{a}_1 x_0| + |\hat{a}_0 x_1|) u^2 \end{aligned}$$

and thus

$$\begin{aligned} |\hat{y}_0 - z_0|^2 + |\hat{y}_1 - z_1|^2 \leqslant \big[&(|\hat{a}_0 x_0| + |\hat{a}_1 x_1| + |\hat{a}_0 x_0 + \hat{a}_1 x_1|)^2 \\ &+ (|\hat{a}_1 x_0| + |\hat{a}_0 x_1| + |\hat{a}_1 x_0 - \hat{a}_0 x_1|)^2\big] u^2 (1 + u)^2. \end{aligned}$$

Applying Lemma 5.1, we find

$$\begin{aligned} |\hat{y}_0 - z_0|^2 + |\hat{y}_1 - z_1|^2 &\leqslant \tfrac{16}{3}\big(\hat{a}_0^2 + \hat{a}_1^2\big)\big(x_0^2 + x_1^2\big) u^2 (1 + u)^2 \\ &= \tfrac{16}{3}(b^2 + \mathcal{O}(u)) u^2 \big(x_0^2 + x_1^2\big), \end{aligned}$$

since $\hat{a}_0^2 + \hat{a}_1^2 = a_0^2 + a_1^2 + \mathcal{O}(u) = b^2 + \mathcal{O}(u)$ by assumption. Therefore we obtain

$$\left\| \begin{pmatrix} \hat{y}_0 - z_0 \\ \hat{y}_1 - z_1 \end{pmatrix} \right\|_2 \leqslant \big(\tfrac{4}{3}\sqrt{3}|b| + \mathcal{O}(u)\big) u \left\| \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \right\|_2.$$

By

$$\begin{pmatrix} z_0 - y_0 \\ z_1 - y_1 \end{pmatrix} = \begin{pmatrix} \Delta a_0 & \Delta a_1 \\ \Delta a_1 & -\Delta a_0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}, \tag{5.5}$$

we conclude that

$$\left\| \begin{pmatrix} z_0 - y_0 \\ z_1 - y_1 \end{pmatrix} \right\|_2 \leqslant \sqrt{2} c_2 u \left\| \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \right\|_2,$$

since the matrix in (5.5) is orthogonal up to a factor and therefore its spectral norm equals

$$\sqrt{(\Delta a_0)^2 + (\Delta a_1)^2} \leqslant \sqrt{2} c_2 u.$$

Using the triangle inequality, we obtain (5.4). □

5.2. Numerical stability of the recursive DCT-II algorithm

Now we consider the fast DCT-II Algorithm 3.1 (involving Algorithm 3.2) which is equivalent to the factorization formula (4.3) up to some scaling factors,

$$\begin{aligned}\sqrt{n} C_n^{\mathrm{II}} &= (P_n(0) A_n(\boldsymbol{\beta}_0)) \cdots (P_n(t-2) A_n(\boldsymbol{\beta}_{t-2})) \\ &\quad \times \big(\sqrt{2} T_n(\boldsymbol{\beta}_{t-1})\big) \cdots \big(\sqrt{2} T_n(\boldsymbol{\beta}_0)\big). \end{aligned} \tag{5.6}$$

Our considerations are now based on the DCT-II algorithm in its iterative form:

For $s = 0$ to $t-1$ compute

$$\mathbf{x}^{(s+1)} := \sqrt{2} T_n(\boldsymbol{\beta}_j) \mathbf{x}^{(s)} \quad (\mathbf{x}^{(0)} := \mathbf{x})$$

and for $s = 0$ to $t-2$ compute

$$\mathbf{x}^{(t+s+1)} := P_n(t-s-2) A_n(\boldsymbol{\beta}_{t-s-2}) \mathbf{x}^{(t+s)}.$$

Then $\mathbf{z} = C_n^{\mathrm{II}} \mathbf{x} = \frac{1}{\sqrt{n}} \mathbf{x}^{(2t-1)}$ is the resulting vector.

The roundoff errors of this algorithm are caused by multiplications with the matrices $S_n(\boldsymbol{\beta}_s) := \sqrt{2} T_n(\boldsymbol{\beta}_s)$, $s = 0, \ldots, t-1$, and $A_n(\boldsymbol{\beta}_s)$, $s = 0, \ldots, t-2$. These matrices have a very simple structure. After suitable permutations, every matrix is block-diagonal with blocks of order $\leqslant 2$. All blocks of order 1 are equal to ± 1. Every block of order 2 is either a (scaled) butterfly matrix

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix},$$

or a scaled rotation matrix/rotation–reflection matrix

$$\begin{pmatrix} a_0 & a_1 \\ -a_1 & a_0 \end{pmatrix}, \quad \begin{pmatrix} a_0 & a_1 \\ a_1 & -a_0 \end{pmatrix}$$

with

$$\begin{aligned} a_0 &= \sqrt{2} \cos \tfrac{(2k+1)\pi}{2^{s+3}}, \quad a_1 = \sqrt{2} \sin \tfrac{(2k+1)\pi}{2^{s+3}}, \\ & s = 0, \ldots, t-2; k = 0, \ldots, 2^s - 1. \end{aligned}$$

