Signal Processing First

Lecture 27

Computing the Spectrum

Discrete Fourier Transform (DFT)

- **Sampling the Fourier Transform:** given an ***aperiodic*** signal $x(n)$, finite or infinite, its continuous DTFT $X(e^{j\omega})$ -- may not be a closed-form function,

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}, \quad \text{DTFT}$$

$$x(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}dw, \quad \text{Inverse DTFT}$$

- Since $X(e^{j\omega})$ is periodic with period 2π, and we are also concerned with memory storage of all continuous samples of $X(e^{j\omega})$ within $\omega \in [0,2\pi)$, let us sample N points in frequency!!

$$\tilde{X}(k) = X(e^{j\omega})\Big|_{\omega=\frac{2\pi k}{N}} = X(e^{j\frac{2\pi k}{N}})$$

Discrete Fourier Transform (DFT)

- What is the corresponding time domain sequence $\tilde{x}(n)$

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

Note that the original inverse DTFT

$$\int_0^{2\pi} \Rightarrow \sum_{k=0}^{N-1}, \qquad dw \Rightarrow \frac{2\pi}{N}$$

- Define $W_N = e^{-j\frac{2\pi}{N}}$

$$\tilde{x}(n) = \frac{1}{N}\sum_{k=0}^{N-1}\tilde{X}(k)W_N^{-kn} = \frac{1}{N}\sum_{k=0}^{N-1}\left[\sum_{m=-\infty}^{\infty}x(m)e^{-j\frac{2\pi km}{N}}\right]W_N^{-kn}$$

$$= \sum_{m=-\infty}^{\infty}x(m)\left[\frac{1}{N}\sum_{k=0}^{N-1}W_N^{-k(n-m)}\right] = \sum_{m=-\infty}^{\infty}x(m)\sum_{r=-\infty}^{\infty}\delta(n-m-rN)$$

$$= x(n) * \sum_{r=-\infty}^{\infty}\delta(n-rN) = \sum_{r=-\infty}^{\infty}x(n-rN)$$

Formal Definition of DFT

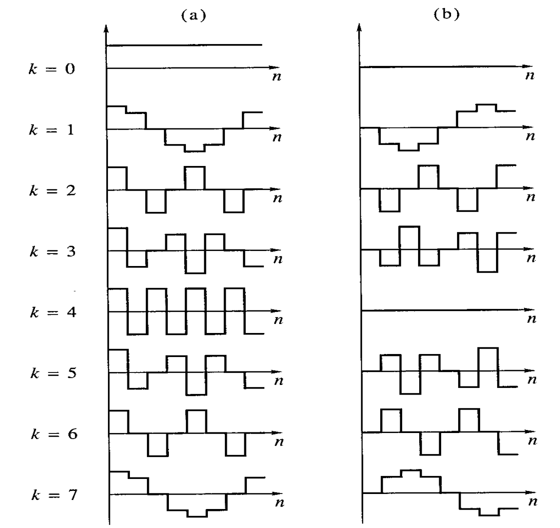
$$DFT \qquad X(k) = \begin{cases} \sum_{n=0}^{N-1} x(n)W_N^{nk}, & 0 \le k \le N\text{-}1 \\ 0, & \text{otherwise} \end{cases}$$

$$IDFT \qquad x(n) = \begin{cases} \frac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-nk}, & 0 \le n \le N\text{-}1 \\ 0, & \text{otherwise} \end{cases}$$

- Major Difference between DFT & DTFT: DFT only define discrete sets of $X(e^{j\omega k})$ N-points, while DTFT defines the continuous set of $X(e^{j\omega})$, from 0 to 2π.
- Both DFT and DTFT uniquely correspond to the set of $x(n)$!

