FFT- THE FAST FOURIER TRANSFORM

Prof. Alina Karabchevsky, www.alinakarabchevsky.com

Introduction to Signal Processing,

School of ECE,

Ben-Gurion University

Reading: Chapter 8 by Proakis and Monolakis

DEMAND FOR COMPUTATION SPEED AND EFFICIENCY

- Motivation for efficient computing: One of the greatest challenges in information processing and computing is to reduce the energy consumption not only for the sake of reducing the tremendous amount of electric power used by the computer-based industry, such as Google, Amazon or Facebook, but also to enable computing devices to operate at a higher frequency without melting as a result of the extra heat: the cooling all the world's data centers account for 1.5% of total global electricity.

FFT

from *SIAM News*, Volume 33, Number 4 May, 2000

The Best of the 20th Century: Editors Name Top 10 Algorithms

By Barry A. Cipra

1965: James **Cooley** of the IBM T.J. Watson Research Center and John **Tukey** of Princeton University and AT&T Bell Laboratories unveil the fast Fourier transform. Easily the most far-reaching algorithm in applied mathematics, the FFT revolutionized signal processing. The underlying idea goes back to Gauss (who needed to calculate orbits of asteroids), but it was the Cooley–Tukey paper that made it clear how easily Fourier transforms can be computed. Like Quicksort, the FFT relies on a divide-and-conquer strategy to reduce an ostensibly O(N^2) chore to an O(N log N) frolic. But unlike Quick-sort, the implementation is (at first sight) nonintuitive and less than straightforward. This in itself gave computer science an impetus to investigate the inherent complexity of computational problems and algorithms.

James Cooley

John Tukey

An Algorithm for the Machine Calculation of Complex Fourier Series

By James W. Cooley and John W. Tukey

An efficient method for the calculation of the interactions of a 2^m factorial experiment was introduced by Yates and is widely known by his name. The generalization to 3^m was given by Box et al. [1]. Good [2] generalized these methods and gave elegant algorithms for which one class of applications is the calculation of Fourier series. In their full generality, Good's methods are applicable to certain problems in which one must multiply an N-vector by an $N \times N$ matrix which can be factored into m sparse matrices, where m is proportional to $\log N$. This results in a procedure requiring a number of operations proportional to $N \log N$ rather than N^2. These methods are applied here to the calculation of complex Fourier series. They are useful in situations where the number of data points is, or can be chosen to be, a highly composite number. The algorithm is here derived and presented in a rather different form. Attention is given to the choice of N. It is also shown how special advantage can be obtained in the use of a binary computer with $N = 2^m$ and how the entire calculation can be performed within the array of N data storage locations used for the given Fourier coefficients.

Consider the problem of calculating the complex Fourier series

$$X(j) = \sum_{k=0}^{N-1} A(k) \cdot W^{jk}, \qquad j = 0, 1, \cdots, N-1, \tag{1}$$

DFT

where the given Fourier coefficients $A(k)$ are complex and W is the principal Nth root of unity,

$$W = e^{2\pi i/N}. \tag{2}$$

A straightforward calculation using (1) would require N^2 operations where "operation" means, as it will throughout this note, a complex multiplication followed by a complex addition.

The algorithm described here iterates on the array of given complex Fourier amplitudes and yields the result in less than $2N \log_2 N$ operations without requiring more data storage than is required for the given array A. To derive the algorithm, suppose N is a composite, i.e., $N = r_1 \cdot r_2$. Then let the indices in (1) be expressed

$$\begin{aligned} j &= j_1 r_1 + j_0, & j_0 &= 0, 1, \cdots, r_1 - 1, & j_1 &= 0, 1, \cdots, r_2 - 1, \\ k &= k_1 r_2 + k_0, & k_0 &= 0, 1, \cdots, r_2 - 1, & k_1 &= 0, 1, \cdots, r_1 - 1. \end{aligned} \tag{3}$$

Then, one can write

$$X(j_1, j_0) = \sum_{k_0} \sum_{k_1} A(k_1, k_0) \cdot W^{jk_1 r_2} W^{jk_0}. \tag{4}$$

Received August 17, 1964. Research in part at Princeton University under the sponsorship of the Army Research Office (Durham). The authors wish to thank Richard Garwin for his essential role in communication and encouragement.

Since

$$W^{jk_1 r_2} = W^{j_0 k_1 r_2}, \tag{5}$$

the inner sum, over k_1, depends only on j_0 and k_0 and can be defined as a new array,

$$A_1(j_0, k_0) = \sum_{k_1} A(k_1, k_0) \cdot W^{j_0 k_1 r_2}. \tag{6}$$

The result can then be written

$$X(j_1, j_0) = \sum_{k_0} A_1(j_0, k_0) \cdot W^{(j_1 r_1 + j_0) k_0}. \tag{7}$$

There are N elements in the array A_1, each requiring r_1 operations, giving a total of Nr_1 operations to obtain A_1. Similarly, it takes Nr_2 operations to calculate X from A_1. Therefore, this two-step algorithm, given by (6) and (7), requires a total of

$$T = N(r_1 + r_2) \tag{8}$$

operations.

It is easy to see how successive applications of the above procedure, starting with its application to (6), give an m-step algorithm requiring

$$T = N(r_1 + r_2 + \cdots + r_m) \tag{9}$$

operations, where

$$N = r_1 \cdot r_2 \cdots r_m. \tag{10}$$

If $r_j = s_j t_j$ with $s_j, t_j > 1$, then $s_j + t_j < r_j$ unless $s_j = t_j = 2$, when $s_j + t_j = r_j$. In general, then, using as many factors as possible provides a minimum to (9), but factors of 2 can be combined in pairs without loss. If we are able to choose N to be highly composite, we may make very real gains. If all r_j are equal to r, then, from (10) we have

$$m = \log_r N \tag{11}$$

and the total number of operations is

$$T(r) = rN \log_r N. \tag{12}$$

If $N = r^m s^n t^p \cdots$, then we find that

$$\begin{aligned} \frac{T}{N} &= m \cdot r + n \cdot s + p \cdot t + \cdots, \\ \log_2 N &= m \cdot \log_2 r + n \cdot \log_2 s + p \cdot \log_2 t + \cdots, \end{aligned} \tag{13}$$

so that

$$\frac{T}{N \log_2 N}$$

is a weighted mean of the quantities

$$\frac{r}{\log_2 r}, \quad \frac{s}{\log_2 s}, \quad \frac{t}{\log_2 t}, \cdots,$$

DIRECT COMPUTATION OF DFT

in general:
complex number

DFT $\quad 0 \le k \le N-1 \quad X[k] = \sum_{n=0}^{N-1} x[n] W_N^{-kn}$

IDFT $\quad 0 \le n \le N-1 \quad x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[k] W_N^{+kn} \quad W_N = e^{j\frac{2\pi}{N}}$

twiddle factor

- IDFT is the same calculation as DFT, therefore we focus on DFT

DIRECT COMPUTATION OF DFT

- In general, DFT and IDFT calculations include multiplication and summation of complex numbers. In practice those calculation are implemented by a computer or DSP which multiply or add real numbers. Therefore, we will write the DFT as

$$X_R(k) = \sum_{n=0}^{N-1}\left[x_R[n]\cos\left(\frac{2\pi}{N}kn\right) + x_I[n]\sin\left(\frac{2\pi}{N}kn\right)\right]$$

Real

$$X_I(k) = \sum_{n=0}^{N-1}\left[-x_R[n]\sin\left(\frac{2\pi}{N}kn\right) + x_I[n]\cos\left(\frac{2\pi}{N}kn\right)\right]$$

Imag

$$e^{-j(\ )} = \cos(\ ) - j\sin(\ )$$

Euler

COMPUTATION COMPLEXITY OF DFT

Direct DFT

ici plain

- Direct computation of DFT, for each values of k, needs $x[n]$ **multiplied** by W_N and then **add** the result from 0 to $N-1$:

$$X_R(k) = \sum_{n=0}^{N-1}\left[x_R[n]\cos\left(\frac{2\pi}{N}kn\right) + x_I[n]\sin\left(\frac{2\pi}{N}kn\right)\right]$$

$$X_I(k) = \sum_{n=0}^{N-1}\left[-x_R[n]\sin\left(\frac{2\pi}{N}kn\right) + x_I[n]\cos\left(\frac{2\pi}{N}kn\right)\right]$$

1) $2N^2$ calculations of **trigonometric functions** - exp: sin, cos (kn).

2) $4N^2$ real **multiplications** – for each k in each row 2N.

3) $2N(2N-1)$ real **additions** – 2N elements in each row needs 2N-1 additions.

4) Overhead – additional calculations due to the saving of indices to the memory and addressing operations.

Example: $(a+jb)(c+jd) = ac + jbc + jad - bd = ac - bd + j(bc+ad) \rightarrow X[k] = \sum_{n=0}^{N-1} x[n]W_N^{-kn}$

For simplicity, we will account the complex operations – usually multiplications.

In case of DFT, there are N^2 since DFT makes multiplication and addition as one operation: **MAC** = **M**ultiply and **Ac**cumulate – for this reason, we count the number of multiplications as the measure of computational complexity.

