



WIKIPEDIA
The Free Encyclopedia

[Main page](#)
[Contents](#)
[Featured content](#)
[Current events](#)
[Random article](#)
[Donate to Wikipedia](#)
[Wikipedia store](#)

Interaction
[Help](#)
[About Wikipedia](#)
[Community portal](#)
[Recent changes](#)
[Contact page](#)


Tools
[What links here](#)
[Related changes](#)
[Upload file](#)
[Special pages](#)
[Permanent link](#)
[Page information](#)
[Wikidata item](#)
[Cite this page](#)

Print/export
[Create a book](#)
[Download as PDF](#)
[Printable version](#)

Languages 


[العربية](#)
[Català](#)
[Čeština](#)
[Deutsch](#)
[Español](#)
[Euskara](#)
[فارسی](#)
[Français](#)
[한국어](#)
[Italiano](#)
[ಕನ್ನಡ](#)
[Қазақша](#)
[Lietuvių](#)
[Magyar](#)
[Nederlands](#)
[日本語](#)
[Polski](#)
[Português](#)
[Română](#)
[Русский](#)
[Simple English](#)
[Српски / srpski](#)
[Svenska](#)

[ไทย](#)
[Українська](#)
[中文](#)

 [Edit links](#)

[Article](#) [Talk](#)

[Read](#) [Edit](#) [More ▾](#)



Discrete cosine transform

From Wikipedia, the free encyclopedia
(Redirected from [Fast 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](#). 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), where 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.

Contents [hide]

- 1 Applications
 - 1.1 JPEG
- 2 Informal overview
- 3 Formal definition
 - 3.1 DCT-I
 - 3.2 DCT-II
 - 3.3 DCT-III
 - 3.4 DCT-IV
 - 3.5 DCT V-VIII
- 4 Inverse transforms
- 5 Multidimensional DCTs
- 6 Computation
- 7 Example of IDCT
- 8 See also
- 9 Notes
- 10 Citations
- 11 References
- 12 Further reading
- 13 External links

Applications [edit]

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.

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.

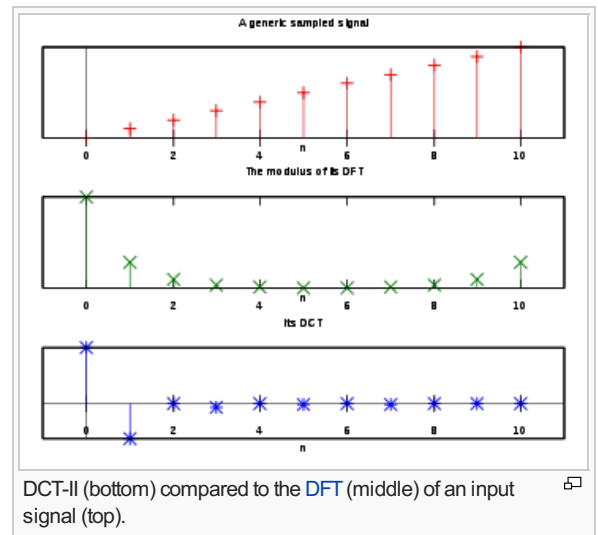
DCTs are also closely related to [Chebyshev polynomials](#), and fast DCT algorithms (below) are used in [Chebyshev](#)

approximation of arbitrary functions by series of Chebyshev polynomials, for example in [Clenshaw–Curtis quadrature](#).

JPEG [edit]

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 8×8 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.



Informal overview [edit]

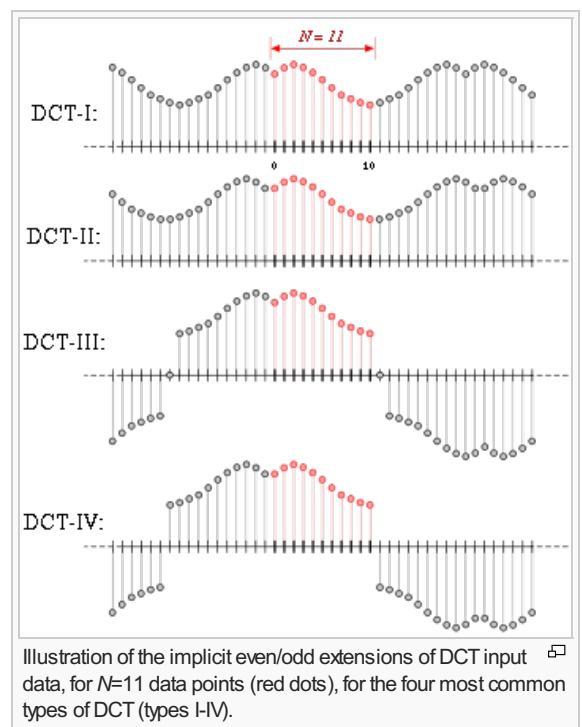
Like any Fourier-related transform, discrete cosine transforms (DCTs) express a function or a signal in terms of a sum of [sinusoids](#) with different [frequencies](#) and [amplitudes](#). Like the [discrete Fourier transform](#) (DFT), a DCT operates on a function at a finite number of discrete data 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](#) than 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).

