1 Lecture Outline

Reading: Chapter 9 DFT/FFT Algorithms

This lecture will cover the following topics

- Derivation of the Radix-2 FFT Algorithm
- FFT Input/Output Data Index Bit Reversal
- Computational Complexity

2 Derivation of the Radix-2 FFT Algorithm

We begin with the N-point DFT of a length $L = N$ signal (and require N to be a power of two)

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}, \quad 0 \le k \le N-1. \tag{1}$$

Note that each $X(k)$ requires N complex multiplications and $N-1$ complex additions. To compute the complete N-point DFT requires N^2 complex multiplications and $N(N-1)$ complex additions. We'll assume multiplications require more resources and simply say the N-point DFT requires N^2 complex multiplications.

First, split the summation into a sum over the even-indexed samples and a sum over the odd-indexed samples

$$\begin{aligned} X(k) &= \sum_{n=0}^{N/2-1} x(2n)e^{-j2\pi k(2n)/N} + \sum_{n=0}^{N/2-1} x(2n+1)e^{-j2\pi k(2n+1)/N} \\ &= \sum_{n=0}^{N/2-1} x(2n)e^{-j2\pi k(2n)/N} + e^{-j2\pi k/N}\sum_{n=0}^{N/2-1} x(2n+1)e^{-j2\pi k(2n)/N} \end{aligned} \tag{2}$$

In order to simplify the notation we define the *twiddle factors*

$$W_N \equiv e^{-j2\pi/N} \tag{3}$$

so that (2) is simply rewritten as

$$X(k) = \sum_{n=0}^{N/2-1} x(2n)W_N^{2kn} + W_N^k \sum_{n=0}^{N/2-1} x(2n+1)W_N^{2kn}, \quad 0 \le k \le N-1. \tag{4}$$

Second, we note that

$$W_N^{2kn} = e^{-j2\pi(2kn)/N} = e^{-j2\pi(kn)/(N/2)} = W_{N/2}^{kn} \tag{5}$$

so that (4) becomes

$$X(k) = \sum_{n=0}^{N/2-1} x(2n)W_{N/2}^{kn} + W_N^k \sum_{n=0}^{N/2-1} x(2n+1)W_{N/2}^{kn}, \quad 0 \le k \le N-1. \tag{6}$$

Now we have two $N/2$ summations whose results can be combined to give us the N-point DFT. The equation above is computed for $0 \le k \le N-1$, however, let's consider just the upper half of the frequency points:

$$X(k+N/2) = \sum_{n=0}^{N/2-1} x(2n)W_{N/2}^{n(k+N/2)} + W_N^{(k+N/2)} \sum_{n=0}^{N/2-1} x(2n+1)W_{N/2}^{n(k+N/2)}, \quad 0 \le k \le N/2-1. \tag{7}$$

We have

$$W_{N/2}^{n(k+N/2)} = e^{-j2\pi kn/(N/2)} e^{-j2\pi n(N/2)/(N/2)} = e^{-j2\pi kn/(N/2)} = W_{N/2}^{kn} \tag{8}$$

and

$$W_N^{(k+N/2)} = e^{-j2\pi k/N} e^{-j2\pi(N/2)/N} = e^{-j2\pi k/N}(-1) = -W_N^k. \tag{9}$$

Thus, using (8) and (9) in (7), the upper half of the frequency points are given by

$$X(k+N/2) = \sum_{n=0}^{N/2-1} x(2n) W_{N/2}^{kn} - W_N^m \sum_{n=0}^{N/2-1} x(2n+1) W_{N/2}^{kn}, \quad 0 \le k \le N/2-1. \tag{10}$$

Comparing the DFT for the lower half of the frequency points (6) and the DFT for the upper half of the frequency points (10), we see only a sign change in the second summation term.

We have a new strategy: compute an $N/2$-point DFT using the even-indexed samples and an $N/2$-point DFT using the odd-indexed samples. Add the DFTs together to get $X(k)$ for $0 \le k \le N/2-1$ and take the difference to get $X(k)$ for $N/2 \le k \le N-1$.

By doing this we can reuse calculations resulting in computation of the N-point DFT using two $N/2$-point DFTs. This requires $2 \cdot (N/2)^2 = N^2/2$ complex multiplications which results in a savings of 50%. The algorithm is illustrated below.

