Fast Fourier Transform

We use the "Divide-and-Conquer" approach. Divide operations into successively smaller ones.

Consider an N-point DFT such that

\[ N = LM \]

Store \( x[n] \) into a 2-D \( L \times M \) array \( x[l,m] \) such that \( n = ML + m \)

where \( l = 0, 1, \ldots (L-1) \) and \( m = 0, 1, \ldots (M-1) \)

Note: a transposition would be \( n = l + mL \)
A similar mapping can be used for the DFT coefficients

\[ r = M \cdot p + q \]  
where  
\[ p = 0, 1, \ldots, (L-1) \]  
\[ q = 0, 1, \ldots, (M-1) \]

on the transpose  
\[ r = p + q \cdot L \]

\[ \overline{X}(p, q) = \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} x[m, l] W_N^{(m \cdot p + q) \cdot (m \cdot L + l)} \]

where  
\[ W_N^{(m \cdot p + q) \cdot (m \cdot L + l)} = W_N^{m \cdot L \cdot m \cdot p} W_N^{m \cdot L \cdot q} W_N^{m \cdot p \cdot l} W_N^{m \cdot q \cdot l} \]

and  
\[ W_N^{m \cdot p} = 1 \]  
\[ W_N^{m \cdot q} = W_N^{m \cdot q} = W_N^{m \cdot q} \]

\[ W_N^{m \cdot p \cdot l} = W_N^{p \cdot l} = W_N^{p \cdot l} \]
so
\[ X(p, q) = \sum_{l=0}^{L-1} \left\{ W_N^{\text{e}^p} \left[ \sum_{m=0}^{M-1} x[l, m] W_M^{mq} \right] \right\} W_L^{lp} \]

1. \( F[l, q] = \sum_{m=0}^{M-1} x[l, m] W_M^{mq} \quad q = 0, 1, \ldots, (M-1) \)
   
   for each row \( l = 0, 1, \ldots, (L-1) \)

2. Form new array
   \[ G[l, q] = W_N^{\text{e}^q} F[l, q] \quad \text{for} \quad l = 0, 1, \ldots, (L-1) \]
   \[ q = 0, 1, \ldots, (M-1) \]
\[ \mathbf{X}(\rho, \varphi) = \sum_{l=0}^{L-1} G(l, \varphi) W_{\ell}^L \]

Computational Complexity

Component 1:
- \( L \) DFTs, each of \( M \) points
  - 1. \( LM^2 \) complex multi (cm)
  - 2. \( LM(M-1) \) complex adds (ca)

Component 2:
- \( LM \) cm

Component 3:
- 1. \( ML^2 \) cm
- 2. \( ML(L-1) \) ca

Total:
- \( N(M+L+1) \) cm
- \( N(M+L-2) \) ca

Compared to \( N^2 \) and \( N(N-1) \) we have order \( NM \)
Ex: Let \( N = 1000 \), \( L = 2 \), \( m = 500 \)

\[
\begin{align*}
\text{Radix - 2, where } N & = \text{number of points} \\
\text{FFT} & \Rightarrow x[n] = x[2n] \\
\text{let } f_1[n] = x[2n] \\
\text{let } f_2[n] = x[2n+1] \\
\text{for } n = 0, 1, \ldots, \frac{N}{2} - 1 \\
\end{align*}
\]
\[ X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} = \sum_{h \text{ even}} x[h] W_N^{k(h/2)} + \sum_{n \text{ odd}} x[n] W_N^{kn} \]

\[ = (N/2)^{-1} \sum_{m=0}^{(N/2)-1} x[2m] W_N^{2mk} + (N/2)^{-1} \sum_{m=0}^{(N/2)-1} x[2m+1] W_N^{k(2m+1)} \]

\[ = \sum_{m=0}^{N/2-1} f_1[m] W_{N/2}^{km} + W_N^k \sum_{m=0}^{N/2-1} f_2[m] W_{N/2}^{km} \]

\[ = F_1[k] + W_N^k F_2[k] \]

\( F_i[k] \) can be subdivided into odd and even until we reach the 2 point DFTs.

![Diagram](image-url)
If \( N \) is not a power of 2 then a mixed radix structure can be applied. For example, a 3-point butterfly has a structure as

\[
\begin{align*}
N & \\
2 & \leftrightarrow 2 \text{ point DFT} \\
3 & \leftrightarrow 3 \text{ point DFT} \\
4 & \leftrightarrow 2 \times 2 \text{ radix 2} \\
5 & \leftrightarrow 5 \text{ point DFT} \\
6 & \leftrightarrow 2 \times 3 \text{ radix 2 and radix 3}
\end{align*}
\]
finally, one eight-point DFT. The combination of the smaller DFTs to form the larger DFT is illustrated in Fig. 6.6 for $N = 8$.

Observe that the basic computation performed at every stage, as illustrated in Fig. 6.6, is to take two complex numbers, say the pair $(a, b)$, multiply $b$ by $W_N^k$, and then add and subtract the product from $a$ to form two new complex numbers $(A, B)$. This basic computation, which is shown in Fig. 6.7, is called a butterfly because the flow graph resembles a butterfly.

In general, each butterfly involves one complex multiplication and two complex additions. For $N = 2^r$, there are $N/2$ butterflies per stage of the computation process and $\log_2 N$ stages. Therefore, as previously indicated the total number of complex multiplications is $(N/2) \log_2 N$ and complex additions is $N \log_2 N$.

Once a butterfly operation is performed on a pair of complex numbers $(a, b)$ to produce $(A, B)$, there is no need to save the input pair $(a, b)$. Hence we can

Since $W_N^k$ store the results ($N$ $2N$ storage we say that.

A serial data sequence $x(0), x(2), \ldots, x(N-1)$ after decimation of the index in natural position $m$ the input data

With the computationally efficient natural form that

Furthermore, the output of the DFT algorithm involves the last $N/2$ computation.

Another algorithm for DFT computation implies a algorithm that involves the last $N/2$ computational steps.
stages of decimation, where each stage involves \( N/2 \) butterflies of the type shown in Fig. 6.10. Consequently, the computation of the \( N \)-point DFT via the decimation-in-frequency FFT algorithm, requires \( (N/2) \log_2 N \) complex multiplications and \( N \log_2 N \) complex additions, just as in the decimation-in-time algorithm. For illustrative purposes, the eight-point decimation-in-frequency algorithm is given in Fig. 6.11.

We observe from Fig. 6.11, that the input data \( x(n) \) occurs in natural order, but the output DFT occurs in bit-reversed order. We also note that the computations are performed in place. However, it is possible to reconfigure the decimation-in-frequency algorithm so that the input sequence occurs in bit-reversed order while the output DFT occurs in normal order. Furthermore, if we abandon the requirement that the computations be done in place, it is also possible to have both the input data and the output DFT in normal order.