

Calculating DFT

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N} = \sum_{n=0}^{N-1} x(n) \underbrace{(e^{-j2\pi/N})^{nk}}_{\substack{\text{frequency index} \\ \text{time index}}}$$

Denote $W_N = e^{-j2\pi/N}$, then $X(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk}$

- Each X(k) requires N complex multiplications and N complex additions.
- Each complex multiplication needs 4 real multiplications and 3 real addition because $(a+jb) \cdot (c+jd) = (ac-bd) + j(bc+ad)$
- There are N different X(k), so we need a total of
 - $4N^2$ real multiplications
 - $4N^2$ real additions

However, by taking advantages of some properties of W_N^m , we can significantly speed out the calculations \Rightarrow Fast Fourier Transform

Interesting Properties of W_N^m :

(1) $W_N^0 = (e^{-j2\pi/N})^0 = e^0 = 1, \quad W_N^N = e^{-j2\pi} = 1$

(2) $W_N^{N+m} = W_N^m$ (periodic)

(3) Assume N is a multiple of 4

$W_N^{N+m} = (W_N^N) \cdot W_N^m = 1 \cdot W_N^m = W_N^m$

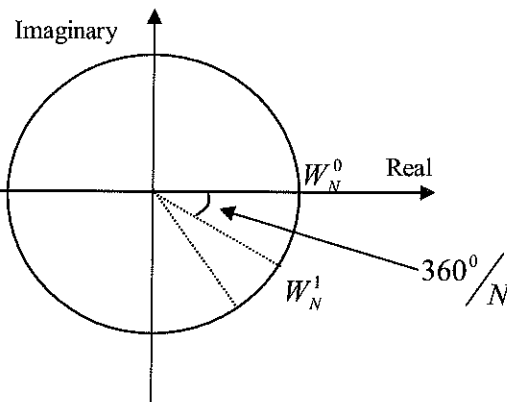
$$\begin{cases} W_N^{N/2} = e^{-j2\pi/(N/2)/N} = e^{-j\pi} = -1 \\ W_N^{N/4} = e^{-j2\pi/(N/4)/N} = e^{-j\pi/2} = -j \\ W_N^{3N/4} = e^{-j2\pi/(3N/4)/N} = e^{-j3\pi/2} = j \end{cases}$$

$W_N = e^{-j\frac{2\pi}{N}}$
 ||
 One slice of pizza

(4) Assume N is even

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

↳ make your slice twice as big and take half as many slices

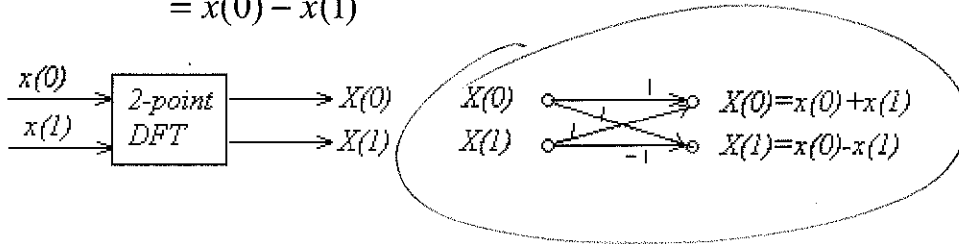


Example 10-3: Two-Point DFT

$$x(0), x(1): X(k) = \sum_{n=0}^1 x(n)W_2^{nk} \quad k = 0,1$$

$$X_0 \Rightarrow X(0) = \sum_{n=0}^1 x(n)W_2^{n0} = \sum_{n=0}^1 x(n) = x(0) + x(1)$$

$$\begin{aligned} X_1 \Rightarrow X(1) &= \sum_{n=0}^1 x(n)W_2^{n1} = \sum_{n=0}^1 x(n)W_2^n \\ &= x(0)W_2^0 + x(1)W_2^1 \\ &= x(0) + x(1)(-1) \\ &= x(0) - x(1) \end{aligned}$$