For an arbitrary input vector $\mathbf{x} \in \mathbb{R}^n$, let $\mathbf{y} := \sqrt{n} C_n^{\mathrm{II}} \mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{z} := \frac{1}{\sqrt{n}} \mathbf{y}$ denotes the exact transformed vector. Further let $\hat{\mathbf{y}} \in \mathbb{R}^n$ be the output vector computed by our DCT-II Algorithm 3.1 (involving Algorithm 3.2) using floating point

arithmetic with unit roundoff u. Finally, let $\hat{\mathbf{z}} := \mathrm{fl}\big(\frac{1}{\sqrt{n}}\hat{\mathbf{y}}\big)$. Since C_n^{II} is non-singular, $\hat{\mathbf{z}}$ can be represented in the form $\hat{\mathbf{z}} = C_n^{\mathrm{II}}(\mathbf{x} + \Delta\mathbf{x})$ with $\Delta\mathbf{x} \in \mathbb{R}^n$. An algorithm for computing $C_n^{\mathrm{II}}\mathbf{x}$ is called *normwise backward stable* (see [10, p. 142]), if there is a positive constant k_n such that

$$\|\Delta\mathbf{x}\|_2 \leqslant \big(k_n u + \mathcal{O}(u^2)\big)\|\mathbf{x}\|_2 \tag{5.7}$$

for all vectors $\mathbf{x} \in \mathbb{R}^n$ and $k_n u \ll 1$. The constant k_n measures the numerical stability. Since C_n^{II} is orthogonal, we conclude that $\|\Delta\mathbf{x}\|_2 = \|C_n^{\mathrm{II}}(\Delta\mathbf{x})\|_2 = \|\hat{\mathbf{z}} - \mathbf{z}\|_2$ and $\|\mathbf{x}\|_2 = \|C_n^{\mathrm{II}}\mathbf{x}\|_2 = \|\mathbf{z}\|_2$. Hence we also have *normwise forward stability* by

$$\|\hat{\mathbf{z}} - \mathbf{z}\|_2 \leqslant \big(k_n u + \mathcal{O}(u^2)\big)\|\mathbf{z}\|_2,$$

if (5.7) is satisfied.

Now let us look closer at the computation steps in our iterative DCT-II algorithm which is equivalent to the factorization (5.6). First, for every $s = 0, \ldots, t-2$, all values

$$\sqrt{2}\cos\tfrac{(2k+1)\pi}{2^{s+3}}, \quad \sqrt{2}\sin\tfrac{(2k+1)\pi}{2^{s+3}}, \quad k = 0, \ldots, 2^s - 1, \tag{5.8}$$

needed in the matrices $\sqrt{2}T_n(\boldsymbol{\beta}_s)$ are precomputed. If cosine and sine are internally computed to higher precision and the results afterwards are rounded towards the next machine number, then we obtain very accurate values of (5.8) with an error constant $c_2 = 1$ (see Lemma 5.2(iii)). We use the matrices $\hat{S}_n(\boldsymbol{\beta}_s)$, $s = 1, \ldots, t-1$, with precomputed entries (5.8) instead of $S_n(\boldsymbol{\beta}_s) = \sqrt{2}T_n(\boldsymbol{\beta}_s)$. Assume that the value $\sqrt{2}/2$ is precomputed with the error constant $c_1 = 1/2$ (see Lemma 5.2(ii)). We use the matrices $\hat{A}_n(\boldsymbol{\beta}_s)$, $s = 1, \ldots, t-2$, with the precomputed scaling factors $\sqrt{2}/2$ instead of $A_n(\boldsymbol{\beta}_s)$. The vectors $\boldsymbol{\beta}_s$ are generated without roundoff errors.

Let $\hat{\mathbf{x}}^{(0)} = \mathbf{x}^{(0)} := \mathbf{x}$. We denote the vectors computed in the iterative DCT-II algorithm by

$$\hat{\mathbf{x}}^{(s+1)} := \mathrm{fl}(\hat{S}_n(\boldsymbol{\beta}_s)\hat{\mathbf{x}}^{(s)}), \quad s = 0, \ldots, t-1,$$

and the corresponding exact vectors by

$$\mathbf{x}^{(s+1)} := S_n(\boldsymbol{\beta}_s)\mathbf{x}^{(s)}, \quad s = 0, \ldots, t-1.$$

Further, we introduce the error vectors $\mathbf{e}^{(s+1)} \in \mathbb{R}^n$ by

$$\hat{\mathbf{x}}^{(s+1)} = S_n(\boldsymbol{\beta}_s)\hat{\mathbf{x}}^{(s)} + \mathbf{e}^{(s+1)}. \tag{5.9}$$

Note that $\mathbf{e}^{(s+1)}$ describes the precomputation error and roundoff error of one step in our DCT-II algorithm. The matrix–vector product $\hat{S}_n(\boldsymbol{\beta}_0)\hat{\mathbf{x}}^{(0)} = S_n(\boldsymbol{\beta}_0)\mathbf{x}$ involves only butterfly operations such that by Lemma 5.2(i)

$$\|\mathbf{e}^{(1)}\|_2 \leqslant \sqrt{2}u\|\mathbf{x}\|_2. \tag{5.10}$$

Every matrix–vector product $\hat{S}_n(\boldsymbol{\beta}_s)\hat{\mathbf{x}}^{(s)}$, $s = 1, \ldots, t-1$, consists of butterfly operations and rotations/rotation–reflections scaled by $\sqrt{2}$ such that by Lemma 5.2(i) and (iii), we obtain

$$\left\|\mathbf{e}^{(s+1)}\right\|_2 \leqslant \left(\tfrac{4}{3}\sqrt{6}+\sqrt{2}+\mathcal{O}(u)\right)u\|\hat{\mathbf{x}}^{(s)}\|_2, \quad s=1,\dots,t-1. \tag{5.11}$$

