Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.341: Discrete-Time Signal Processing

OpenCourseWare 2006

Lecture 20

The Goertzel Algorithm and the Chirp Transform

Reading: Sections 9.0 - 9.2 and 9.6 in Oppenheim, Schafer & Buck (OSB).

In the previous lecture we discussed a well-known class of algorithms for computing the DFT efficiently. While the fast Fourier transform's various incarnations have gained considerable popularity, careful selection of an appropriate algorithm for computing the DFT in practice need not be limited to choosing between these so-called "fast" implementations. We'll therefore focus in this lecture on two other techniques for computing the DFT: the Goertzel algorithm and the chirp transform.

Before going further, think about the following question:

> Given a signal $x[n]$ with corresponding 8-point DFT $X[k]$, what is the most efficient way, in terms of total multiplications required, to compute $X[5]$ from $x[n]$?

To start, note that using a radix-2 FFT algorithm isn't the best choice; judicious application of the canonical DFT equation is enough to beat it. Computing $X[5]$ requires roughly $8 \log_2 8 = 24$ complex multiplications when employing a radix-2 FFT algorithm, since $X[k]$ for $k = 0, 1, \ldots, 7$ must first be computed. However, calculating $X[5] = \sum_{n=0}^{7} x[n] e^{-j(2\pi/8)5n}$ using OSB Equation 8.11 requires only 8 complex multiplications and is strikingly simple compared to the radix-2 FFT algorithms, depicted in OSB Figure 9.10. This example illustrates the point that while the FFT algorithms may be useful for a wide class of problems, they are by no means the most efficient choice in all situations.

1 The Goertzel Algorithm

We'll now discuss the Goertzel Algorithm, an efficient method (in terms of multiplications) for computing $X[k]$ for a given k. The derivation of the algorithm, which is developed in OSB Section 9.2, begins by noting that the DFT can be formulated in terms of a convolution.

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

$$
\begin{aligned}
&= \sum_{r=0}^{N-1} x[r] W_N^{-k(N-r)} \\
&= \left(x[n] * W_N^{-nk} \right) \Big|_{n=N} \\
&= \left[x[n] * \left(W_N^{-nk} u[n] \right) \right] \Big|_{n=N}
\end{aligned}
$$

Specifically, processing a signal $x[n]$ through an LTI filter with impulse response $h[n] = W_N^{-nk} u[n]$ and evaluating the result, $y_k[n]$, at $n = N$ will give the corresponding N-point DFT coefficient $X[k] = y_k[N]$. This LTI filtering process is illustrated below.

$$
x[n] \longrightarrow \boxed{h[n] = W_N^{-nk} u[n]} \longrightarrow y_k[n] = x[n] * \left(W_N^{-nk} u[n] \right)
$$

Representing the filter by its z-transform, we have

$$
x[n] \longrightarrow \boxed{\sum_{m=0}^{\infty} W_N^{-mk} z^{-m}} \longrightarrow y_k[n],
$$

and since

$$
\begin{aligned}
H(z) &= \sum_{m=0}^{\infty} W_N^{-mk} z^{-m} \\
&= \frac{1 - W_N^{-k} z^{-1}}{1 - W_N^{-k} z^{-1}} \sum_{m=0}^{\infty} W_N^{-mk} z^{-m} \\
&= \frac{\sum_{m=0}^{\infty} W_N^{-mk} z^{-m} - \sum_{m=0}^{\infty} W_N^{-(m+1)k} z^{-(m+1)}}{1 - W_N^{-k} z^{-1}} \\
&= \frac{W_N^0 z^0}{1 - W_N^{-k} z^{-1}} \\
&= \frac{1}{1 - W_N^{-k} z^{-1}},
\end{aligned}
$$

the filtering operation can be equivalently performed by the system

$$
x[n] \longrightarrow \boxed{\frac{1}{1 - W_N^{-k} z^{-1}}} \longrightarrow y_k[n],
$$

with the associated flow graph depicted in OSB Figure 9.10.