COMPUTATIONAL COMPLEXITY סיבוכיות חישוב

Efficient Computation of the DFT: FFT Algorithms

TABLE 8.1 Comparison of Computational Complexity for the Direct Computation of the DFT Versus the FFT Algorithm

Number of Points, N	Complex Multiplications in Direct Computation, N^2	Complex Multiplications in FFT Algorithm, $(N/2)\log_2 N$	Speed Improvement Factor
4	16	4	4.0
8	64	12	5.3
16	256	32	8.0
32	1,024	80	12.8
64	4,096	192	21.3
128	16,384	448	36.6
256	65,536	1,024	64.0
512	262,144	2,304	113.8
1,024	1,048,576	5,120	204.8

Image, signal, speech processing

HOW TO MAKE DFT MORE EFFICIENT? THE 2-POINT DFT

- The direct calculation of DFT is not efficient since it does not consider the properties of W_N such as:
- Symmetry: $W_N^{k+N/2} = -W_N^k$
- Periodicity: $W_N^{k+N} = W_N^k$
- Example: 2-point DFT → $N = 2$
- Assume we have only two samples
- Only two frequency components
  - DC
  - Alternation

$$X_0 = \sum_{n=0}^{1} x_n (W_2)^{-n0} = x_0 + x_1 \quad \text{DC}$$

$k=0$ $\quad 1 \quad 1$

$$X_1 = \sum_{n=0}^{1} x_n (W_2)^{-n1} = x_0 - x_1 \quad \text{AC}$$

$k=1$ $\quad 1 \quad -1$

DFT 2x2

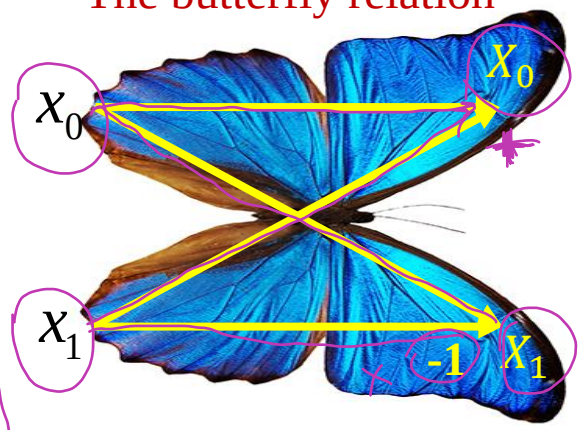
$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\boxed{W_2 = e^{j\frac{2\pi}{2}} = e^{j\pi} = -1}$$

$N=2$

Computational complexity:
$N^2 \rightarrow \frac{N}{2}\log_2 N$

The butterfly relation



$= x_0 + x_1$

$= x_0 + (-1) \cdot x_1$

W_2

THE 4-POINT DFT

- Three frequency components
  - DC
  - Alternating (high frequency)
  - Intermediate

$$X_0 = \sum_{n=0}^{3} x_n \left(W_4\right)^{-n0} = x_0 + x_1 + x_2 + x_3$$

$$X_1 = \sum_{n=0}^{3} x_n \left(W_4\right)^{-n1} = x_0 - jx_1 - x_2 + jx_3$$

$$X_2 = \sum_{n=0}^{3} x_n \left(W_4\right)^{-n2} = x_0 - x_1 + x_2 - x_3$$

$$X_3 = \sum_{n=0}^{3} x_n \left(W_4\right)^{-n3} = x_0 + jx_1 - x_2 - jx_3$$

$$\boxed{W_4 = e^{j\frac{2\pi}{4}} = e^{j\frac{\pi}{2}} = j}$$

THE 4-POINT DFT

- We can see the intuitive significance of each component

$$X_0 = \sum_{n=0}^{3} x_n \left(W_4\right)^{-n0} = x_0 + x_1 + x_2 + x_3 \longrightarrow \text{How big the signal is}$$

$$X_1 = \sum_{n=0}^{3} x_n \left(W_4\right)^{-n1} = x_0 - jx_1 - x_2 + jx_3 \longrightarrow \text{Differences in off-by-one}$$

$$X_2 = \sum_{n=0}^{3} x_n \left(W_4\right)^{-n2} = x_0 - x_1 + x_2 - x_3 \longrightarrow \text{How much it alternates}$$

$$X_3 = \sum_{n=0}^{3} x_n \left(W_4\right)^{-n3} = x_0 + jx_1 - x_2 - jx_3$$

NOTICING REPEATED TERMS

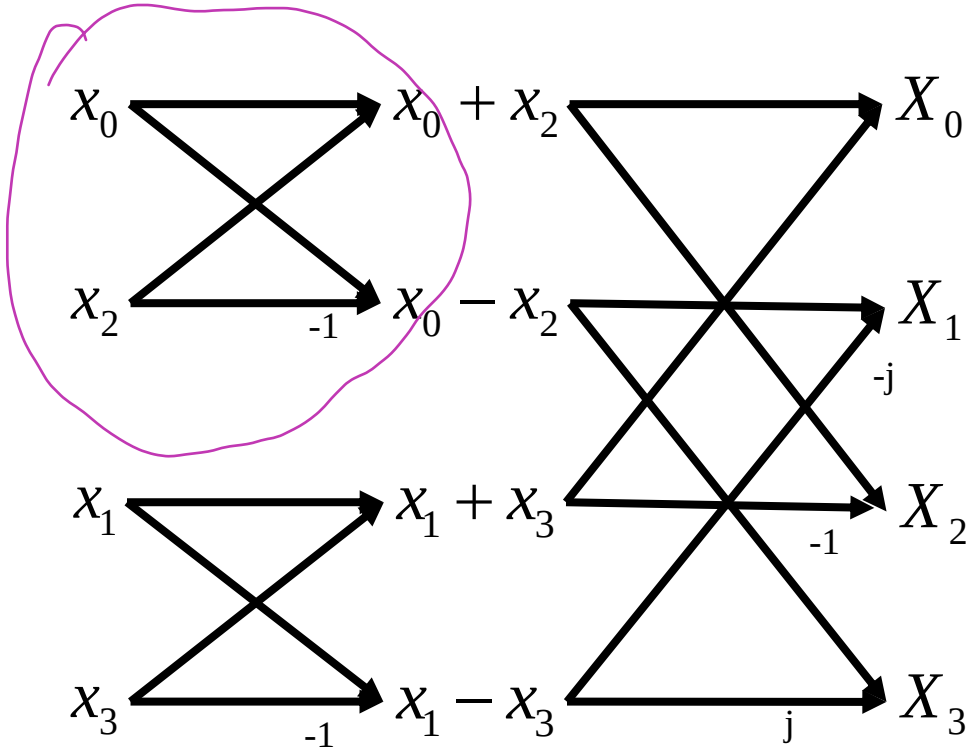
- By rearranging terms, we can create a more 'efficient' computation

$$X_0 = [x_0 + x_2] + [x_1 + x_3]$$

$$X_1 = [x_0 - x_2] - j[x_1 - x_3]$$

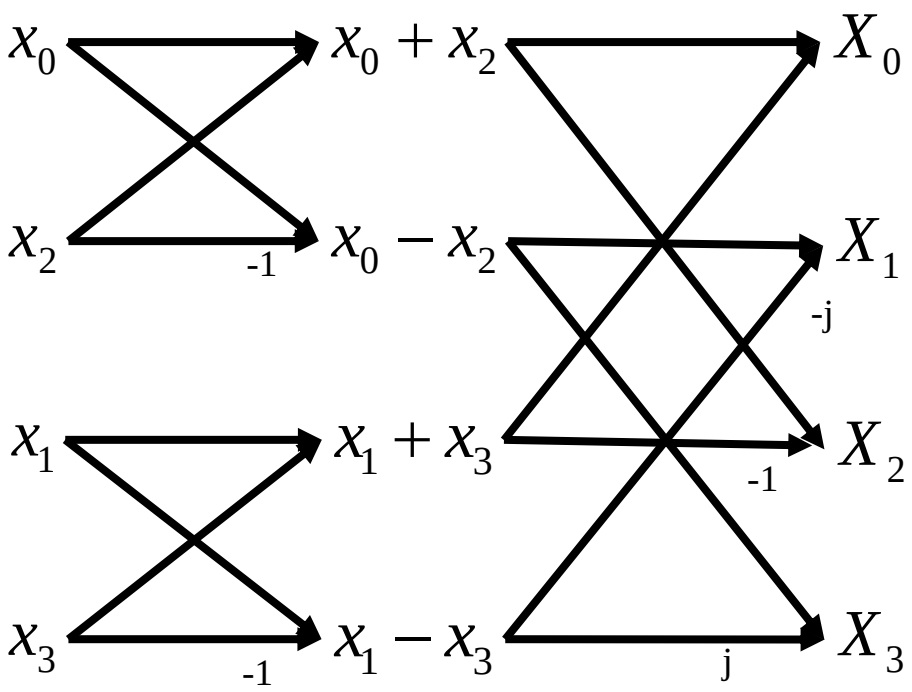
$$X_2 = [x_0 + x_2] - [x_1 + x_3]$$

$$X_3 = [x_0 - x_2] + j[x_1 - x_3]$$



NOTICING REPEATED TERMS

- View this result in terms of frequencies
  - Split the data
  - DC in both halves gives DC of total
  - High frequency of one half gives intermediate frequency of total
  - Comparison of two halves gives highest frequency



GENERALIZING TO N ELEMENTS = RADIX-2 FFT DECIMATION IN TIME דילול בזמן

- **Radix** (**=root**) is the most popular with sequences of length $N = 2^r$, $r \in \mathbb{Z}$.
- We **decimate** (split) the sequence $x[n]$ to two sequences of even and odd indices:

$$f_1[n] = x[2n]$$

$$x[n] = f_1 + f_2$$

$$f_2[n] = x[2n+1] \quad 0 \le n \le \frac{N}{2} - 1$$

- We will say that sequences $f_1[n]$ and $f_2[n]$ are decimated in time out of $x[n]$.
- We will write $X[k]$ as:

$$X[k] = \sum_{n=0}^{N-1} x[n](W_N)^{-nk} =$$

DFT

$$\sum_{n \text{ even}} x[n]W_N^{-kn} + \sum_{m \text{ odd}} x[n]W_N^{-kn} =$$

$$= \sum_{r=0}^{\frac{N}{2}-1} x[2r](W_N)^{-2rk} + \sum_{r=0}^{\frac{N}{2}-1} x[2r+1](W_N)^{-(2r+1)k}$$

