Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.341: Discrete-Time Signal Processing

OpenCourseWare 2006

Lecture 19
FFT Algorithms

Reading: Sections 9.1, 9.3 and 9.4 in Oppenheim, Schafer & Buck (OSB).

The DFT of a finite-length sequence of length N is

$$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \tag{1}$$

where $W_N = e^{-j2\pi/N}$. The IDFT is

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

The DFT and the IDFT involve same operations except the sign of the exponent and the scale factor, so we will only focus on the computation of the DFT, and it is straightforward to apply the same analysis to IDFT.

When $x[n]$ is complex, the term $(x[n]W_N^{nk})$ requires one complex multiplication, which is equivalent to 4 real multiplications and 2 real additions. In order to compute $X[k]$ for $k = 0, 1, \ldots,$ $N-1$, we need N^2 complex multiplications and $N(N-1) \approx N^2$ complex additions.

Here are some approaches to reduce computation:

- Don't do multiplies by unity.
- If $x[n]$ is real, we can calculate two transforms at one, which is illustrated in OSB Problem 9.31.
- Use the Goertzel algorithm (covered in the next lecture) which reduces the number of multiplications by 2.

However, in all of the above cases, the number of multiplications and additions (MAD) is still proportional to N^2.

FFT Algorithms

In general, when we use FFT algorithms to compute an N-point DFT, the number of complex MAD's is proportional to $N(p_1 + p_2 + \ldots + p_v)$, where $N = p_1 \cdot p_2 \cdot \cdots \cdot p_v$.

Example:

> $N = 2^v \quad \Rightarrow \quad p_1 = p_2 = \ldots = p_v = 2$
> Number of MAD's : proportional to $N \cdot 2v \propto N \log_2 N$.
> If $N = 2^{10} = 1024$, without using FFT algorithms, number of MAD's to compute the N-point DFT is proportional to $N^2 \approx 10^6$, while the FFT needs MAD's proportional to $N \log_2 N \approx 10^4$.

In this lecture, instead of discussing many variations of FFT algorithms in detail, we will focus on the two basic schemes and only consider the case that N is an integer power of 2, i.e. $N = 2^v$. For the details of the derivation and various alternative forms, see OSB Sections 9.3 and 9.4.

Decimation-in-Frequency

Consider computing separately the even and odd numbered frequency samples.

For the even numbered samples:

$$\begin{aligned} X[2r] &= \sum_{n=0}^{N-1} x[n] W_N^{n(2r)} \\ &= \sum_{n=0}^{(N/2)-1} \{x[n] + x[n + \frac{N}{2}]\} W_{N/2}^{nr}, \qquad r = 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{2}$$

This is equivalent to adding the first half and the last half of the input sequence, and taking the $(N/2)$-point DFT. Adding the two halves of the sequence corresponds to time aliasing, which is reasonable because we are undersampling in the frequency domain when computing only the even-numbered frequency samples.

Similarly, for the odd-numbered samples:

$$\begin{aligned} X[2r+1] &= \sum_{n=0}^{N-1} x[n] W_N^{n(2r+1)} \\ &= \sum_{n=0}^{(N/2)-1} \{x[n] - x[n + \frac{N}{2}]\} W_N^{n} W_{N/2}^{nr}. \end{aligned} \tag{3}$$

This is equivalent to the $(N/2)$-point DFT of $(x[n] - x[n + \frac{N}{2}])W_N^n$.

The procedure suggested by Equation 2 and 3 is illustrated in OSB Figure 9.17 for the case $N = 8$.

Recall that N^2 complex multiplications were required to compute the N-point DFT using Equation 1. By comparison, using the procedure in the figure above, we need two (N/2)-point DFTs and additional (N/2) complex multiplications. Therefore, the number of multiplications required to compute the N-point DFT using Equation 2 and 3 is

$$2 \cdot (\frac{N}{2})^2 + \frac{N}{2},$$

which is smaller than N^2. The number of additions is also reduced to approximately $2 \cdot (\frac{N}{2})^2 + N$.