Example 10-4: Four-point DFT of $x(0), x(1), x(2), x(3)$

$$X(k) = \sum_{n=0}^3 x(n)W_4^{nk} \quad k = 0,1,2,3,$$

$W_N = e^{\frac{-j2\pi}{N}}$
Twiddle factor

DFT '0' Sample

$$X(0) = \sum_{n=0}^3 x(n)W_4^{n0} = \sum_{n=0}^3 x(n) = x(0) + x(1) + x(2) + x(3)$$

$$\begin{aligned} X(1) &= \sum_{n=0}^3 x(n)W_4^n = x(0)W_4^0 + x(1)W_4^1 + x(2)W_4^2 + x(3)W_4^3 \\ &= x(0) - jx(1) - x(2) + jx(3) = (x(0) - x(2)) + j(x(3) - x(1)) \end{aligned}$$

$$\begin{aligned} X(2) &= \sum_{n=0}^3 x(n)W_4^{2n} = x(0)W_4^0 + x(1)W_4^2 + x(2)W_4^4 + x(3)W_4^6 \\ &= x(0) + x(1)(-1) + x(2)(1) + x(3)W_4^2 \\ &= (x(0) - x(1)) + x(2) - x(3) \end{aligned}$$

$W_4^0 = e^{\frac{-j2\pi}{4} \cdot 0} = 1$
 $W_4^1 = e^{\frac{-j2\pi}{4} \cdot 1} = -j$

$$\begin{aligned}
 X(3) &= \sum_{n=0}^3 x(n)W_4^{3n} = x(0)W_4^0 + x(1)W_4^3 + x(2)W_4^6 + x(3)W_4^9 \\
 &= x(0) + x(1)W_4^3 + x(2)(1)W_4^2 + x(3)W_4^1 \\
 &= x(0) + jx(1) + (-1)x(2) + (-j)x(3) \\
 &= x(0) + jx(1) - x(2) - jx(3)
 \end{aligned}$$

I. Reuse calculation

II. 2-pt DFT of $\{x(0), x(2)\}$
2-pt DFT of $\{x(1), x(3)\}$

$$\begin{aligned}
 X(0) &= [x(0) + x(2)] + [x(1) + x(3)] \\
 X(1) &= [x(0) - x(2)] + (-j)[x(1) - x(3)] \\
 X(2) &= [x(0) + x(2)] - [x(1) + x(3)] \\
 X(3) &= [x(0) - x(2)] + j[x(1) - x(3)]
 \end{aligned}$$

If we denote $z(0) = x(0), z(1) = x(2) \Rightarrow Z(0) = z(0) + z(1) = x(0) + x(2)$
 $Z(1) = z(0) - z(1) = x(0) - x(2)$

Two-point DFT of $[x(0) \ x(2)]$

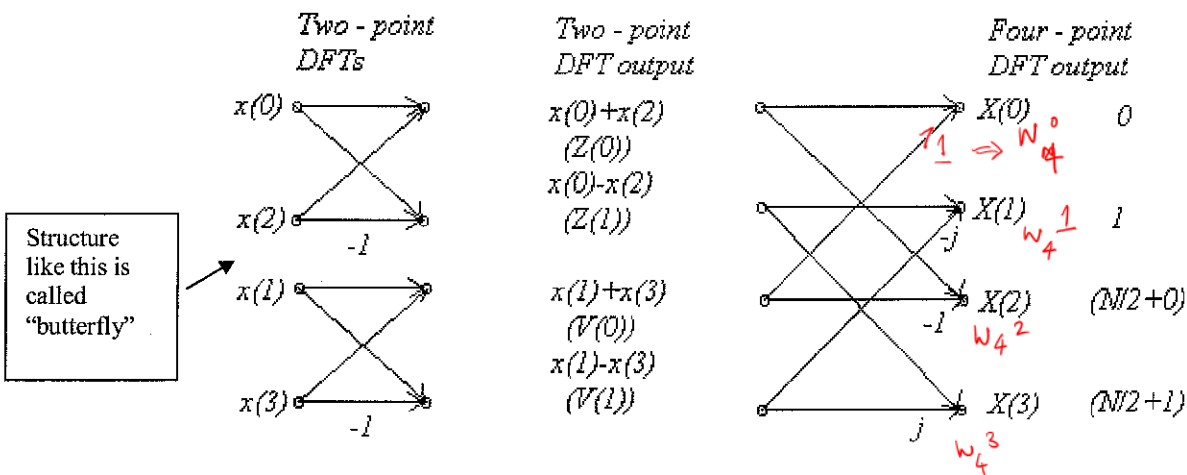
$v(0) = x(1), v(1) = x(3) \Rightarrow V(0) = v(0) + v(1) = x(1) + x(3)$
 $V(1) = v(0) - v(1) = x(1) - x(3)$

