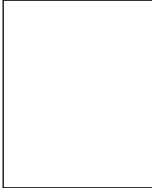


# On Fast Fourier Transform

## A Popular Tool for Spectrum Analysis

*V U Reddy*



V U Reddy is with the Electrical Communication Engineering Department, Indian Institute of Science. His research areas are adaptive signal processing, multirate filtering and wavelets, space-time signal processing for wireless communication.

**Fast Fourier transform (FFT) is an efficient algorithm for computing the discrete Fourier transform. The discovery of the FFT algorithm paved the way for widespread use of digital methods of spectrum estimation which influenced the research in almost every field of engineering and science. In this article, we will first introduce the continuous-time Fourier transform (CFT), discrete-time Fourier transform and discrete Fourier transform (DFT) and then present an example to illustrate the relation between CFT and DFT. In particular, we bring out the fact that the DFT is a tool to estimate the samples of the CFT at uniformly spaced frequencies. Next, we introduce the FFT algorithm giving certain key steps in its development.**

### Fourier Transform

The Fourier transform, named after the French mathematician Jean Baptiste Joseph, Baron de Fourier (1768–1830), is a mathematical tool to convert a time-domain signal into a frequency-domain signal. It has been extensively used by communication engineers, physicists and statisticians. It can simplify the mathematics, and also it makes the phenomenon of interest easier to understand.

Let  $h(t)$  denote a continuous-time signal. If  $h(t)$  satisfies the Dirichlet conditions, its Fourier transform, defined by

$$H \tag{1}$$

exists, where  $\Omega$  represents frequency in radians/second ( $\Omega = 2\pi f$ , with  $f$  denoting the frequency in cycles / second).



Suppose we sample the signal  $h(t)$  with the spacing between any two consecutive samples as  $T$  seconds. Let  $h(nT)$  denote the discrete-time signal obtained by sampling. Here, we assume that  $T$  is chosen in accordance with the sampling theorem which guarantees the recovery of the signal  $h(t)$  from its sampled version. The Fourier transform of the discrete-time signal is given by

$$H(\omega) = \sum_{n=-\infty}^{\infty} h(nT)e^{-j\omega nT} = \sum_{n=-\infty}^{\infty} h(nT)e^{-j\Omega nT} \tag{2}$$

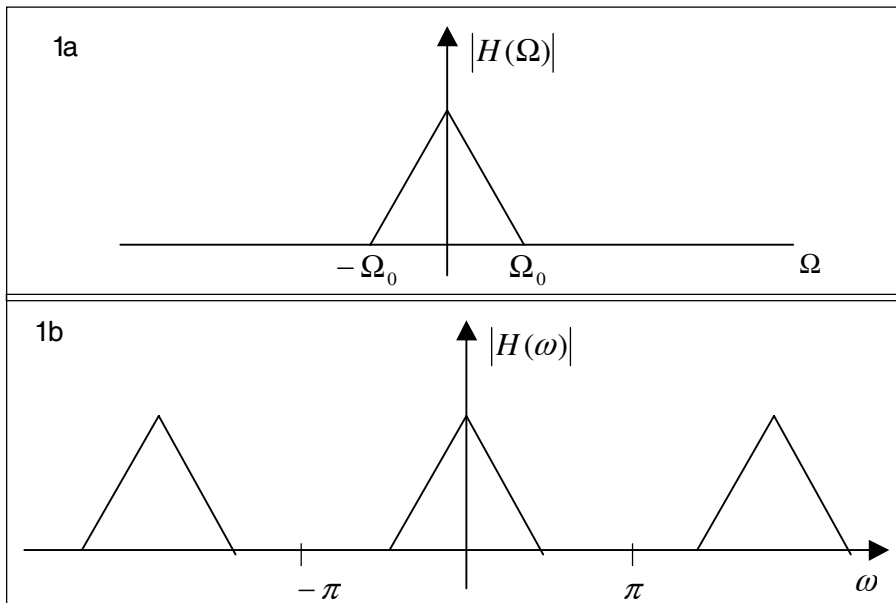
where  $\omega (= \Omega T)$  denotes normalised frequency in radians. It is easy to see from (2) that

