Elec4621:
Signal Processing 2
Chapter 4: The Discrete Fourier Transform

Dr. D. S. Taubman

March 18, 2011

1 Definition

Recall that the Discrete Time Fourier Transform (DTFT) is defined on sequences of infinite extent according to

$$\hat{x}(\omega) = \sum_{n=-\infty}^{\infty} x[n] e^{-jn\omega}, \quad \text{for } -\pi \leq \omega \leq \pi \tag{1}$$

with the inverse DTFT given by

$$x[n] = \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} d\boldsymbol{\omega} \cdot \hat{x}(\boldsymbol{\omega}) e^{jn\omega} \tag{2}$$

Although $x[n]$ is a sequence defined only at integer coordinates, n, its DTFT, $\hat{x}(\omega)$, is a continuous function over $\omega \in [-\pi, \pi]$. Moreover, if $x\,[n]$ is obtained by Nyquist sampling a continuous signal, $x\,(t)$, $\hat{x}\,(\omega)$ is both the DTFT of $x\,[n]$ and the FT (Fourier Transform) of $x\,(t)$.

By contrast, the Discrete Fourier Transform (DFT) involves a finite number of discrete frequencies. It is defined only for sequences, $x[n]$, of finite support, e.g., $0 \leq n < N$, where N is the length of the sequence. It is not necessary for us to start the indices, n, from 0, but this is the most common convention. With this convention, the N-point DFT and its inverse are given by the following equations:

$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi nk}{N}}, \quad 0 \leq k < N \\ x\,[n] &= \frac{1}{N} \sum_{k=0}^{N-1} X\,[k]\, e^{-j\frac{2\pi nk}{N}}, \quad 0 \leq n < N \end{aligned}$$

The complex exponentials involved in the forward and inverse DFT are used so commonly in signal processing that it becomes convenient to define the special symbol,

$$W_N \stackrel{\Delta}{=} e^{j\frac{2\pi}{N}}$$

Noting that $W_N^k = \left(e^{j\frac{2\pi}{N}}\right)^k = e^{j\frac{2\pi k}{N}}$, the DFT and its inverse may be expressed as

$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n] W_N^{-nk} \\ x[n] &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{nk} \end{aligned}$$

Note also that $\left\{W_N^k\right\}_{k=0,1,\ldots,N-1}$ are the N complex roots of unity.

We may also express the DFT in vector notation as follows. Let $\mathbf{w}_k$ denote the vector (sequence) whose elements are

$$w_k[n] = W_N^{nk} = e^{j\frac{2\pi nk}{N}}$$

Then the DFT may be expressed as

$$X[k] = \langle \mathbf{x}, \mathbf{w}_k \rangle$$

and the inverse DFT may be expressed as

$$\mathbf{x} = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \mathbf{w}_k$$

This formulation provides great insight into the DFT operation. It indicates that the vectors, $\mathbf{w}_k$, form a basis for the set of all length N input sequences, $\mathbf{x}$. Moreover, this basis is orthogonal, so that $\langle \mathbf{w}_i, \mathbf{w}_j \rangle = 0$ for $i \neq j$.

It is often convenient to work with orthonormal basis sets. To this end, define

$$\mathbf{w'}_k = \frac{1}{\sqrt{N}} \mathbf{w}_k$$

These form an orthonormal basis for the set of input sequences, as may be seen by rewriting the DFT in a normalized form, with

$$X'[k] = \frac{1}{\sqrt{N}} X[k] = \langle \mathbf{x}, \mathbf{w'}_k \rangle \tag{3}$$

and

$$\begin{aligned} \mathbf{x} &= \sum_{k=0}^{N-1} X'[k] \mathbf{w'}_k \\ &= \sum_{k=0}^{N-1} \langle \mathbf{x}, \mathbf{w'}_k \rangle \mathbf{w'}_k \end{aligned} \tag{4}$$

In this form we can clearly see that the input vector (sequence), $\mathbf{x}$ is just being written as the sum of its projections, $\langle \mathbf{x}, \mathbf{w}'_k \rangle \mathbf{w}'_k$, onto the orthonormal basis vectors, $\mathbf{w}'_k$.

2 Relationship Between DFT and DTFT

The DFT and DTFT are closely related; in fact, under appropriate conditions, they are equivalent, which is the foundation for much of the usefulness of the DFT. Let $x[n]$ denote a finite support sequence, which is equal to zero everywhere except in the region, $0 \leq n < N$. The N-point DFT of the finite sequence, restricted to this region may be written

$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi nk}{N}} \\ &= \left(\sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \right) \Bigg|_{\omega=\frac{2\pi}{N}k} \end{aligned} \tag{5}$$

So the DFT, $X[k]$, is a sampling of the spectrally continuous DTFT, $\hat{x}(\boldsymbol{\omega})$ at the locations $\omega = \frac{2\pi}{N}k$.

