# Discrete cosine transform

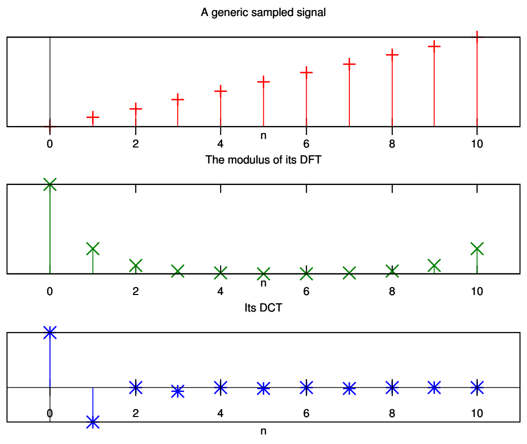
A **discrete cosine transform** (**DCT**) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in science and engineering, from lossy compression of audio (e.g. MP3) and images (e.g. JPEG) (where small high-frequency components can be discarded), to spectral methods for the numerical solution of partial differential equations. The use of cosine rather than sine functions is critical for compression, since it turns out (as described below) that fewer cosine functions are needed to approximate a typical signal, whereas for differential equations the cosines express a particular choice of boundary conditions.

In particular, a DCT is a Fourier-related transform similar to the discrete Fourier transform (DFT), but using only real numbers. The DCTs are generally related to Fourier Series coefficients of a periodically and symmetrically extended sequence whereas DFTs are related to Fourier Series coefficients of a periodically extended sequence. DCTs are equivalent to DFTs of roughly twice the length, operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even), whereas in some variants the input and/or output data are shifted by half a sample. There are eight standard DCT variants, of which four are common.

The most common variant of discrete cosine transform is the type-II DCT, which is often called simply "the DCT".[1][2] Its inverse, the type-III DCT, is correspondingly often called simply "the inverse DCT" or "the IDCT". Two related transforms are the discrete sine transform (DST), which is equivalent to a DFT of real and *odd* functions, and the modified discrete cosine transform (MDCT), which is based on a DCT of *overlapping* data. Multidimensional DCTs (MD DCTs) are developed to extend the concept of DCT on MD Signals. There are several algorithms to compute MD DCT. A new variety of fast algorithms are also developed to reduce the computational complexity of implementing DCT.

## 1 Applications

The DCT, and in particular the DCT-II, is often used in signal and image processing, especially for lossy compression, because it has a strong "energy compaction" property:[1][2] in typical applications, most of the signal information tends to be concentrated in a few low-frequency components of the DCT. For strongly correlated Markov processes, the DCT can approach the compaction efficiency of the Karhunen-Loève transform (which is optimal in the decorrelation sense). As explained below, this stems from the boundary conditions implicit in the cosine functions.



*DCT-II (bottom) compared to the DFT (middle) of an input signal (top).*

A related transform, the *modified* discrete cosine transform, or MDCT (based on the DCT-IV), is used in AAC, Vorbis, WMA, and MP3 audio compression.

DCTs are also widely employed in solving partial differential equations by spectral methods, where the different variants of the DCT correspond to slightly different even/odd boundary conditions at the two ends of the array.

DCTs are also closely related to Chebyshev polynomials, and fast DCT algorithms (below) are used in Chebyshev approximation of arbitrary functions by series of Chebyshev polynomials, for example in Clenshaw–Curtis quadrature.

### 1.1 JPEG

Main article: JPEG § Discrete cosine transform

The DCT is used in JPEG image compression, MJPEG, MPEG, DV, Daala, and Theora video compression. There, the two-dimensional DCT-II of $N \times N$ blocks are computed and the results are quantized and entropy coded. In this case, $N$ is typically 8 and the DCT-II formula is applied to each row and column of the block. The result is an $8 \times 8$ transform coefficient array in which the

$(0,0)$ element (top-left) is the DC (zero-frequency) component and entries with increasing vertical and horizontal index values represent higher vertical and horizontal spatial frequencies.

Multidimensional DCTs (MD DCTs) have several applications mainly 3-D DCT-II has several new applications like Hyperspectral Imaging coding systems,[3] variable temporal length 3-D DCT coding,[4] video coding algorithms,[5] adaptive video coding [6] and 3-D Compression.[7] Due to enhancement in the hardware, software and introduction of several fast algorithms, the necessity of using M-D DCTs is rapidly increasing. DCT-IV has gained popularity for its applications in fast implementation of real-valued polyphase filtering banks,[8] lapped orthogonal transform[9][10] and cosine-modulated wavelet bases.[11]

## 2 Informal overview

Like any Fourier-related transform, discrete cosine transforms (DCTs) express a function or a signal in terms of a sum of sinusoids with different frequencies and amplitudes. Like the discrete Fourier transform (DFT), a DCT operates on a function at a finite number of discrete data points. The obvious distinction between a DCT and a DFT is that the former uses only cosine functions, while the latter uses both cosines and sines (in the form of complex exponentials). However, this visible difference is merely a consequence of a deeper distinction: a DCT implies different boundary conditions from the DFT or other related transforms.

The Fourier-related transforms that operate on a function over a finite domain, such as the DFT or DCT or a Fourier series, can be thought of as implicitly defining an *extension* of that function outside the domain. That is, once you write a function $f(x)$ as a sum of sinusoids, you can evaluate that sum at any $x$ , even for $x$ where the original $f(x)$ was not specified. The DFT, like the Fourier series, implies a periodic extension of the original function. A DCT, like a cosine transform, implies an even extension of the original function.