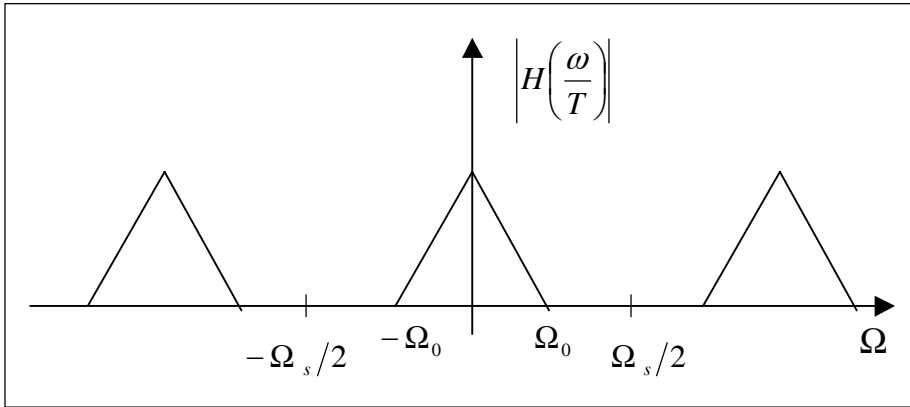
$$H(\omega + 2n\pi) = H(\omega) \tag{3}$$

where  $n$  is any integer. Equation (3) shows that  $H(\omega)$  is a periodic function in  $\omega$  with period  $2\pi$ . We now present the relation between the Fourier transforms of the continuous-time signal and its discrete-time version with an example.

Suppose  $H(\Omega)$  is as depicted in Figure 1a. Then Figure 1b depicts the Fourier transform of the discrete-time signal  $h(nT)$  on

**Figure 1a. (top) Fourier transform of  $h(t)$**   
**Figure 1b. (bottom) Fourier transform of  $h(nT)$  (on normalised frequency scale)**





**Figure 1(c).** Fourier transform of  $h(nT)$  (on absolute frequency scale).

normalized frequency scale while *Figure 1c* shows it on absolute frequency scale. In the representation of *Figures 1b and 1c*, there is an implicit assumption that the sampling frequency  $\Omega_s$  is more than  $2\Omega_0$ .

### Discrete Fourier Transform

We now introduce discrete Fourier transform (DFT). Suppose  $h(nT)$  is a finite-duration sequence of  $N$  samples and we denote it by  $h(nT)$ ,  $0 \leq n \leq N-1$ . (If  $h(nT)$  is an infinite-duration sequence, we can choose  $N$  samples of this by truncating it. We will comment on the implication of this truncation later). Construct a periodic sequence  $\tilde{h}(nT)$  by periodically extending  $h(nT)$  such that

$$\tilde{h}(nT) = h(nT) \quad \text{for } 0 \leq n \leq N-1 \quad (4)$$

where  $k = \dots, -2, -1, 0, 1, 2, \dots$ . We can then use discrete Fourier series representation for  $\tilde{h}(nT)$ , analogous to continuous-time periodic signals. Let  $H(k/NT)$  denote the discrete Fourier series coefficient corresponding to the frequency  $2\pi k/NT$ . Since  $NT$  is the fundamental period, we can write

$$\tilde{h}(nT) = \sum_k H(k/NT) e^{i2\pi k n / N} \quad (5)$$

and

Here we develop the DFT through discrete Fourier series representation.