Similarly, we can compute the (N/2)-point DFTs using the (N/4)-point DFTs as illustrated in OSB Figure 9.18. Now the number of multiplications required is

$$4 \cdot (\frac{N}{4})^2 + 2 \cdot \frac{N}{4} + \frac{N}{2}.$$

We can continue the decomposition until we are left with only 2-point transforms which can be computed easily with the butterfly structure shown in OSB Figure 9.19.

In general, using the complete decimation-in-frequency decomposition, we need $(N/2)\log_2 N$ complex multiplications and $N \log_2 N$ complex additions to compute the N-point DFT. The flow graph for the case of an 8-point DFT is given in OSB Figure 9.20.

The flow graph above suggests a useful way of storing the original data and the results of the computation. If we store the input ($x[n]$s) in the same order as they appear in the figure from top to bottom, the output($X[k]$s) will appear in bit-reversed order. That is, the binary digits are sorted starting with the least significant bit instead of starting with the most significant bit as in normal order sorting.

Other rearrangements of the flow graph are discussed in OSB Section 9.4.2.

Decimation-in-Time

The decimation-in-frequency algorithm decomposes the output sequence $X[k]$ into successively smaller subsequences. We can also decompose the input sequence $x[n]$ into smaller subsequences, starting from decomposing into the even- and odd- numbered samples.

$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n] W_N^{nk} \\ &= \sum_{n\ even} x[n] W_N^{nk} + \sum_{n\ odd} x[n] W_N^{nk} \\ &= G[k] + W_N^k H[k], \end{aligned} \tag{4}$$

where $G[k]$ is the $(N/2)$-point DFT of the even numbered $x[n]$, and $H[k]$ is the $(N/2)$-point DFT of the odd numbered $x[n]$. Note that both $G[k]$ and $H[k]$ are periodic in k with period $(N/2)$, so when computing the value of $X[N/2]$, we can use $G[0]$ and $H[0]$, and so on. OSB Figure 9.3 depicts the computation using Equation 4 for $N = 8$.

Now, the number of complex multiplications and complex additions are both reduced to

$$2 \cdot (\frac{N}{2})^2 + N,$$

which is less than N^2 for $N > 2$. Similarly to the decimation-in-frequency algorithm, we can continue the decomposition until we are left with only two inputs as depicted in OSB Figure 9.5 and 9.7.

OSB Figure 9.8 shows the basic butterfly structure in the flow graph. Since

$$W_N^{(r+N/2)} = W_N^r W_N^{(N/2)} = -W_N^r,$$

we can further reduce the number of multiplications by replacing the structure in OSB Figure 9.8 with the one in OSB Figure 9.9.

The flow graph of an 8-point DFT using the butterfly computation of OSB Figure 9.9 is shown in OSB Figure 9.10. Notice that the input is in bit-reverse order and the output is in normal order. We can rearrange nodes to sort the input in normal order and the output in bit-reverse order as shown in OSB Figure 9.14.

In Lecture 6, we learned that the transposition of a flow graph is obtained by interchanging the input and output and reversing the direction of signal flow. Generally, for each decimation-in-time FFT algorithm flow graph, there exist a decimation-in-frequency algorithm obtained by transposing the original flow graph. One example is shown in OSB Figure 9.10 and 9.20. The flow graph in OSB Figure 9.10 depicts a decimation-in-time algorithm, and OSB Figure 9.20 shows a decimation-in-frequency flow graph. You can check that the flow graph in the right is the transpose of the flow graph in the left.

As in the decimation-in-frequency algorithm, if we rearrange nodes to sort both the input and output in normal order, we can not accomplish in-place computation. A flow graph that does not allow in-place computation, but nonetheless useful, is shown in OSB Figure 9.16. The same geometry for each stage permits sequential data accessing and storage.