However, because DCTs operate on *finite*, *discrete* sequences, two issues arise that do not apply for the continuous cosine transform. First, one has to specify whether the function is even or odd at *both* the left and right boundaries of the domain (i.e. the min-*n* and max-*n* boundaries in the definitions below, respectively). Second, one has to specify around *what point* the function is even or odd. In particular, consider a sequence *abcd* of four equally spaced data points, and say that we specify an even *left* boundary. There are two sensible possibilities: either the data are even about the sample *a*, in which case the even extension is *dcbabcd*, or the data are even about the point *halfway* between *a* and the previous point, in which case the even extension is *dcbaabcd* (*a* is repeated).

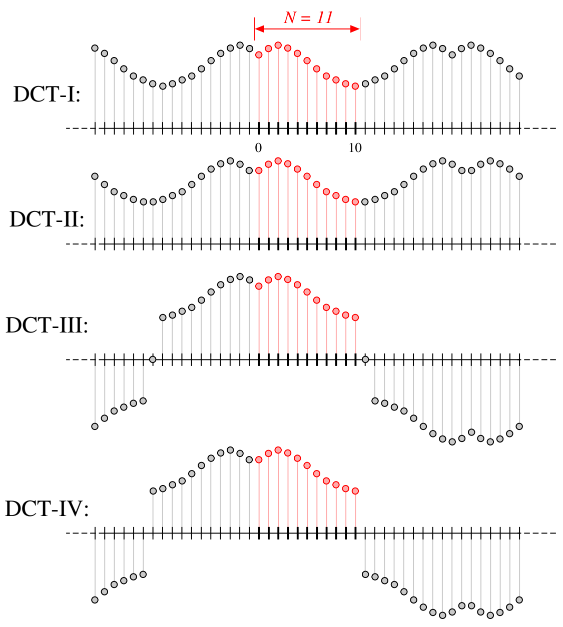


*Illustration of the implicit even/odd extensions of DCT input data, for* N=*11 data points (red dots), for the four most common types of DCT (types I-IV).*

These choices lead to all the standard variations of DCTs and also discrete sine transforms (DSTs). Each boundary can be either even or odd (2 choices per boundary) and can be symmetric about a data point or the point halfway between two data points (2 choices per boundary), for a total of $2 \times 2 \times 2 \times 2 = 16$ possibilities. Half of these possibilities, those where the *left* boundary is even, correspond to the 8 types of DCT; the other half are the 8 types of DST.

These different boundary conditions strongly affect the applications of the transform and lead to uniquely useful properties for the various DCT types. Most directly, when using Fourier-related transforms to solve partial differential equations by spectral methods, the boundary conditions are directly specified as a part of the problem being solved. Or, for the MDCT (based on the type-IV DCT), the boundary conditions are intimately involved in the MDCT's critical property of time-domain aliasing cancellation. In a more subtle fashion, the boundary conditions are responsible for the "energy compactification" properties that make DCTs useful for image and audio compression, because the boundaries affect the rate of convergence of any Fourier-like series.

In particular, it is well known that any discontinuities in a function reduce the rate of convergence of the Fourier series, so that more sinusoids are needed to represent the function with a given accuracy. The same principle governs the usefulness of the DFT and other transforms for signal compression; the smoother a function is, the fewer terms in its DFT or DCT are required to represent it accurately, and the more it can be compressed. (Here, we think of the DFT or DCT as approximations for the

Fourier series or cosine series of a function, respectively, in order to talk about its "smoothness".) However, the implicit periodicity of the DFT means that discontinuities usually occur at the boundaries: any random segment of a signal is unlikely to have the same value at both the left and right boundaries. (A similar problem arises for the DST, in which the odd left boundary condition implies a discontinuity for any function that does not happen to be zero at that boundary.) In contrast, a DCT where *both* boundaries are even *always* yields a continuous extension at the boundaries (although the slope is generally discontinuous). This is why DCTs, and in particular DCTs of types I, II, V, and VI (the types that have two even boundaries) generally perform better for signal compression than DFTs and DSTs. In practice, a type-II DCT is usually preferred for such applications, in part for reasons of computational convenience.

# 3 Formal definition

Formally, the discrete cosine transform is a linear, invertible function $f : \mathbb{R}^N \to \mathbb{R}^N$ (where $\mathbb{R}$ denotes the set of real numbers), or equivalently an invertible $N \times N$ square matrix. There are several variants of the DCT with slightly modified definitions. The $N$ real numbers $x_0$, ..., $xN_{-1}$ are transformed into the $N$ real numbers $X_0$, ..., $XN_{-1}$ according to one of the formulas:

## 3.1 DCT-I

$$X_k = \frac{1}{2}(x_0+(-1)^k x_{N-1}) + \sum_{n=1}^{N-2} x_n \cos\left[\frac{\pi}{N-1} nk\right]$$

Some authors further multiply the $x_0$ and $xN_{-1}$ terms by √2, and correspondingly multiply the $X_0$ and $XN_{-1}$ terms by 1/√2. This makes the DCT-I matrix orthogonal, if one further multiplies by an overall scale factor of $\sqrt{\frac{2}{N-1}}$ , but breaks the direct correspondence with a real-even DFT.

The DCT-I is exactly equivalent (up to an overall scale factor of 2), to a DFT of $2N-2$ real numbers with even symmetry. For example, a DCT-I of $N$=5 real numbers *abcde* is exactly equivalent to a DFT of eight real numbers *abcdedcb* (even symmetry), divided by two. (In contrast, DCT types II-IV involve a half-sample shift in the equivalent DFT.)

Note, however, that the DCT-I is not defined for $N$ less than 2. (All other DCT types are defined for any positive $N$.)

Thus, the DCT-I corresponds to the boundary conditions: *xn* is even around *n* = 0 and even around *n* = *N*−1; similarly for *Xk*.

