Chapter 6: DFT/FFT Transforms and Applications

6.1 DFT and its Inverse

DFT: It is a transformation that maps an N-point Discrete-time (DT) signal $x[n]$ into a function of the N complex discrete harmonics. That is, given $x[n]; \quad n = 0,1,2,\cdots,N-1$, an N-point Discrete-time signal $x[n]$ then DFT is given by **(analysis equation):**

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2p}{N}nk} \qquad for \quad k = 0,1,2,\cdots,N-1 \tag{6.1}$$

and the inverse DFT (**IDFT**) is given by **(synthesis equation):**

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{+j\frac{2p}{N}nk} \qquad for \quad n = 0,1,2,\cdots,N-1 \tag{6.2}$$

Proof:

$$X(k).e^{j\frac{2p}{N}nk} = e^{j\frac{2p}{N}nk}.\sum_{p=0}^{N-1} x[p] e^{-j\frac{2p}{N}pk} = \sum_{p=0}^{N-1} x[p] e^{-j\frac{2p}{N}(n-p)k}$$

Sum both sides over *0* to *N-1*:

$$\sum_{k=0}^{N-1} X(k).e^{j\frac{2p}{N}nk} = e^{j\frac{2p}{N}nk}.\sum_{p=0}^{N-1} x[p] e^{-j\frac{2p}{N}pk} = \sum_{k=0}^{N-1}\sum_{p=0}^{N-1} x[p] e^{-j\frac{2p}{N}(n-p)k} = \sum_{p=0}^{N-1}\sum_{k=0}^{N-1} x[p] e^{-j\frac{2p}{N}(n-p)k} \tag{6.3}$$

Using the property:

$$\sum_{k=0}^{N-1} e^{j\frac{2p}{N}(n-p)k} = \begin{cases} 0 & n \neq p \\ N & n = p \end{cases} \tag{6.4}$$

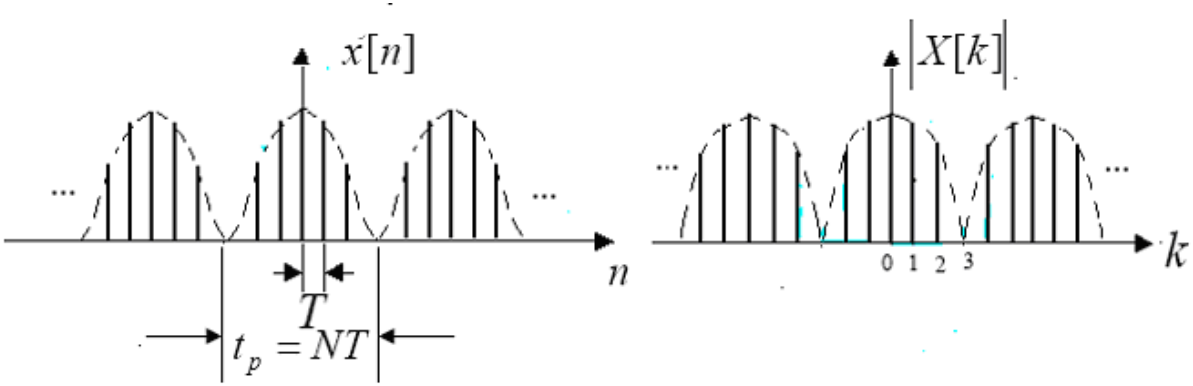
(6.3) sums to $N.x[n]$, division by N yields (6.2).

1. These two equations form DFT pair.
2. They have both N-point resolution both in the discrete-time domain and discrete-frequency domain.
3. Always the scaling factor $1/N$ is associated with the synthesis equation (inverse DFT).
4. $X(k)$ is periodic in N or equivalently in $\Omega_k = 2p/N$; that is

$$X(k) = X(\Omega_k) = X(\Omega_k + 2p) = X(\frac{2p}{N}(k+N)) = X(k+N) \tag{6.5}$$

5. $x[n]$ determined from (6.2) is also periodic in N;

$$x[n] = x[n+N] \tag{6.6}$$



Example 6.1: Compute the DFT of the following two sequences:

$h[n] = \{1,3,-1,-2\}$ and $x[n] = \{1,2,0,-1\}$

where $N = 4 \Rightarrow e^{j\frac{2p}{N}} = e^{j\frac{2p}{4}} = e^{j\frac{p}{2}} = j$

Let us use this information in (6.1) to compute DFT values:

$$H(k) = \sum_{n=0}^{3} h[n] e^{-j\frac{p}{2}nk} \quad for \quad k = 0,1,2,3$$

$$H(0) = h[0] + h[1] + h[2] + h[3] = 1$$
$$H(1) = h[0] + h[1]e^{-jp/2} + h[2].e^{-jp} + h[3].e^{-j3p/2} = 2 - j5$$
$$H(2) = h[0] + h[1]e^{-jp} + h[2].e^{-j2p} + h[3].e^{-j3p} = -1$$
$$H(3) = h[0] + h[1]e^{-j3p/2} + h[2].e^{-j3p} + h[3].e^{-j9p/2} = 2 + j5$$

