Note Title 04-11-2013

EE5330 Nov. 4, 2013

The Fast Fourier Transform (FFT) Algorithm:

Recall the following DFT definition, where the notation $W_N = e^{-j\frac{2\pi}{N}}$ is used. The suffix N in W_N is dropped when there is no ambiguity. It is convenient to use x_k instead of $x[k]$.

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W & W^2 & \cdots & W^{N-1} \\ 1 & W^2 & W^4 & \cdots & W^{2(N-1)} \\ \vdots & & & & \\ 1 & W^{N-1} & & \cdots & W^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{N-1} \end{bmatrix}$$

Each row involves N multiplies (neglecting the fact that this is not true for the first row and first column). Since there are N such rows, the no. of multiplies is $N \cdot N$, or N^2. Also, for each X_k, we require $N-1$ additions, and totally there are $N \cdot (N-1)$ additions, or, roughly N^2 additions. Thus, the straightforward computation of the DFT requires N^2 multiplications and N^2 additions approximately.

Decimation-in-Time FFT Algorithm:

Consider breaking up $\underline{x}$ into its odd & even indices before computing the DFT. That is

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi k n}{N}} = \sum_{n=0}^{N-1} x_k W_N^{kn} \qquad k = 0, 1, \ldots, N-1.$$

$$= \sum_{r=0}^{\frac{N}{2}-1} x_{2r} e^{-j\frac{2\pi k}{N} 2r} + \sum_{r=0}^{\frac{N}{2}-1} x_{2r+1} e^{-j\frac{2\pi k}{N}(2r+1)}$$

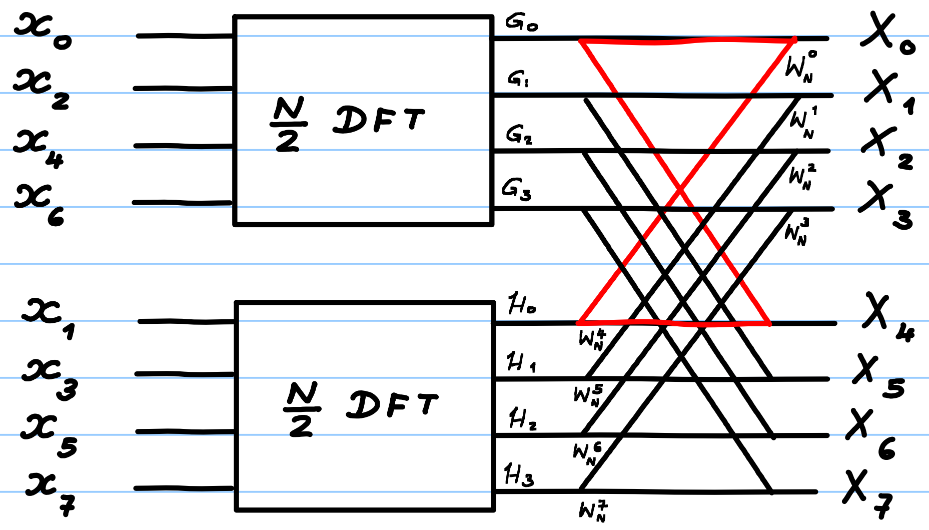
$$= \sum_{r=0}^{\frac{N}{2}-1} g_r e^{-j\frac{2\pi k r}{N/2}} + e^{-j\frac{2\pi k}{N}} \sum_{r=0}^{\frac{N}{2}-1} h_r e^{-j\frac{2\pi k r}{N/2}}$$

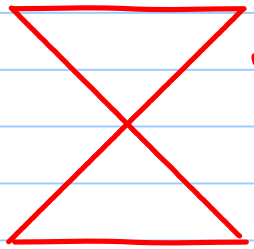
$$= \sum_{r=0}^{\frac{N}{2}-1} g_r W_{\frac{N}{2}}^{rk} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} h_r W_{\frac{N}{2}}^{rk}$$

These are two $\frac{N}{2}$-point DFTs !

$$= G_k + W_N^k H_k \qquad k = 0, 1, 2, \ldots, N-1$$

Since $\{G_k\}_{k=0}^{\frac{N}{2}-1}$ & $\{H_k\}_{k=0}^{\frac{N}{2}-1}$ are $\frac{N}{2}$-point DFTs, they are periodic with period $\frac{N}{2}$. Thus $G_{\frac{N}{2}} = G_0$, and so on. The above set of equations can be represented using the following block diagram.





"Butterfly"

Recall that the no. of mult./adds for an N-point transform is N^2.
The computation derived above has two $\frac{N}{2}$ transforms. They require $2\cdot\left(\frac{N}{2}\right)^2$ multiplies. In addition, combining them via W_N^k requires N multiplies. Thus, $N^2 \longrightarrow 2.\left(\frac{N}{2}\right)^2 + N$ is the reduction in the no. of multiplies.

<u>Example</u>

If $N=8$, $N^2 = 64$, whereas $2.\left(\frac{N}{2}\right)^2 + N = 2\cdot 16 + 8 = 40 < 64$. Thus, this "divide and conquer" approach leads to computational savings.

The sequences $\underline{g}$ and $\underline{h}$ can further divided into their odd and even indices and the above strategy can be exploited once more.
If N is a power of 2, the division by two can be carried out $\log_2 N$ times.

The computational savings follow the same pattern:

$$N^2 \longrightarrow 2\underbrace{\left(\frac{N}{2}\right)}^2 + N \longrightarrow 2\left[2\left(\frac{N}{4}\right)^2 + \frac{N}{2}\right] + N$$

$$= 4\left(\frac{N}{4}\right)^2 + \underbrace{2N}_{\text{second stage}}$$

For $N = 8$, we get $64 \rightarrow 40 \rightarrow 32$

In general, if N is a power of 2, we can continue to divide by two until we reach segments of length two. The no. of such segments is $\log_2 N$.

$$N^2 \longrightarrow 2\left(\frac{N}{2}\right)^2 + N \longrightarrow 4\left(\frac{N}{4}\right)^2 + N + N \longrightarrow 4\left[2\left(\frac{N}{8}\right)^2 + \frac{N}{4}\right] + N + N$$

$$= 8\left(\frac{N}{8}\right)^2 + \underbrace{3N \quad \text{third stage}}$$

Thus, if there are $\log_2 N$ segments, the no. mults./adds will be $\boxed{N \cdot \log_2 N}$ approximately.

Thus, from $O(N^2)$ mult./adds, we have come to $N\log_2 N$ mult./adds. If $N = 1024$, the savings is hundred-fold, i.e., two orders of magnitude!

The FFT algorithm is the principal reason why DSP is as practical and as powerful as it has become today. But for the presence of such FFT-type algorithms, DSP would have remained only as an academic curiosity!