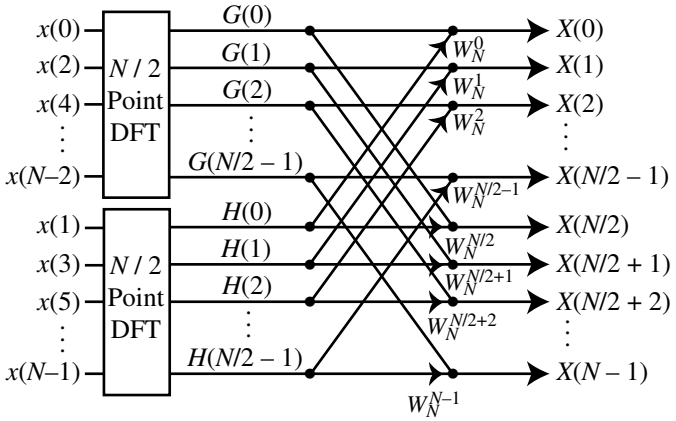

Figure 1: p. 508 Figure 9.8.2, N-point DFT as two $N/2$-point DFTs.

The next obvious step is to compute each of the $N/2$-point DFTs with a pair of $N/4$-point DFTs using the appropriate input samples. This process can be continued until we are left with an array of 2-point DFTs and a *butterfly* pattern of signal flows as shown below. This is why the number of points in our FFTs are constrained to be some power of 2 and why the FFT algorithm is referred to as the *radix-2 FFT*.

3 FFT Input/Output Data Index Bit Reversal

The algorithm presented is known as the *decimation-in-time, radix 2 FFT*. The DIT phrase refers to how we broke the DFT input samples into even and odd parts. This time decimation leads to the scrambled order of the input data's index n in Figure 3. This pattern can be understood with a bit-reversed address table.

Note that the algorithm presented has bit-reversed input samples and leads to a normally-ordered output frequency points. There are simple adjustments that can be made that lead to an algorithm with normally-ordered input samples and bit-reversed output frequency points.

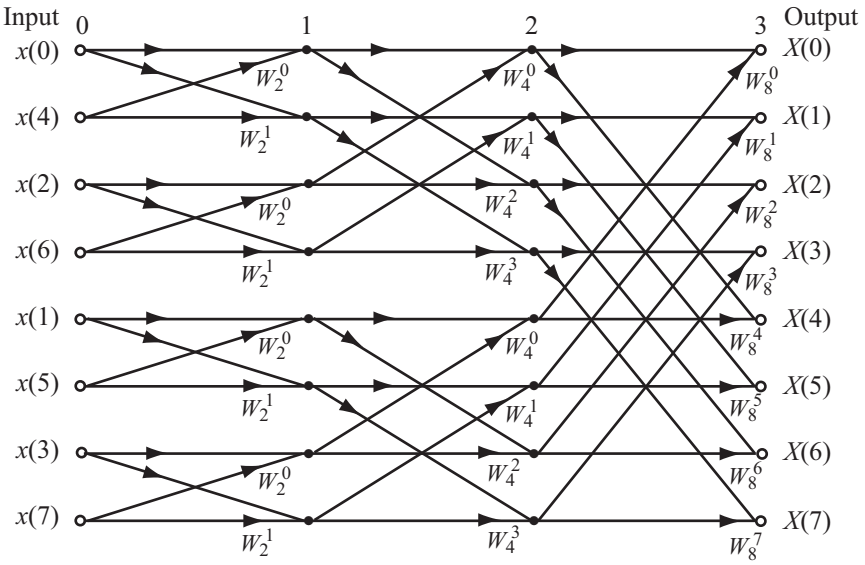

Figure 2: p. 510 Figure 9.8.4, N-point DFT as consecutive stages of $N/2$-point DFTs.

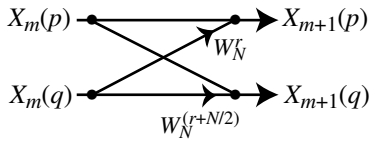

Figure 3: basic butterfly unit

Figure 4: Bit-reversed address table

4 Computational Complexity

As shown before, the N-point DFT requires N^2 complex multiplications. AS it turns out the FFT requires $\frac{N}{2}\log_2 N$ complex multiplications which can be a huge savings when N is large. For example when $N = 512$, the DFT requires $200\times$ more complex multiplications than those needed by the FFT. When $N = 8192$, the factor is $1000\times$.