The DFT Basis Vector for N = 8

Real Part



Imaginary Part

Frequency Sampling Theorem

- If $x(n)$ is time limited (i.e., of finite duration) to [0, N-1], then N samples of X(z) on the unit circle completely determine $X(z)$, for all z!
- **Proof:** *since N samples of X(z) on unit circle is equivalent to* $X(e^{j\omega k})$ *whose IDFT exactly recovers N-sample x(n) without any aliasing error, therefore we can base on that information to derive the X(z).*

$$X(z) = \sum_{n=0}^{N-1} \tilde{x}(n) z^{-n}$$

$$= \sum_{n=0}^{N-1} \left\{ \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) W_N^{-kn} \right\} z^{-n} = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) \left\{ \sum_{n=0}^{N-1} W_N^{-kn} z^{-n} \right\}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) \left\{ \frac{1 - W_N^{-kn} z^{-N}}{1 - W_N^{-k} z^{-1}} \right\} = \frac{1 - z^{-N}}{N} \sum_{k=0}^{N-1} \frac{\tilde{X}(k)}{1 - W_N^{-k} z^{-1}}$$

Example (DTFT)

$$x(n) = \begin{cases} 1, & 0 \le n \le 3 \\ 0, & \text{else} \end{cases}$$

$$X(e^{j\omega}) = DTFT\{x(n)\} = \frac{\sin(2\omega)}{\sin(\xi/2)} e^{-j\frac{3\omega}{2}}$$

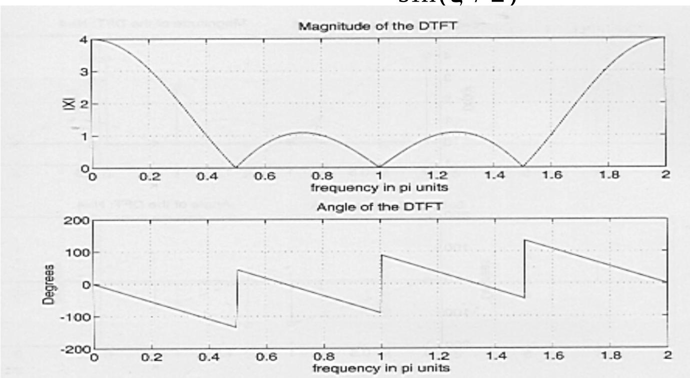


FIGURE 5.4 *The DTFT plots in Example 5.6*

Example (DFT by Sampling DTFT)

- Let us sample only 4 points on $X(e^{j\omega k})$: $X(k)$=[4,0,0,0]

$$\tilde{x}(n) = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,....]$$

$$x(n) = [1,1,1,1]$$

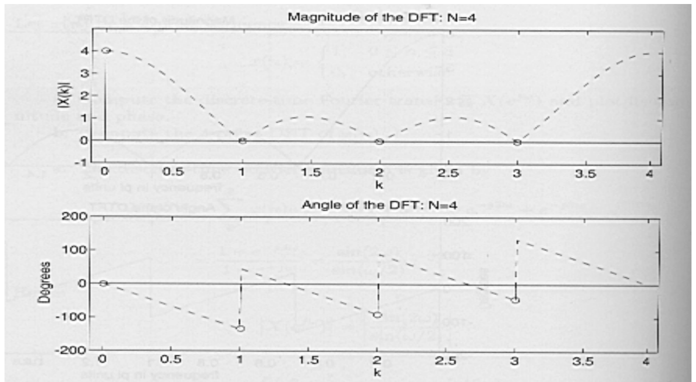


FIGURE 5.5 *The DFT plots of Example 5.6*

Example (DFT by Sampling DTFT)

- Let us sample 8 points of $X(e^{j\omega k})$:

$$\tilde{x}(n) = [1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,....]$$

$$x(n) = [1,1,1,1,0,0,0,0]$$

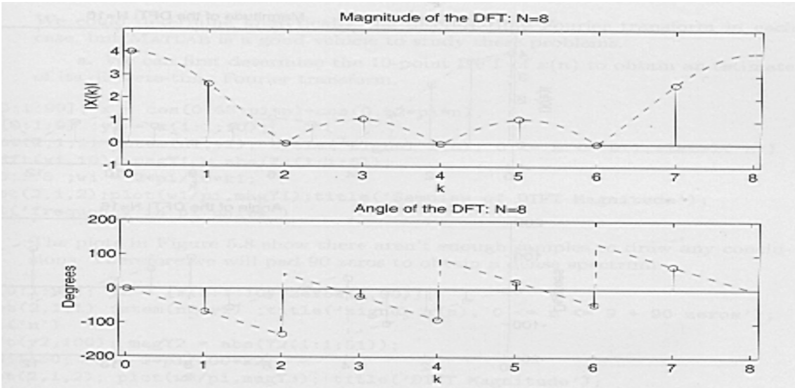
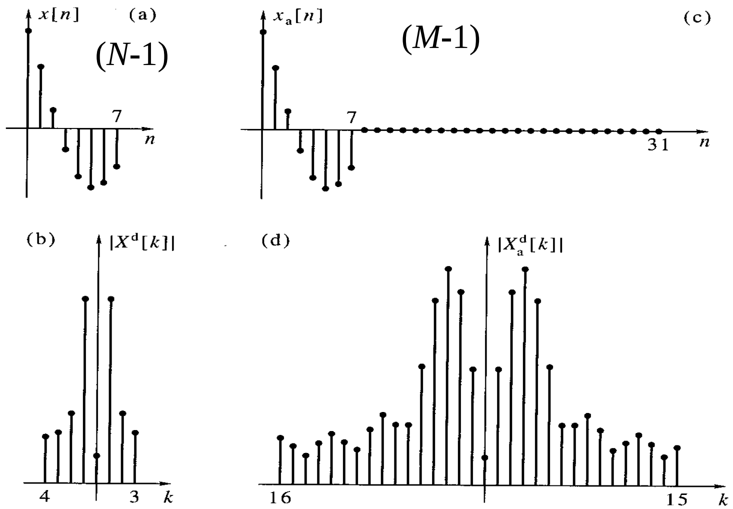


FIGURE 5.6 *The DFT plots of Example 5.7: N = 8*

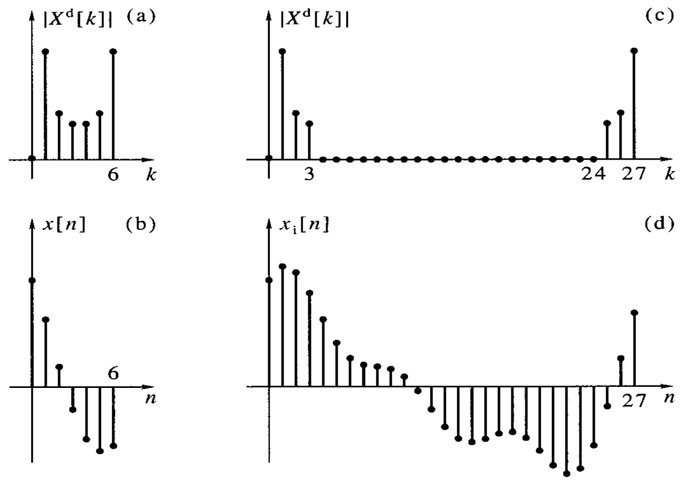
Important Observations

- Zero padding allows “lengthing” the finite duration x(n), i.e., more frequency samples on $X(k)$.
- To obtain DTFT of the true x(n), we don’t need to go through z-domain interpolation, i.e., from finite X(k) to get $X(z)$, based on the frequency sampling theorem, then replace z by $e^{j\omega}$. We just need to do a lot of zero padding!
- Zero padding gives “higher density spectrum”, but not “higher resolution spectrum” -- since the corresponding $X(e^{j\omega})$ has been fixed.
- To obtain better resolution spectrum, we need more non-zero data in the time domain.

Zero-Padding in the Time Domain



Zero-Padding in the Frequency Domain



Useful Properties of DFT

- Linearity $ax_1(n)+bx_2(n) \Rightarrow aX_1(k)+bX_2(k)$
- Circular Folding (place the x(n) and X(k) circularly) $x\left((-n)\right)_N \Rightarrow X\left((-k)\right)_N$
- Conjugation $x^*(n) \Rightarrow X^*\left((-k)\right)_N$
- Conjugate Symmetry of Real Signal x(n)

$$X(k) = X^*\left((-k)\right)_N$$

- Circular Convolution

$$x_1(n) * x_2(n) \Rightarrow X_1(k)X_2(k)$$

Inverse DFT by Forward DFT

- Computing an IDFT by using direct DFT routine. Some data processing (e.g., conjugating) is required.