Now we introduce the vectors computed in our DCT-II algorithm

$$\hat{\mathbf{x}}^{(t+s+1)} := \mathrm{fl}\left(P_n(t-s-2)\hat{A}_n(\boldsymbol{\beta}_{t-s-2})\hat{\mathbf{x}}^{(t+s)}\right), \quad s=0,\dots,t-2,$$

the corresponding exact vectors

$$\mathbf{x}^{(t+s+1)} := P_n(t-s-2)A_n(\boldsymbol{\beta}_{t-s-2})\mathbf{x}^{(t+s)}, \quad s=0,\dots,t-2,$$

and the corresponding error vectors $\mathbf{e}^{(t+s+1)} \in \mathbb{R}^n$ by

$$\hat{\mathbf{x}}^{(t+s+1)} := P_n(t-s-2)A_n(\boldsymbol{\beta}_{t-s-2})\hat{\mathbf{x}}^{(t+s)} + \mathbf{e}^{(t+s+1)}. \tag{5.12}$$

Every matrix–vector product $\hat{A}_n(\boldsymbol{\beta}_{t-s-2})\hat{\mathbf{x}}^{(t+s)}$, $s=0,\dots,t-3$, consists of identities, minus identities, and scaled butterfly operations (with precomputed scaling factor $\sqrt{2}/2$) such that by Lemma 5.2(ii) we can estimate

$$\left\|\mathbf{e}^{(t+s+1)}\right\|_2 \leqslant \left(2+\tfrac{1}{2}\sqrt{2}+\mathcal{O}(u)\right)u\|\hat{\mathbf{x}}^{(t+s)}\|_2, \quad s=0,\dots,t-3. \tag{5.13}$$

Note that by $\hat{A}_n(\boldsymbol{\beta}_0) = I_n$ we have $\mathbf{e}^{(2t-1)} = \mathbf{0}$.

Finally, we scale the result of our DCT-II algorithm by $\mathbf{z} := 2^{-t/2}\mathbf{x}^{(2t-1)}$. Let $\hat{\mathbf{z}} := \mathrm{fl}(2^{-t/2}\hat{\mathbf{x}}^{(2t-1)})$. For even t, this scaling by a power of 2 does not produce an additional roundoff error such that

$$\|\hat{\mathbf{z}}-\mathbf{z}\|_2 = 2^{-t/2}\left\|\hat{\mathbf{x}}^{(2t-1)}-\mathbf{x}^{(2t-1)}\right\|_2.$$

For odd t, we precompute $\mathrm{fl}(2^{-t/2}) = 2^{-(t+1)/2}\mathrm{fl}(\sqrt{2})$ with $|\mathrm{fl}(\sqrt{2})-\sqrt{2}| \leqslant u$. Then by (5.1) we obtain that for $j=0,\dots,n-1$,

$$\hat{z}_j = 2^{-(t+1)/2}\mathrm{fl}\left(\sqrt{2}\right)\hat{x}_j^{(2t-1)}\left(1+\delta_j^{\times}\right)$$

with $|\delta_j^{\times}| \leqslant u$. But this implies that

$$\left\|\hat{\mathbf{z}}-2^{-t/2}\hat{\mathbf{x}}^{(2t-1)}\right\|_2 \leqslant 2^{-t/2}\left\|\hat{\mathbf{x}}^{(2t-1)}\right\|_2 u\left(1+\tfrac{1}{2}\sqrt{2}+\mathcal{O}(u)\right).$$

Finally, by triangle inequality it follows that

$$\|\hat{\mathbf{z}}-\mathbf{z}\|_2 \leqslant \tfrac{1}{\sqrt{n}}\left\|\hat{\mathbf{x}}^{(2t-1)}\right\|_2 u\left(1+\tfrac{1}{2}\sqrt{2}+\mathcal{O}(u)\right)+\tfrac{1}{\sqrt{n}}\left\|\hat{\mathbf{x}}^{(2t-1)}-\mathbf{x}^{(2t-1)}\right\|_2 \tag{5.14}$$

which is also true for even t.

We are now ready to estimate the total roundoff error $\|\hat{\mathbf{z}}-\mathbf{z}\|_2$ of our fast DCT-II algorithm under the assumption that $\sqrt{2}$ and the trigonometric values (5.8) in the factor matrices are precomputed with error bound u.