Similarly,

$$X(0) = x[0] + x[1] + x[2] + x[3] = 2$$
$$X(1) = x[0] + x[1]e^{-jp/2} + x[2].e^{-jp} + x[3].e^{-j3p/2} = 1 - j3$$
$$X(2) = x[0] + x[1]e^{-jp} + x[2].e^{-j2p} + x[3].e^{-j3p} = 0$$
$$X(3) = x[0] + x[1]e^{-j3p/2} + x[2].e^{-j3p} + x[3].e^{-j9p/2} = 1 + j3$$

- Watch for the conjugate symmetry of terms; i.e., complex harmonics come in pairs.

6.2 Matrix Representation of DFT

Let us write the variables involved in matrix form: $x = [x[0], x[1], \cdots, x[N-1]]^T$ and $W_N = e^{-j2p/N}$:

$$X(k) = \sum_{n=0}^{N-1} x[n].W_N^{kn} \quad for \quad k = 0,1,2,\cdots,N-1 \tag{6.7}$$

Then the weight matrix is simply:

$$W = \begin{bmatrix} W_N^0 & W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^1 & W_N^2 & \cdots & W_N^{N-1} \\ \vdots & \vdots & & \cdots & \vdots \\ W_N^0 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} \tag{6.8}$$

which is normally referred as DFT matrix and the resulting transform vector becomes:

$$X = [X(0), X(1), \cdots, X(N-1)]^T = W.x \tag{6.9}$$

- $[W]_{nk} = [W]_{kn}$ (6.10a)

- $W = W^T$ (6.10b)
- $W_N^{-1} = W_N^*$; where * stands for the complex conjugate. (6.10c)

With these we can write the inverse DFT (IDFT) as follows:

$$x[n] = \frac{1}{N}\sum_{n=0}^{N-1} X(k).W_N^{-kn} \quad for \quad n = 0,1,2,\cdots,N-1 \tag{6.11a}$$

$$x = \frac{1}{N}W^*.X = W^{-1}.X \tag{6.11b}$$

$$W^{-1} = \frac{1}{N}.W^* \qquad W^*.W = N.I_N \qquad \text{with } I_N : NxN \text{ identity matrix} \tag{6.12}$$

6.3 Relation Between DFT and z-Transforms

Let us re-write the DFT definition:

$$X(k) = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2p}{N}nk} = \sum_{n=0}^{N-1} x[n]e^{-j\Omega}\Big|_{\Omega=\frac{2p}{N}k} \tag{6.13}$$

$$= X(\Omega)\big|_{\Omega=\frac{2p}{N}k}$$

The DFT of $x[n]$ is its DTFT evaluated at N equally spaced points in the range $[0,2p)$. For a sequence for which both the DTFT and the z-transform exist, we see that:

$$X(k) = X(z)\big|_{z=e^{j\frac{2p}{N}.k}} \tag{6.14}$$

Example 6.2: Consider a discrete-time pulse signal $x[nT] = u[(n+1)T] - u[(n-3)T] \quad where\, T = 0.2\ s.$ (a) Use a six-point DFT to compute $X(k)$. (b) Compute the IDFT of $X(k)$.

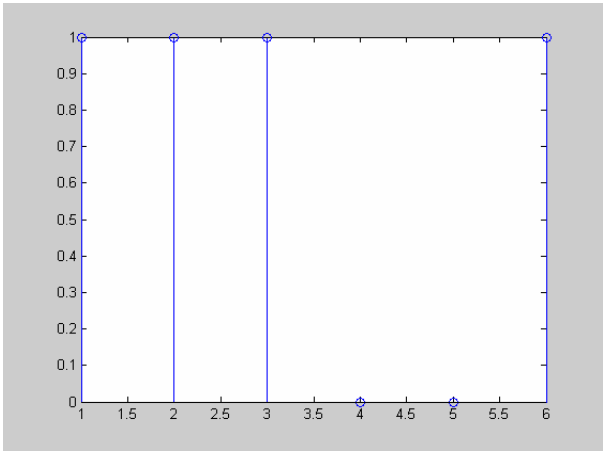
Let us start the samples at $t = -0.2$ then the six samples of the periodic extension would be $x[n] = [\overline{1},1,1,0,0,1]$. Then the script is simply:

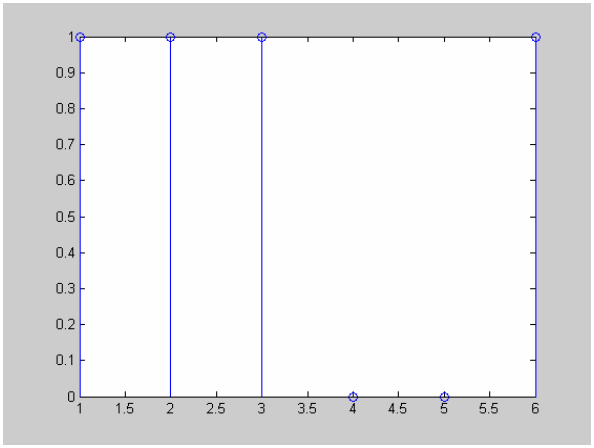
```
% Example 6.2
% Part (a)
x=[1,1,1,0,0,1]; N=size(x,2); T=0.2;
stem(x);
X=fft(x);
disp(X);
```