GENERALIZING TO N ELEMENTS = RADIX-2 FFT

- Assume N is even

$r = 2^{\checkmark}$

$$X[k] = \underbrace{\sum_{r=0}^{\frac{N}{2}-1} f_1[r](W_N)^{-2rk}}_{even} + W_N^{-k} \underbrace{\sum_{r=0}^{\frac{N}{2}-1} f_2[r](W_N)^{-2rk}}_{odd}, k = 0, \dots, N-1$$

while $W_N^{-2kr} = e^{-j\frac{2\pi}{N}2kr} = e^{-j\frac{2\pi}{N/2}kr} = W_{N/2}^{-kr}$

If N is even!

$$X[k] = \underbrace{\sum_{r=0}^{\frac{N}{2}-1} x[2r](W_{N/2})^{-rk}}_{F_1[k]} + W_N^{-k} \underbrace{\sum_{r=0}^{\frac{N}{2}-1} x[2r+1](W_{N/2})^{-2rk}}_{F_2[k]}, k = 0, \dots, N-1$$

DFT of N/2 length *even*

DFT of N/2 length *odd*

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

We obtain: $= F_1[k] + W_n^{-k} F_2[k], \quad 0 \le k \le N-1$

Note: $F_1[k]$ and $F_2[k]$ are N/2 points DFT of $f_1[r]$ and $f_2[r]$

GENERALIZING TO N-ELEMENTS

- Since $f_1[n]$ and $f_2[n]$ are periodic with period $N/2$:

$$F_1\left[k+\frac{N}{2}\right] = F_1[k], \qquad F_2\left[k+\frac{N}{2}\right] = F_2[k]$$

- In addition, $W_N^{-\left(k+\frac{N}{2}\right)} = W_n^{-k} \cdot W_n^{-\frac{N}{2}} \rightarrow W_N^{-\left(k+\frac{N}{2}\right)} = -W_n^{-k}$

=-1

- Therefore, we can decimate the $X[k]$ to two regions:

$$\begin{cases} X[k] = F_1[k] + W_N^{-k} F_2[k] & 0 \le k \le \frac{N}{2} - 1 \\ X\left[k+\frac{N}{2}\right] = F_1[k] - W_N^{-k} F_2[k] & 0 \le k \le \frac{N}{2} - 1 \end{cases}$$

COMPUTATIONAL COMPLEXITY

$$\begin{cases} X[k] = F_1[k] + W_N^{-k} F_2[k] & 0 \le k \le \frac{N}{2} - 1 \\ X\left[k + \frac{N}{2}\right] = F_1[k] - W_N^{-k} F_2[k] & 0 \le k \le \frac{N}{2} - 1 \end{cases}$$

One DFT of length N: N^2 complex multiplications calculated from 2 DFTs of length $\frac{N}{2} \rightarrow 2\left(\frac{N}{2}\right)^2$

And $N/2$ multiplications with exponential function $= N/2$

In summary: $N^2/2 + N/2$ which is approximately half of the calculations when N is large.

COMPUTATIONAL COMPLEXITY

- We will add G

$$G_1[k] = F_1[k]$$
$$G_2[k] = W_N^{-k}[k]F_2[k]$$

- We can write

$$\begin{cases} X[k] = G_1[k] + G_2[k] \\ X\left[k + \frac{N}{2}\right] = G_1[k] - G_2[k] \quad 0 \le k \le \frac{N}{2} - 1 \end{cases}$$

- The addition and the subtraction are actually DFT with length of 2

$$W_2^0 = 1, \qquad W_2^{-1} = -1$$

$x(n) \rightarrow$

DECIMATION-IN-TIME ALGORITHM: FIRST STEP

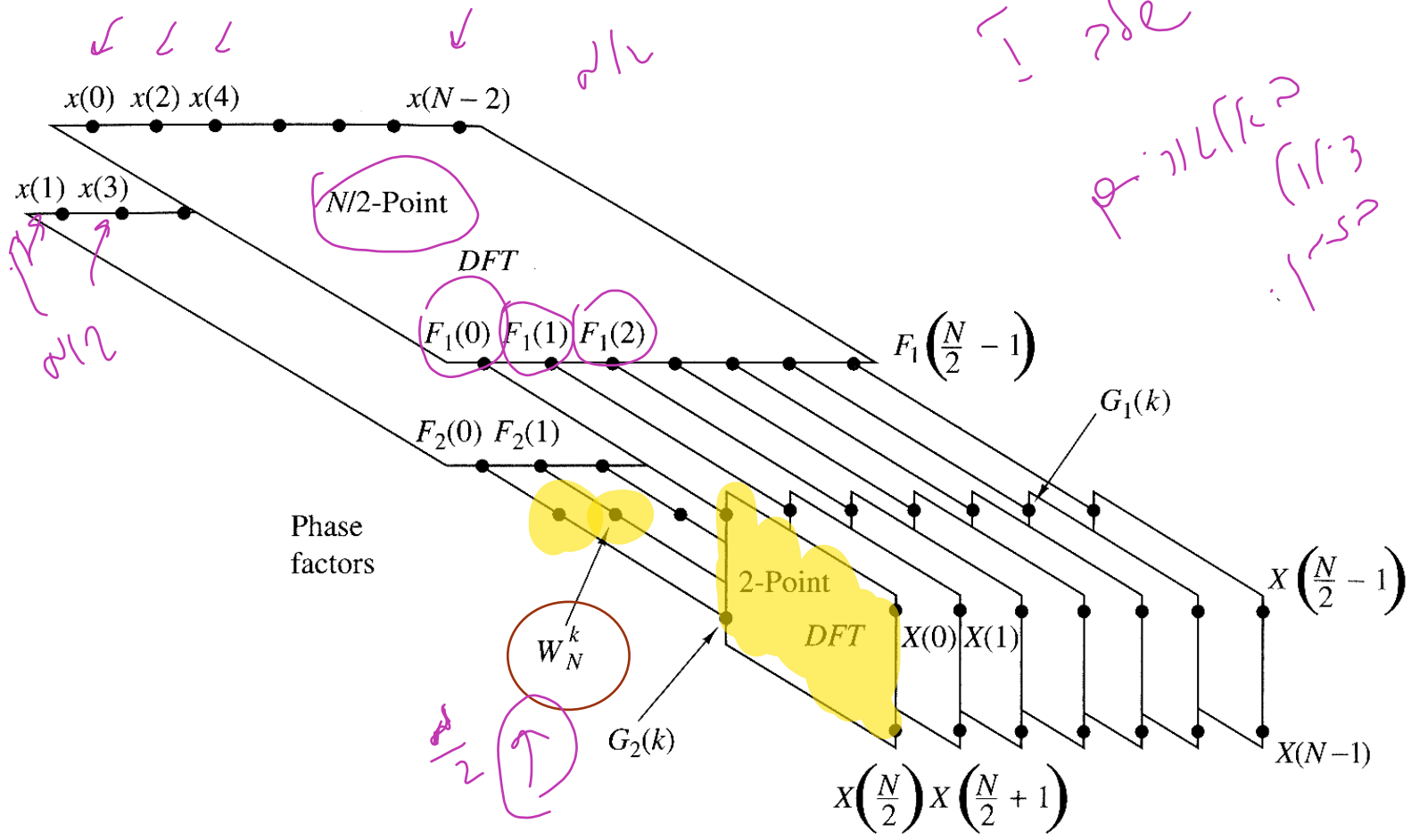


Figure 8.1.4 First step in the decimation-in-time algorithm.

- Decimation of DFT length N to DFT length N/2.
- In each step there are N/2 multiplications by exponent.
- The decimation-in-time is repeated until we reach which do not include multiplications.
- Overall $\log_2 N$ steps
- $(N/2)\log_2 N$ multiplications

THE RADIX-2 FFT AS BUILDING BLOCKS N=8

- FFT: Fast Fourier Transform

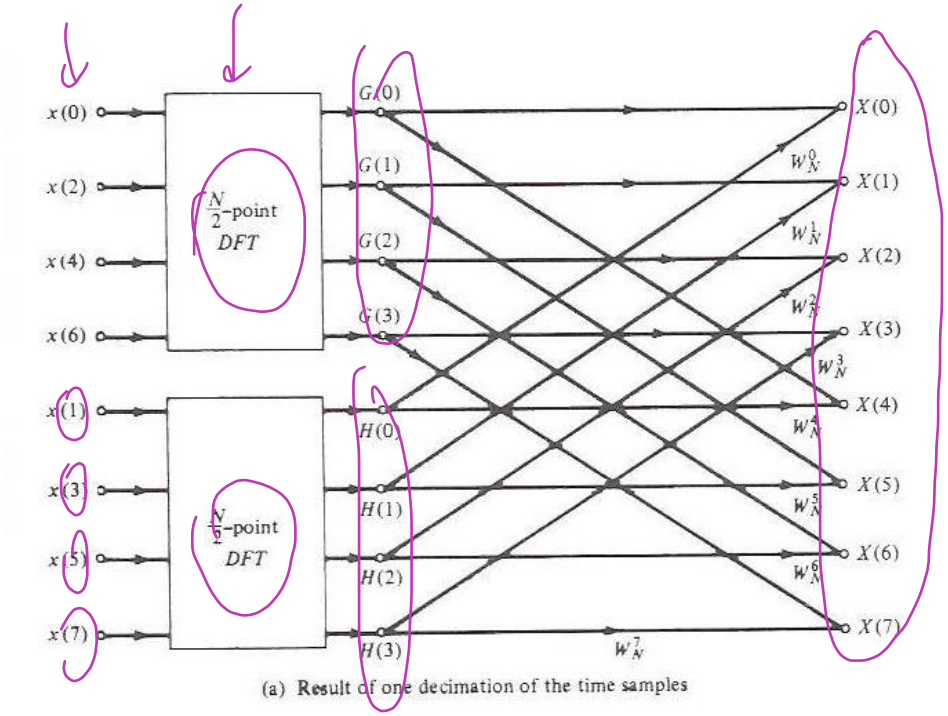
$$X[k] = G[k] + (W_N)^k H[k]$$

$$G[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r] \left(W_{\frac{N}{2}}\right)^{rk}$$