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

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

$$x(n) = \frac{1}{N}\left(\sum_{k=0}^{N-1} X^*(k)W_N^{nk}\right)^*$$

Computation of DFT

- Recall DFT definition: $X(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk}, \quad 0 \le k \le N-1$
- For each *k,* we need *N* complex multiplications. For all *N* point of *X*(*k*), we need N^2 complex multiplications. Also we need storage of *N* complex X(k) and all $\{W_N^{nk}\}$.
- Some properties of W_N^{nk} can be exploited:

$$W_N^{k(N-n)} = W_N^{-kn} = \left(W_N^{kn}\right)^*$$

$$W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n}$$

- Other useful properties:

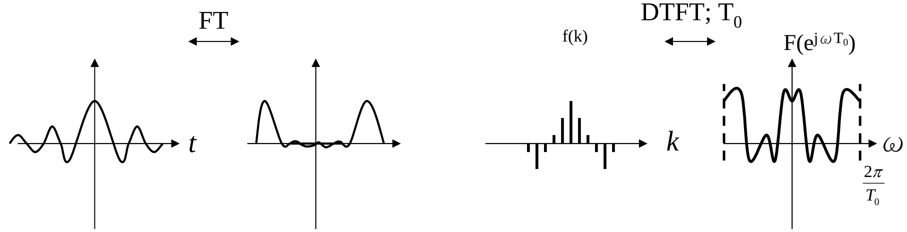
$$W_N^{(k+N/2)n} = W_N^{kn}W_N^{nN/2} = W_N^{kn}e^{-jn\pi} = \begin{cases} W_N^{kn} & \text{if } n \text{ even} \\ -W_N^{kn} & \text{if } n \text{ odd} \end{cases}$$

$$W_N^{2kn} = W_{N/2}^{kn}$$

FOUR CLASSES OF FOURIER TRANSFORM

Continuous in frequency

Continuous in Time | *Discrete in time-Periodic in frequency*



$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(j\omega)e^{j\omega t}d\omega$$

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$$

Fourier transform

$$f(k) = \frac{T_0}{2\pi}\int_{-\pi/T_0}^{\pi/T_0} F(e^{j\omega T_0})e^{jk\omega T_0}d\omega$$

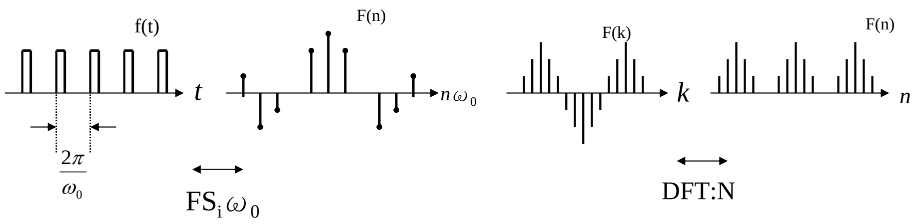
$$F(e^{j\omega T_0}) = \sum_{k=-\infty}^{\infty} f(k)e^{-jk\omega T_0}$$

Discrete time Fourier Transform

FOUR CLASSES OF FOURIER TRANSFORM (CONT.)

Discrete in frequency – periodic in time

Continuous in Time | *Discrete in time-Periodic in frequency*



$$f(t) = \sum_{n=-\infty}^{\infty} F(n) e^{jn\omega_0 t}$$

$$F(n) = \frac{\omega_0}{2\pi} \int_{-\pi/\omega_0}^{\pi/\omega_0} f(t) e^{-jn\omega_0 t} dt$$

Fourier series

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

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

Discrete Fourier Transform

DIRECT COMPUTATION OF THE DFT

For a complex-valued sequence of *N* points the DFT may be expressed as

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

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

- The direct computation requires:
- $2N^2$ evaluations of trigonometric functions.
- $4N^2$ real multiplications.
- $4N(N-1)$ real additions.
- A number of indexing and addressing operations.

```
C
C  DFT SUBROUTINE
C   ISEL = 0 : DFT
C   ISEL = 1 : INVERSE DFT
C
    SUBROUTINE DFT(N, XR, XI, XFR, XFI, ISEL)
    DIMENSION XR(N), XI(N), XFR(N), XFI(N)
    WN = 6.2831853 / FLOAT(N)
    IF (ISEL.EQ.1) WN = - WN
    DO 20 K = 1, N
      XFR(K) = 0,
      XFI(K) = 0,
      KM1 = K - 1
      DO 20 I = 1, N
        IM1 = I - 1
        ARG = WN * KM1 * IM1
        C = COS(ARG)
        S = SIN(ARG)
        XFR(K) = XFR(K) + XR(I)*C + XI(I)*S
        XFI(K) = XFI(K) - XR(I)*S + XI(I)*C
10    CONTINUE
      IF (ISEL - 1) 20, 30, 20
30    XFR(K) = XFR(K) / FLOAT(N)
20    XFI(K) = XFI(K) / FLOAT(N)
    CONTINUE
    RETURN
    END
```