## 3.2 DCT-II

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad k = 0, \dots, N-1.$$

The DCT-II is probably the most commonly used form, and is often simply referred to as "the DCT".[1][2]

This transform is exactly equivalent (up to an overall scale factor of 2) to a DFT of $4N$ real inputs of even symmetry where the even-indexed elements are zero. That is, it is half of the DFT of the $4N$ inputs $y_n$ , where $y_{2n} = 0$ , $y_{2n+1} = x_n$ for $0 \le n < N$ , $y_{2N} = 0$ , and $y_{4N-n} = y_n$ for $0 < n < 2N$ .

Some authors further multiply the $X_0$ term by 1/√2 and multiply the resulting matrix by an overall scale factor of $\sqrt{\frac{2}{N}}$ (see below for the corresponding change in DCT-III). This makes the DCT-II matrix orthogonal, but breaks the direct correspondence with a real-even DFT of half-shifted input. This is the normalization used by Matlab, for example. In many applications, such as JPEG, the scaling is arbitrary because scale factors can be combined with a subsequent computational step (e.g. the quantization step in JPEG[12]), and a scaling can be chosen that allows the DCT to be computed with fewer multiplications.[13][14]

The DCT-II implies the boundary conditions: *xn* is even around *n* = −1/2 and even around *n* = *N*−1/2; *Xk* is even around *k* = 0 and odd around *k* = *N*.

## 3.3 DCT-III

$$X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n \cos\left[\frac{\pi}{N} n\left(k+\frac{1}{2}\right)\right] \qquad k = 0, \dots, N-1.$$

Because it is the inverse of DCT-II (up to a scale factor, see below), this form is sometimes simply referred to as "the inverse DCT" ("IDCT").[2]

Some authors divide the $x_0$ term by √2 instead of by 2 (resulting in an overall $x_0$/√2 term) and multiply the resulting matrix by an overall scale factor of $\sqrt{\frac{2}{N}}$ (see above for the corresponding change in DCT-II), so that the DCT-II and DCT-III are transposes of one another. This makes the DCT-III matrix orthogonal, but breaks the direct correspondence with a real-even DFT of half-shifted output.

The DCT-III implies the boundary conditions: *xn* is even around *n* = 0 and odd around *n* = *N*; *Xk* is even around *k* = −1/2 and odd around *k* = *N*−1/2.

## 3.4 DCT-IV

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\right] \qquad k = 0, \dots, N-1.$$

The DCT-IV matrix becomes orthogonal (and thus, being clearly symmetric, its own inverse) if one further multiplies by an overall scale factor of $\sqrt{2/N}$ .

A variant of the DCT-IV, where data from different transforms are *overlapped*, is called the modified discrete cosine transform (MDCT).[15]

The DCT-IV implies the boundary conditions: $xn$ is even around $n = -1/2$ and odd around $n = N-1/2$; similarly for $Xk$.

### 3.5 DCT V-VIII

DCTs of types I-IV treat both boundaries consistently regarding the point of symmetry: they are even/odd around either a data point for both boundaries or halfway between two data points for both boundaries. By contrast, DCTs of types V-VIII imply boundaries that are even/odd around a data point for one boundary and halfway between two data points for the other boundary.

In other words, DCT types I-IV are equivalent to real-even DFTs of even order (regardless of whether $N$ is even or odd), since the corresponding DFT is of length $2(N-1)$ (for DCT-I) or $4N$ (for DCT-II/III) or $8N$ (for DCT-IV). The four additional types of discrete cosine transform[16] correspond essentially to real-even DFTs of logically odd order, which have factors of $N \pm ½$ in the denominators of the cosine arguments.

However, these variants seem to be rarely used in practice. One reason, perhaps, is that FFT algorithms for odd-length DFTs are generally more complicated than FFT algorithms for even-length DFTs (e.g. the simplest radix-2 algorithms are only for even lengths), and this increased intricacy carries over to the DCTs as described below.

(The trivial real-even array, a length-one DFT (odd length) of a single number $a$, corresponds to a DCT-V of length $N = 1$.)

## 4 Inverse transforms

Using the normalization conventions above, the inverse of DCT-I is DCT-I multiplied by $2/(N-1)$. The inverse of DCT-IV is DCT-IV multiplied by $2/N$. The inverse of DCT-II is DCT-III multiplied by $2/N$ and vice versa.[2]

Like for the DFT, the normalization factor in front of these transform definitions is merely a convention and differs between treatments. For example, some authors multiply the transforms by $\sqrt{2/N}$ so that the inverse does not require any additional multiplicative factor. Combined with appropriate factors of √2 (see above), this can be used to make the transform matrix orthogonal.

## 5 Multidimensional DCTs

Multidimensional variants of the various DCT types follow straightforwardly from the one-dimensional definitions: they are simply a separable product (equivalently, a composition) of DCTs along each dimension.

### 5.1 M-D DCT-II

For example, a two-dimensional DCT-II of an image or a matrix is simply the one-dimensional DCT-II, from above, performed along the rows and then along the columns (or vice versa). That is, the 2D DCT-II is given by the formula (omitting normalization and other scale factors, as above):

$$\begin{aligned} X_{k_1,k_2} &= \sum_{n_1=0}^{N_1-1}\left(\sum_{n_2=0}^{N_2-1} x_{n_1,n_2}\cos\left[\frac{\pi}{N_2}\left(n_2+\frac{1}{2}\right)k_2\right]\right)\cos\Big[\frac{\pi}{N_1} \\ &= \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x_{n_1,n_2}\cos\left[\frac{\pi}{N_1}\left(n_1+\frac{1}{2}\right)k_1\right]\cos\Big[\frac{\pi}{N_2}\Big( \end{aligned}$$

The inverse of a multi-dimensional DCT is just a separable product of the inverse(s) of the corresponding one-dimensional DCT(s) (see above), e.g. the one-dimensional inverses applied along one dimension at a time in a row-column algorithm.