$$H[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] \left(W_{\frac{N}{2}}\right)^{rk}$$

even

odd



(a) Result of one decimation of the time samples

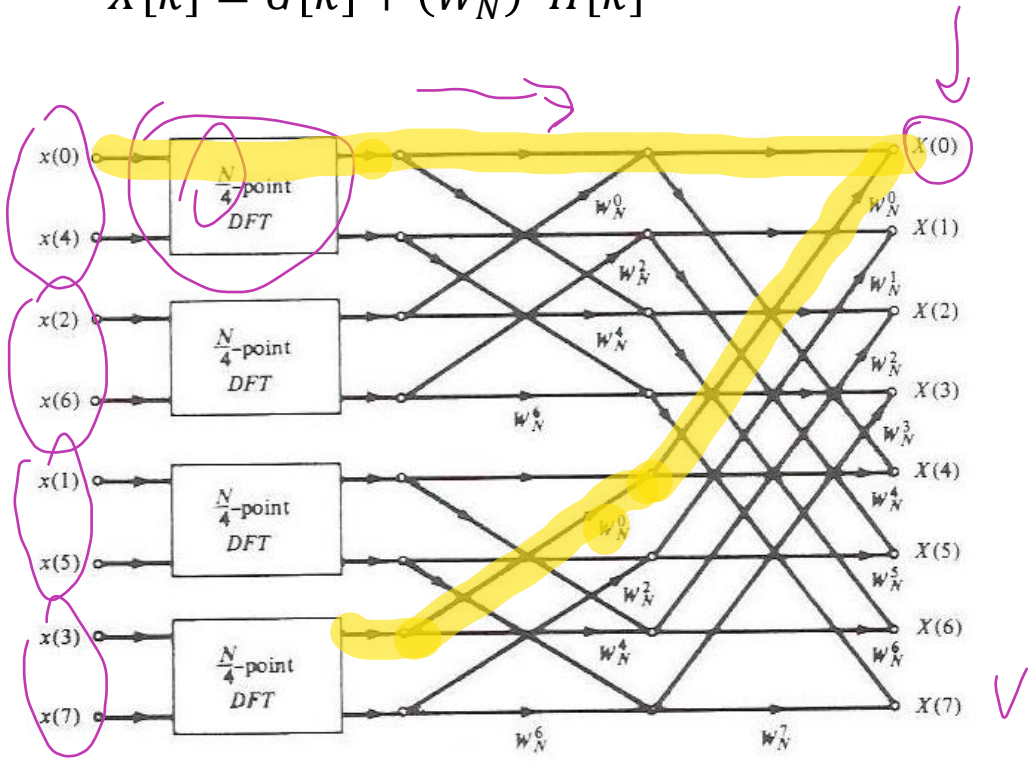
THE RADIX-2 FFT AS BUILDING BLOCKS N=8

- FFT: Fast Fourier Transform

$$X[k] = G[k] + (W_N)^k H[k]$$

$$G[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r] \left(W_{\frac{N}{2}}\right)^{rk}$$

$$H[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] \left(W_{\frac{N}{2}}\right)^{rk}$$



(b) Result of applying two decimations

0 → N-1

THE RADIX-2 FFT AS BUILDING BLOCKS N=8 $= 2^{r=3}$

- FFT: Fast Fourier Transform

$$X[k] = G[k] + (W_N)^k H[k]$$

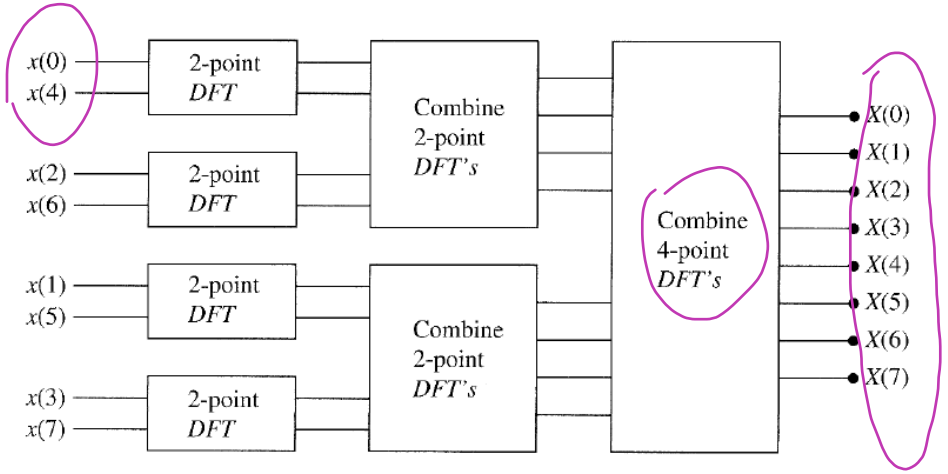
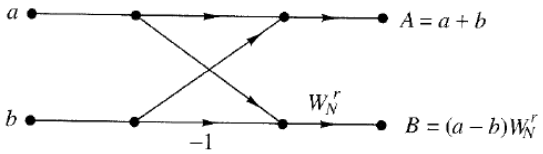


Figure 8.1.5 Three stages in the computation of an $N = 8$-point DFT.

Figure 8.1.10 Basic butterfly computation in the decimation-in-frequency FFT algorithm.



$$G[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r] \left(W_{\frac{N}{2}}\right)^{rk}$$

$$H[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] \left(W_{\frac{N}{2}}\right)^{rk}$$

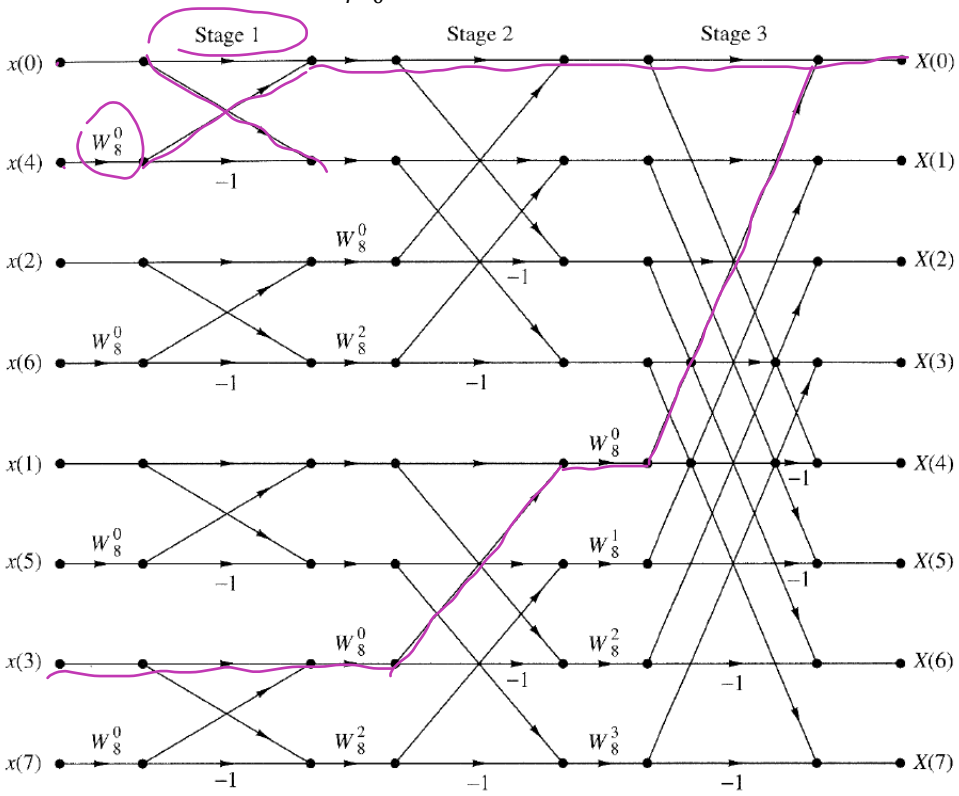
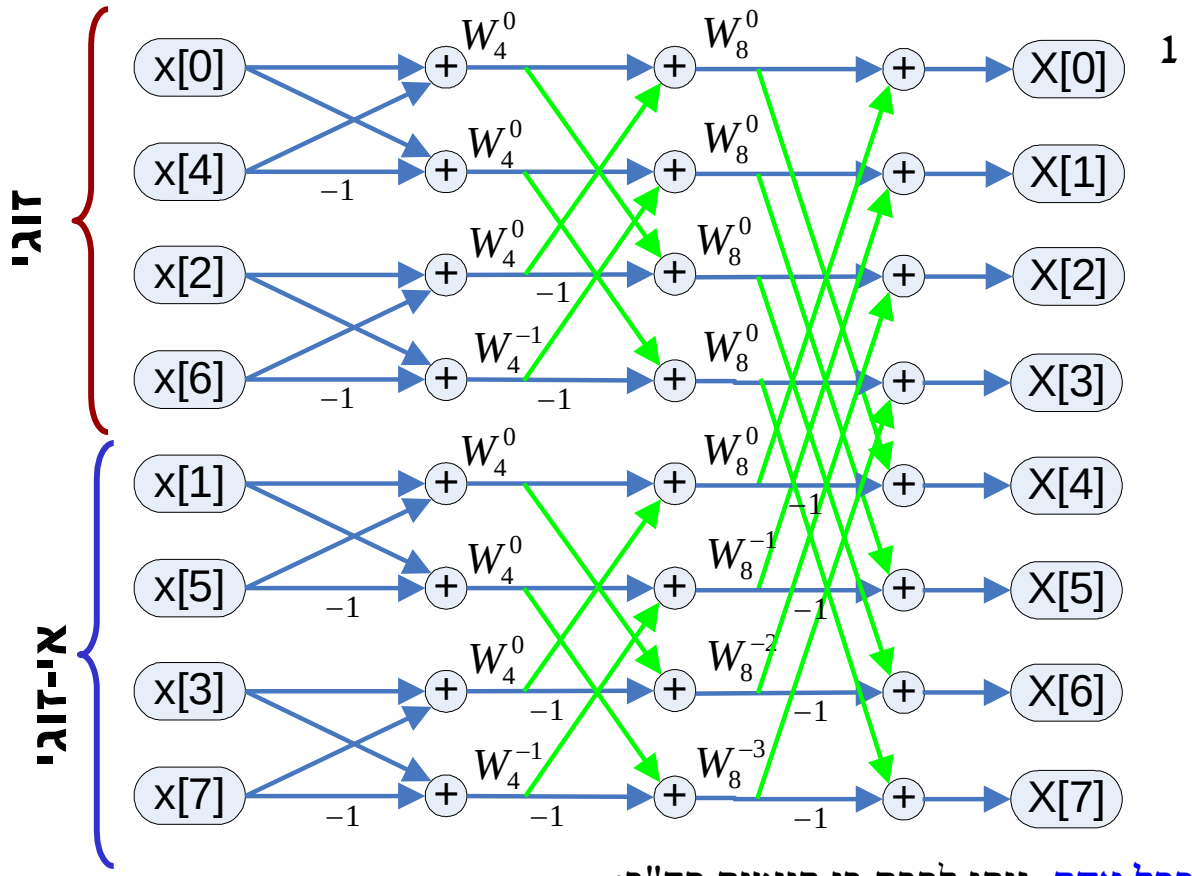