$$(6)$$

To determine the range of values over which the summations are to be carried out, consider the following. From (6), we note that

$$H((k + N)/NT) = H(k/NT). \tag{7}$$

This suggests that the discrete Fourier series coefficients  $H(k/NT)$  are periodic with period  $N$ . In view of this fact and the relation (4), the summations in (5) and (6) are from 0 to  $N-1$ . Since  $\tilde{h}(nT)$  is periodic with period  $N$ , we choose the first  $N$  samples, i.e., 0 to  $N-1$ , and in view of (4) they are identical to  $h(nT)$ . Thus (5) and (6) can be expressed as

$$, n = 0, 1, \dots, N-1 \tag{8}$$

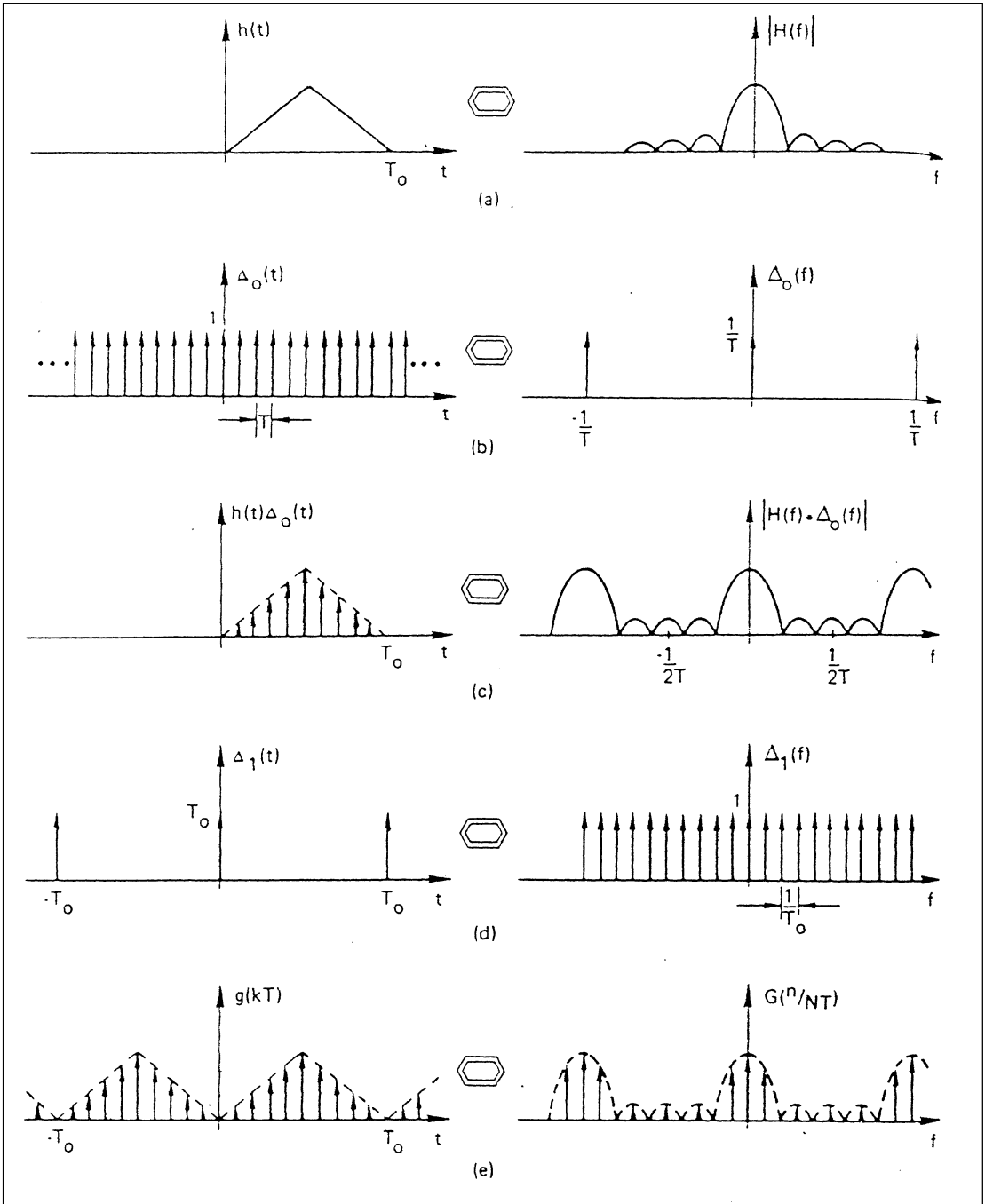
$$H(k/NT) = \sum_{n=0}^{N-1} h(nT) e^{-j2\pi nk/N}, k = 0, 1, \dots, N-1 \tag{9}$$

where we have introduced a normalizing factor  $1/N$  in the relation for  $h(nT)$ . This factor is inserted so that when we substitute for  $H(k/NT)$  in the RHS of (8) and simplify, we get  $h(nT)$ , same as the LHS. Note that this factor can be inserted in the relation for  $H(k/NT)$  instead, or we can insert a factor  $1/\sqrt{N}$  in both the relations.

