

1.6.2 FFT (continued).

Lec 17 10/11/01 Mon

①

$$X^{(N)}(k) = \underbrace{X_0^{(N/2)}(k)}_{N/2\text{-periodic}} + \underbrace{W_N^k X_1^{(N/2)}(k)}_{N\text{-periodic}}, \quad k=0, 1, \dots, N-1$$

where

$$X_0^{(N/2)}(k) = \frac{N}{2}\text{-pt DFT of even-numbered samples of } x(n)$$

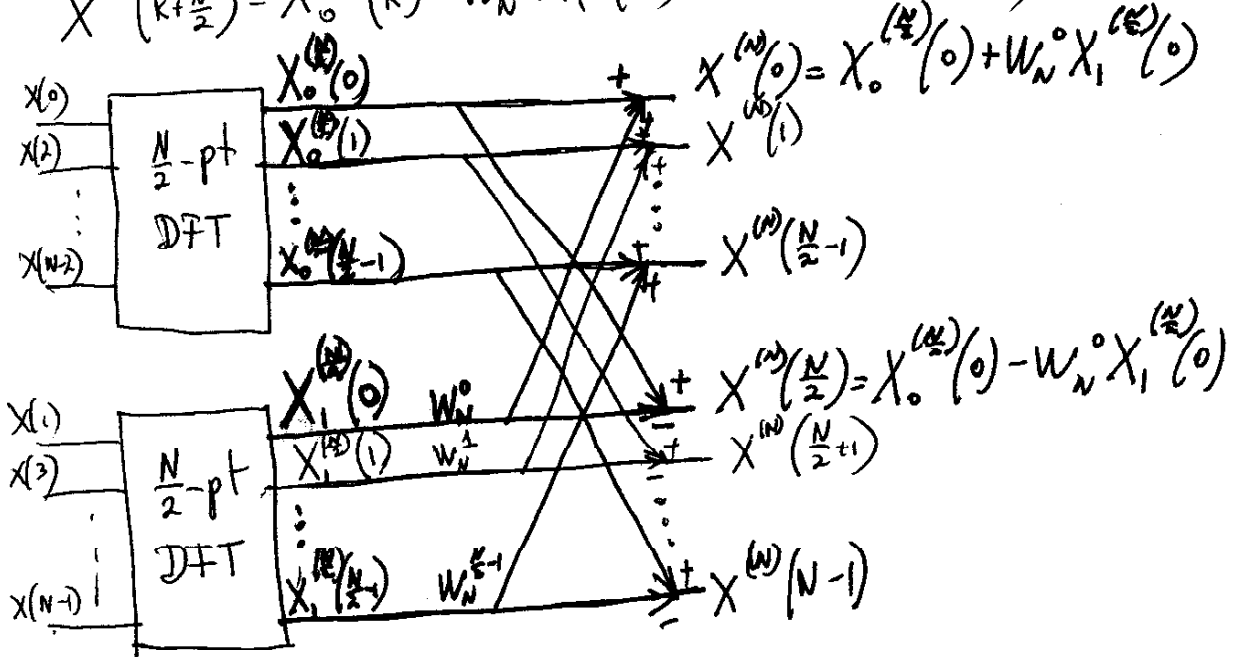
$$X_1^{(N/2)}(k) = \frac{N}{2}\text{-pt DFT of odd-numbered samples of } x(n).$$

~~Note:~~ $W_N = e^{-j\frac{2\pi}{N}}$

Note: $W_N^{k+\frac{N}{2}} = e^{-j\frac{2\pi}{N}(k+\frac{N}{2})} = e^{-j(\frac{2\pi k}{N} + \pi)} = -e^{-j\frac{2\pi k}{N}} = -W_N^k$