Figure 8.1.6 Eight-point decimation-in-time FFT algorithm.

EXAMPLE: 8-POINT DECIMATION-IN-TIME FFT



1. בצע/י דילול בזמן
2. מהי כמות השלבים?
3. מהי כמות הפרפרים בכל שלב?
4. מהי סיבוכיות החישוב עבור FFT?

<u>פתרון:</u>

2. ניתן לראות כי קיבלנו $\text{steps} = log_2 N$

3. אם נספור את כמות הפרפרים בכל שלב,

נראה כי כמותם היא $\frac{N}{2}$ בכל שלב.

4. מכיוון ש"פרפר" אחד מוגדר על ידי **שתי** פעולות **חיבור** ופעולת **כפל אחת**, ניתן לסכם כי קיימים סה"כ:

$N \log_2 N$ סכומים קומפלקסיים + $\frac{N}{2} log_2 N$ מכפלות קומפלקסיות.

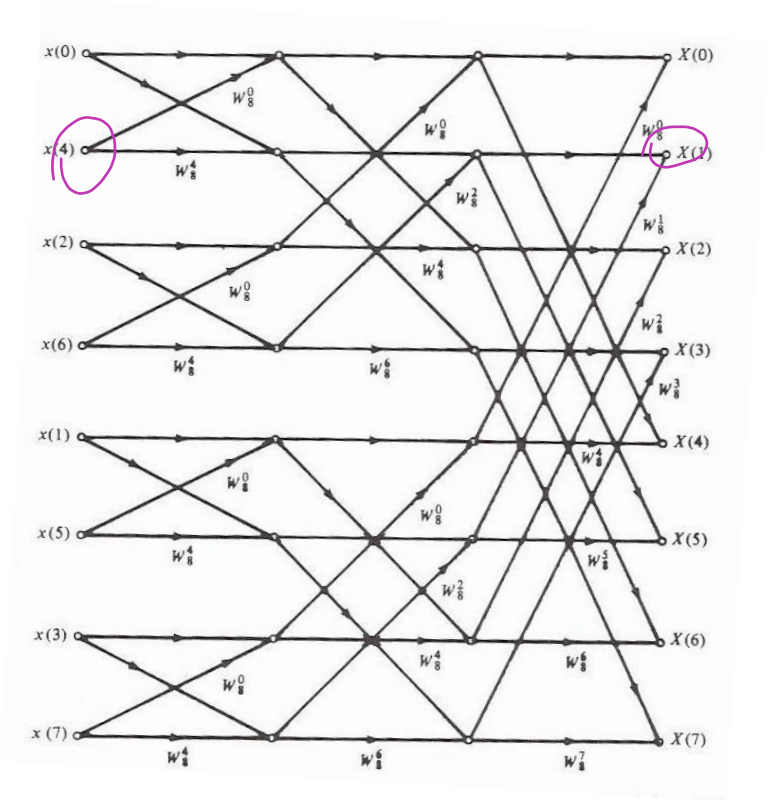
THE FFT AS BUILDING BLOCKS

- FFT: Fast Fourier Transform

$$X[k] = G[k] + (W_N)^k H[k]$$

$$G[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r] \left(W_{\frac{N}{2}}\right)^{rk}$$

$$H[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] \left(W_{\frac{N}{2}}\right)^{rk}$$



IN-PLACE COMPUTATIONS VIA BIT REVERSAL: 3-STAGE EVEN-ODD DECOMPOSITION

For **N=8**

input	Binary numbers	output
$x[0]$	$000 \rightarrow 000$	$x[0]$
$x[1]$	$001 \rightarrow 100$	$x[4]$
$x[2]$	$010 \rightarrow 010$	$x[2]$
$\vdots$	$\vdots$	$\vdots$
$x[7]$	$111 \rightarrow 111$	$x[7]$

$(n_2 n_1 n_0)$	$\rightarrow$	$(n_1 n_0 n_2)$	$\rightarrow$	$(n_0 n_1 n_2)$
(0 0 0)	$\rightarrow$	(0 0 0)	$\rightarrow$	(0 0 0)
(0 0 1)	$\rightarrow$	(0 1 0)	$\rightarrow$	(1 0 0)
(0 1 0)	$\rightarrow$	(1 0 0)	$\rightarrow$	(0 1 0)
(0 1 1)	$\rightarrow$	(1 1 0)	$\rightarrow$	(1 1 0)
(1 0 0)	$\rightarrow$	(0 0 1)	$\rightarrow$	(0 0 1)
(1 0 1)	$\rightarrow$	(0 1 1)	$\rightarrow$	(1 0 1)
(1 1 0)	$\rightarrow$	(1 0 1)	$\rightarrow$	(0 1 1)
(1 1 1)	$\rightarrow$	(1 1 1)	$\rightarrow$	(1 1 1)

MSB $\longleftrightarrow$ LSB

Most **S**ignificant **B**it

Least **S**ignificant **B**it

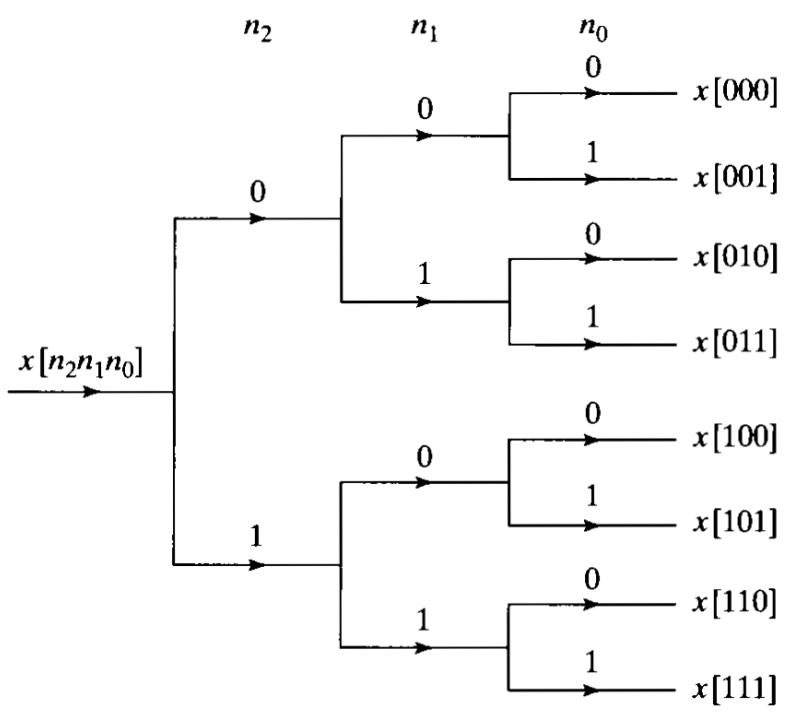


Figure 9.12 Tree diagram depicting normal-order sorting.