Two-point DFT of $[x(1) \ x(3)]$

Our four-point DFT:

$$\begin{aligned}
 X(0) &= Z(0) + V(0) \\
 X(1) &= Z(1) + (-j)V(1) \\
 X(2) &= Z(0) - V(0) \\
 X(3) &= Z(1) + jV(1)
 \end{aligned}$$

4 point DFT = 4 Point FFT \rightarrow 2 \cdot [2pt FFT] + (4 Adds) + (4 Mults) = 2 \cdot (4 ops)



Key point: We compute 4-point DFT based on two 2-point DFTs

$4N^2$

Decimation-in-Time FFT Algorithm

$x(0), x(1), \dots, x(N-1)$ $N = 2^m = 2^{10} = 1024$ Samples

Separate into even and odd samples:

$g(r) = x(2r) \Rightarrow x(0), x(2), x(4), \dots, x(1024)$
 $h(r) = x(2r+1) \Rightarrow x(1), x(3), x(5), \dots, x(1023)$

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n)W_N^{kn} \\ &= \sum_{r=0}^{N/2-1} g(r)W_N^{k(2r)} + \sum_{r=0}^{N/2-1} h(r)W_N^{k(2r+1)} \quad (k = 0, 1, \dots, N-1) \\ &= \sum_{r=0}^{N/2-1} g(r)W_N^{2kr} + W_N^k \sum_{r=0}^{N/2-1} h(r)W_N^{2kr} \end{aligned}$$

$W_N^k \cdot W_N^{2kr}$

Using property 4:

$$\begin{aligned} W_N^{2kr} &= (e^{-j2\pi/N})^{2kr} = (e^{-j2\pi/(N/2)})^{kr} = W_{N/2}^{kr} \\ \Rightarrow X(k) &= \sum_{r=0}^{N/2-1} g(r)W_{N/2}^{kr} + W_N^k \sum_{r=0}^{N/2-1} h(r)W_{N/2}^{kr} \\ &= G(k) + W_N^k H(k) \end{aligned}$$

Even

odd

where $G(k)$ and $H(k)$ are the $N/2$ point DFT of $g(r)$ and $h(r)$ respectively.

DFT
↓
1024

$\leftarrow X(k) = G(k) + W_N^k H(k) \quad k = 0, 1, \dots, N-1 \quad \leftarrow$

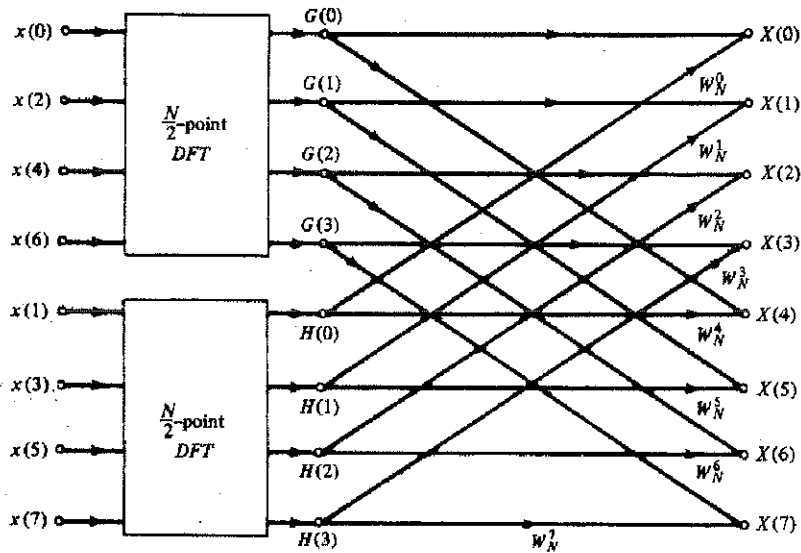
$$\begin{aligned} G(k) &= \sum_{r=0}^{N/2-1} g(r)W_{N/2}^{kr} = \sum_{r=0}^{N/2-1} x(2r)W_{N/2}^{kr} \\ H(k) &= \sum_{r=0}^{N/2-1} h(r)W_{N/2}^{kr} = \sum_{r=0}^{N/2-1} x(2r+1)W_{N/2}^{kr} \end{aligned}$$