Theorem 5.3. *Let $n = 2^t$ ($t \geqslant 3$). Assume that $\sqrt{2}$ and the values (5.8) are precomputed with absolute error bound u. Then the fast* DCT-II *Algorithm 3.1 (involving Algorithm 3.2) and including the final scaling with $1/\sqrt{n}$ is normwise backward stable with the constant*

$$k_n = \left(\tfrac{4}{3}\sqrt{3} + \tfrac{1}{2}\sqrt{2} + 3\right)(\log_2 n - 1) \approx 6.016508(\log_2 n - 1).$$

Proof. First we estimate the roundoff error $\|\hat{\mathbf{x}}^{(2t-1)} - \mathbf{x}^{(2t-1)}\|_2$. Applying (5.9) and (5.12) repeatedly, we obtain

$$\begin{aligned}\hat{\mathbf{x}}^{(2t-1)} = \mathbf{x}^{(2t-1)} &+ P_n(0)A_n(\boldsymbol{\beta}_0)\cdots P_n(t-2)A_n(\boldsymbol{\beta}_{t-2})S_n(\boldsymbol{\beta}_{t-1})\cdots S_n(\boldsymbol{\beta}_1)\mathbf{e}^{(1)}\\ &+ \cdots + P_n(0)A_n(\boldsymbol{\beta}_0)\cdots P_n(t-2)A_n(\boldsymbol{\beta}_{t-2})S_n(\boldsymbol{\beta}_{t-1})\mathbf{e}^{(t-1)}\\ &+ P_n(0)A_n(\boldsymbol{\beta}_0)\cdots P_n(t-2)A_n(\boldsymbol{\beta}_{t-2})\mathbf{e}^{(t)}\\ &+ \cdots + P_n(0)A_n(\boldsymbol{\beta}_0)\mathbf{e}^{(2t-2)}. \end{aligned} \tag{5.15}$$

The matrices $S_n(\boldsymbol{\beta}_s)$, $s = 0, \ldots, t-1$, are orthogonal up to a factor and have the spectral norm $\|S_n(\boldsymbol{\beta}_s)\|_2 = \sqrt{2}$. The matrices $P_n(s)A_n(\boldsymbol{\beta}_s)$, $s = 0, \ldots, t-2$, are orthogonal such that $\|P_n(s)A_n(\boldsymbol{\beta}_s)\|_2 = 1$. By (5.9) and (5.12) we can estimate

$$\begin{aligned}\|\hat{\mathbf{x}}^{(s+1)}\|_2 &\leqslant \sqrt{2}\|\hat{\mathbf{x}}^{(s)}\|_2 + \|\mathbf{e}^{(s+1)}\|_2, \quad s = 0, \ldots, t-1,\\ \|\hat{\mathbf{x}}^{(t+s+1)}\|_2 &\leqslant \|\hat{\mathbf{x}}^{(t+s)}\|_2 + \|\mathbf{e}^{(t+s+1)}\|_2, \quad s = 0, \ldots, t-2.\end{aligned}$$

Thus by (5.11) and (5.13) we see that

$$\begin{aligned}\|\hat{\mathbf{x}}^{(s+1)}\|_2 &\leqslant \left(\sqrt{2} + \mathcal{O}(u)\right)\|\hat{\mathbf{x}}^{(s)}\|_2, \quad s = 0, \ldots, t-1,\\ \|\hat{\mathbf{x}}^{(t+s+1)}\|_2 &\leqslant (1 + \mathcal{O}(u))\|\hat{\mathbf{x}}^{(t+s)}\|_2, \quad s = 0, \ldots, t-2.\end{aligned}$$

Since $\hat{\mathbf{x}}^{(0)} = \mathbf{x}$ this implies

$$\|\hat{\mathbf{x}}^{(s+1)}\|_2 \leqslant \left(2^{(s+1)/2} + \mathcal{O}(u)\right)\|\mathbf{x}\|_2, \quad s = 0, \ldots, t-1, \tag{5.16}$$

$$\|\hat{\mathbf{x}}^{(t+s+1)}\|_2 \leqslant \left(2^{t/2} + \mathcal{O}(u)\right)\|\mathbf{x}\|_2, \quad s = 0, \ldots, t-2. \tag{5.17}$$

From (5.11), (5.13), (5.16), and (5.17) it follows that

$$\begin{aligned}\|\mathbf{e}^{(s+1)}\|_2 &\leqslant 2^{(s+1)/2}\left(\tfrac{4}{3}\sqrt{3} + 1 + \mathcal{O}(u)\right)u\|\mathbf{x}\|_2, \quad s = 0, \ldots, t-1,\\ \|\mathbf{e}^{(t+s+1)}\|_2 &\leqslant 2^{t/2}\left(2 + \tfrac{1}{2}\sqrt{2} + \mathcal{O}(u)\right)u\|\mathbf{x}\|_2, \quad s = 0, \ldots, t-3.\end{aligned}$$

We obtain from (5.15) that

$$\begin{aligned}\|\hat{\mathbf{x}}^{(2t-1)} - \mathbf{x}^{(2t-1)}\|_2 &\leqslant \|S_n(\boldsymbol{\beta}_{t-1})\cdots S_n(\boldsymbol{\beta}_1)\|_2\|\mathbf{e}^{(1)}\|_2 + \cdots\\ &\quad + \|S_n(\boldsymbol{\beta}_{t-1})\|_2\|\mathbf{e}^{(t-1)}\|_2 + \|\mathbf{e}^{(t)}\|_2 + \cdots + \|\mathbf{e}^{(2t-2)}\|_2\\ &\leqslant \left(\sqrt{2}\right)^{t-1}\|\mathbf{e}^{(1)}\|_2 + \cdots + \sqrt{2}\|\mathbf{e}^{(t-1)}\|_2\\ &\quad + \|\mathbf{e}^{(t)}\|_2 + \cdots + \|\mathbf{e}^{(2t-2)}\|_2\end{aligned}$$