The *3-D DCT-II* is only the extension of *2-D DCT-II* in three dimensional space and mathematically can be calculated by the formula

$$X_{k_1,k_2,k_3} = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\sum_{n_3=0}^{N_3-1} x_{n_1,n_2,n_3}\cos\left[\frac{\pi}{N_1}\left(n_1+\frac{1}{2}\right)k_1\right]\cos\Big[\frac{\pi}{N}$$

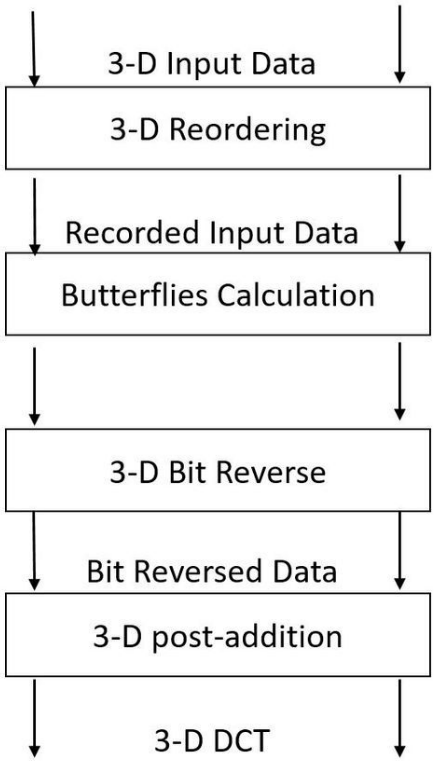
The inverse of **3-D DCT-II** is **3-D DCT-III** and can be computed from the formula given by

$$x_{n_1,n_2,n_3} = \sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}\sum_{k_3=0}^{N_3-1} X_{k_1,k_2,k_3}\cos\left[\frac{\pi}{N_1}\left(n_1+\frac{1}{2}\right)k_1\right]\cos\Big[\frac{\pi}{N}$$

Technically, computing a two-, three- (or -multi) dimensional DCT by sequences of one-dimensional DCTs along each dimension is known as a *row-column* algorithm. As with multidimensional FFT algorithms, however, there exist other methods to compute the same thing while performing the computations in a different order (i.e. interleaving/combining the algorithms for the different dimensions). Owing to the rapid growth in the applications based on the 3-D DCT, several fast algorithms are developed for the computation of 3-D DCT-II. Vector-Radix algorithms are applied for computing M-D DCT to reduce the computational complexity and to increase the computational speed. To compute 3-D DCT-II efficiently, a fast algorithm, Vector-Radix Decimation in Frequency (VR DIF) algorithm was developed.

### 5.1.1 3-D DCT-II VR DIF

In order to apply the VR DIF algorithm the input data is to be formulated and rearranged as follows.[17][18] The transform sixe *N x N x N* is assumed to be *2*.



*The four basic stages of computing 3-D DCT-II using VR DIF Algorithm.*

$\tilde{x}(n_1,n_2,n_3) = x(2n_1,2n_2,2n_3)$
$\tilde{x}(n_1,n_2,N-n_3-1) = x(2n_1,2n_2,2n_3+1)$
$\tilde{x}(n_1,N-n_2-1,n_3) = x(2n_1,2n_2+1,2n_3)$
$\tilde{x}(n_1,N-n_2-1,N-n_3-1) = x(2n_1,2n_2+1,2n_3+1)$
$\tilde{x}(N-n_1-1,n_2,n_3) = x(2n_1+1,2n_2,2n_3)$
$\tilde{x}(N-n_1-1,n_2,N-n_3-1) = x(2n_1+1,2n_2,2n_3+1)$
$\tilde{x}(N-n_1-1,N-n_2-1,n_3) = x(2n_1+1,2n_2+1,2n_3)$
$\tilde{x}(N-n_1-1,N-n_2-1,N-n_3-1 = x(2n_1+1,2n_2+1,2n_3+1)$
where $0 \le n_1,n_2,n_3 \le \frac{N}{2}-1$

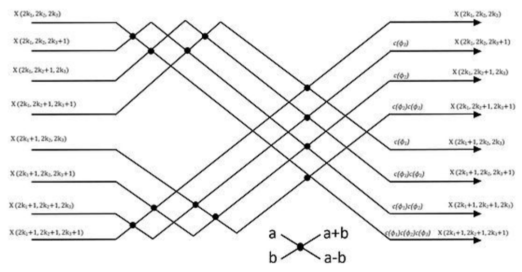
The figure to the adjacent shows the four stages that are involved in calculating 3-D DCT-II using VR DIF algorithm. The first stage is the 3-D reordering using the index mapping illustrated by the above equations. The second stage is the butterfly calculation. Each butterfly calculates eight points together as shown in the figure just below., where $c(\phi_i) = \cos(\phi_i)$ .

The original 3-D DCT-II now can be written as

$$X(k_1,k_2,k_3) = \sum_{n_1=1}^{N-1}\sum_{n_2=1}^{N-1}\sum_{n_3=1}^{N-1} \tilde{x}(n_1,n_2,n_3)\cos(\phi k_1)\cos(\phi k_2)\cos(\phi k_3$$

where $\phi_i = \frac{\pi}{2N}(4N_i+1)$, and $i = 1,2,3$ .