Equations (8) and (9) constitute the so-called discrete Fourier transform (DFT) pair, and drawing the analogy from the continuous-time Fourier series, we can treat (8) as the synthesis equation and (9) as the analysis equation. We now present some examples to illustrate the various steps involved in going from continuous-time Fourier transform to DFT and how they are related to each other.

Consider *Figure 2* which shows the various steps involved in the case of a time-limited waveform [1]. The left half of the figure





**Figure 2. Discrete Fourier transform of a time-limited waveform. (From E O Breigham, *The Fast Fourier Transform*, Prentice Hall Inc., 1974)**



The DFT is used to estimate the Fourier transform of the given continuous-time signal at uniformly spaced frequencies.

depicts the time-domain signals while the right half gives their corresponding frequency-domain versions. If  $h(t)$  is time limited, its Fourier transform is not band-limited. Consequently, sampling process results in aliasing in the frequency domain. Here, the periodic impulse train shown in *Figure 2b* is the sampling signal. Now, the periodic extension of discrete-time sequence shown in *Figure 2e* can be viewed as the convolution of the signal in *Figure 2c* with that of *Figure 2d*. This operation is equivalent to multiplication of the Fourier transform of the  $h(t)\Delta_0(t)$  with that of  $\Delta_1(t)$  and the result is shown in the right half of *Figure 2e*. The discrete Fourier series coefficients  $G(n/NT)$ , in one period, are the DFT coefficients of  $h(t)\Delta_0(t)$ . Note that the coefficients  $G(n/NT)$ , in one period, provide the estimates of the samples of the Fourier transform of the original continuous-time waveform. How good these estimates are (i.e., how close the coefficients  $G(n/NT)$  are to the samples of the Fourier transform of  $h(t)$ ) depends on several factors, the discussion of which is outside the scope of this article. In any case, the DFT is a tool to estimate the Fourier transform of the given continuous-time signal at uniformly spaced frequencies in the interval  $-\Omega_s/2$  to  $\Omega_s/2$ .

If the given continuous-time signal is not time limited, we need to truncate the discrete-time sequence. This truncation results in smearing of the Fourier transform of the discrete-time sequence and as a result, the DFT coefficients of the resulting finite duration sequence are poorer estimates of the samples of the Fourier transform of the continuous-time signal. The inquisitive reader may refer to figure 6.8 in [1]. We will now introduce the fast Fourier transform algorithm giving certain key steps in its development.

### Fast Fourier Transform

For convenience, we express the finite duration discrete-time sequence and the DFT coefficients given in (8) and (9) as ,

$$h(n) = \frac{1}{N} \sum_{k=0}^{N-1} H(k) W_N^{-nk} , n = 0, 1, \dots, N-1 \quad (10)$$



$$H(k) = \sum_{n=0}^{N-1} h(n)W_N^{nk}, \quad k=0, 1, \dots, N-1 \quad (11)$$

where  $W_N = \exp\left(\frac{-j2\pi}{N}\right)$ . Consider (11). The direct evaluation of (11) (or (10)) requires  $N^2$  multiplications and  $N(N-1)$  additions. If we assume  $N \gg 1$  then the number of additions required can be taken as  $N^2$ . Suppose,  $H(n)$ ,  $n=0, 1, \dots, N-1$ , is a sequence of complex-valued samples. The number of complex multiplications and complex additions grow exponentially with  $N$ . If we wish to compute the DFT in real time, as may be required in some practical applications, we may have to limit the values of  $N$  for want of computational power. However, the underlying application may not permit this. This necessitated the search for techniques to cut down the number of arithmetic operations (additions and multiplications).