Computation of DFT

- Recall DFT definition: $X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad 0 \le k \le N-1$
- For each *k*, we need *N* complex multiplications. For all *N* point of *X*(*k*), we need *N*2 complex multiplications. Also we need storage of *N* complex X(k) and all $\{W_N^{nk}\}$.
- Some properties of W_N^{nk} can be exploited:

$$W_N^{k(N-n)} = W_N^{-kn} = \left(W_N^{kn}\right)^*$$

$$W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n}$$

- Other useful properties:

$$W_N^{(k+N/2)n} = W_N^{kn} W_N^{nN/2} = W_N^{kn} e^{-jn\pi} = \begin{cases} W_N^{kn} & \text{if } n \text{ even} \\ -W_N^{kn} & \text{if } n \text{ odd} \end{cases}$$

$$W_N^{2kn} = W_{N/2}^{kn}$$

Decimation in Time (DIT) FFT

- How about group the data into even and odd parts:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad 0 \le k \le N-1$$

$$= \sum_{\substack{n=even \\ n=2r}} x(n) W_N^{nk} + \sum_{\substack{n=odd \\ n=2r+1}} x(n) W_N^{nk}$$

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

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

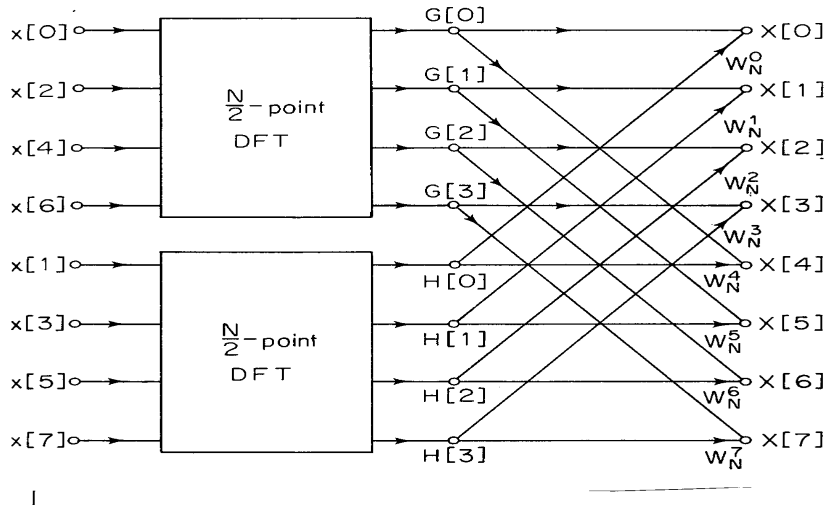
$$= G(k) + W_N^k H(k)$$

DIT FFT (Cont.)

- $G(k)$ is $N/2$ points DFT of the even numbered data: x(0), x(2), x(4), …., x(N-2), assuming N is even. Note that $G(k)$ is defined over $k = 0,1,\ldots,N/2-1$
- Similarly, $H(k)$ is the $N/2$ points DFT of the odd numbered data: x(1), x(3), …, x(N-1). Also $k = 0,1,\ldots,N/2-1$.
- Since $G(k)$ and $H(k)$ are of length $N/2$, how can we create $X(k)$ of length N? $G(k)=G(k+N/2)$ and $H(k)=H(k+N/2)$,

$$X(k) = G(k) + W_N^k H(k), \qquad k = 0,1,\ldots,N-1$$

DIT FFT (Cont.)



DIT FFT (Cont.)

- Note that the number of complex multiplications is more or less reduced by half:

$$N^2 \Rightarrow 2\left(\frac{N}{2}\right)^2 + N = \frac{N^2}{2} + N \approx \frac{N^2}{2}$$

- How about the computation of each $N/2$-point DFT?