RECURSIVE CALCULATION OF THE EXPONENT

חישוב רקורסיבי של האקספוננט:

בחישוב DFT או FFT נדרשים איברי מערך פורייה
חשוב. ניתן לחשב אותם מראש ולבנות בטבלאות

- While calculating DFT or FFT, one needs elements of the Fourier matrix, W_N^{kn}

או לחשב באופן רקורסיבי:

$W_N^{-(n+1)} = W_N^{-n} \cdot W_N^{-1}$

1) One can calculate them in advance and prepare in tables or 2) Calculate in a recursive manner $W_N^{-(n+1)} = W_N^{-n}$
W_N^{-1} - requires regular multiplications
W_N^{-1} as compared to cos, sin – disadvantage is in accumulating error

כלומר נדרשים רק W_N^{-1}, ומאז מבצעים רב לקבל כל המקדמים.

$8 = N$

SURVEY: RADIX 2 FFT

$\frac{N}{2} = 4$ $\frac{N}{2} = 4$

- EasyPolls:

How many complex multiplications are required to compute X[k] from x[n] using the FIRST STAGE ONLY of an N=8, Radix-2 FFT?

- 64 $8^2 = N^2 \rightarrow$ DFT ✗
- 36 DFT DFT $4^2 + 4^2 + 4 \leftarrow$ ✓
- 32 $4^2 + 4^2 = 32$ מכפלות
- 12 מכפלות $\frac{N}{2} \log_2 N = \frac{8}{2} \cdot \log_2 8 = 12$ ✗ full FFT

results vote

DECIMATION IN FREQUENCY

נניח כי אורך הסיגנל במישור התדר הוא N כאשר: $N = 2^r$, ו r מספר טבעי. את התמרת הIDFT נקבל באופן הבא: $X^d[k] = \sum_{n=0}^{N-1} x[n] {W_N}^{-nk}$

כעת נבצע את הפירוק לשני סכומים: $X^d[k] = \sum_{n=0}^{\frac{N}{2}-1} x[n] {W_N}^{-kn} + \sum_{n=\frac{N}{2}}^{N-1} x[n] {W_N}^{-kn}$

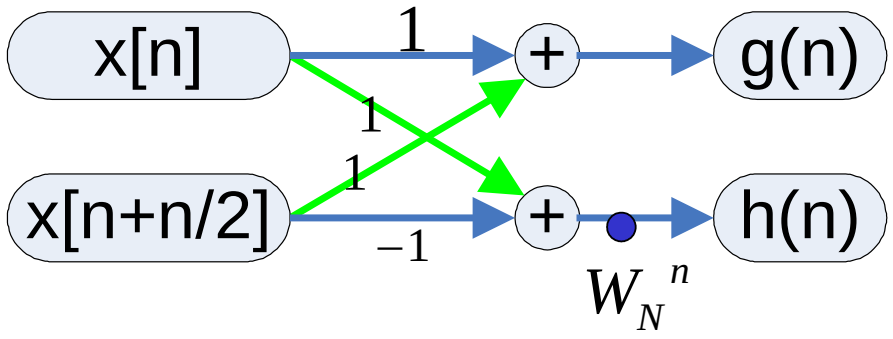
$$= \sum_{n=0}^{\frac{N}{2}-1} x[n] {W_N}^{-kn} + \sum_{n=0}^{\frac{N}{2}-1} x\left[n + \frac{N}{2}\right] {W_N}^{-k(n+N/2)} =$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x[n] + (-1)^k x\left[n + \frac{N}{2}\right] {W_N}^{-kn} =$$

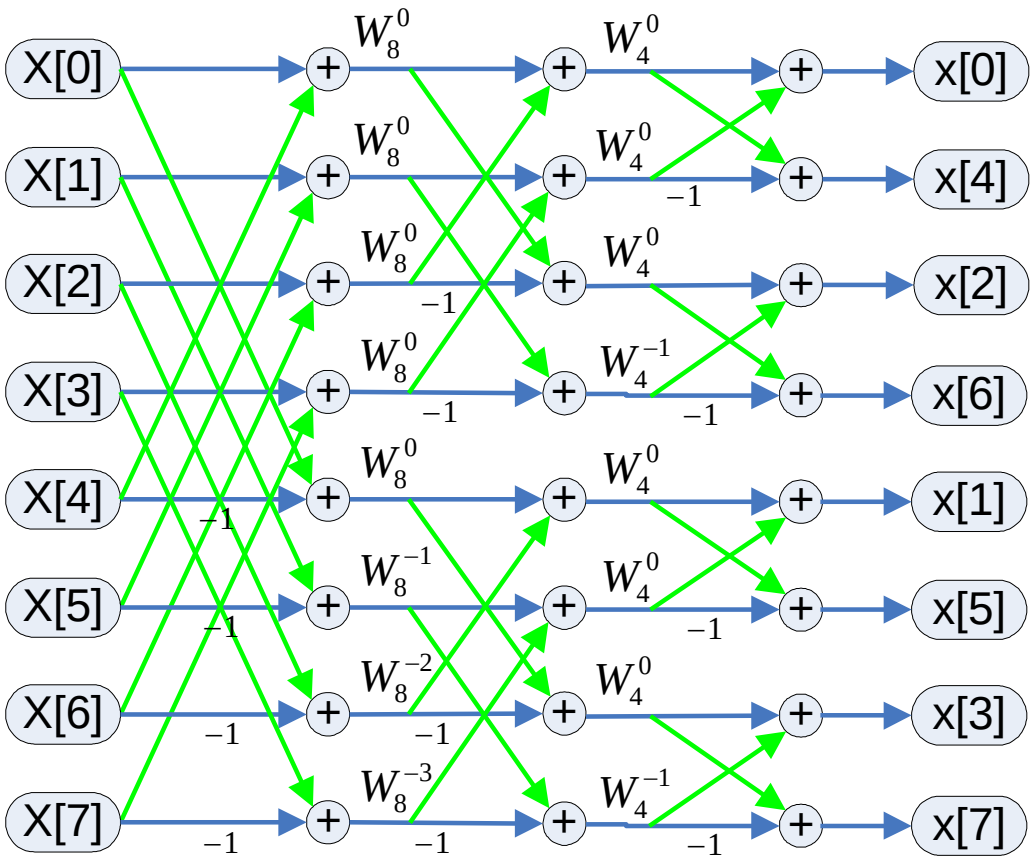
שני איברי ההתמרה הסמוכים נראים באופן הבא: $X^d[2m] = \sum_{n=0}^{\frac{N}{2}-1} \left[x[n] + x\left[n + \frac{N}{2}\right]\right] {W_{N/2}}^{-mn} = \sum_{n=0}^{\frac{N}{2}-1} g[n] {W_{N/2}}^{-mn}$

$$X^d[2m+1] = \sum_{n=0}^{\frac{N}{2}-1} \left[x[n] - x\left[n + \frac{N}{2}\right]\right] {W_N}^{-n} {W_{N/2}}^{-mn} = {W_N}^{-n} \sum_{n=0}^{\frac{N}{2}-1} h[n] {W_{N/2}}^{-mn}$$

BUTTERFLY RELATION FOR IFFT



EXAMPLE: IFFT BUTTERFLY RELATION FOR N=8



דוגמא: N=8:

בצע/י דילול בתדר.

פתרון:

CT - DIVIDE AND CONQUER APPROACH

שיטת הפרד ומשול לפי קולייי וטוקיי

- We assume that N is not a prime number (מספר ראשוני) and can be represented as a multiplication of two other two integers $\boxed{N = ML}$
- If N cannot be represented by multiplication $N = ML \rightarrow$ apply **zero padding**
- Vector $x[n]$ can be presented as a matrix of $L \times M$ or $M \times L$ elements:

Vector $\quad x[n],\ 0 \le n \le N-1$

Matrix $\quad x[l,m],\ 0 \le l \le L-1, \quad 0 \le m \le M-1$

row column

$x[n]$ $\quad 0 \le n \le N-1 \rightarrow N = ML$

$n = l + mL$ column distributions

$n = m + lM$ row distributions

CT: 2D DATA ARRAY TO STORE THE SEQUENCE X(N)

x(n)

$$N = LM$$

$n \longrightarrow$	0	1		…	$N-1$
	$x(0)$	$x(1)$	$x(2)$	…	$x(N-1)$

(a)

row index l \ column index m	0	1		$M-1$
0	$x(0, 0)$	$x(0, 1)$	…	
1	$x(1, 0)$	$x(1, 1)$	…	
2	$x(2, 0)$	$x(2, 1)$	…	
⋮	⋮	⋮	…	⋮
$L-1$			…	

(b)

$l \leq L-1$ and $0 \leq m \leq M-1$

Figure 8.1.1 Two dimensional data array for storing the sequence $x(n)$, $0 \leq n \leq N-1$.

CT: ARRANGEMENT OF DATA ARRAY ROW-WISE

For example, suppose that we select the mapping

$$n = Ml + m \tag{8.1.9}$$

This leads to an arrangement in which the first row consists of the first M elements of $x(n)$, the second row consists of the next M elements of $x(n)$, and so on, as illustrated in Fig. 8.1.2(a). On the other hand, the mapping

$$n = l + mL \tag{8.1.10}$$

stores the first L elements of $x(n)$ in the first column, the next L elements in the second column, and so on, as illustrated in Fig. 8.1.2(b).