Most approaches to improving the efficiency of the computation of the DFT exploits the following special properties of the quantities  $W_N^{kn}$  :

$$W_N^{k(N-n)} = \left(W_N^{kn}\right)^* \quad \text{Property 1}$$

$$W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n} \quad \text{Property 2}$$

Using the first property, the number of multiplications can be reduced by approximately a factor of 2. The second property, which is the periodicity of the complex sequence  $W_N^{kn}$  can be used in achieving significantly greater reductions of the computations.

In 1965, Cooley and Tukey [2] published an algorithm for the computation of the DFT which is applicable when  $N$  is a composite number, i.e., when  $N$  is a product of two or more integers. Subsequently, several computational algorithms have been discovered and all these algorithms have come to be known as fast Fourier transform (FFT) algorithms. We may point out here that though the discovery of FFT algorithm is attributed to



The FFT is not a transform like CFT or DFT; it is a fast algorithm for computing the DFT.

Cooley and Tukey, similar computationally efficient algorithms have been known and used earlier. The interested reader may refer to [2]. We now develop the FFT algorithm for the special case when  $N=2^m$ , where  $m$  is an integer.

Since  $N$  is an even integer, we can decompose the sequence into two subsequences, one consisting of the even numbered points and the other odd-numbered points. Then,  $H(k)$  can be expressed as

$$\begin{aligned}
 H(k) &= \sum_{n=\text{even}} h(n)W_N^{nk} + \sum_{n=\text{odd}} h(n)W_N^{nk} \\
 &= \sum_{r=0}^{\frac{N}{2}-1} h(2r)W_N^{2rk} + \sum_{r=0}^{\frac{N}{2}-1} h(2r+1)W_N^{(2r+1)k} \\
 &= \sum_{r=0}^{\frac{N}{2}-1} h(2r)\left(W_N^2\right)^{rk} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} h(2r+1)\left(W_N^2\right)^{rk} \quad (12)
 \end{aligned}$$

Note that  $W_N^2 = e^{\frac{-j2\pi \cdot 2}{N}} = e^{\frac{-j2\pi}{N/2}} = W_{N/2}$ . Using this in (12) and denoting  $a(r)=h(2r)$  and  $b(r)=h(2r+1)$ , we can express

$$H(k) = \sum_{r=0}^{\frac{N}{2}-1} a(r)W_{N/2}^{rk} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} b(r)W_{N/2}^{rk} = A(k) + W_N^k B(k) \quad (13)$$

where  $A(k)$  and  $B(k)$  are the  $N/2$ -point DFTs of  $a(r)$  and  $b(r)$ . Recall that  $L$ -point DFT is periodic with period  $L$ . Hence, we can obtain the  $N$  DFT coefficients  $H(k)$ ,  $k=0, 1, \dots, N-1$ , from

$$H(k) = A(k) + W_N^k B(k), \quad k = 0, 1, \dots, N/2 - 1 \quad (14a)$$

$$H(k + N/2) = A(k) + W_N^{k+N/2} B(k), \quad k = 0, 1, \dots, N/2 - 1 \quad (14b)$$

To see if we have made any savings by going through this step, consider  $N=32$ . The direct computation requires 1024 additions