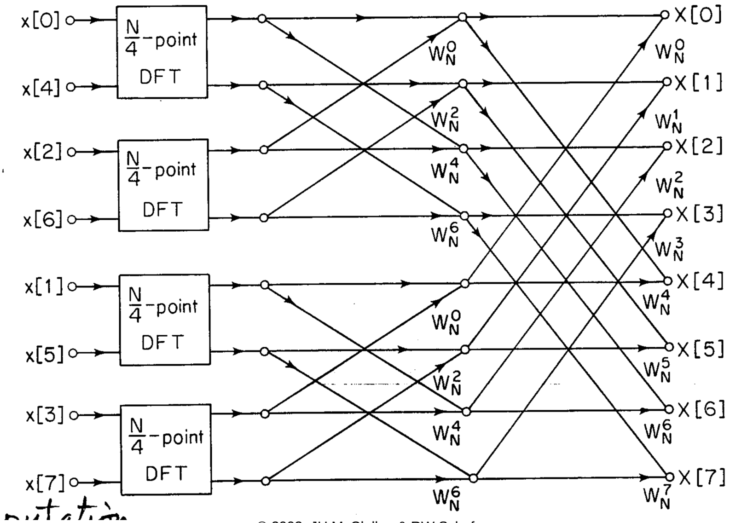
$$G(k) = \sum_{r=0}^{N/2-1} g(r) W_{N/2}^{rk}, \qquad g(r) = x(2r)$$

$$= \sum_{l=0}^{N/4-1} g(2l) W_{N/2}^{2lk} + \sum_{l=0}^{\frac{N}{4}-1} g(2l+1) W_{N/2}^{2(l+1)k}$$

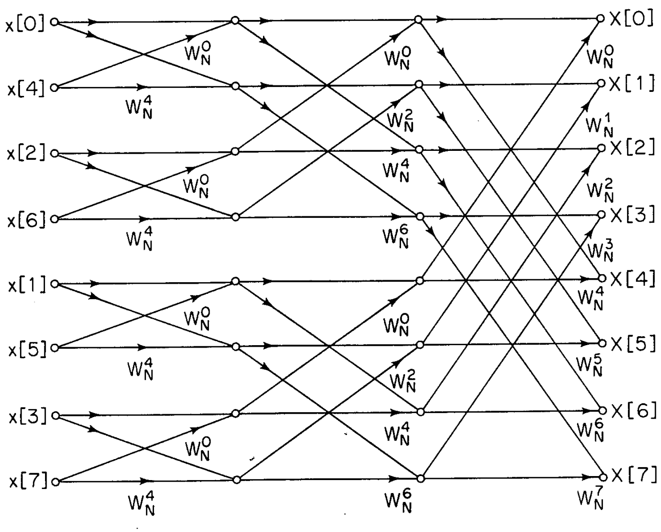
$$= \sum_{l=0}^{N/4-1} g(2l) W_{N/4}^{lk} + W_{N/2}^{k} \sum_{l=0}^{N/4-1} g(2l+1) W_{N/4}^{lk}$$

$$= Q_g(k) + W_{N/2}^k P_g(k)$$

DIT FFT (Cont.)



8-Point DIT FFT

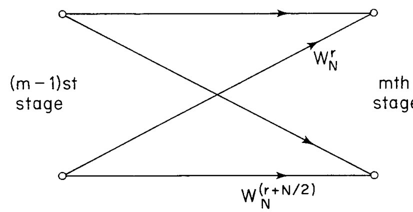


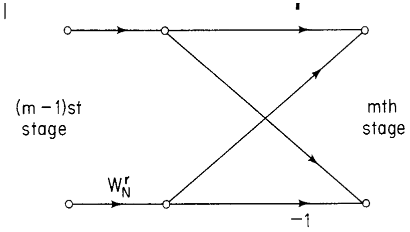
Final Counts of DIT FFT

- Each stage, N complex multiplications are required, and $\log_2 N$ stages are decomposed, therefore

$$N^2 \Rightarrow N \log_2 \mathrm{N} \;\; \text{complex multiplications}$$

- Further reduction on "butterfly", since $W_N^{r+\frac{N}{2}} = -W_N^r$





reduced to $N^2 \Rightarrow \frac{N}{2} \log_2 N$

Final Counts of DIT FFT (Cont.)