Although this relationship is mathematically correct, it violates the convention that $\hat{x}(\omega)$ is defined only on $-\pi \leq \omega \leq \pi$, which is required to impart the correct interpretation to discrete frequency[1]. To correct this problem, define

$$x \bmod 2\pi \triangleq x - 2\pi \left\lfloor \frac{x}{2\pi} + \frac{1}{2} \right\rfloor$$

where $\lfloor x \rfloor$ is the floor function, rounding its argument down to the largest integer, no larger than x. Note that $-\pi \leq x \bmod 2\pi < \pi$. Now, since $x \bmod 2\pi$ differs from x by a multiple of 2π, the mathematical equivalence in equation (5) also gives

$$X[k] = \left(\sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \right) \Bigg|_{\omega=\left(\frac{2\pi}{N}k\right) \bmod 2\pi} = \hat{x}(\omega)\big|_{\omega=\left(\frac{2\pi}{N}k\right) \bmod 2\pi} \tag{6}$$

The fact that the DFT is a sampled version of the DTFT and that the DFT is invertible means that this sampled representation must contain all the information in the spectrally continuous DTFT, provided of course, that the signal is zero outside the region over which the DFT is taken. In the general case, where $x[n]$ has infinite support, we can still sample the spectrally continuous

[1] The interpretation of frequency in the discrete domain derives from the fact that the FT and the DTFT are identical for Nyquist band-limited signals. Although some people prefer to define the DTFT as having a periodically repeating spectrum, this confuses the relationship between discrete and continuous signals – it requires the introduction of impulse sequences and other mathematical anomalies that are best avoided.

DTFT, $\hat{x}(\omega)$, and associate these samples with a hypothetical DFT, defined by $X'[k] = \hat{x}(\omega)|_{\omega=\left(\frac{2\pi}{N}k\right) \bmod 2\pi}$. In this case, however, the inverse DFT of $X'[k]$ is given by

$$x'[n] = \sum_m x[n+mN], \quad 0 \le n < N$$

This is a time-domain aliasing relationship. It is the analog of frequency aliasing which occurs when a signal is sampled too coarsely in the time domain; in this case, the aliasing occurs when a signal is sampled too coarsely in the frequency domain. Time-domain aliasing can easily occur if we are not careful when using the DFT (usually through an FFT implementation) to implement filtering operations, as we shall see shortly.

3 Filtering with the DFT

The DFT provides an efficient tool for implementing FIR digital filters. This fact is a direct consequence of the relationship between the DFT and the DTFT. Let $h[n]$ denote the impulse response of a causal FIR filter of order M. Thus $h[n]$ has region of support $0 \le n \le M$. We would like to filter an input signal, $x[n]$, to obtain

$$y[n] = \sum_{m=0}^{M} h[m]x[n-m]$$

To begin, we shall suppose that the input signal, $x[n]$, also has finite support with N_x samples, given by $0 \le n < N_x$. It follows, that after filtering, the output signal, $y[n]$, will be zero outside the region $0 \le n < N_x + M$. Thus, selecting

$$N = N_x + M$$

we see that both $x[n]$ and $y[n]$ are zero outside the region $0 \le n < N$, meaning that their N-point DFT's are sampled copies of their respective DTFT's. The DFT/DTFT equivalence relationship of equation (6) may then be invoked, yielding

$$\begin{aligned} Y[k] &= \hat{y}(\omega)|_{\omega=\left(\frac{2\pi}{N}k\right) \bmod 2\pi} \\ &= \left[\hat{x}(\omega)\,\hat{h}(\omega)\right]_{\omega=\left(\frac{2\pi}{N}k\right) \bmod 2\pi} \\ &= X[k] \times \left[\hat{h}(\omega)\right]_{\omega=\left(\frac{2\pi}{N}k\right) \bmod 2\pi} \\ &= X[k]\,H[k], \quad 0 \le k < N \end{aligned}$$

Notice that we have not appealed to any specific properties of the DFT itself, other than its relationship to the DTFT. It is the DTFT's convolution theorem which is at work here. We will discuss a separate, related convolution theorem for the DFT which applies to "circular" convolution; however, it is important to distinguish between circular convolution and true convolution – the latter is usually the goal in filtering applications.