Row-wise $\quad n = Ml + m$

$l \backslash m$	0	1	2		$M-1$
0	$x(0)$	$x(1)$	$x(2)$	$\dots$	$x(M-1)$
1	$x(M)$	$x(M+1)$	$x(M+2)$	$\dots$	$x(2M-1)$
2	$x(2M)$	$x(2M+1)$	$x(2M+2)$	$\dots$	$x(3M-1)$
	$\vdots$	$\vdots$	$\vdots$	$\dots$	$\vdots$
$L-1$	$x((L-1)M)$	$x((L-1)M+1)$	$x((L-1)M+2)$	$\dots$	$x(LM-1)$

CT: ARRANGEMENT OF DATA ARRAY COLUMN-WISE

Column-wise $\quad n = l + mL$

$l \backslash m$	0	1	2		$M-1$
0	$x(0)$	$x(L)$	$x(2L)$	$\cdots$	$x((M-1)L)$
1	$x(1)$	$x(L+1)$	$x(2L+1)$	$\cdots$	$x((M-1)L+1)$
2	$x(2)$	$x(L+2)$	$x(2L+2)$	$\cdots$	$x((M-1)L+2)$
	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
$L-1$	$x(L-1)$	$x(2L-1)$	$x(3L-1)$	$\cdots$	$x(LM-1)$

CT- DIVIDE AND CONQUER APPROACH IN FREQUENCY DOMAIN

- The same approach is applied to $X[k]$ in frequency domain:

Vector $X[k], \quad 0 \le k \le N-1$

Matrix $X[p,q], \; 0 \le p \le L-1, \quad 0 \le q \le M-1$

row column

mapping $k = p + qL$ is an arrangement in column

mapping $k = q + pM$ is an arrangement in row

MATRIX FORMULATION OF DFT

$N = M \cdot L$

- From the definition: $X[k] = \sum_{n=0}^{N-1} x[n] W_N^{-kn}$, $0 \le k \le N-1$
- Let arrange columns in $x[n]$ as $n = l + mL$
- Let arrange rows in $X[k]$ as $k = q + pM$

DFT

$$X[p,q] = \sum_{m=0}^{M-1} \sum_{l=0}^{L-1} x[l,m] W_N^{-(\overset{k}{q+pM})(\overset{n}{l+mL})}$$

We simplify the exponent: $W_N^{(q+pM)(l+mL)} = W_N^{ql} W_N^{mLq} W_N^{Mpl} W_N^{MLmp}$ $= 1$

- $W_N^{MLmp} = W_N^{Nmp} = 1$; $W_N^{mqL} = W_{N/L}^{mq} = W_M^{mq}$; $W_N^{Mpl} = W_{N/M}^{pl} = W_L^{pl}$

MATRIX FORMULATION OF DFT

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

( ) – DFT length L

[ ] - DFT length M on row l of x

MATRIX FORMULATION OF DFT

- We divide the calculation to steps:

1) Calculate M-point DFT on rows of $x[l,m]$

$N = M \cdot L$

DFT on index m: $F[l,q] = \sum_{m=0}^{M-1} x[l,m] W_M^{-mq}$, $0 \le l \le L-1$

2) Calculate multiplications

$$G[l,q] = W_N^{-ql} F[l,q], \qquad 0 \le l \le L-1, \qquad 0 \le q \le M-1$$

3) Calculate L-point DFT on columns of $G[l,q]$

$$X[p,q] = \sum_{l=0}^{L-1} G[l,q] W_L^{-lp}, \qquad 0 \le q \le M-1$$

MATRIX FORMULATION OF DFT EXAMPLE: L=5, M=3, N=LM=15

- x[l,m] is the matrix 5x3
- Step 1: DFT on rows of x[l,m]
- Step 2: Multiply by W_N
- Step 3: DFT on columns of the result

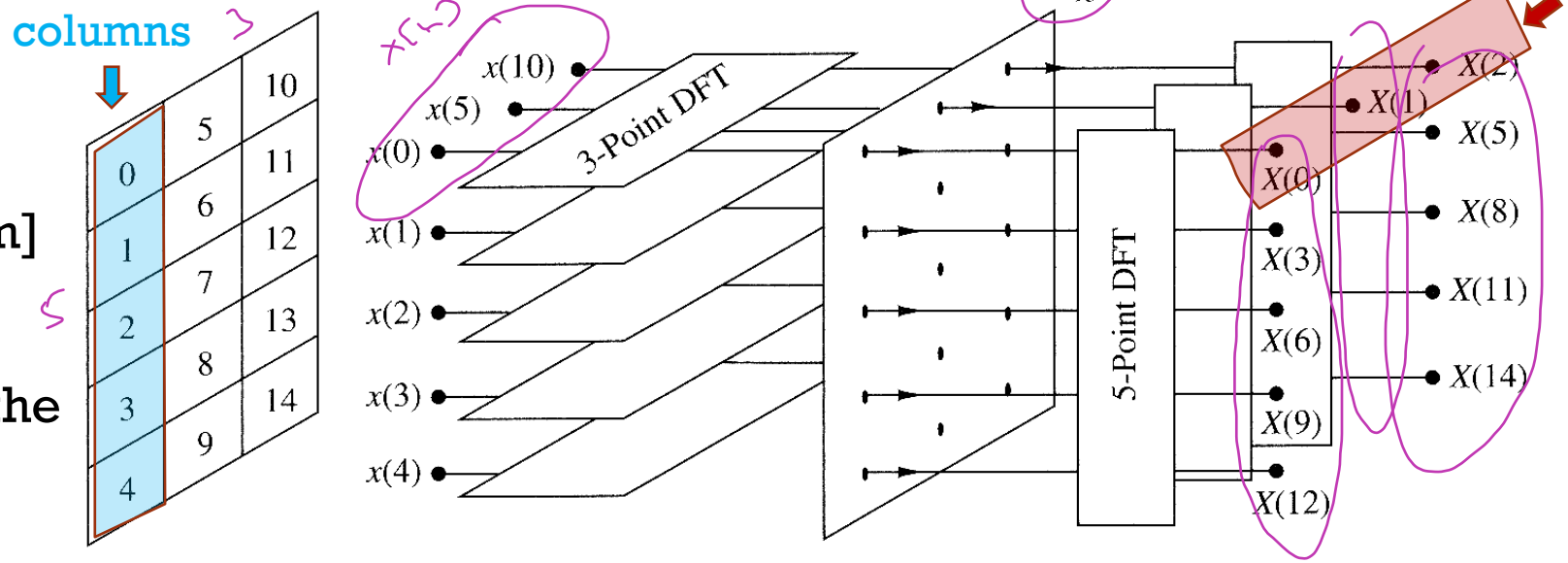


Figure 8.1.3 Computation of $N = 15$-point DFT by means of 3-point and 5-point DFTs.

MATRIX FORMULATION OF DFT: COMPUTATIONAL EFFICIENCY

Multiplications:

- Step 1: L DFT length of M = LM^2
- Step 2: LxM multiplications
- Step 3: M DFT length of L = ML^2

- $LM^2+ML^2+LM= LM(M+L+1)$

  $= N(M+L+1)$

- As compared to N^2 in direct computation
- In our example: M=3, L=5, N=15

Direct computation N^2: $15^2=225$

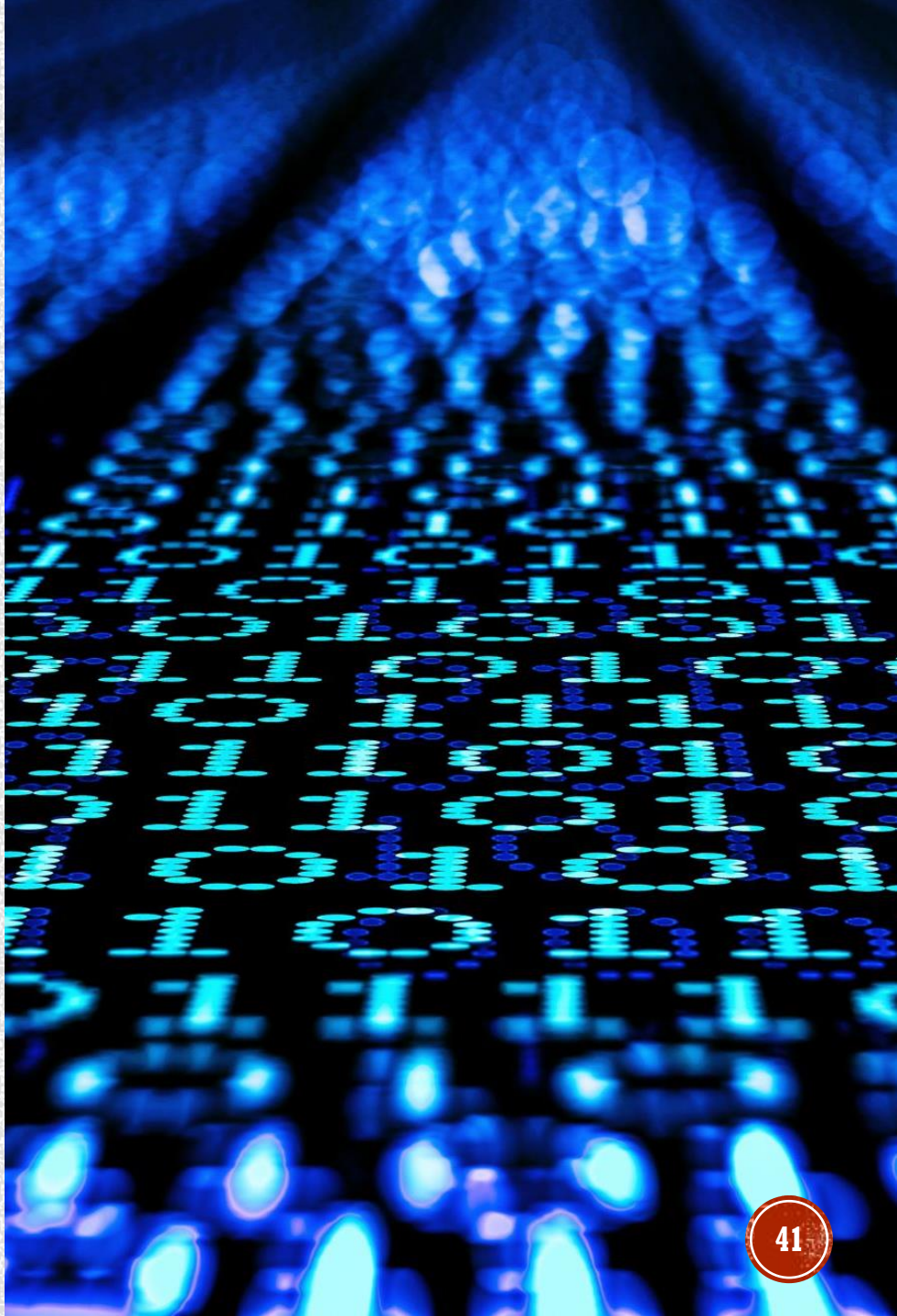
CT decimation $15(5+3+1)=135$

MATRIX FORMULATION OF DFT

- Calculation of $X[k]$ from $X[p,q]$ is performed by taking rows and converting them to one vector.
- If N is decomposed to more than two multiplications, one can continue the decimation process: $N = r_1 r_2 \dots r_v$
- The decimation process can take place $(v-1)$ times.