If the even and the odd parts of $k_1, k_2$ and $k_3$ and are considered, the general formula for the calculation of the 3-D DCT-II can be expressed as



*The single butterfly stage of VR DIF algorithm.*

$$X(k_1,k_2,k_3) = \sum_{n_1=1}^{\frac{N}{2}-1}\sum_{n_2=1}^{\frac{N}{2}-1}\sum_{n_1=1}^{\frac{N}{2}-1} \tilde{x}_{ijl}(n_1,n_2,n_3)\cos(\phi(2k_1+i)\cos(\phi(2k$$

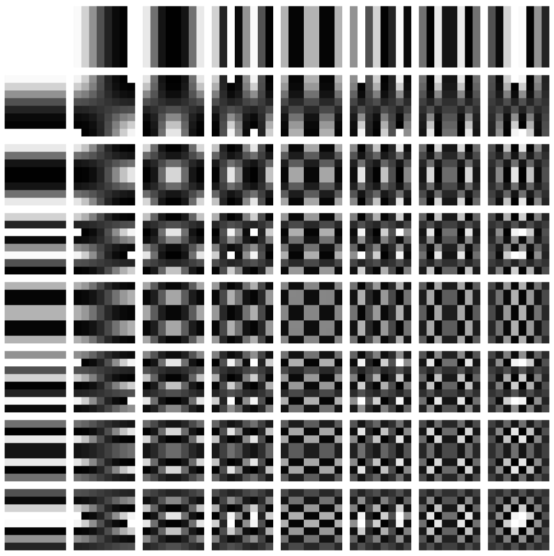
where

$$\tilde{x}_{ijl}(n_1,n_2,n_3) = \tilde{x}(n_1,n_2,n_3) + (-1)^l\tilde{x}\left(n_1,n_2,n_3+\frac{n}{2}\right)$$
$$+(-1)^j\tilde{x}\left(n_1,n_2+\frac{n}{2},n_3\right) + (-1)^{j+l}\tilde{x}\left(n_1,n_2+\frac{n}{2},n_3+\frac{n}{2}\right)$$
$$+(-1)^i\tilde{x}\left(n_1+\frac{n}{2},n_2,n_3\right) + (-1)^{i+j}\tilde{x}\left(n_1+\frac{n}{2}+\frac{n}{2},n_2,n_3\right)$$
$$+(-1)^{i+l}\tilde{x}\left(n_1+\frac{n}{2},n_2,n_3+\frac{n}{3}\right)$$
$$+(-1)^{i+j+l}\tilde{x}\left(n_1+\frac{n}{2},n_2+\frac{n}{2},n_3+\frac{n}{2}\right) \text{ where } i,j,l = 0 \text{ or } 1.$$

**Arithmetic complexity** The whole 3-D DCT calculation needs $[\log_2 N]$ stages, and each stage involves $N^3/8$ butterflies. The whole 3-D DCT requires $[(N^3/8)\log_2 N]$ butterflies to be computed. Each butterfly requires seven real multiplications (including trivial multiplications) and 24 real additions (including trivial additions). Therefore, the total number of real multiplications needed for this stage is $\left[(7/8)N^3\log_2 N\right]$ , and the total number of real additions i.e. including the post-additions (recursive additions) which can be calculated directly after the butterfly stage or after the bit-reverse stage are given by[18] $\underbrace{\left[\frac{3}{2}N^3\log_2 N\right]}_{\text{Real}} + \underbrace{\left[\frac{3}{2}N^3\log_2 N - 3N^3 + 3N^2\right]}_{\text{Recursive}} = \left[\frac{9}{2}N^3\log_2 N - 3N^3 + 3N^2\right]$ .

The conventional method to calculate MD-DCT-II is using a Row-Column-Frame (RCF) approach which is computationally complex and less productive on most advanced recent hardware platforms. The number of multiplications required to compute VR DIF Algorithm when compared to RCF algorithm are quite a few in number. The number of Multiplications and additions involved in RCF approach are given by $\left[\frac{3}{2}N^3\log_2 N\right]$ and $\left[\frac{9}{2}N^3\log_2 N - 3N^3 + 3N^2\right]$ respectively. From Table 1, it can be seen that the total number

of multiplications associated with the 3-D DCT VR algorithm is less than that associated with the RCF approach by more than 40%. In addition, the RCF approach involves matrix transpose and more indexing and data swapping than the new VR algorithm. This makes the 3-D DCT VR algorithm more efficient and better suited for 3-D applications that involve the 3-D DCT-II such as video compression and other 3-D image processing applications. The main consideration in choosing a fast algorithm is to avoid computational and structural complexities. As the technology of computers and DSPs advances, the execution time of arithmetic operations (multiplications and additions) is becoming very fast, and regular computational structure becomes the most important factor.[19] Therefore, although the above proposed 3-D VR algorithm does not achieve the theoretical lower bound on the number of multiplications,[20] it has a simpler computational structure as compared to other 3-D DCT algorithms. It can be implemented in place using a single butterfly and possesses the properties of the Cooley–Tukey FFT algorithm in 3-D. Hence, the 3-D VR presents a good choice for reducing arithmetic operations in the calculation of the 3-D DCT-II while keeping the simple structure that characterize butterfly style Cooley–Tukey FFT algorithms.



*Two-dimensional DCT frequencies from the JPEG DCT*

The image to the right shows a combination of horizontal and vertical frequencies for an 8 x 8 ( $N_1 = N_2 = 8$ ) two-dimensional DCT. Each step from left to right and top to bottom is an increase in frequency by 1/2 cycle.

For example, moving right one from the top-left square yields a half-cycle increase in the horizontal frequency. Another move to the right yields two half-cycles. A move down yields two half-cycles horizontally and a half-cycle vertically. The source data (8x8) is transformed to a linear combination of these 64 frequency squares.