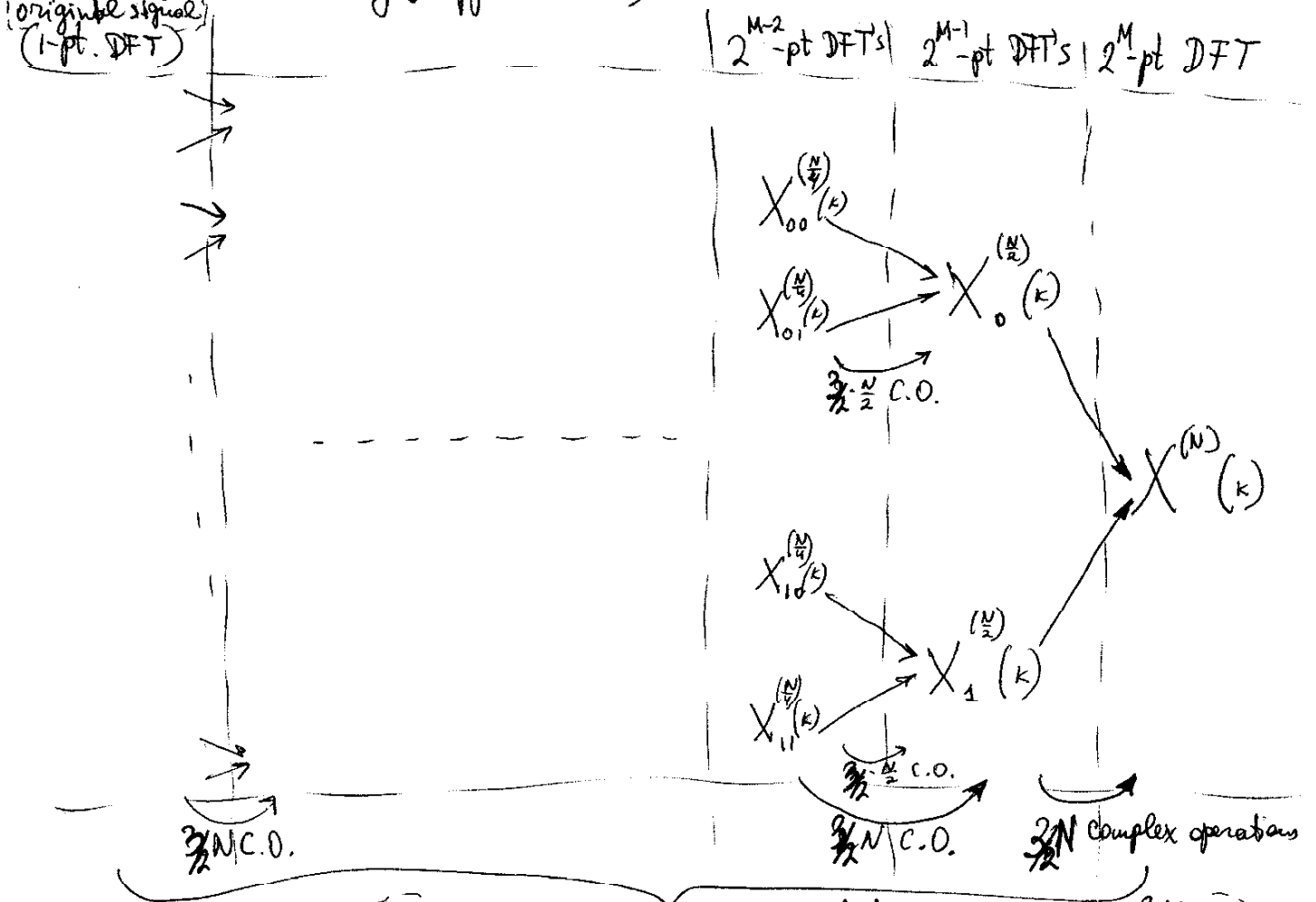
So, $X^{(N)}(k) = X_0^{(N/2)}(k) + W_N^k X_1^{(N/2)}(k)$ for $k=0, 1, \dots, \frac{N}{2}-1$ Eq. (12)

$X^{(N)}(k+\frac{N}{2}) = X_0^{(N/2)}(k) - W_N^k X_1^{(N/2)}(k)$ " Lab 6.2



$\frac{N}{2}$ complex multiplications (actually, slightly fewer)
 N complex additions

Repeat recursively (suppose $N=2^M$):



$M = \log_2 N$ stages of computation, each requires $\frac{3}{2}N$ C.O.
 Total computational complexity is $O(N \log N)$.

What is a 1-point DFT?

$$X(0) = \sum_{n=0}^{N-1} x(n) e^{-j \frac{2\pi \cdot 0}{N} \cdot n} = X(0) - \text{samples of the original signal}$$

- Remarks:
- For large N , this is much faster than $O(N^2)$.
 - Terminology: decimation-in-time radix-2 FFT.
 - $O(N \log N)$ essentially means $CN \log N$. There are many variations of FFT aimed at reducing the constant C . E.g., if $N=3^M$, it may be better to use radix-3 FFT.