3.1 Overlap-Add-Save Method

We have shown that true filtering of a finite length sequence, $x[n]$, by a finite support filter, $h[n]$, is equivalent to multiplication of their N-piont DFT's, so long as N is at least as large as the sum of the filter order, M, and the input sequence length, N_x. In Section 5, we will demonstrate a fast implementation for the DFT (known as the FFT) which allows an N-point DFT to be computed using only $N \log_2 N$ real-valued multiplications. These two observations are the basis of a number of related strategies for implementing large FIR filters efficiently.

The DFT size, N, is generally limited by practical considerations such as delay, memory and (to a lesser extent) computation, while the input signals of interest often have unbounded length. In this section, we describe the overlap-add-save method for filtering an unbounded input sequence, $x[n]$, using bounded DFT operations.

The DFT size, N, is fixed, and the input signal is broken into finite length segments, $x_p[n]$, each with $N_x = N - M$ samples. Specifically,

$$x_p[n] = \begin{cases} x[n + pN_x] & 0 \le n < N_x \\ 0 & \text{otherwise} \end{cases}$$

Evidently, $x[n]$ is the sum of its translated segments,

$$x[n] = \sum_p x_p[n - pN_x] \tag{7}$$

This segmented representation of $x[n]$ is illustrated in Figure 1.

From our earlier discussion, we know that each segment, $x_p[n]$, may be filtered using the N-point DFT, yielding filtered output $y_p[n]$ with

$$Y_p[k] = X_p[k] H[k], \quad 0 \le k < N$$

From equation (7), together with linearity and time invariance of filtering, we see that the complete filtered signal may be formed as a sum of the translated, filtered segments as

$$y[n] = \sum_p y_p[n - pN_x]$$

Importantly, the support of each filtered segment, $y_p[n]$, is not restricted to the same interval, $0 \le n < N_x$, but occupies all N samples. Thus, the final M samples from each filtered output segment overlap the first M samples from the next filtered output segment, as shown in Figure 1. The overlapping contributions must be added.

The complete filtering procedure may be summarized as follows:

1. Determine the segment length, N_x, from $N_x = N - M$.
2. For each input segment, $x_p[n]$,

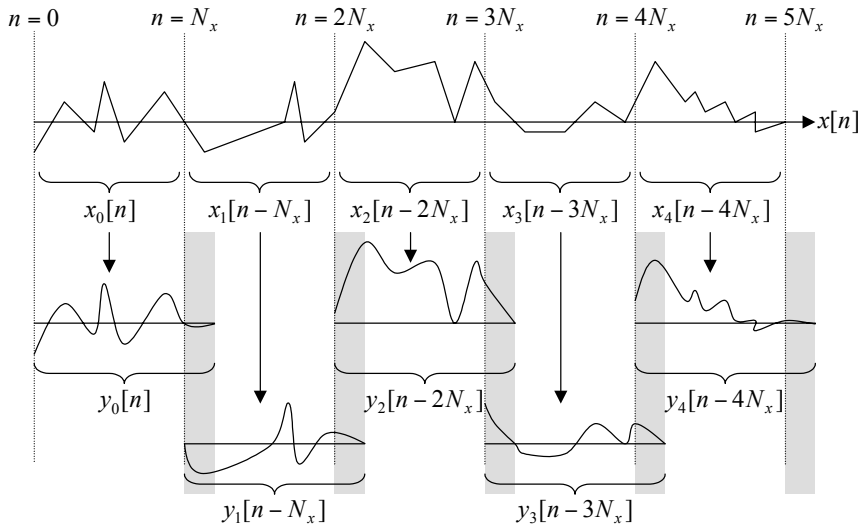

Figure 1: *Segmentation of an unbounded input signal, $x[n]$, into segments of length N_x, for DFT-based filtering using the overlap-add-save method. The overlapping portions of the filtered output sections are identified by the shaded bars.*

(a) Compute the N-point DFT, $X_p[k]$. Note that the last M samples of $x_p[n]$ are always zero; it has at most N_x non-zero samples.

(b) Find $Y_p[k]$ from $Y_p[k] = X_p[k] \times H[k]$. The factors, $H[k]$, are fixed for all segments and generally computed ahead of time.

(c) Find $y_p[n]$ by taking the inverse DFT of $Y_p[k]$.

(d) Add the last M samples of $y_{p-1}[n]$ (saved in the previous segment's "save" step) to the first M samples of $y_p[n]$, yielding the modified filtered output, $y'_p[n]$.

(e) Output the first N_x samples of $y'_p[n]$, saving the remaining M samples in a temporary buffer for use in step (2.d) when processing the next segment.

Evidently, the algorithm calls for an M-sample save buffer, in addition to an N-sample buffer for performing the various DFT operations.

