# Experiment No. 06

### **Name: Rebecca Dias Roll No.: 19 Class: BE CMPN A PID: 182027**

**Aim:** To perform the DFT of a discrete time signal in Python.

# Theory:

## Study about Fourier transform:

(Continuous) Fourier transform

Most often, the unqualified term Fourier transform refers to the transform of functions of a continuous real argument, and it produces a continuous function of frequency, known as a frequency distribution. One function is transformed into another, and the operation is reversible. When the domain of the input (initial) function is time (t), and the domain of the output (final) function is ordinary frequency, the transform of function s(t) at frequency f is given by the complex number:



Evaluating this quantity for all values of f produces the frequency-domain function. Then s(t) can be represented as a recombination of complex exponentials of all possible frequencies:

which is the inverse transform formula. The complex number, S (f), conveys both amplitude and phase of frequency f.

See Fourier transform for much more information, including:

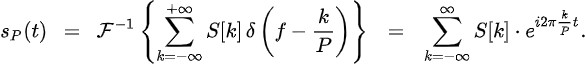
* + conventions for amplitude normalization and frequency scaling/units
  + transform properties
  + tabulated transforms of specific functions
  + an extension/generalization for functions of multiple dimensions, such as images.

## Study about Fourier series:

The Fourier transform of a periodic function, *sP*(*t*), with period *P*, becomes a [Dirac](https://en.wikipedia.org/wiki/Dirac_comb) [comb](https://en.wikipedia.org/wiki/Dirac_comb) function, modulated by a sequence of complex [coefficients](https://en.wikipedia.org/wiki/Coefficients)**:**



The inverse transform, known as Fourier series, is a representation of sP(t) in terms of a summation of a potentially infinite number of harmonically related sinusoids or complex exponential functions, each with an amplitude and phase specified by one of the coefficients:



Any sP(t) can be expressed as a periodic summation of another function, s(t):



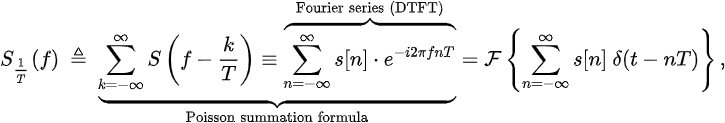
and the coefficients are proportional to samples of S( f ) at discrete intervals of 1/p.



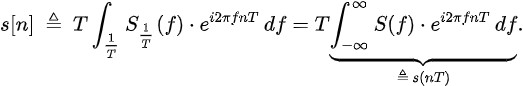
Note that any s(t) whose transform has the same discrete sample values can be used in the periodic summation. A sufficient condition for recovering s(t) (and therefore S( f )) from just these samples (i.e. from the Fourier series) is that the non-zero portion of s(t) be confined to a known interval of duration P, which is the frequency domain dual of the Nyquist–Shannon sampling theorem.

## Study about DFT and DTFT:

**Discrete-time Fourier transform (DTFT]**

The DTFT is the mathematical dual of the time-domain Fourier series. Thus, a convergent periodic summation in the frequency domain can be represented by a Fourier series, whose coefficients are samples of a related continuous time function:

which is known as the DTFT. Thus the DTFT of the s[n] sequence is also the Fourier transform of the modulated Dirac comb function. [B]



Parameter *T* corresponds to the sampling interval, and this Fourier series can now be recognized as a form of the [Poisson summation formula](https://en.wikipedia.org/wiki/Poisson_summation_formula). Thus we have the important result that when a discrete data sequence, *s*[*n*], is proportional to samples of an underlying continuous

function, *s*(*t*), one can observe a periodic summation of the continuous Fourier

transform, *S*(f ). Note that any *s*(*t*) with the same discrete sample values produces the same DTFT But under certain idealized conditions one can theoretically

recover *S*(f ) and *s*(*t*) exactly. A sufficient condition for perfect recovery is that the non-zero portion of *S*(f ) be confined to a known frequency interval of width 1/*T*. When that interval

is [−1/2*T*, 1/2*T*], the applicable reconstruction formula is the [Whittaker–Shannon interpolation](https://en.wikipedia.org/wiki/Whittaker%E2%80%93Shannon_interpolation_formula) [formula.](https://en.wikipedia.org/wiki/Whittaker%E2%80%93Shannon_interpolation_formula) This is a cornerstone in the foundation of [digital signal processing.](https://en.wikipedia.org/wiki/Digital_signal_processing)

Another reason to be interested in *S*1/T( *f* ) is that it often provides insight into the amount of [aliasing](https://en.wikipedia.org/wiki/Aliasing) caused by the sampling process.