### 5.2 MD-DCT-IV

The M-D DCT-IV is just an extension of 1-D DCT-IV on to M dimensional domain. The 2-D DCT-IV of a matirx or an image is given by

$$X_{k,l} = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} x_{n,m} \cos\left(\frac{(2n+1)(2k+1)\pi}{4N}\right) \cos\left(\frac{(2n+1)(2l+1}{4M}\right.$$

We can compute the MD DCT-IV using the regular row-column method or we can use the polynomial transform method[21] for the fast and efficient computation. The main idea of this algorithm is to use the Polynomial Transform to convert the multidimensional DCT into a series of 1-D DCTs directly. MD DCT-IV also has several applications in various fiields.

## 6 Computation

Although the direct application of these formulas would require O($N^2$) operations, it is possible to compute the same thing with only O($N$ log $N$) complexity by factorizing the computation similarly to the fast Fourier transform (FFT). One can also compute DCTs via FFTs combined with O($N$) pre- and post-processing steps. In general, O($N$ log $N$) methods to compute DCTs are known as fast cosine transform (FCT) algorithms.

The most efficient algorithms, in principle, are usually those that are specialized directly for the DCT, as opposed to using an ordinary FFT plus O($N$) extra operations (see below for an exception). However, even "specialized" DCT algorithms (including all of those that achieve the lowest known arithmetic counts, at least for power-of-two sizes) are typically closely related to FFT algorithms—since DCTs are essentially DFTs of real-even data, one can design a fast DCT algorithm by taking an FFT and eliminating the redundant operations due to this symmetry. This can even be done automatically (Frigo & Johnson, 2005). Algorithms based on the Cooley–Tukey FFT algorithm are most common, but any other FFT algorithm is also applicable. For example, the Winograd FFT algorithm leads to minimal-multiplication algorithms for the DFT, albeit generally at the cost of more additions, and a similar algorithm was proposed by

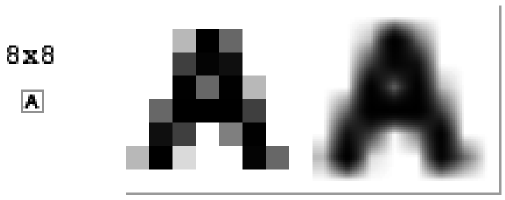
Feig & Winograd (1992) for the DCT. Because the algorithms for DFTs, DCTs, and similar transforms are all so closely related, any improvement in algorithms for one transform will theoretically lead to immediate gains for the other transforms as well (Duhamel & Vetterli 1990).

While DCT algorithms that employ an unmodified FFT often have some theoretical overhead compared to the best specialized DCT algorithms, the former also have a distinct advantage: highly optimized FFT programs are widely available. Thus, in practice, it is often easier to obtain high performance for general lengths $N$ with FFT-based algorithms. (Performance on modern hardware is typically not dominated simply by arithmetic counts, and optimization requires substantial engineering effort.) Specialized DCT algorithms, on the other hand, see widespread use for transforms of small, fixed sizes such as the $8 \times 8$ DCT-II used in JPEG compression, or the small DCTs (or MDCTs) typically used in audio compression. (Reduced code size may also be a reason to use a specialized DCT for embedded-device applications.)

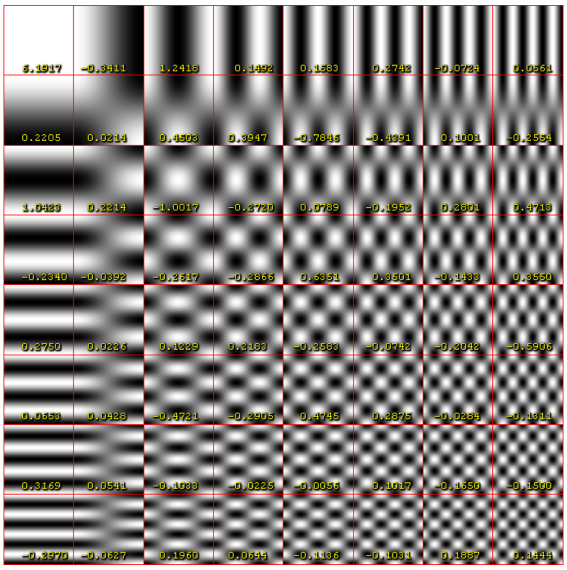
In fact, even the DCT algorithms using an ordinary FFT are sometimes equivalent to pruning the redundant operations from a larger FFT of real-symmetric data, and they can even be optimal from the perspective of arithmetic counts. For example, a type-II DCT is equivalent to a DFT of size $4N$ with real-even symmetry whose even-indexed elements are zero. One of the most common methods for computing this via an FFT (e.g. the method used in FFTPACK and FFTW) was described by Narasimha & Peterson (1978) and Makhoul (1980), and this method in hindsight can be seen as one step of a radix-4 decimation-in-time Cooley–Tukey algorithm applied to the "logical" real-even DFT corresponding to the DCT II. (The radix-4 step reduces the size $4N$ DFT to four size- $N$ DFTs of real data, two of which are zero and two of which are equal to one another by the even symmetry, hence giving a single size- $N$ FFT of real data plus $O(N)$ butterflies.) Because the even-indexed elements are zero, this radix-4 step is exactly the same as a split-radix step; if the subsequent size- $N$ real-data FFT is also performed by a real-data split-radix algorithm (as in Sorensen et al. 1987), then the resulting algorithm actually matches what was long the lowest published arithmetic count for the power-of-two DCT-II ( $2N \log_2 N - N + 2$ real-arithmetic operations[lower-alpha 1]). So, there is nothing intrinsically bad about computing the DCT via an FFT from an arithmetic perspective—it is sometimes merely a question of whether the corresponding FFT algorithm is optimal. (As a practical matter, the function-call overhead in invoking a separate FFT routine might be significant for small $N$ , but this is an implementation rather than an algorithmic question since it can be solved by unrolling/inlining.)