To understand the computational advantages of DFT-based filtering, observe that the algorithm generates N_x new samples after processing each segment. To generate these samples, an N-point DFT and an N-point IDFT must be performed, having a combined cost of $2N \log_2 N$ real-valued multiplications (see Section 5). Multiplication of $X_p[k]$ by $H[k]$ need only be performed over half of the DFT coefficients, due to conjugate symmetry, $Y_p[k] = Y_p^*[N-k]$, for a total

cost of $N/2$ complex-valued multiplications, or $2N$ real-valued multiplications. The total number of multiplications[2] per output sample is then

$$\frac{2N \log_2 2N}{N_x} = \frac{2N}{N-M} \log_2 2N$$

Example 1 *A 1024-point DFT/IDFT processor is available for use in a filtering application, involving an order $M = 128$ FIR filter. The total number of multiplications required per output sample is*

$$\frac{2048}{1024-128} \log_2 2048 = 25.14$$

By constrast, direct application of the filter requires 128 multiplications per output sample.

Some consideration should confirm the fact that DFT-based techniques are most interesting when working with very high order FIR filters. As an example, to simulate the acoustic environment of a typical music hall might require filters with many thousands of taps. In this case, the potential computational savings from a DFT-based implementation can be enormous.

4 Properties of the DFT

Conjugate Symmetry: When the input sequence, $x[n]$, is real-valued, as is almost invariably the case for image processing applications,

$$X[k] = X^*[N-k]$$

where the superscript, *, denotes complex conjugation as always.

Circular Convolution: Recall that the DFT is defined only on $0 \le n < N$. The circular convolution of two sequences, $x[n]$ and $h[n]$, on this interval is defined by

$$\begin{aligned} y[n] &= \sum_{k=0}^{N-1} x\,[k]\, h\,[(n-k) \bmod N] \\ &= \sum_{k=0}^{N-1} h\,[k]\, x\,[(n-k) \bmod N] \end{aligned}$$

In the DFT domain, circular convolution is always equivalent to multiplication, i.e.

$$Y[k] = X[k] \cdot H[k]$$

but the conditions under which circular convolution and true convolution are identical are those discussed in Section 3.

[2] A similar number of additions is required, but multiplications are generally more costly than additions, as discussed in the previous lecture.

Dual of Convolution: Suppose we multiply two sequences, $x[n]$ and $y[n]$, in the time domain to obtain $u[n] = x[n] \cdot y[n]$. In the DFT domain, this operation is eqivalent to circular convolution. Specifically,

$$U[k] = \frac{1}{N} \sum_{p=0}^{N-1} X[p]\, Y[(k-p) \bmod N] \tag{8}$$

Energy: Parseval's relation states, essentially, that the energy or power in the time domain is equal to the energy in the Fourier domain. For the DFT, Parseval's relation is

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2$$

This may be derived easily from the normalized vector space interpretation of the DFT and its inverse in equations (3) and (4).

Extended Parseval Relationship: An extended form of Parseval's relationship, for two arbitrary sequences, $x[n]$ and $y[n]$, states that

$$\sum_{n=0}^{N-1} x[n]\, y^*[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]\, Y^*[k]$$

5 Fast Algorithms for the DFT

Although fast DFT algorithms can be developed for arbitrary N, the savings are greatest and the development simplest when N is a power of 2. For this reason, let $N = 2^r$ for some r. The basic FFT (Fast Fourier Transform) algorithm is based on the observation that $X[k]$ may be written as

$$\begin{aligned} X[k] &= \sum_{n=0}^{2^r-1} x[n] \exp\left(\frac{-j2\pi kn}{2^r}\right) \\ &= \sum_{n=0}^{2^{r-1}-1} x[2n] \exp\left(\frac{-j2\pi kn}{2^{r-1}}\right) \\ &+ \sum_{n=0}^{2^{r-1}-1} x[2n+1] \exp\left(\frac{-j2\pi kn}{2^{r-1}}\right) \cdot \exp\left(\frac{-j2\pi k}{2^r}\right) \end{aligned}$$

Now let $x_e[n] = x[2n]$ denote the even sub-sequence of $x[n]$ and let $x_o[n] = x[2n+1]$ denote the odd sub-sequence. The DFT of each of these sub-sequences, $X_e[k]$ and $X_o[k]$, is defined for $0 \le k < 2^{r-1}$. The above formula becomes

$$X[k] = \begin{cases} X_e[k] + W_{2^r}^{-k} X_o[k] & \text{for } 0 \le k < 2^{r-1}, \\ X_e[k-2^{r-1}] + W_{2^r}^{-k} X_o[k-2^{r-1}] & \text{for } 2^{r-1} \le k < 2^r. \end{cases}$$

where we have used the notation,

$$W_N = e^{j\frac{2\pi}{N}}$$

introduced previously.