Question: $X(k)$ needs $G(k), H(k), k=0 \dots N-1$

How do we obtain $G(k), H(k)$, for $k > N/2-1$?

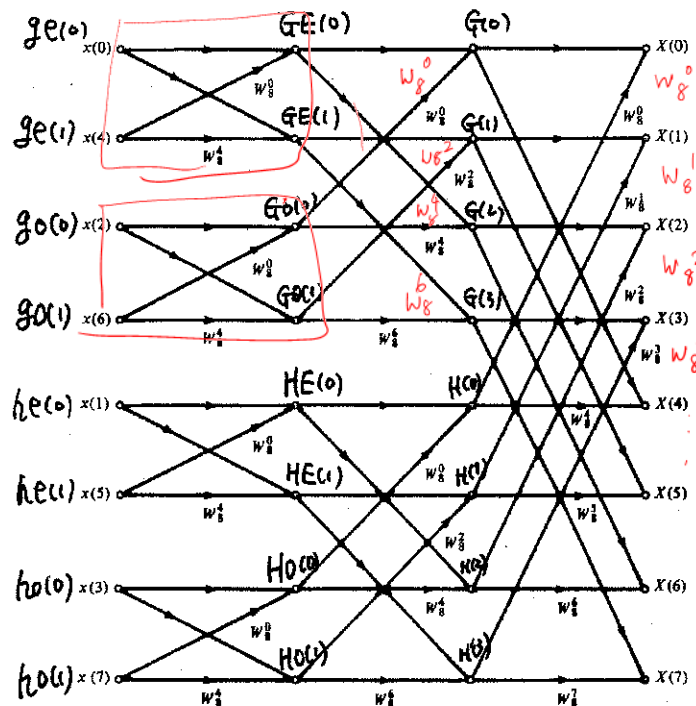
$G(k) = G(N/2+k) \quad k \leq N/2-1$

$H(k) = H(N/2+k) \quad k \leq N/2-1$



(a) Result of one decimation of the time samples

Such recursion can be carried on by considering the even and odd samples of the sequence at each stage. For a 8-point FFT, the resulting decomposition looks like:



What is the big deal?

- (1) Very regular structure \Rightarrow easy hardware (and software) implementation
- (2) Big complexity savings!!

Why? Let's say you want to compute 1024-point DFT.

Using direct computation, we need $4N^2 \sim 4$ million operations

Using the above strategy:

Let $C(N)$ = total operations for 1024-pt DFT

$$\begin{aligned}
 C(1024) &= 2*1024 + 2*C(512) \quad : \text{One stage butterfly} + \text{Two 512-DFT} \\
 &= 2*1024 + 2*[2*512+2*C(256)] \\
 &= 4*1024 + 4*[2*256+2*C(128)] \\
 &= 6*1024 + 8*[2*128+2*C(64)] \\
 &= 8*1024 + 16*[2*64+2*C(32)] \\
 &= 10*1024 + 32*[2*32+2*C(16)] \\
 &= 12*1024 + 64*[2*16+2*C(8)] \\
 &= 14*1024 + 128*[2*8+2*C(4)] \\
 &= 16*1024 + 256*C(2) \\
 &= 16*1024 + 256*4 = 17408 \text{ operations} \sim \text{A factor of 100 savings!}
 \end{aligned}$$

In general, $C(N) \sim N(\log_2 N)$. The great savings come from the reduction of a factor N to $(\log_2 N)$, thanks to the recursion