## 7 Example of IDCT

Consider this 8x8 grayscale image of capital letter A.



*Original size, scaled 10x (nearest neighbor), scaled 10x (bilinear).*

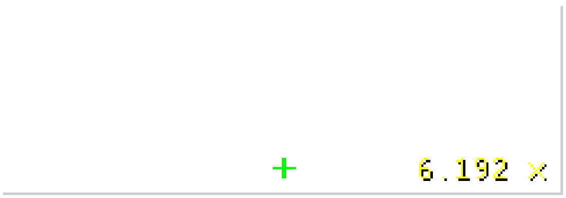


*Basis functions of the discrete cosine transformation with corresponding coefficients (specific for our image).*

$$\textit{DCT of the image} = \begin{bmatrix} 6.1917 & -0.3411 & 1.2418 & 0.1492 & 0.1583 \\ 0.2205 & 0.0214 & 0.4503 & 0.3947 & -0.7846 \\ 1.0423 & 0.2214 & -1.0017 & -0.2720 & 0.0789 \\ -0.2340 & -0.0392 & -0.2617 & -0.2866 & 0.6351 \\ 0.2750 & 0.0226 & 0.1229 & 0.2183 & -0.2583 \\ 0.0653 & 0.0428 & -0.4721 & -0.2905 & 0.4745 \\ 0.3169 & 0.0541 & -0.1033 & -0.0225 & -0.0056 \\ -0.2970 & -0.0627 & 0.1960 & 0.0644 & -0.1136 \end{bmatrix}$$

.

Each basis function is multiplied by its coefficient and then this product is added to the final image.



*On the left is the final image. In the middle is the weighted function (multiplied by a coefficient) which is added to the final image. On the right is the current function and corresponding coefficient. Images are scaled (using bilinear interpolation) by factor 10×.*

## 8 See also

- JPEG—Contains a potentially easier to understand example of DCT transformation
- Modified discrete cosine transform
- Discrete sine transform
- Discrete Fourier transform
- List of Fourier-related transforms
- Discrete wavelet transform

## 9 Notes

[1] The precise count of real arithmetic operations, and in particular the count of real multiplications, depends somewhat on the scaling of the transform definition. The $2N \log_2 N - N + 2$ count is for the DCT-II definition shown here; two multiplications can be saved if the transform is scaled by an overall $\sqrt{2}$ factor. Additional multiplications can be saved if one permits the outputs of the transform to be rescaled individually, as was shown by Arai, Agui & Nakajima (1988) for the size-8 case used in JPEG.

## 10 Citations

[1] Ahmed, N.; Natarajan, T.; Rao, K. R. (January 1974), “Discrete Cosine Transform”, *IEEE Transactions on Computers*, **C–23** (1): 90–93, doi:10.1109/T-C.1974.223784

[2] Rao, K; Yip, P (1990), *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Boston: Academic Press, ISBN 0-12-580203-X

[3] Abousleman, G. P.; Marcellin, M. W.; Hunt, B. R. (January 1995), “Compression of hyperspectral imagery using 3-D DCT and hybrid DPCM/DCT”, *IEEE Trans. Geosci. Remote. Sens.*, **33**: 26–34, doi:10.1109/36.368225

[4] Chan, Y.; Siu, W. (May 1997), “Variable temporal-length 3-D discrete cosine transform coding”, *IEEE Trans. Image Processing.*, **6**: 758–763, doi:10.1109/83.568933

[5] Song, J.; SXiong, Z.; Liu, X.; Liu, Y., “An algorithm for layered video coding and transmission”, *Proc. Fourth Int. Conf./Exh. High Performace Comput. Asia-Pacific Region*, **2**: 700–703

[6] Tai, S.-C,; Gi, Y.; Lin, C.-W. (September 2000), “An adaptive 3-D discrete cosine transform coder for medical image compression”, *IEEE. Trans. Inform. Technol. Biomed.*, **4**: 259–263, doi:10.1109/4233.870036

[7] Yeo, B.; Liu, B. (May 1995), “Volume rendering of DCT-based compressed 3D scalar data”, *IEEE Trans. Comput. Graphics.*, **1**: 29–43, doi:10.1109/2945.468390

[8] CHAN, S.C., LIU, W., and HO, K.L.: ‘Perfect reconstruction modulated filter banks with sum of powers-of-two coefficients’. Proceedings of Inte.n Symp. Circuits and syst., 28-3 1 May 2000, Geneva, Switzerland,p p. 28-31

[9] Queiroz, R. L.; Nguyen, T. Q. (1996). “Lapped transforms for efficient transform/subband coding”. *IEEE Trans. Signal Process.* **44** (5): 497–507.

[10] Malvar, H. S. (1992). *Signal processing with lapped transforms.* Englewood Cliffs, NJ: Prentice Hall.

[11] Chan, S. C.; Luo, L.; Ho, K. L. (1998). “M-Channel compactly supported biorthogonal cosine-modulated wavelet bases”. *IEEE Trans. Signal Process.* **46** (2): 1142–1151.

[12] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard.* New York: Van Nostrand Reinhold, 1993.

[13] Y. Arai, T. Agui, and M. Nakajima, “A fast DCT-SQ scheme for images,” *Trans. IEICE*, vol. 71, no. 11, pp. 1095–1097, 1988.

[14] X. Shao and S. G. Johnson, “Type-II/III DCT/DST algorithms with reduced number of arithmetic operations,” *Signal Processing*, vol. 88, pp. 1553–1564, June 2008.