and hence by (5.10) and the above estimates for $\|\mathbf{e}^{(s+1)}\|_2$ and $\|\mathbf{e}^{(t+s+1)}\|_2$ that

$$\begin{aligned}&\left\|\hat{\mathbf{x}}^{(2t-1)} - \mathbf{x}^{(2t-1)}\right\|_2 \\ &\quad\leqslant 2^{t/2}u\left(\left(\tfrac{4}{3}\sqrt{3}+\tfrac{1}{2}\sqrt{2}+3\right)(t-1) - 1 - \tfrac{1}{2}\sqrt{2} + \mathcal{O}(u)\right)\|\mathbf{x}\|_2. \qquad (5.18)\end{aligned}$$

For the final scaling $\mathbf{z} = 2^{-t/2}\mathbf{x}^{(2t-1)}$, let $\hat{\mathbf{z}} = \mathrm{fl}(2^{-t/2}\hat{\mathbf{x}}^{(2t-1)})$. By (5.14), (5.17) and (5.18) we get the final estimate

$$\|\hat{\mathbf{z}} - \mathbf{z}\|_2 \leqslant u\left(\left(\tfrac{4}{3}\sqrt{3}+\tfrac{1}{2}\sqrt{2}+3\right)(t-1) + \mathcal{O}(u)\right)\|\mathbf{x}\|_2.$$

This completes the proof. □

5.3. Numerical stability of the other recursive DCT algorithms

The numerical stability of the further recursive DCT algorithms of Section 3 can now be shown analogously, as it is done in the last subsection for the DCT-II algorithm.

Theorem 5.4. *Let $n = 2^t$ $(t \geqslant 3)$. Assume that the values*

$$\sqrt{2},\ \sqrt{2}\cos\tfrac{(2k+1)\pi}{2^{s+3}},\ \sqrt{2}\sin\tfrac{(2k+1)\pi}{2^{s+3}}$$

with $k = 0, \ldots, 2^s - 1$ and $s = 0, \ldots, t-1$ are precomputed with the absolute error bound u. Then the fast DCT-IV *Algorithm 3.2 (involving Algorithm 3.1) and including the final scaling with $1/\sqrt{n}$ is normwise backward stable with the constant*

$$k_n = \left(\tfrac{4}{3}\sqrt{3}+\tfrac{1}{2}\sqrt{2}+3\right)(\log_2 n - 1) \approx 6.016508(\log_2 n - 1).$$

Proof. We apply Algorithm 3.2 and the proof of Theorem 5.3. For arbitrary $\mathbf{x} \in \mathbb{R}^n$, let $\mathbf{y} = \sqrt{n}C_n^{\mathrm{IV}}\mathbf{x}$, where $\sqrt{n}C_n^{\mathrm{IV}}$ can be factorized by (2.8) in the form

$$\sqrt{n}C_n^{\mathrm{IV}} = P_n^{\mathrm{T}} A_n(1)\left(\sqrt{n_1}C_{n_1}^{\mathrm{II}} \oplus \sqrt{n_1}C_{n_1}^{\mathrm{II}}\right) S_n(1)$$

with $S_n(1) = \sqrt{2}T_n(1)$. In step 1 of Algorithm 3.2 we compute $\hat{\mathbf{u}} = \mathrm{fl}(\hat{S}_n(1)\mathbf{x})$. The corresponding error vector $\mathbf{e}^{(1)}$ is defined by

$$\hat{\mathbf{u}} = S_n(1)\mathbf{x} + \mathbf{e}^{(1)}.$$

By Lemma 5.2(iii) (with $b = \sqrt{2}$ and $c_2 = 1$) we obtain that

$$\left\|\mathbf{e}^{(1)}\right\|_2 \leqslant \left(\tfrac{4}{3}\sqrt{6} + \sqrt{2} + \mathcal{O}(u)\right)u\|\mathbf{x}\|_2$$

and hence $\|\hat{\mathbf{u}}\|_2 \leqslant (\sqrt{2} + \mathcal{O}(u))\|\mathbf{x}\|_2$.

In step 2 of Algorithm 3.2 we get the computed vector $\hat{\mathbf{v}} = \mathrm{fl}\left(\left(\sqrt{n_1}C_{n_1}^{\mathrm{II}} \oplus \sqrt{n_1}C_{n_1}^{\mathrm{II}}\right)\hat{\mathbf{u}}\right)$ with the corresponding error vector $\mathbf{e}^{(2)}$ defined by

$$\hat{\mathbf{v}} = \left(\sqrt{n_1}C_{n_1}^{\mathrm{II}} \oplus \sqrt{n_1}C_{n_1}^{\mathrm{II}}\right)\hat{\mathbf{u}} + \mathbf{e}^{(2)}.$$