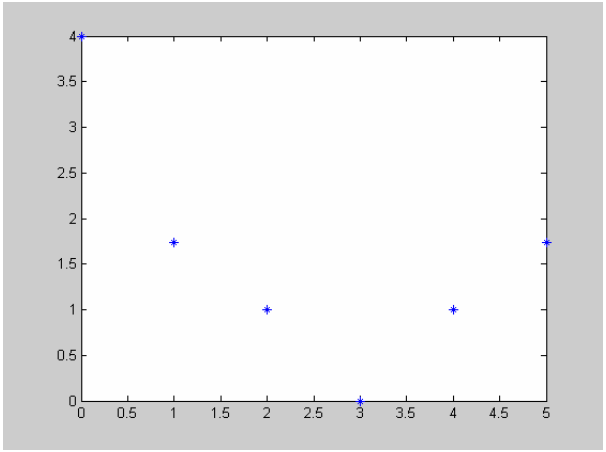
```
Answers>>  Columns 1 through 4
  4.0000            1.5000 - 0.8660i  -0.5000 + 0.8660i        0

  Columns 5 through 6
  -0.5000 - 0.8660i   1.5000 + 0.8660i
```

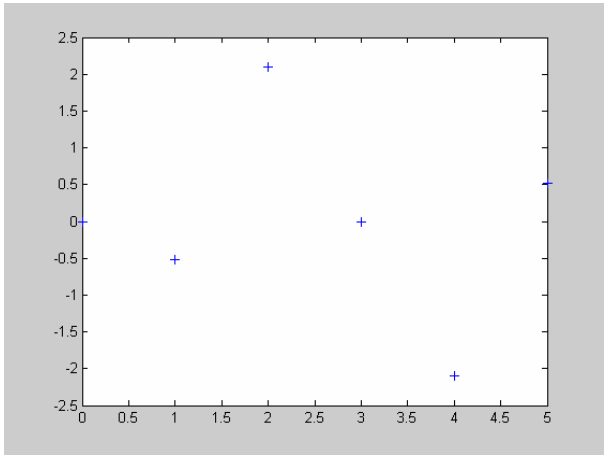
```
Mag=abs(X); Phase=angle(X);
% Plots; figure;
plot(n,Mag,'*');  figure;
plot(n,Phase,'+');
```

```
% Part (b)
xr=ifft(X);
figure;
stem(xr);
```









Input signal is exactly recovered by means of a full DFT and IDFT process.

Example 6.3: Consider the discrete-time representation of a cosine signal:

$$x[n] = A.Cos(\frac{2p}{N}n) = 0.5.A.(e^{j\frac{2p}{N}n} + e^{-j\frac{2p}{N}n})$$

$$X(k) = A.\sum_{n=0}^{N-1}(\frac{e^{j\frac{2p}{N}n} + e^{-j\frac{2p}{N}n}}{2}).e^{-j\frac{2p}{N}nk} = \frac{A}{2}\sum_{n=0}^{N-1} e^{j\frac{2p}{N}n(1-k)} + \frac{A}{2}\sum_{n=0}^{N-1} e^{-j\frac{2p}{N}n(1+k)}$$

$$= \frac{A}{2}\sum_{n=0}^{N-1}(e^{j\frac{2p}{N}(1-k)})^n + \frac{A}{2}\sum_{n=0}^{N-1}(e^{-j\frac{2p}{N}(1-k)})^n = \frac{A}{2}\frac{1-e^{j2p(1-k)}}{1-e^{j\frac{2p}{N}(1-k)}} + \frac{A}{2}\frac{1-e^{j2p(1+k)}}{1-e^{j\frac{2p}{N}(1+k)}} \qquad for\ k = 0,1,2,\cdots,N-1$$

- First term is 0 when $k \neq 1$ and it is $NA/2$ *for* $k = 1$.
- Second term is 0 when $k \neq N-1$ and it is $NA/2$ *for* $k = N-1$.
- Final result becomes: $X(k) = \frac{NA}{2}[d(k-1) + d(k-(N-1))]$

where the first term represents the positive frequency term and the other one is the mirror image as expected.