[15] Malvar 1992

[16] Martucci 1994

[17] S. C. Chan and K. L. Ho, “Direct methods for computing discrete sinusoidal transforms,” in Proc. Inst. Elect. Eng. Radar Signal Process., vol. 137, Dec. 1990, pp. 433–442.

[18] O. Alshibami and S. Boussakta, “Three-dimensional algorithm for the 3-D DCT-III,” in Proc. Sixth Int. Symp. Commun., Theory Applications, July 2001, pp. 104–107.

[19] G. Bi, G. Li, K.-K. Ma, and T. C. Tan, “On the computation of two-dimensional DCT,” IEEE Trans. Signal Process., vol. 48, pp. 1171–1183, Apr. 2000.

[20] E. Feig, “On the multiplicative complexity of discrete \cosine transforms,”IEEE Trans. Inform. Theory, vol. 38, pp. 1387–1390, Aug. 1992.

[21] Nussbaumer, H. J. (1981). *Fast Fourier transform and convolution algorithms* (1st ed.). New York: Springer-Verlag.

## 11 References

- Narasimha, M.; Peterson, A. (June 1978). “On the Computation of the Discrete Cosine Transform”. *IEEE Transactions on Communications.* **26** (6): 934–936. doi:10.1109/TCOM.1978.1094144.
- Makhoul, J. (February 1980). “A fast cosine transform in one and two dimensions”. *IEEE Transactions on Acoustics, Speech, and Signal Processing.* **28** (1): 27–34. doi:10.1109/TASSP.1980.1163351.

- Sorensen, H.; Jones, D.; Heideman, M.; Burrus, C. (June 1987). "Real-valued fast Fourier transform algorithms". *IEEE Transactions on Acoustics, Speech, and Signal Processing*. **35** (6): 849–863. doi:10.1109/TASSP.1987.1165220.
- Arai, Y.; Agui, T.; Nakajima, M. (November 1988). "A fast DCT-SQ scheme for images". *IEICE Transactions*. **71** (11): 1095–1097.
- Plonka, G.; Tasche, M. (January 2005). "Fast and numerically stable algorithms for discrete cosine transforms". *Linear Algebra and its Applications*. **394** (1): 309–345. doi:10.1016/j.laa.2004.07.015.
- Duhamel, P.; Vetterli, M. (April 1990). "Fast fourier transforms: A tutorial review and a state of the art". *Signal Processing*. **19** (4): 259–299. doi:10.1016/0165-1684(90)90158-U.
- Ahmed, N. (January 1991). "How I came up with the discrete cosine transform". *Digital Signal Processing*. **1** (1): 4–9. doi:10.1016/1051-2004(91)90086-Z.
- Feig, E.; Winograd, S. (September 1992). "Fast algorithms for the discrete cosine transform". *IEEE Transactions on Signal Processing*. **40** (9): 2174–2193. doi:10.1109/78.157218.
- Malvar, Henrique (1992), *Signal Processing with Lapped Transforms*, Boston: Artech House, ISBN 0-89006-467-9
- Martucci, S. A. (May 1994). "Symmetric convolution and the discrete sine and cosine transforms". *IEEE Transactions on Signal Processing*. **42** (5): 1038–1051. doi:10.1109/78.295213.
- Oppenheim, Alan; Schafer, Ronald; Buck, John (1999), *Discrete-Time Signal Processing* (2nd ed.), Upper Saddle River, N.J: Prentice Hall, ISBN 0-13-754920-2
- Frigo, M.; Johnson, S. G. (February 2005). "The Design and Implementation of FFTW3" (PDF). *Proceedings of the IEEE*. **93** (2): 216–231. doi:10.1109/JPROC.2004.840301.
- Boussakta, Said.; Alshibami, Hamoud O. (April 2004). "Fast Algorithm for the 3-D DCT-II". *IEEE TRANSACTIONS ON SIGNAL PROCESSING*. **52** (4): 992–1000. doi:10.1109/TSP.2004.823472.
- Cheng, L. Z.; Zeng, Y. H. (2003). "New fast algorithm for multidimensional type-IV DCT". *IEEE TRANSACTIONS ON SIGNAL PROCESSING*. **51** (1): 213–220. doi:10.1109/TSP.2002.806558.

## 12 Further reading

- Wen-Hsiung Chen; Smith, C.; Fralick, S. (September 1977). "A Fast Computational Algorithm for the Discrete Cosine Transform". *IEEE Transactions on Communications*. **25** (9): 1004–1009. doi:10.1109/TCOM.1977.1093941.
- Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007), "Section 12.4.2. Cosine Transform", *Numerical Recipes: The Art of Scientific Computing* (3rd ed.), New York: Cambridge University Press, ISBN 978-0-521-88068-8

## 13 External links

- "discrete cosine transform". *PlanetMath*.
- Syed Ali Khayam: The Discrete Cosine Transform (DCT): Theory and Application
- Implementation of MPEG integer approximation of 8x8 IDCT (ISO/IEC 23002-2)
- Matteo Frigo and Steven G. Johnson: *FFTW*, http://www.fftw.org/. A free (GPL) C library that can compute fast DCTs (types I-IV) in one or more dimensions, of arbitrary size.
- Takuya Ooura: General Purpose FFT Package, http://www.kurims.kyoto-u.ac.jp/~{ }ooura/fft.html. Free C & FORTRAN libraries for computing fast DCTs (types II-III) in one, two or three dimensions, power of 2 sizes.
- Tim Kientzle: Fast algorithms for computing the 8-point DCT and IDCT, http://drdobbs.com/parallel/184410889.
- LTFAT is a free Matlab/Octave toolbox with interfaces to the FFTW implementation of the DCTs and DSTs of type I-IV.

# 14 Text and image sources, contributors, and licenses

## 14.1 Text

## 14.2 Images

## 14.3 Content license