Noting that

$$
H(z) = \frac{1}{1 - W_N^{-k} z^{-1}}
$$

$$
\begin{aligned}
&= \frac{1 - W_N^k z^{-1}}{1 - W_N^k z^{-1}} \frac{1}{1 - W_N^{-k} z^{-1}} \\
&= \frac{1 - W_N^k z^{-1}}{1 - \left(2\cos\frac{2\pi k}{N}\right) z^{-1} + z^{-2}},
\end{aligned}
$$

the filtering operation can also be equivalently performed by

$$
x[n] \longrightarrow \boxed{\frac{1 - W_N^k z^{-1}}{1 - \left(2\cos\frac{2\pi k}{N}\right) z^{-1} + z^{-2}}} \longrightarrow y_k[n],
$$

with the associated flow graph depicted in OSB Figure 9.20.

Since we're only evaluating the output of this filter at $y_k[N]$, the multiplier $-W_N^k$ need only be used at time $n = N$. With this in mind, the algorithm requires N real multiplications and a single complex multiplication to compute $X[k]$ for a given k. This compares favorably to N complex multiplications as required by the canonical DFT equation and approximately $N \log_2 N$ complex multiplications as required by the radix-2 FFT algorithms, when computing $X[k]$ for a given k. A final note about the Goertzel algorithm: since it is not restricted to computing $\sum_{n=0}^{N-1} x[n] W_N^{nk}$ for integer values of k, the algorithm has the convenient property that it can efficiently compute $X(e^{j\omega})$ for arbitrary choice of ω.

2 The Chirp Transform Algorithm

The chirp transform algorithm, which is derived in detail in OSB Subsection 9.6.2, computes $X(e^{j\omega})$ for uniformly-spaced samples of ω anywhere along the unit circle, as depicted in OSB Figure 9.25. The algorithm therefore computes

$$
\boxed{X(e^{j(\omega_0 + k\Delta\omega)}) = \sum_{n=0}^{N-1} x[n] e^{-j(\omega_0 + k\Delta\omega)n}, \quad k = 0, 1, \ldots, M-1},
$$

or equivalently,

$$
X(e^{j\omega_k}) = \sum_{n=0}^{N-1} x[n] e^{-j\omega_k n}, \quad k = 0, 1, \ldots, M-1,
$$

where

$$
\omega_k = \omega_0 + k\Delta\omega, \quad k = 0, 1, \ldots, M-1.
$$

Defining $W = e^{-j\Delta\omega}$, this becomes

$$
\begin{aligned}
X(e^{j\omega_k}) &= \sum_{n=0}^{N-1} e^{-j\omega_0 n} W^{nk} \\
&= \sum_{n=0}^{N-1} x[n] e^{-j\omega_0 n} W^{n^2/2} W^{k^2/2} W^{-(k-n)^2/2} \\
&= W^{n^2/2} \left[\left(e^{-j\omega_0 n} W^{n^2/2} x[n] \right) * W^{-n^2/2} \right].
\end{aligned}
$$

By defining $g[n] = e^{-j\omega_0 n} W^{n^2/2} x[n]$, the above equation is equivalently represented as

$$X(e^{j\omega_k}) = W^{n^2/2} \left(g[n] * W^{-n^2/2} \right),$$

which leads to the block diagram depicted in OSB Figure 9.26. This system isn't realizable as-is, since it is neither causal nor stable. If, however, we restrict ourselves to operating on a finite-length causal signal $x[n]$, the system can be implemented, since we're also only interested its output $y[n]$ over a given, finite time interval. There are several possible causal realizations of this, and in each case the implementation relies on appropriate bookkeeping.

There may also be some question as to how implementing $X(e^{j\omega_k})$ as in OSB Figure 9.26 could be practically useful. Note however that for a given $\Delta\omega$, the analysis parameter ω_0 appears only at the first modulator block. As long as the frequency sample spacing $\Delta\omega$ is fixed, the system can therefore be easily adjusted to compute for frequency samples beginning at any point on the unit circle. Another advantage of this technique is that it computes DTFT samples using a fixed convolution block, a property which is convenient when using discrete-time analog hardware, such as charge-coupled devices (CCDs).