Now let us try to implement that using DFT with 48-points and 127-point of sampled versions:

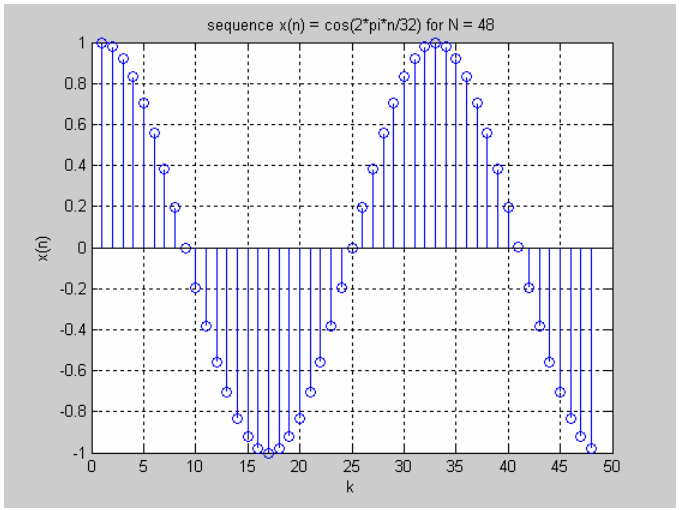
```
% E1xample 6.3 Sampled Cosine with N=48
N = 48;
n =[0:1:N-1];
k = n;
M = 32;
xn = cos(2*pi*n/M);
Xk = fft(xn);
magXk = abs(Xk);
PhaseXk = angle(Xk);
```

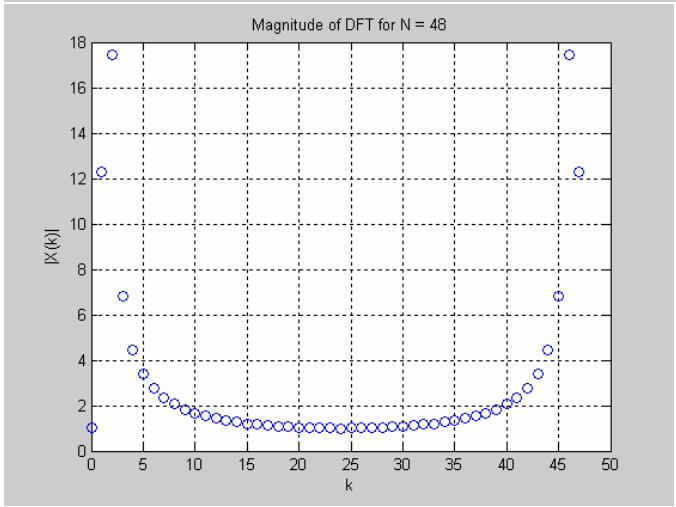
```
% Plots
axis([1 2 3 4]); axis;
stem(xn); xlabel('k');ylabel('x(n)');
title('sequence x(n) = cos(2*pi*n/32) for N = 48');
grid;
figure; axis([0,47,0,18]);
plot(n,magXk,'o');
```

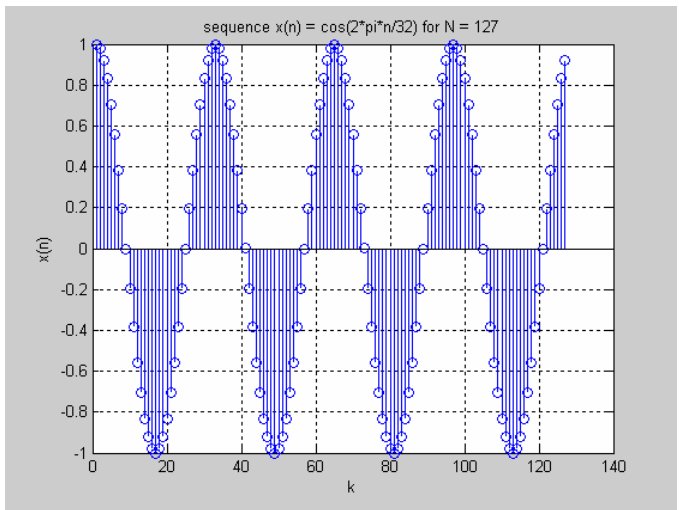
```
% Now try it with
N = 127;
n =[0:1:N-1];
k = n;
M = 32;
xn = cos(2*pi*n/M);
Xk = fft(xn);
magXk = abs(Xk);
PhaseXk = angle(Xk);
```

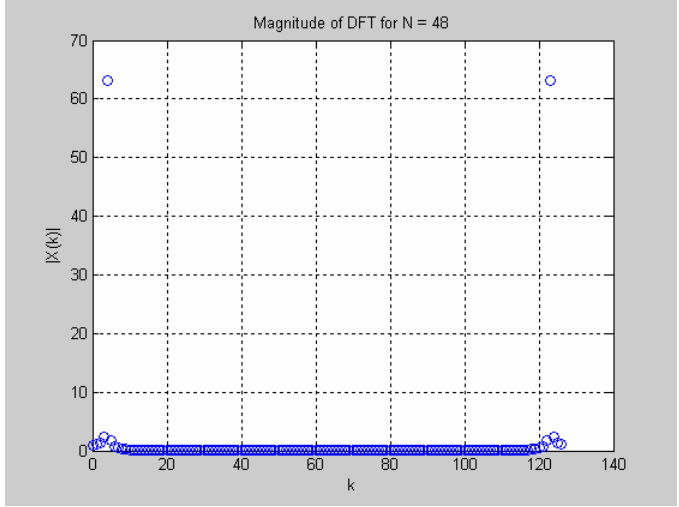
```
% Plots
axis([1 2 3 4]); axis;
stem(xn); xlabel('k');ylabel('x(n)');
title('sequence x(n) = cos(2*pi*n/32) for N = 127');
grid;
figure; axis([0,47,0,18]);
plot(n,magXk,'o');
```