4. ~~Let $W_N = e^{-j \frac{2\pi}{N}}$. Since $x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{nk}$, an FFT alg. for DFT can be converted into an IFFT alg. for IFFT by~~

- changing the sign on all phase factors
- dividing the final output by N

4. Note: $\left\{ \frac{1}{N} \text{DFT}[x^*(n)] \right\}^* = \left\{ \frac{1}{N} \sum_{n=0}^{N-1} x^*(n) e^{-j \frac{2\pi k}{N} n} \right\}^*$

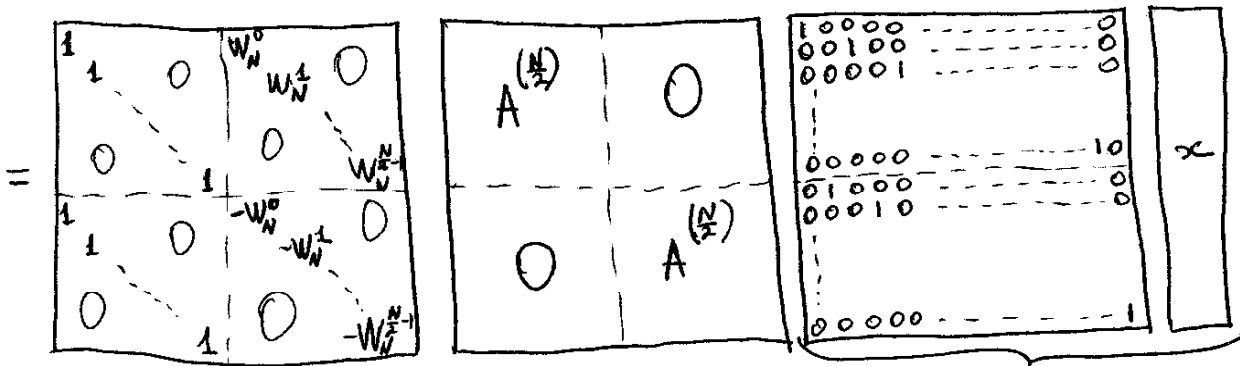
$$= \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{j \frac{2\pi k}{N} n},$$



which is IDFT.

Thus, FFT can be used to compute IDFT.

Recall that DFT is multiplication by a matrix:

$$\begin{matrix} \boxed{X} \\ N \times 1 \end{matrix} = \begin{matrix} \boxed{A^{(N)}} \\ N \times N \end{matrix} \begin{matrix} \boxed{x} \\ N \times 1 \end{matrix}$$



We've essentially reduced one  to two  - almost a factor of two reduction in the number of operations.