באופן כ' מתפרק לגורמים יש שני מרכיבים
אפשר להמשיך בתהליך הפירוק וכך להפוך אותו
לפעולה:

$N = r_1 r_2 \dots r_v$
הפירוק יתבצע $(v-1)$ פעמים



NOTES

- Let analyze the rows of matrices x, X:
- In X – the rows represent $X[k]$
- In x – the rows represent $x[n]$ but after decimation each 5 samples

-> Therefore, this algorithm is named **decimation in time**. If we arrange by rows x in time, we obtain **decimation in frequency** because the rows of $X[k]$ will be decimated.

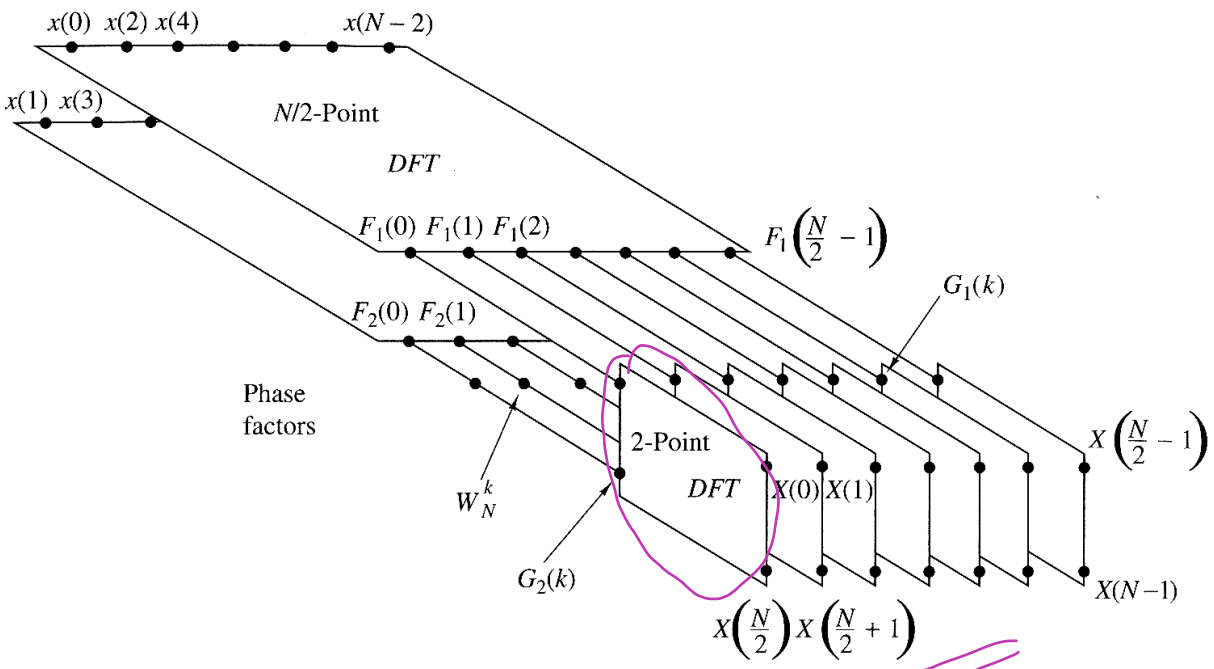


Figure 8.1.4 First step in the decimation-in-time algorithm.

When $N = a^r$, a and $r \in \mathbb{Z}$, it is named <u>Radix-a FFT</u> if a=2-> <u>Radix-2 FFT</u>

root

SURVEY: CT ALGORITHMS

- EasyPolls:

How many complex multiplications requires the CT algorithms for N=3x3x3?

$N=27$

- 63
- 297
- 351
- 414
- 729

results vote

SURVEY: CT ANSWER

$N^2=27 = 3^{x^9}$

- L=3, M=9
- CT: $LM(L+M+1)=LM^2+ML^2+ML$

- 63 = wrong answer: $LxM=3x3$->$LM^2+ML^2+ML=3x3^2+3x3^2+3x3$
- 297 = correct answer: $LxM=3x9$->$LM^2+ML^2+ML=3x9^2+9x3^2+3x9=3x63+81+27$

  $3x3^2+3x3^2+3x3=63$

27

- 351 = wrong answer: $LxM=3x9$->$LM^2+ML^2+ML=3x9^2+9x3^2+3x9=3x81+81+27$
- 414 = wrong answer: $LxM=3x9$->351+63
- 729 = wrong answer: direct calculation N^2->27^2

direct calculation

FFT: MATRIX FORMULATION

- **Single stage of decimation of FFT is $N = 2 \times {}^{N}/_{2}$**

$$\begin{cases} X[k] = F_1[k] + W_N^{-k} F_2[k] \\ X\left[k + \frac{N}{2}\right] = F_1[k] - W_N^{-k} F_2[k] \end{cases}, \qquad k = 0, \dots, \frac{N}{2} - 1$$

We will write this equation in **matrix form**:

$$\underline{X_1} = \underline{F_1} + \underline{\underline{W_N}}\,\underline{F_2}$$

$$\underline{X_2} = \underline{F_1} - \underline{\underline{W_N}}\,\underline{F_2}$$

While $\underline{X_1}, \underline{X_2}, \underline{F_1}, \underline{F_2}$ are vectors of length $N/2$ and the matrix $\underline{\underline{W_N}}$ is the diagonal matrix

$$\underline{\underline{W_N}} = \begin{array}{cc} & N/2 \\ N/2 & \begin{bmatrix} W_n^0 & & 0 \\ & \ddots & \\ 0 & & W_N^{-\left(\frac{N}{2}-1\right)} \end{bmatrix} \end{array}$$

ADD TWO EQUATIONS

- We add two equations: $\overset{1}{\underset{N}{\begin{bmatrix} \underline{X_1} \\ \underline{X_2} \end{bmatrix}}} = \begin{bmatrix} \underline{F_1} \\ \underline{F_1} \end{bmatrix} + \begin{bmatrix} \underline{\underline{W_N}} & 0 \\ 0 & -\underline{\underline{W_N}} \end{bmatrix} \begin{bmatrix} \underline{F_2} \\ \underline{F_2} \end{bmatrix}$ -this is inefficient form since $\underline{F_1}$, $\underline{F_2}$ appear twice

- An efficient form: $\begin{bmatrix} \underline{X_1} \\ \underline{X_2} \end{bmatrix} = \begin{bmatrix} & A & \end{bmatrix} \begin{bmatrix} \underline{F_1} \\ \underline{F_2} \end{bmatrix}$, what is A? $A = \begin{bmatrix} \underline{\underline{I}} & \underline{\underline{W_N}} \\ \underline{\underline{I}} & -\underline{\underline{W_N}} \end{bmatrix}$

- Now, we can calculate $\underline{F_1}, \underline{F_2}$ by decomposition $N/2 = 2 \times N/4$

$$F_1 = \overset{\text{N/4xN/4}}{\underset{\text{N/2}}{\begin{bmatrix} \underline{\underline{I}} & \overset{\text{N/2}}{\underline{\underline{W_{N/2}}}} \\ \underline{\underline{I}} & -\underline{\underline{W_{N/2}}} \end{bmatrix}}} \times \overset{1}{\begin{bmatrix} \underline{F_{11}} \\ \underline{F_{12}} \end{bmatrix}}$$

N/4

ADD TWO EQUATIONS

- $$\underline{X} = \overset{N}{\underbrace{\begin{bmatrix} \underline{\underline{I}} & \overset{N}{\underline{\underline{W_N}}} \\ \underline{\underline{I}} & \underline{\underline{-W_N}} \end{bmatrix}}_{\text{Matrix A}}} \begin{bmatrix} \underline{\underline{I}} & \underline{\underline{W_{N/2}}} & 0 & 0 \\ \underline{\underline{I}} & \underline{\underline{-W_{N/2}}} & 0 & 0 \\ 0 & 0 & \underline{\underline{I}} & \underline{\underline{W_{N/2}}} \\ 0 & 0 & \underline{\underline{I}} & \underline{\underline{-W_{N/2}}} \end{bmatrix} \overset{1}{\begin{bmatrix} \underline{F_{11}} \\ \underline{F_{12}} \\ \underline{F_{21}} \\ \underline{F_{22}} \end{bmatrix}} \; N$$

N/4

Step I

Step II

H.W.

- For $N = 4$, develop an expression for Radix-2 FFT in a matrix form and present the elements of each matrix