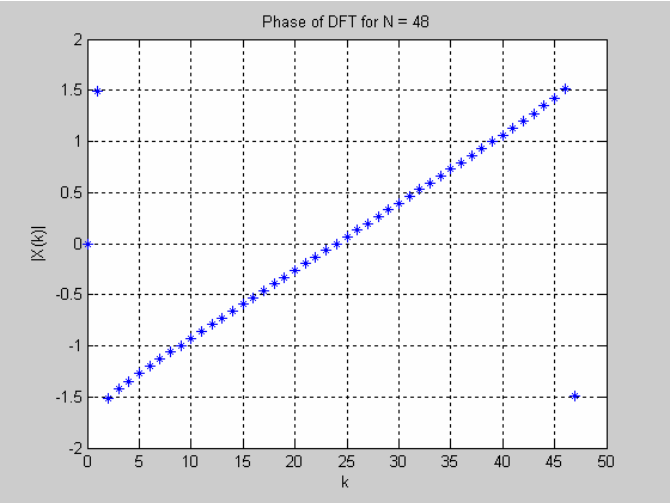
xlabel('k');ylabel('|X(k)|');
title('Magnitude of DFT for N = 48'); grid;
figure; axis([0,47,-2,2]);
plot(n,PhaseXk,'*');
xlabel('k');ylabel('|X(k)|');
title('Phase of DFT for N = 48');
grid;axis;
	xlabel('k');ylabel('|X(k)|');
	title('Magnitude of DFT for N = 48');grid;
	figure; axis([0,47,-2,2]);
	plot(n,PhaseXk,'*');
	xlabel('k');ylabel('|X(k)|');
	title('Phase of DFT for N = 127');
	grid;axis;

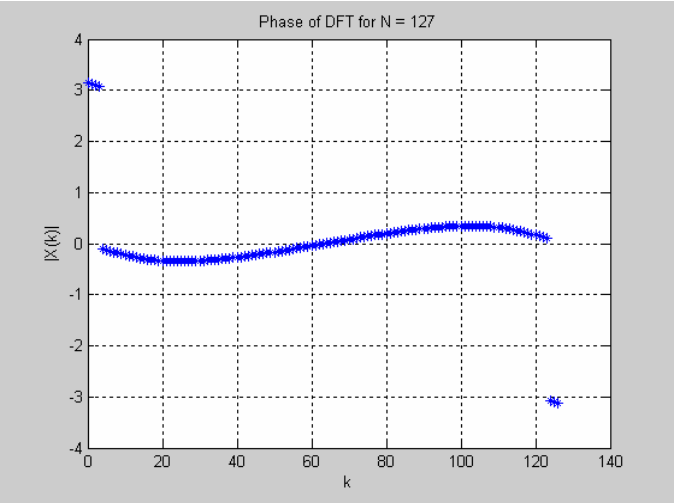












Even though we were expecting two harmonics we have ended up plots needs some explanation:

- Sampled signals are becoming more like a continuous signal as N increases.
- Magnitude plots are not located at two harmonics but expand over the frequency range, which is called "Leakage" in the business.
- Symmetry is respect to the center of the plots rather than 2-sided spectral halves, which is due to the periodic behavior of the DFT.

6.4 Properties of DFT

Linearity:

$$DFT\{a.x[n] + b[y[n]\} = a.X(k) + b.Y(k) \tag{6.15}$$

Time-Shift: For any real integer n_0,

$$DFT\{x[n+n_0]\} = \sum_{n=0}^{N-1} x[n+n_0].e^{-j\frac{2p}{N}kn}$$

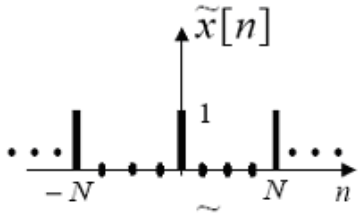
$$= \sum_{<N>} x[m].e^{-j\frac{2p}{N}k(m-n_0)} = e^{j\frac{2p}{N}kn_0} . \sum_{<N>} x[m].e^{-j\frac{2p}{N}km} \qquad (6.16)$$

$$= e^{j\frac{2p}{N}kn_0}.X(k)$$

Example 6.4: Consider the following sequence:

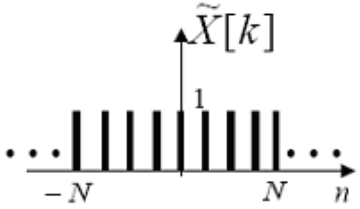
$$\tilde{x}[n] = \sum_{r=-\infty}^{\infty} \delta[n + rN]$$

since $\tilde{x}[n] = \delta[n] \quad for \ \ 0 \le n \le N-1$



then $\tilde{X}[k] = \sum_{n=0}^{N-1} \delta[n] W_N^{kn} = 1$

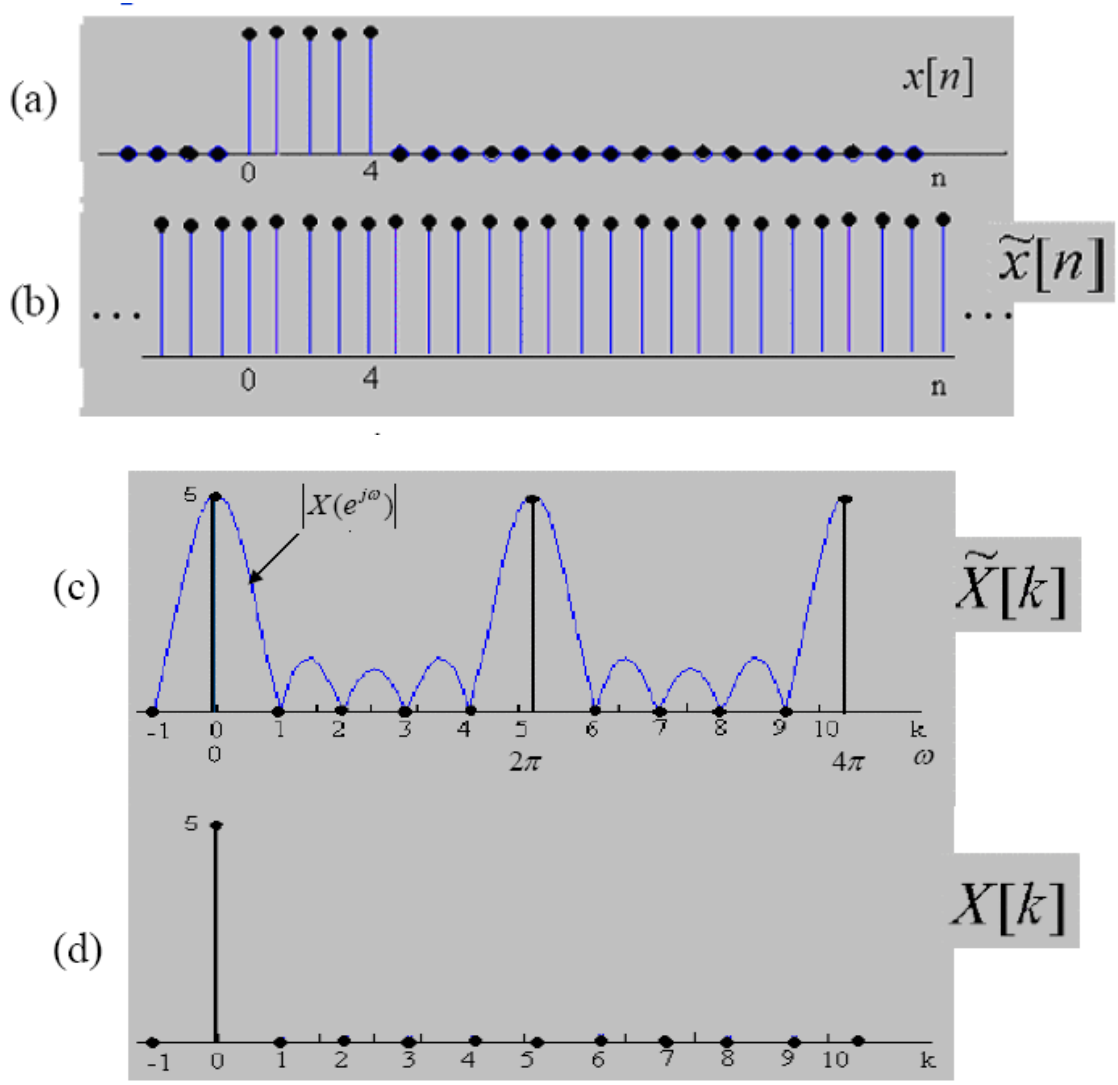
$$\tilde{x}[n] = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}[k] W_N^{-kn}$$



$$\tilde{x}[n] = \sum_{r=-\infty}^{\infty} \delta[n+rN] = \frac{1}{N} \sum_{k=0}^{N-1} W_N^{-kn}$$

