scipy.fftpack.dct

scipy.fftpack.dct(*x*, *type=2*, *n=None*, *axis=-1*, *norm=None*, *overwrite_x=False*) **[source] (http://github.com/scipy/scipy/blob/v0.14.0/scipy/fftpack/realtransforms.py#L25)**

Return the Discrete Cosine Transform of arbitrary type sequence x.

Parameters:	**x** : *array_like*	
		The input array.
	type : *{1, 2, 3}, optional*	
		Type of the DCT (see Notes). Default type is 2.
	n : *int, optional*	
		Length of the transform.
	axis : *int, optional*	
		Axis over which to compute the transform.
	norm : *{None, 'ortho'}, optional*	
		Normalization mode (see Notes). Default is None.
	overwrite_x : *bool, optional*	
		If True the contents of x can be destroyed. (default=False)
Returns:	**y** : *ndarray of real*	
		The transformed input array.

See also:

idct **(scipy.fftpack.idct.html#scipy.fftpack.idct)** Inverse DCT

Notes

For a single dimension array `x`, `dct(x, norm='ortho')` is equal to MATLAB `dct(x)`.

There are theoretically 8 types of the DCT, only the first 3 types are implemented in scipy. 'The' DCT generally refers to DCT type 2, and 'the' Inverse DCT generally refers to DCT type 3.

Type I

There are several definitions of the DCT-I; we use the following (for `norm=None`):

```
                                       N-2
y[k] = x[0] + (-1)**k x[N-1] + 2 * sum x[n]*cos(pi*k*n/(N-1))
                                       n=1
```

Only None is supported as normalization mode for DCT-I. Note also that the DCT-I is only supported for input size > 1

Type II

There are several definitions of the DCT-II; we use the following (for `norm=None`):

```
          N-1
y[k] = 2* sum x[n]*cos(pi*k*(2n+1)/(2*N)), 0 <= k < N.
          n=0
```

If `norm='ortho'`, `y[k]` is multiplied by a scaling factor *f*:

```
f = sqrt(1/(4*N)) if k = 0,
f = sqrt(1/(2*N)) otherwise.
```

Which makes the corresponding matrix of coefficients orthonormal (`OO' = Id`).

Type III

There are several definitions, we use the following (for `norm=None`):

```
                  N-1
y[k] = x[0] + 2 * sum x[n]*cos(pi*(k+0.5)*n/N), 0 <= k < N.
                  n=1
```

or, for `norm='ortho'` and 0 <= k < N:

```
                                    N-1
y[k] = x[0] / sqrt(N) + sqrt(2/N) * sum x[n]*cos(pi*(k+0.5)*n/N)
                                    n=1
```

The (unnormalized) DCT-III is the inverse of the (unnormalized) DCT-II, up to a factor *2N*. The orthonormalized DCT-III is exactly the inverse of the orthonormalized DCT-II.

References

[R29] 'A Fast Cosine Transform in One and Two Dimensions', by J. Makhoul, *IEEE Transactions on acoustics, speech and signal processing* vol. 28(1), pp. 27-34, http://dx.doi.org/10.1109/TASSP.1980.1163351 (http://dx.doi.org/10.1109/TASSP.1980.1163351) (1980).

[R30] Wikipedia, "Discrete cosine transform", http://en.wikipedia.org/wiki/Discrete_cosine_transform (http://en.wikipedia.org/wiki/Discrete_cosine_transform)

Examples

The Type 1 DCT is equivalent to the FFT (though faster) for real, even-symmetrical inputs. The output is also real and even-symmetrical. Half of the FFT input is used to generate half of the FFT output:

```
>>> fft(array([4., 3., 5., 10., 5., 3.])).real
array([ 30.,  -8.,   6.,  -2.,   6.,  -8.])
>>> dct(array([4., 3., 5., 10.]), 1)
array([ 30.,  -8.,   6.,  -2.])
```


Previous topic

scipy.fftpack.irfft (scipy.fftpack.irfft.html)

Next topic

scipy.fftpack.idct (scipy.fftpack.idct.html)