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.

Formal definition [\[edit\]](#)

Formally, the discrete cosine transform is a [linear](#), invertible [function](#) $f : \mathbb{R}^N \rightarrow \mathbb{R}^N$ (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, \dots, x_{N-1} are transformed into the N real numbers X_0, \dots, X_{N-1} according to one of the formulas:

DCT-I [\[edit\]](#)

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

Some authors further multiply the x_0 and x_{N-1} terms by $\sqrt{2}$, and correspondingly multiply the X_0 and X_{N-1} terms by $1/\sqrt{2}$. This makes the DCT-I matrix [orthogonal](#), if one further multiplies by an overall scale factor of $\sqrt{2/(N-1)}$, 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: x_n is even around $n=0$ and even around $n=N-1$; similarly for X_k .

DCT-II [\[edit\]](#)

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

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 $0 \leq n < N$, $y_{2N} = 0$, and $y_{4N-n} = y_n$ for $0 < n < 2N$.

Some authors further multiply the X_0 term by $1/\sqrt{2}$ and multiply the resulting matrix by an overall scale factor of $\sqrt{2/N}$ (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^[3]), and a scaling that can be chosen that allows the DCT to be computed with fewer multiplications.^{[4][5]}

The DCT-II implies the boundary conditions: x_n is even around $n=-1/2$ and even around $n=N-1/2$; X_k is even around $k=0$ and odd around $k=N$.

DCT-III [\[edit\]](#)

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

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 $\sqrt{2}$ instead of by 2 (resulting in an overall $x_0/\sqrt{2}$ term) and multiply the resulting matrix by an overall scale factor of $\sqrt{2/N}$ (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: x_n is even around $n=0$ and odd around $n=N$; X_k is even around $k=-1/2$ and odd around $k=N-1/2$.

DCT-IV ^[edit]

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

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) (Malvar, 1992).

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

DCT V-VIII ^[edit]

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 (Martucci, 1994) correspond essentially to real-even DFTs of logically odd order, which have factors of $N\pm\frac{1}{2}$ 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 $N=1$.)

Inverse transforms ^[edit]

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 $\sqrt{2}$ (see above), this can be used to make the transform matrix [orthogonal](#).

Multidimensional DCTs ^[edit]

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.

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):

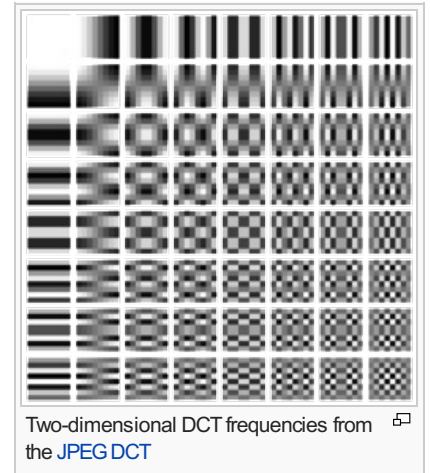
$$\begin{aligned}
 X_{k_1, k_2} &= \sum_{n_1=0}^{N_1-1} \left(\sum_{n_2=0}^{N_2-1} x_{n_1, n_2} \cos \left[\frac{\pi}{N_2} \left(n_2 + \frac{1}{2} \right) k_2 \right] \right) \cos \left[\frac{\pi}{N_1} \left(n_1 + \frac{1}{2} \right) k_1 \right] \\
 &= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n_1,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

Technically, computing a two- (or multi-) dimensional DCT by sequences of one-dimensional DCTs along each dimension is known as a *row-column* algorithm (after the two-dimensional case). 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).

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 image to the right shows combination of horizontal and vertical frequencies for an 8×8 ($N_1 = N_2 = 8$) 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 (8×8) is transformed to a [linear combination](#) of these 64 frequency squares.



Two-dimensional DCT frequencies from the [JPEG DCT](#)

Computation [\[edit\]](#)

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](#)).

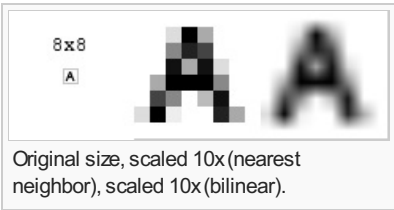
While DCT algorithms that employ an unmodified FFT often have some theoretical overhead compared to the best specialized DCT algorithms, the former also have a distinct advantage: highly optimized FFT programs are widely available. Thus, in practice, it is often easier to obtain high performance for 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 8×8 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^[a]). 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.)