In order to compute $\hat{\mathbf{v}}$, we apply the fast DCT-II Algorithm 3.1 of length n_1 two times. Using (5.18), we can estimate

$$\left\|\mathbf{e}^{(2)}\right\|_2 \leqslant 2^{(t-1)/2}u\left(\left(\tfrac{4}{3}\sqrt{3}+\tfrac{1}{2}\sqrt{2}+3\right)(t-2)-1-\tfrac{1}{2}\sqrt{2}+\mathcal{O}(u)\right)\|\hat{\mathbf{u}}\|_2$$

and thus

$$\|\hat{\mathbf{v}}\|_2 \leqslant \left(\sqrt{n_1}+\mathcal{O}(u)\right)\|\hat{\mathbf{u}}\|_2 \leqslant \left(\sqrt{n}+\mathcal{O}(u)\right)\|\mathbf{x}\|_2.$$

Let $\hat{\mathbf{y}}$ be the computed vector $\mathrm{fl}(P_n^{\mathrm{T}}\hat{A}_n(1)\hat{\mathbf{v}})$ with the corresponding error vector $\mathbf{e}^{(3)}$ explained by

$$\hat{\mathbf{y}} = P_n^{\mathrm{T}}A_n(1)\hat{\mathbf{v}} + \mathbf{e}^{(3)}.$$

From Lemma 5.2(ii) (with $a = \sqrt{2}/2$ and $c_1 = 1/2$) it follows that

$$\|\mathbf{e}^{(3)}\|_2 \leqslant \left(2+\tfrac{1}{2}\sqrt{2}+\mathcal{O}(u)\right)u\|\hat{\mathbf{v}}\|_2.$$

By

$$\hat{\mathbf{y}} = \mathbf{y} + P_n^{\mathrm{T}}A_n(1)\left(\sqrt{n_1}C_{n_1}^{\mathrm{II}} \oplus \sqrt{n_1}C_{n_1}^{\mathrm{II}}\right)\mathbf{e}^{(1)} + P_n^{\mathrm{T}}A_n(1)\mathbf{e}^{(2)} + \mathbf{e}^{(3)}$$

we find the estimate

$$\begin{aligned}\|\hat{\mathbf{y}}-\mathbf{y}\|_2 &\leqslant \sqrt{n_1}\|\mathbf{e}^{(1)}\|_2 + \|\mathbf{e}^{(2)}\|_2 + \|\mathbf{e}^{(3)}\|_2\\ &\leqslant \sqrt{n}\left(\left(\tfrac{4}{3}\sqrt{3}+\tfrac{1}{2}\sqrt{2}+3\right)(t-1)-1-\tfrac{1}{2}\sqrt{2}+\mathcal{O}(u)\right)u\|\mathbf{x}\|_2.\end{aligned}$$

Finally, we scale this result by $1/\sqrt{n}$ and obtain the wanted constant k_n by (5.14). □

Remark 5.5. Let $n = 2^t$ $(t \geqslant 3)$. In [2], it has been shown that for computing $\mathbf{z} = C_n^{\mathrm{II}}\mathbf{x}$ one has normwise backward stability with

(i) $k_n = \sqrt{2}n^{3/2}$ for classical matrix–vector computation,
(ii) $k_n = \left(4\sqrt{2}+2\right)\log_2 n + \sqrt{2}$ for an FFT-based DCT-II algorithm, and
(iii) $k_n = 2\sqrt{3}(n-1)$ for the real fast algorithm [19] based on polynomial arithmetic.

In these algorithms, the non-trivial entries of the factor matrices were assumed to be precomputed exactly. As shown in Theorems 5.3 and 5.4, the fast DCT-II and DCT-IV algorithms are extremely stable with a constant which is comparable with the constant for the FFT-based DCT-II algorithm.

The similarly small constant

$$\begin{aligned}k_n &= \left(\tfrac{4}{3}\sqrt{3}+\tfrac{1}{2}\sqrt{2}+3\right)(\log_2 n-2)+2+\tfrac{3}{2}\sqrt{2}\\ &\approx 6.016508\log_2 n - 7.911695\end{aligned}$$

is achieved for the DCT-I Algorithm 3.6 (involving Algorithms 3.7 and 3.8) under the assumption that the values

$$\sqrt{2},\ \sqrt{2}\cos\tfrac{k\pi}{2^{t+1}},\ \sqrt{2}\sin\tfrac{k\pi}{2^{t+1}} \quad (k=1,\dots,2^{t-1}-1)$$

are precomputed with the absolute error bound u and including the final scaling with $1/\sqrt{n}$. Under the same assumption, the fast DCT-III Algorithm 3.7 (involving Algorithms 3.6 and 3.8) possesses the stability constant

$$\begin{aligned} k_n &= \left(\tfrac{4}{3}\sqrt{3}+\tfrac{1}{2}\sqrt{2}+3\right)(\log_2 n-1)+3+\tfrac{3}{2}\sqrt{2} \\ &\approx 6.016508\log_2 n-0.895188 \end{aligned}$$

after the final scaling with $1/\sqrt{n}$. The corresponding proofs use again matrix factorizations and are omitted here.