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

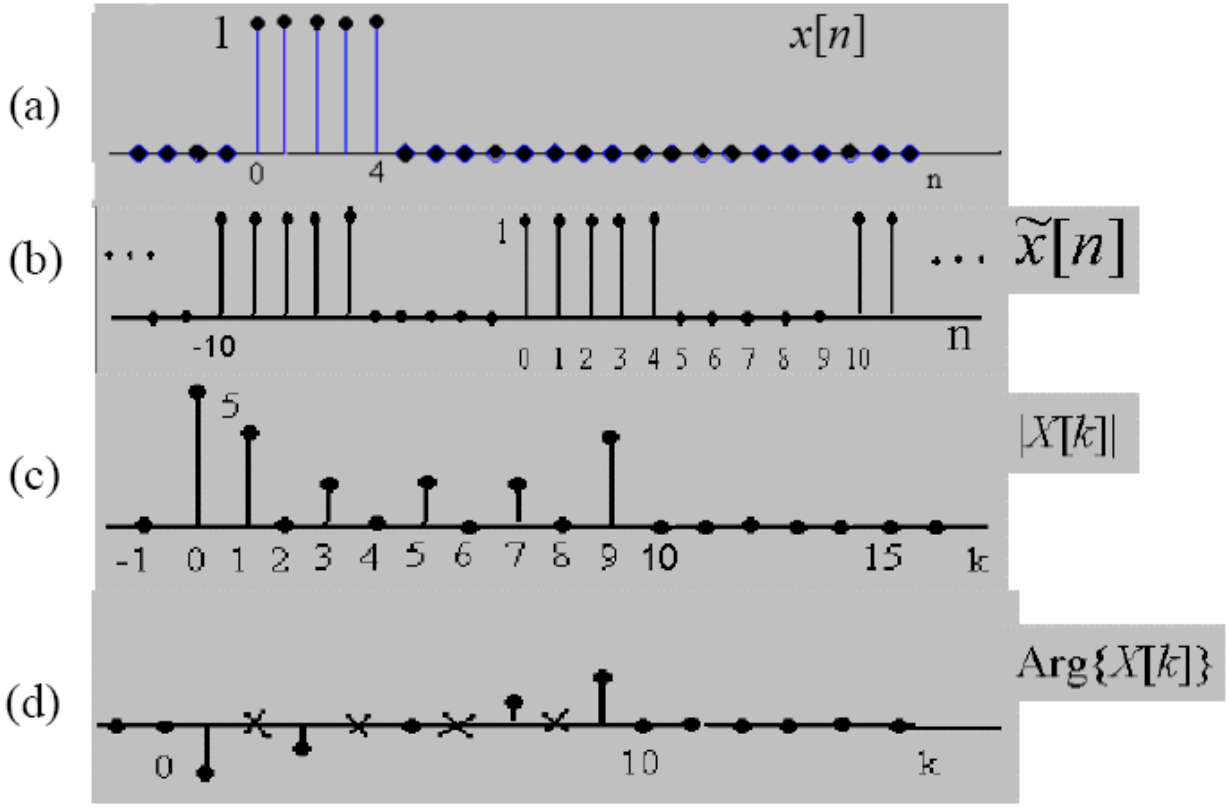
Example 6.5: Consider the following sequence for: $N = 5$



where

$$\tilde{X}(k) = \sum_{n=0}^{N-1} W_N^{nk} = \sum_{n=0}^{N-1} e^{-j2pk/N} = \frac{1-e^{-j2pk}}{1-e^{-j2pk/N}} = \begin{cases} N & k = 0, \pm N, \pm 2N, \cdots \\ 0 & Otherwise \end{cases}$$

Let us now change it to: $N = 10$



Circular (Periodic) Convolution:

$$Y(k) = X(k).H(k) \tag{6.17}$$

$$\begin{aligned} y[n] = IDFT\{Y(k)\} &= \frac{1}{N}\sum_{k=0}^{N-1} Y(k).e^{j\frac{2p}{N}nk} = \frac{1}{N}\sum_{k=0}^{N-1} X(k).H(k).e^{j\frac{2p}{N}nk} \\ &= \frac{1}{N}\sum_{k=0}^{N-1}\{\sum_{m=0}^{N-1} h(m).e^{-j\frac{2p}{N}mk}\}.X(k).e^{j\frac{2p}{N}nk} = \sum_{m=0}^{N-1} h[m].x[n-m] \end{aligned} \tag{6.18}$$

Example 6.6: Perform the periodic (circular) convolution of two sequences in Example 6.1:

$h[n] = \{1,3,-1,-2\}$ and $x[n] = \{1,2,0,-1\}$

The output is the product of the two sets found earlier:

$$Y(0) = X(0).H(0) = 2$$
$$Y(1) = X(1).H(1) = -13 - j11$$
$$Y(2) = X(2).H(2) = 0$$
$$Y(3) = X(3).H(3) = -13 + j11$$

The IDFT would yield the output in discrete-time domain:

$$y(0) = \frac{1}{4}[Y(0) + Y(1) + Y(2) + Y(3)] = -6$$

$$y(1) = \frac{1}{4}[Y(0) + Y(1).e^{jp/2} + Y(2)e^{j2p} + Y(3)e^{j3p/2}] = 7$$

$$y(2) = \frac{1}{4}[Y(0) + Y(1).e^{jp} + Y(2)e^{jp} + Y(3)e^{j3p/2}] = 6$$

$$y(3) = \frac{1}{4}[Y(0) + Y(1).e^{j3p/2} + Y(2)e^{j3p} + Y(3)e^{j9p/2}] = -5$$

Recall: Example 3.8: Consider the following system and signal sequences:

$$x[n] = \{1,2,0,-1\}$$
$$h[n] = \{1,3,-1,-2\}$$

- These two sequences have a common period of 4 samples.
- It is not difficult to see that the output sequence $y[n]$ will be again 4 samples long in the interval $0 \le n \le 3$ and repeat itself.