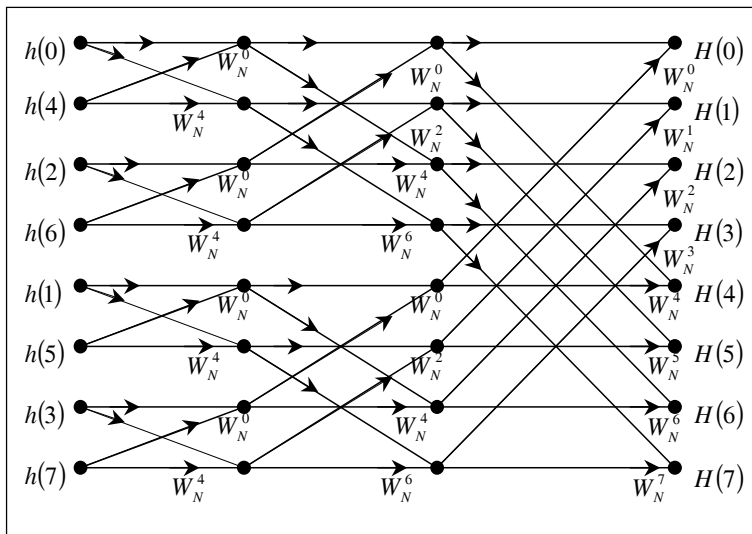
and 1024 multiplications. If we instead compute 16 point DFTs directly and use (14), we need  $256+256+32 = 544$  multiplications and additions. Note the saving we have already made. We can apply the above step to the computation of  $A(k)$  and  $B(k)$ . That is, we divide each subsequence into two, consisting of even-numbered and odd-numbered points. To elaborate this, consider  $N = 8$ . In the first step of the algorithm, given above, we divide  $\{h(0), h(1), \dots, h(7)\}$  into  $\{h(0), h(2), h(4), h(6)\}$  and  $\{h(1), h(3), h(5), h(7)\}$ . In the second step of the algorithm, we decompose the even-numbered subsequence into  $\{h(0), h(4)\}$  and  $\{h(2), h(6)\}$  and the odd-numbered subsequence into  $\{h(1), h(5)\}$  and  $\{h(3), h(7)\}$ . After the second step of the algorithm, the number of arithmetic operations required to compute the  $N$ -point DFT through  $N/4$ -point DFTs and combining them is  $4(64) + 32 + 32 = 320$  for  $N=32$ . Note the reduction from 544 to 320 by applying the above step for the second time. We can continue this until each subsequence is a 2-point sequence. Further decomposition into 1-point sequences does not help since the DFT of a single sample is the sample itself.

*Figure 3* gives the flow graph of 8-point FFT algorithm. This algorithm is known as decimation-in-time FFT since the sequences are decimated in the time domain. In this case, the input time samples are in the so-called bit-reversed order and the output DFT coefficients are in the natural order. We can develop another form by applying the decimation to the DFT coefficients, i.e., applying the decimation to frequency domain samples. The corresponding algorithm is called decimation-in-frequency FFT. In this case, the input time samples are in natural order while the output DFT coefficients are in bit-reversed order.

We note the following from the flow graph of *Figure 3*. The algorithm consists of 3 stages ( $m$  stages when  $N=2^m$ ) and at each stage, we require  $N$  multiplications and  $N$  additions. Thus for  $N=2^m$ , the FFT algorithm requires  $N \log_2 N$  multiplications and the same number of additions. A close observation of the flow graph, however, reveals that the multiplications with  $W_N^0$ ,



**Figure 3. Flow graph of 8-point decimation-in-time FFT algorithm.**



$W_N^4$ ,  $W_N^2$  and  $W_N^6$  for  $N=8$  need not be computed since  $W_N^0 = 1$ ,  $W_N^4 = -1$ ,  $W_N^2 = -j$  and  $W_N^6 = j$ . Thus, a further saving can be achieved in the number of multiplications. Some manipulations are also possible with regard to the additions so that the actual number of multiplications and additions required is less than  $M \log_2 N$ .

Another point to be noted from the flow graph of *Figure 3* is the in-place computation. That is, each stage of computation takes  $N$  complex numbers (real numbers in the first stage only if the input time samples are real valued) and transforms them into another set of  $N$  complex numbers. Thus the FFT algorithm does not require any more storage than what is needed for storing the data. The algorithm outlined above is known as radix-2 FFT algorithm. For various other forms of the algorithm and the form it takes when  $N$  is a prime number, the reader may refer to [2].

*Address for Correspondence*  
 V U Reddy  
 Electrical Communication  
 Engineering  
 Indian Institute of Science,  
 Bangalore 560012, India

**Suggested Reading**

- [1] E O Brigham. *The Fast Fourier Transform*. Prentice-Hall Inc, 1974.
- [2] Selected papers on *Fast Fourier Transform in Digital Signal Processing*. the IEEE Press. Selected Reprint Series, 1972.