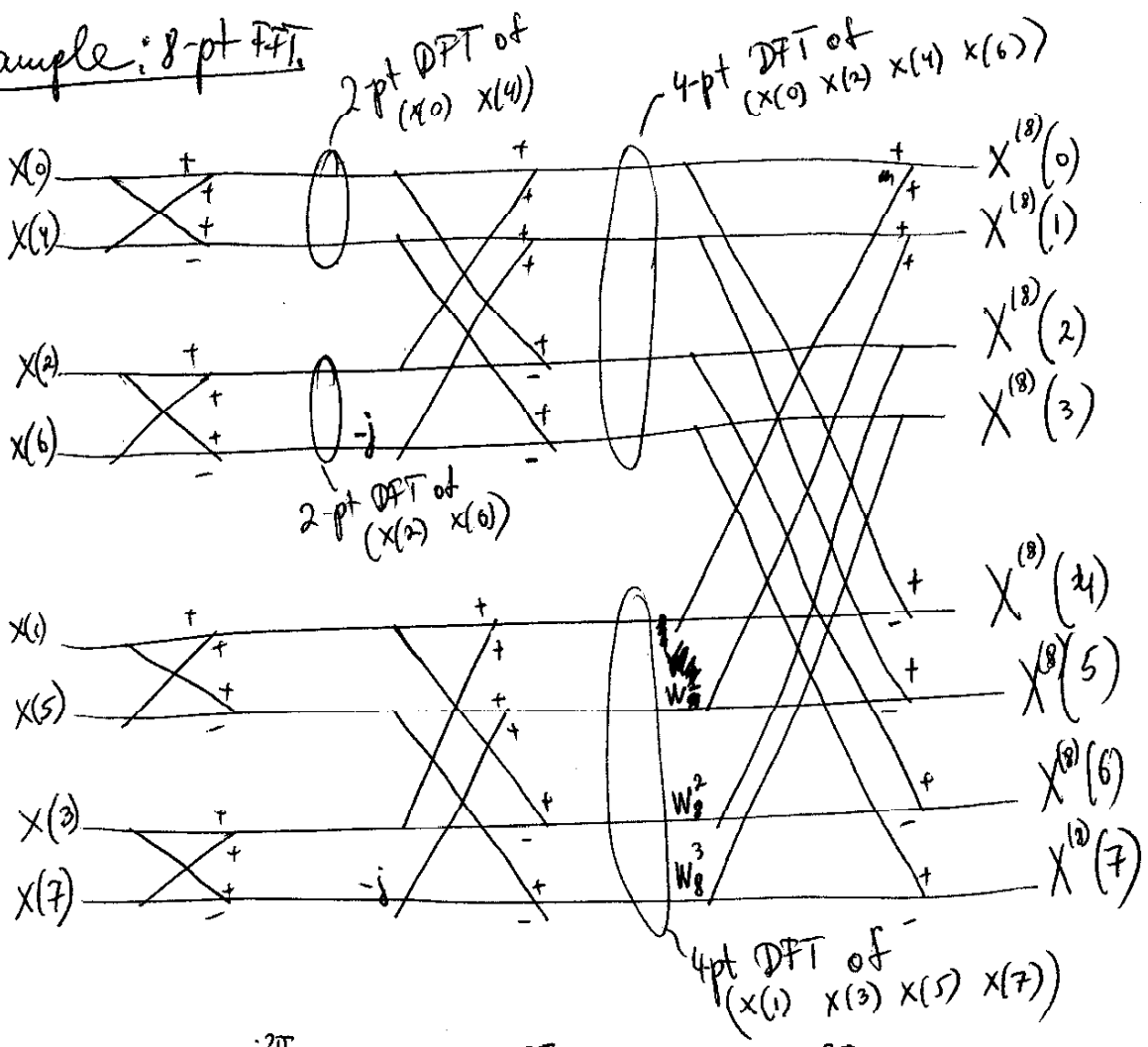
This is yet another illustration of the fact that most of what we do in DSP is linear algebra.

- "Matlab" stands for "Matrix laboratory";
 - its basic data structures are matrices and vectors
- ⇒ EE 438 labs are matrix manipulations
 ⇒ in order to really understand DSP, one needs to be fluent in linear algebra.

- x(0)
- x(2)
- x(4)
- ⋮
- x(N-2)
- x(1)
- x(3)
- ⋮
- x(N-1)

- X₀^(N/2)(0)
- ⋮
- X₀^(N/2)(N/2-1)
- X₁^(N/2)(0)
- ⋮
- X₁^(N/2)(N/2-1)

Example: 8-pt FFT



$$W_2 = e^{-j\frac{2\pi}{2}} = -1 \quad W_4 = e^{-j\frac{2\pi}{4}} = -j \quad W_8 = e^{-j\frac{2\pi}{8}}$$

$$\text{Let } A_k = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & e^{-j\frac{2\pi k}{N}} \\ 1 & -e^{-j\frac{2\pi k}{N}} \end{pmatrix}$$

(6)

Then ~~$A_k A_k^H = \frac{1}{2} \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$~~

$$\langle A_k X, A_k Y \rangle = (A_k Y)^H (A_k X)$$

$$= Y^H A_k^H A_k X =$$

$$= Y^H \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ e^{j\frac{2\pi k}{N}} & -e^{j\frac{2\pi k}{N}} \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & e^{-j\frac{2\pi k}{N}} \\ 1 & -e^{-j\frac{2\pi k}{N}} \end{pmatrix} X$$

$$= Y^H \frac{1}{2} \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} X = Y^H X = \langle X, Y \rangle$$

i.e., multiplication by A_k preserves distances and angles (rotation, reflection) — see HW 3 Prob. ---

FFT:

A is written as a product of $\frac{N}{2} \log N$ $(\sqrt{2} A_k)^s$, each operating on a pair of coordinates — decomposition into elementary planar transformations.