Considering the numerical stability of the MDCT-II Algorithm 3.4, we obtain only a constant $k_n=\mathcal{O}\left(\sqrt{n}\log_2 n\right)$. Hence the MDCT-II algorithm is much less stable than the above DCT-II Algorithm 3.1, but behaves better than the polynomial DCT-II algorithms considered in [2]. The larger constant k_n is due to the fact that all matrices $A'_n(\boldsymbol{\beta}_s)$ and $S'_n(\boldsymbol{\beta}_s)$, $s=1,\dots,t-2$, as well as $S_n(\boldsymbol{\beta}_{t-1})$ are not longer orthogonal, but have the spectral norm $\sqrt{2}$. In particular we obtain

Theorem 5.6. *Let $n=2^t$ $(t\geqslant 3)$. Assume that*

$$\sqrt{2},\ \sqrt{2}\cos\tfrac{\pi}{8},\ \sqrt{2}\sin\tfrac{\pi}{8}$$

are precomputed with absolute error bound u and that the values

$$\cos\tfrac{(2k+1)\pi}{2^{s+3}},\quad \sin\frac{(2k+1)\pi}{2^{s+3}}$$

with $k=0,\dots,2^s-1$ and $s=1,\dots,t-2$ are precomputed with absolute error bound $u/2$. Then the fast MDCT-II *Algorithm 3.4 is normwise backward stable with the constant*

$$\begin{aligned} k_n &= \sqrt{n}\left(\left(\tfrac{1}{3}\sqrt{6}+\tfrac{1}{4}\sqrt{2}+\tfrac{3}{4}\right)(\log_2 n-1)+\tfrac{2}{3}\sqrt{3}+\tfrac{1}{4}\sqrt{2}+1\right) \\ &\approx 1.920050\sqrt{n}\log_2 n+0.588204\sqrt{n}. \end{aligned}$$

Proof. The proof follows similar lines than that of Theorem 5.3. Using analogous notations as in the proof of Theorem 5.3 for $\mathbf{x}^{(s)}$, $\hat{\mathbf{x}}^{(s)}$, $\mathbf{e}^{(s)}$, $s=0,\dots,2t-1$, where the matrices $A_n(\boldsymbol{\beta}_s)$, $S_n(\boldsymbol{\beta}_s)$, $s=0,\dots,t-2$, are replaced by $A'_n(\boldsymbol{\beta}_s)$, $S'_n(\boldsymbol{\beta}_s)$, we find by $\|S'_n(\boldsymbol{\beta}_s)\|_2=\|P_n(s)A'_n(\boldsymbol{\beta}_s)\|_2=\sqrt{2}$ for $s=0,\dots,t-2$ and $\|S_n(\boldsymbol{\beta}_{t-1})\|_2=\sqrt{2}$ that

$$\left\|\hat{\mathbf{x}}^{(s+1)}\right\|_2\leqslant\sqrt{2}\left\|\hat{\mathbf{x}}^{(s)}\right\|_2+\left\|\mathbf{e}^{(s+1)}\right\|_2,\quad s=0,\dots,2t-2.$$

By Lemma 5.2 we obtain that

$$\begin{aligned} \left\|\mathbf{e}^{(s+1)}\right\|_2 &\leqslant \left(\tfrac{4}{3}\sqrt{3}+\tfrac{1}{2}\sqrt{2}+\mathcal{O}(u)\right)u\|\hat{\mathbf{x}}^{(s)}\|_2,\quad s=0,\dots,t-2, \\ \left\|\mathbf{e}^{(t)}\right\|_2 &\leqslant \left(\tfrac{4}{3}\sqrt{6}+\sqrt{2}+\mathcal{O}(u)\right)u\|\hat{\mathbf{x}}^{(t-1)}\|_2, \\ \left\|\mathbf{e}^{(t+s+1)}\right\|_2 &\leqslant \left(\sqrt{2}+1+\mathcal{O}(u)\right)u\|\hat{\mathbf{x}}^{(t+s)}\|_2,\quad s=0,\dots,t-3, \end{aligned}$$

and by $A_n'(\boldsymbol{\beta}_0) = I_n$ we have $\mathbf{e}^{(2t-1)} = \mathbf{0}$. Thus, with

$$\|\hat{\mathbf{x}}^{(s+1)}\|_2 \leqslant \big(2^{(s+1)/2} + \mathcal{O}(u)\big)\|\mathbf{x}\|_2, \quad s = 0, \ldots, 2t-2,$$

it follows that

$$\begin{aligned}
\|\mathbf{e}^{(s+1)}\|_2 &\leqslant 2^{s/2}\big(\tfrac{4}{3}\sqrt{3} + \sqrt{2} + \mathcal{O}(u)\big)u\|\mathbf{x}\|_2, \quad s = 0, \ldots, t-2,\\
\|\mathbf{e}^{(t)}\|_2 &\leqslant 2^{(t-1)/2}\big(\tfrac{4}{3}\sqrt{6} + \sqrt{2} + \mathcal{O}(u)\big)u\|\mathbf{x}\|_2,\\
\|\mathbf{e}^{(t+s+1)}\|_2 &\leqslant 2^{(s+t)/2}\big(\sqrt{2} + \mathcal{O}(u)\big)u\|\mathbf{x}\|_2, \quad s = 0, \ldots, t-3.
\end{aligned}$$