For $N = 2^v$, this decimation can be performed $v = \log_2 N$ times.

The total number of

- Complex multiplications : $(N/2) * \log_2 N$
- Complex additions : $N * \log_2 N$

N	Complex multiplications in Direct Computation N^2	Complex multiplications in FFT algorithm, $(N/2)\log_2 N$
4	16	4
8	64	12
16	256	32
32	1,024	80
64	4,096	192
128	16,384	448
256	65,535	1,024
512	262,144	2,304
1,024	1,048,576	5,120

Decimation in Frequency (DIF) FFT

- Let us now consider to group the 1st and 2nd halves:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad 0 \le \mathrm{k} \le \mathrm{N-1}$$

$$= \sum_{n=0}^{N/2-1} x(n) W_N^{nk} + \sum_{n=N/2}^{N-1} x(n) W_N^{nk}$$

$$= \sum_{n=0}^{N/2-1} x(n) W_N^{nk} + \sum_{n=0}^{N/2-1} x(n+N/2) W_N^{(n+N/2)k}$$

$$= \sum_{n=0}^{N/2-1} x(n) W_N^{nk} + W_N^{Nk/2} x(n+N/2) W_N^{nk}$$

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

DIF FFT (Cont.)

- Consider k to be even: $k = 2r$

$$X(2r) = \sum_{n=0}^{N/2-1}\left[x(n) + x(n+\frac{N}{2})\right]W_N^{2rn}$$
$$= \sum_{n=0}^{N/2-1}\left[x(n) + x(n+\frac{N}{2})\right]W_{N/2}^{rn}$$
$$= \sum_{n=0}^{N/2-1} a[n]W_{N/2}^{rn}$$

- Similarly, k to be odd: $k = 2r + 1$

$$X(2r+1) = \sum_{n=0}^{N/2-1}\left[x(n) - x(n+N/2)\right]W_N^{n}W_{N/2}^{rn}$$
$$= \sum_{n=0}^{N/2-1} b(n)W_N^{n}W_{N/2}^{rn}$$

DIF FFT (Cont.)

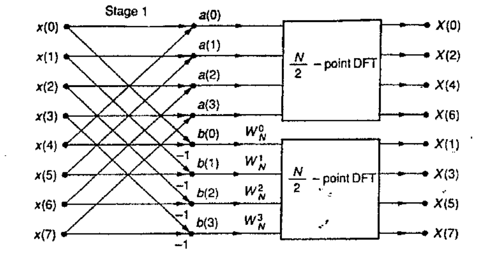
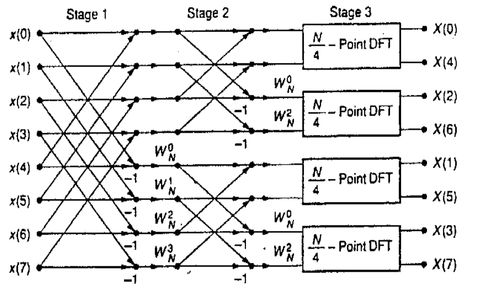


FIGURE 6.2 Decomposition of N-point DFT into two (N/2)-point DFT's, for N = 8.



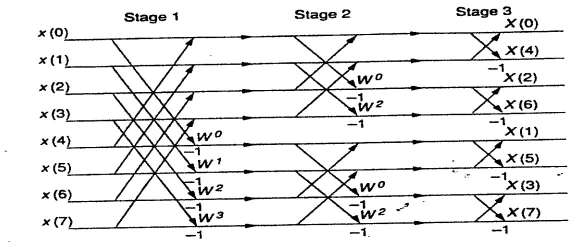


FIGURE 6.5 Eight-point FFT flow graph using decimation-in-frequency.

Similarly, the final counts $= \frac{N}{2}\log_2 N$

Important Issues of FFT

- Not necessarily for $N = 2^v$ (power of two) length. Any **prime factor decomposition** can save computation time, e.g., N =15 = 3x5.
- **In place computation:** after moving to a new stage, the data can be overwritten to save the storage memory.
- How to calculate the **bit-reversal**:

0	000	⇔	000	0
1	001	⇔	100	4
2	010	⇔	010	2
3	011	⇔	110	6
4	100	⇔	001	1
5	101	⇔	101	5
6	110	⇔	011	3
7	111	⇔	111	7