# **Discrete Fourier Transform ([numpy.fft](https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.fft.html" \l "module-numpy.fft))**

## **Background information**

Fourier analysis is fundamentally a method for expressing a function as a sum of periodic components, and for recovering the function from those components. When both the function and its Fourier transform are replaced with discretized counterparts, it is called the **discrete** **Fourier transform** (DFT).

The DFT has become a mainstay of numerical computing in part because of a very fast algorithm for computing it, called the Fast Fourier Transform (FFT), which was known to **Gauss** (1805) and was brought to light in its current form by Cooley and Tukey. [[NR]](https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.fft.html" \l "nr)provide an accessible introduction to Fourier analysis and its applications.

Because the discrete Fourier transform separates its input into components that contribute at discrete frequencies, it has a great number of applications in digital signal processing, e.g., for filtering, and in this context the discretized input to the transform is customarily referred to as a*signal*, which exists in the*time domain*. The output is called a*spectrum*or*transform*and exists in the*frequency domain*.

## **Implementation details**

There are many ways to define the DFT, varying in the sign of the exponent, normalization, etc. In this implementation, the DFT is defined as



The DFT is in general defined for complex inputs and outputs, and a single-frequency component at linear frequencyis represented by a complex exponential, whereis the sampling interval.

The values in the result follow so-called “standard” order: IfA=fft(a,n), thenA[0]contains the zero-frequency term (the sum of the signal), which is always purely real for real inputs. ThenA[1:n/2]contains the positive-frequency terms, andA[n/2+1:]contains the negative-frequency terms, in order of decreasingly negative frequency. For an even number of input points,A[n/2]represents both positive and negative Nyquist frequency, and is also purely real for real input. For an odd number of input points,A[(n-1)/2]contains the largest positive frequency, whileA[(n+1)/2]contains the largest negative frequency. The routinenp.fft.fftfreq(n)returns an array giving the frequencies of corresponding elements in the output. The routinenp.fft.fftshift(A)shifts transforms and their frequencies to put the zero-frequency components in the middle, andnp.fft.ifftshift(A)undoes that shift.

When the input*a*is a time-domain signal andA=fft(a),np.abs(A)is its amplitude spectrum andnp.abs(A)\*\*2is its power spectrum. The phase spectrum is obtained bynp.angle(A).

The inverse DFT is defined as



It differs from the forward transform by the sign of the exponential argument and the default normalization by.

## **Normalization**

The default normalization has the direct transforms unscaled and the inverse transforms are scaled by. It is possible to obtain unitary transforms by setting the keyword argumentnormto"ortho"(default is*None*) so that both direct and inverse transforms will be scaled by.

## **Real and Hermitian transforms**

When the input is purely real, its transform is Hermitian, i.e., the component at frequencyis the complex conjugate of the component at frequency, which means that for real inputs there is no information in the negative frequency components that is not already available from the positive frequency components. The family of**[rfft](https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.fft.rfft.html" \l "numpy.fft.rfft)**functions is designed to operate on real inputs, and exploits this symmetry by computing only the positive frequency components, up to and including the Nyquist frequency. Thus,ninput points producen/2+1complex output points. The inverses of this family assumes the same symmetry of its input, and for an output ofnpoints usesn/2+1input points.

Correspondingly, when the spectrum is purely real, the signal is Hermitian. The**[hfft](https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.fft.hfft.html" \l "numpy.fft.hfft)**family of functions exploits this symmetry by usingn/2+1complex points in the input (time) domain fornreal points in the frequency domain.

In higher dimensions, FFTs are used, e.g., for image analysis and filtering. The computational efficiency of the FFT means that it can also be a faster way to compute large convolutions, using the property that a convolution in the time domain is equivalent to a point-by-point multiplication in the frequency domain.

## **Higher dimensions**

In two dimensions, the DFT is defined as



which extends in the obvious way to higher dimensions, and the inverses in higher dimensions also extend in the same way.