Hence,

$$\begin{aligned}
&\|\hat{\mathbf{x}}^{(2t-1)} - \mathbf{x}^{(2t-1)}\|_2\\
&\quad\leqslant \|P_n(1)A_n'(\boldsymbol{\beta}_1)\cdots P_n(t-2)A_n'(\boldsymbol{\beta}_{t-2})S_n(\boldsymbol{\beta}_{t-1})S_n'(\boldsymbol{\beta}_{t-2})\cdots S_n'(\boldsymbol{\beta}_1)\|_2\\
&\qquad\times \|\mathbf{e}^{(1)}\|_2 + \cdots + \|P_n(1)A_n'(\boldsymbol{\beta}_1)\|_2\|\mathbf{e}^{(2t-3)}\|_2 + \|\mathbf{e}^{(2t-2)}\|_2\\
&\leqslant \sum_{s=1}^{2t-2} 2^{t-1-s/2}\|\mathbf{e}^{(s)}\|_2\\
&\leqslant 2^t u\big(\big(\tfrac{1}{3}\sqrt{6} + \tfrac{1}{4}\sqrt{2} + \tfrac{3}{4}\big)(t-1) + \tfrac{2}{3}\sqrt{3} - \tfrac{1}{4}\sqrt{2} + \mathcal{O}(u)\big)\|\mathbf{x}\|_2.
\end{aligned}$$

After scaling with $2^{-t/2}$ we find the result by (5.14). □

Acknowledgment

The authors would like thank one of the referees for the very useful comments and hints to improve that paper. Further we thank K. Ihsberner for several corrections.

References

[1] H. Ahmed, T. Natarajan, K.R. Rao, Discrete cosine transform, IEEE Trans. Comput. 23 (1974) 90–93.
[2] G. Baszenski, U. Schreiber, M. Tasche, Numerical stability of fast cosine transforms, Numer. Funct. Anal. Optim. 21 (2000) 25–46.
[3] V. Britanak, A unified discrete cosine and discrete sine transform computation, Signal Process. 43 (1995) 333–339.
[4] W.H. Chen, C.H. Smith, S. Fralick, A fast computational algorithm for the discrete cosine transform, IEEE Trans. Comm. 25 (1977) 1004–1009.
[5] L. Cheng, H. Hu, Y. Luo, Integer discrete cosine transform and its fast algorithm, Electron. Lett. 37 (2001) 64–65.
[6] I. Daubechies, W. Sweldens, Factoring wavelet transforms into lifting steps, J. Fourier Anal. Appl. 4 (1998) 247–269.
[7] E. Feig, A scaled DCT algorithm, Proc. SPIE 1244 (1990), 2–13.

[8] E. Feig, S. Winograd, Fast algorithms for the discrete cosine transform, IEEE Trans. Signal Process. 40 (1992) 2174–2193.
[9] M.T. Heideman, D.H. Johnson, C.S. Burrus, Gauss and the history of the fast Fourier transform, Arch. Hist. Exact Sci. 34 (1985) 265–277.
[10] N.J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, Philadelphia, PA, 1996.
[11] H.S. Hou, A fast recursive algorithm for computing the discrete cosine transform, IEEE Trans. Acoust. Speech Signal Process. 35 (1987) 1455–1461.
[12] B. Lee, A new algorithm to compute the discrete cosine transform, IEEE Trans. Acoust. Speech Signal Process. 32 (1984) 1243–1245.
[13] L. Loeffler, A. Lightenberg, G.S. Moschytz, Practicle fast 1-d DCT algorithms with 11 multiplications, Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (1989) 989–991.
[14] M. Püschel, J.M. Moura, The algebraic approach to the discrete cosine and sine transforms and their fast algorithms, SIAM J. Comput. 32 (2003) 1280–1316.
[15] K.R. Rao, P. Yip, Discrete Cosine Transform: Algorithms, Advantages, Applications, Academic Press, Boston, 1990.
[16] C. Runge, H. König, Lectures on Numerical Arithmetic, Springer, Berlin, 1924 (in German).
[17] U. Schreiber, Fast and numerically stable trigonometric transforms, Thesis, University of Rostock, 1999 (in German).
[18] A.N. Skodras, C.A. Christopolous, Split-radix fast cosine transform algorithm, Int. J. Electr. 74 (1993) 513–522.
[19] G. Steidl, Fast radix-p discrete cosine transform, Appl. Algebra Engrg. Comm. Comput. 3 (1992) 39–46.
[20] G. Steidl, M. Tasche, A polynomial approach to fast algorithms for discrete Fourier-cosine and Fourier-sine transforms, Math. Comput. 56 (1991) 281–296.
[21] G. Strang, The discrete cosine transform, SIAM Rev. 41 (1999) 135–147.
[22] C.W. Sun, P. Yip, Split-radix algorithms for DCT and DST, Proc. Asilomar Conf. Signals Systems Comput., Pacific Grove, 1989, pp. 508–512.
[23] M. Tasche, H. Zeuner, Roundoff error analysis for fast trigonometric transforms, in: G. Anastassiou (Ed.), Handbook of Analytic-Computational Methods in Applied Mathematics, Chapman & Hall/CRC Press, Boca Raton, 2000, pp. 357–406.
[24] C.F. Van Loan, Computational Framework for the Fast Fourier Transform, SIAM, Philadelphia, PA, 1992.
[25] Z. Wang, Fast algorithms for the discrete W transform and the discrete Fourier transform, IEEE Trans. Acoust. Speech Signal Process. 32 (1984) 803–816.
[26] Z. Wang, On computing the discrete Fourier and cosine transforms, IEEE Trans. Acoust. Speech Signal Process. 33 (1985) 1341–1344.
[27] H. Zeuner, A general theory of stochastic roundoff error analysis with applications to DFT and DCT, J. Comput. Anal. Appl., in press.