Finally, noting that $W_{2^r}^k = -W_{2^r}^{2^{r-1}+k}$, we may express the 2^r-point DFT of $x[n]$ in terms of the 2^{r-1}-point DFT's of its even and odd sub-sequences as follows

$$\begin{pmatrix} X[k] \\ X\left[k+2^{r-1}\right] \end{pmatrix} = \begin{pmatrix} X_e[k] + W_{2^r}^{-k} X_o[k] \\ X_e[k] - W_{2^r}^{-k} X_o[k] \end{pmatrix}, \quad 0 \le k < 2^{r-1} \tag{9}$$

Now suppose that a 2^r-point FFT requires C_r operations per sample, or a total of $2^r C_r$ opreations. The following recursive description of the algorithm allows us to compute its complexity in a straightforward manner:

1. Compute $X_e[k]$ and $X_o[k]$ for $0 \le k < 2^{r-1}$. The number of operations required for this step is $2^r C_{r-1}$.

2. Multiply $X_o[k]$ by $W_{2^r}^{-k}$ for $0 \le k < 2^{r-1}$, to obtain $X_0'[k]$. We assume that the factors, $W_{2^r}^{-k}$, have been computed and stored in advance, which is reasonable because they do not depend upon the signal, $x[n]$. This step requires 2^{r-1} complex-valued multiplications.

3. For $0 \le k < 2^{r-1}$ compute $X[k] = X_e[k] + X_o'[k]$ and $X\left[k+2^{r-1}\right] = X_e[k] - X_o'[k]$. This step requires 2^r complex-valued additions/subtractions[3].

In terms of complex-valued multiplications, the operation count thus satisfies

$$\begin{aligned} 2^r C_r &= 2^r C_{r-1} + \frac{1}{2} 2^r \\ &= 2^r C_{r-2} + \frac{1}{2} 2^r + \frac{1}{2} 2^r \\ &\vdots \\ &= \frac{r}{2} 2^r \end{aligned}$$

The number of complex-valued additions is twice as large, i.e. $r2^r$. These complexities are usually expressed in terms of the dimension, $N = 2^r$. So that the number of complex-valued multiplications required to compute the N-point DFT using this simple FFT algorithm is $\frac{N}{2}\log_2(N)$, while the number of complex-valued additions is $N\log_2(N)$. Compare this with the direct computation of the FFT which requires N^2 complex-valued multiplications and additions.

[3]Since the complexity of additions is identical to the complexity of subtractions, we lump them together and generally refer to both as additions.

5.1 Real-Valued Sequences

In the above formulation, the input sequence, its DFT and all numerical operations are complex-valued. In practice, we normally work with real-valued input sequences, $x[n]$, whose DFT satisfies the conjugate symmetry property, $X[k] = X^*[N-k]$. We can exploit this symmetry to reduce the overall complexity.

One simple way to do this is to observe that at each stage of the FFT algorithm we need only compute $X[k]$ over the range $0 \le k < \frac{N}{2}$, using values of $X_e[k]$ and $X_o[k]$ in the range $0 \le k < \frac{N}{4}$. The relevant update relationships are readily derived from equation (9) as

$$\begin{aligned}\begin{pmatrix} X[k] \\ X\left[2^{r-1}-k\right] \end{pmatrix} &= \begin{pmatrix} X_e[k] + W_{2^r}^{-k} X_o[k] \\ X_e^*[k] - W_{2^r}^{k} X_o^*[k] \end{pmatrix} \\ &= \begin{pmatrix} X_e[k] + W_{2^r}^{-k} X_o[k] \\ X_e^*[k] - \left(W_{2^r}^{-k} X_o[k]\right)^* \end{pmatrix}^*, \quad 0 \le k < 2^{r-2}\end{aligned}$$

The complexity of these calculations is identical to that of the full complex-valued calculations described previously, except that there are only half as many of them. Finally, observing that a complex-valued multiplication requires four real-valued multiplications and a complex-valued addition requires two real-valued additions, we find that the complexity of an N-point real-valued FFT is

$$\begin{aligned} &N \log_2 N \text{ real-valued multiplications, and} \\ &N \log_2 N \text{ real-valued additions}\end{aligned}$$

It is worth noting that there are many variations on the basic FFT algorithm, as described here. In fact, there is at least one entire book, which deals exclusively with fast Fourier transform algorithms. Common to all these algorithms is the fact that the complexity of an N-point DFT has order $N \log_2 N$.