Applications of the DTFT are not limited to sampled functions. See Discrete-time Fourier transform for more information on this and other topics, including**:**

* + normalized frequency units
  + windowing (finite-length sequences)
  + transform properties
  + tabulated transforms of specific functions

## Discrete Fourier transform (DFT)

Similar to a Fourier series, the DTFT of a periodic sequence, sN[n], with period N, becomes a Dirac comb function, modulated by a sequence of complex coefficients (see DTFT § Periodic data):

The *S*[*k*] sequence is what is customarily known as the **DFT** of one cycle of *s*N. It is also *N*- periodic, so it is never necessary to compute more than *N* coefficients. The inverse transform, also known as a [discrete Fourier series,](https://en.wikipedia.org/wiki/Discrete_Fourier_series) is given by:



When *s*N[*n*] is expressed as a [periodic summation](https://en.wikipedia.org/wiki/Periodic_summation) of another function:

the coefficients are proportional to samples of *S*1/T( *f* ) at discrete intervals of 1/*P* = 1/*NT*:



Conversely, when one wants to compute an arbitrary number (*N*) of discrete samples of one cycle of a continuous DTFT, *S*1/T( *f* ), it can be done by computing the relatively simple DFT of *s*N[*n*], as defined above. In most cases, *N* is chosen equal to the length of non-zero portion of *s*[*n*]. Increasing *N*, known as *zero-padding* or *interpolation*, results in more closely spaced samples of one cycle of *S*1/T( *f* ). Decreasing *N*, causes overlap (adding) in the time-domain

(analogous [to aliasing](https://en.wikipedia.org/wiki/Aliasing)), which corresponds to decimation in the frequency domain. (see Discrete- time Fourier transform § L=N×I) In most cases of practical interest, the *s*[*n*] sequence represents a longer sequence that was truncated by the application of a finite-length window

function or FIR filter array.

The DFT can be computed using a fast Fourier transform (FFT) algorithm, which makes it a practical and important transformation on computers.

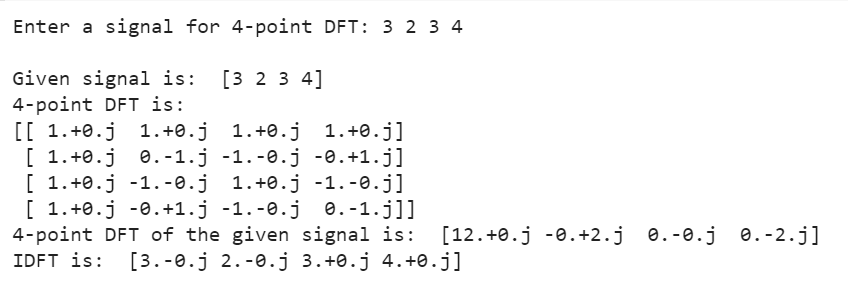
See Discrete Fourier transform for much more information, including:

* transform properties
* applications
* tabulated transforms of specific functions

# Implementation:

Code:

|  |
| --- |
| import numpy as np  from scipy.linalg import dft  def calculate(m,x):    y1= []    y = []    for i in range(len(m)):      prod = 0      for j in range(len(x)):        prod = prod + (m[i][j]\*x[j])      y1.append(prod)    y = np.array(y1)    return y  np.set\_printoptions(precision=2, suppress=True)  x4 = np.array(list(map(int, input("Enter a signal for 4-point DFT: ").split())))  m4 = dft(4)  print('\nGiven signal is: ',x4)  print('4-point DFT is:')  print(m4)  r = calculate(m4,x4)  print('4-point DFT of the given signal is: ',r)  m4inv = np.conj(m4)  print('IDFT is: ',calculate(m4inv,r)/4)  print('\n') |



|  |
| --- |
| m8 = dft(8)  x8 = np.array(list(map(int, input("Enter a signal for 8-point DFT: ").split())))  print('Given signal is: ',x8)  print('8-point DFT is:')  print(m8)  r = calculate(m8,x8)  print('8-point DFT of the given signal is: ',r)  m8inv = np.conj(m8)  print('IDFT is: ',calculate(m8inv,r)/8) |

# 

# Conclusion:

We have successfully studied and implemented the concepts of Fourier Transform, Fourier series and DFT and got the outputs in python.