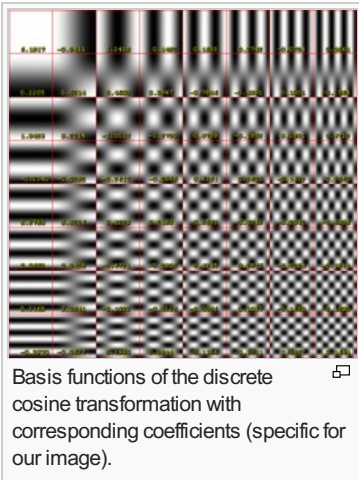
Example of IDCT [\[edit\]](#)

Consider this 8x8 grayscale image of capital letter A.

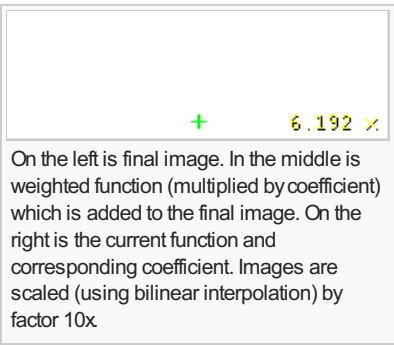


DCT of the image.

6.1917	−0.3411	1.2418	0.1492	0.1583	0.2742	−0.0724	0.0561
0.2205	0.0214	0.4503	0.3947	−0.7846	−0.4391	0.1001	−0.2554
1.0423	0.2214	−1.0017	−0.2720	0.0789	−0.1952	0.2801	0.4713
−0.2340	−0.0392	−0.2617	−0.2866	0.6351	0.3501	−0.1433	0.3550
0.2750	0.0226	0.1229	0.2183	−0.2583	−0.0742	−0.2042	−0.5906
0.0653	0.0428	−0.4721	−0.2905	0.4745	0.2875	−0.0284	−0.1311
0.3169	0.0541	−0.1033	−0.0225	−0.0056	0.1017	−0.1650	−0.1500
−0.2970	−0.0627	0.1960	0.0644	−0.1136	−0.1031	0.1887	0.1444



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



See also [\[edit\]](#)

- [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](#)

Notes [\[edit\]](#)

- ^{**^**} 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.

Citations [\[edit\]](#)

- ^{**^**} ^{**a**} ^{**b**} ^{**c**} [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](#)
- ^{**^**} ^{**a**} ^{**b**} ^{**c**} ^{**d**} ^{**e**} [Rao, K](#); Yip, P (1990), *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Boston: Academic Press, ISBN 0-12-580203-X
- ^{**^**} W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*. New York: Van Nostrand Reinhold, 1993.
- ^{**^**} Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *Trans. IEICE*, vol. 71, no. 11, pp. 1095–1097, 1988.
- ^{**^**} 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.

References [\[edit\]](#)

- 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.
- 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](#) .

Further reading [\[edit\]](#)

- 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





External links [\[edit\]](#)



- [discrete cosine transform](#) at [PlanetMath.org](#).

Wikimedia Commons has

- 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.
- 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.

v · t · e		Data compression methods	[hide]
Lossless	Entropy type	Unary · Arithmetic · Golomb · Huffman (Adaptive · Canonical · Modified) · Range · Shannon · Shannon–Fano · Shannon–Fano–Elias · Tunstall · Universal (Exp-Golomb · Fibonacci · Gamma · Levenshtein)	
	Dictionary type	Byte pair encoding · DEFLATE · Lempel–Ziv (LZ77 / LZ78 (LZ1 / LZ2) · LZJB · LZMA · LZO · LZRW · LZS · LZSS · LZW · LZWL · LZX · LZ4 · Statistical)	
	Other types	BWT · CTW · Delta · DMC · MTF · PAQ · PPM · RLE	
Audio	Concepts	Bit rate (average (ABR) · constant (CBR) · variable (VBR)) · Companding · Convolution · Dynamic range · Latency · Nyquist–Shannon theorem · Sampling · Sound quality · Speech coding · Sub-band coding	
	Codec parts	A-law · μ -law · ACELP · ADPCM · CELP · DPCM · Fourier transform · LPC (LAR · LSP) · MDCT · Psychoacoustic model · WLP	
Image	Concepts	Chroma subsampling · Coding tree unit · Color space · Compression artifact · Image resolution · Macroblock · Pixel · PSNR · Quantization · Standard test image	
	Methods	Chain code · DCT · EZW · Fractal · KLT · LP · RLE · SPIHT · Wavelet	
Video	Concepts	Bit rate (average (ABR) · constant (CBR) · variable (VBR)) · Display resolution · Frame · Frame rate · Frame types · Interlace · Video characteristics · Video quality	
	Codec parts	Lapped transform · DCT · Deblocking filter · Motion compensation	
Theory	Entropy · Kolmogorov complexity · Lossy · Quantization · Rate–distortion · Redundancy · Timeline of information theory		
<div><div> Compression formats ·  Compression software (codecs)</div></div>			

Categories: [Digital signal processing](#) | [Fourier analysis](#) | [Discrete transforms](#)

This page was last modified on 4 September 2015, at 06:25.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