Let us verify that with a circular convolution table.

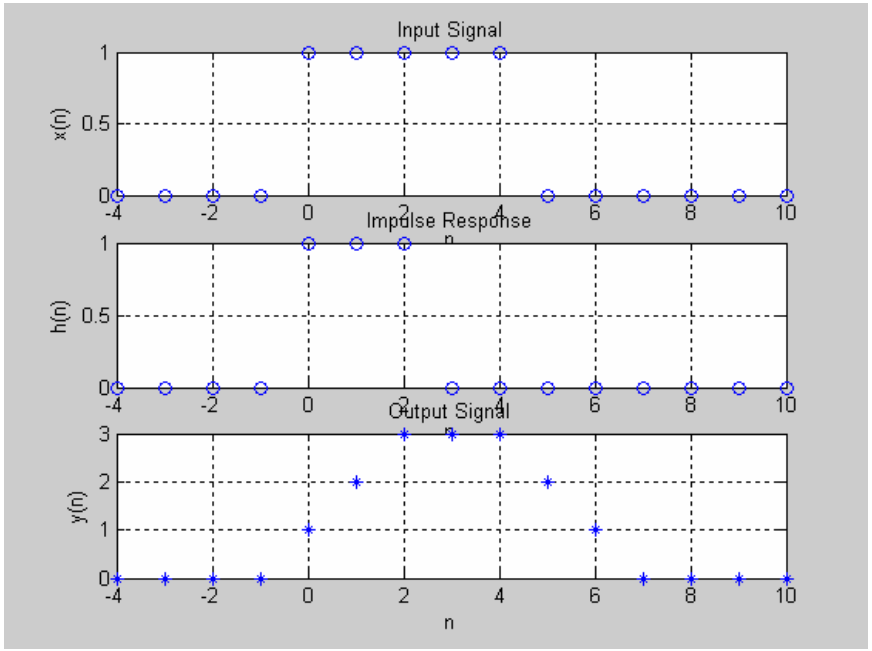
n	**0**	**1**	**2**	**3**
x[n]	1	2	0	-1
x[n-1]	-1	1	2	0
x[n-2]	0	-1	1	2
x[n-3]	2	0	-1	1
h[0]x[n]	1	2	0	-1
h[1]x[n-1]	-3	3	6	0
h[2]x[n-2]	0	1	-1	-2
h[3]x[n-3]	-4	0	2	-2
y_c[n]	**-6**	**6**	**7**	**-5**

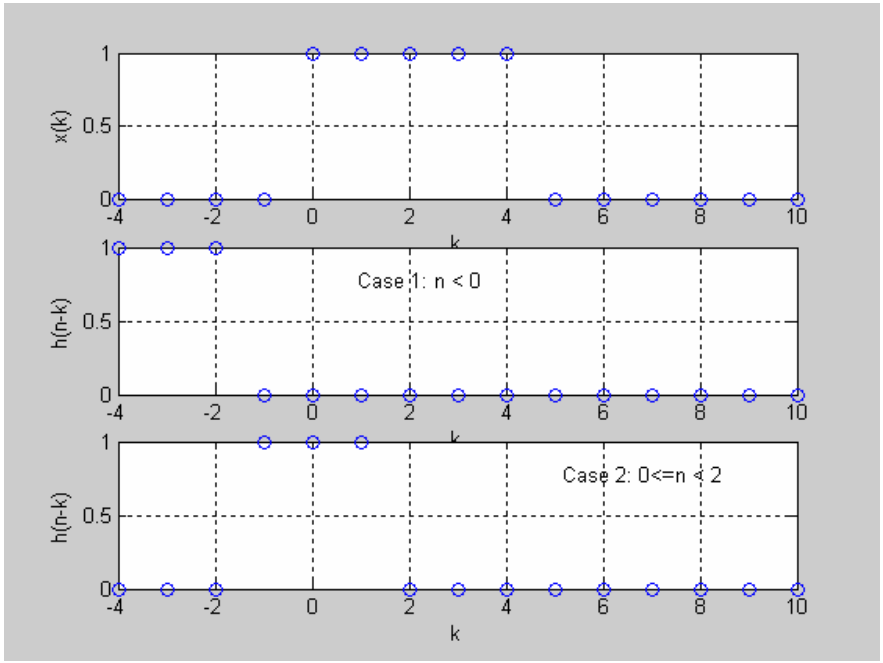
The last row or the output is: $y[n] = y_c[n+4] = \{-6,6,7,-5\}$**, which identical to the answer we got in Chapter 3.**

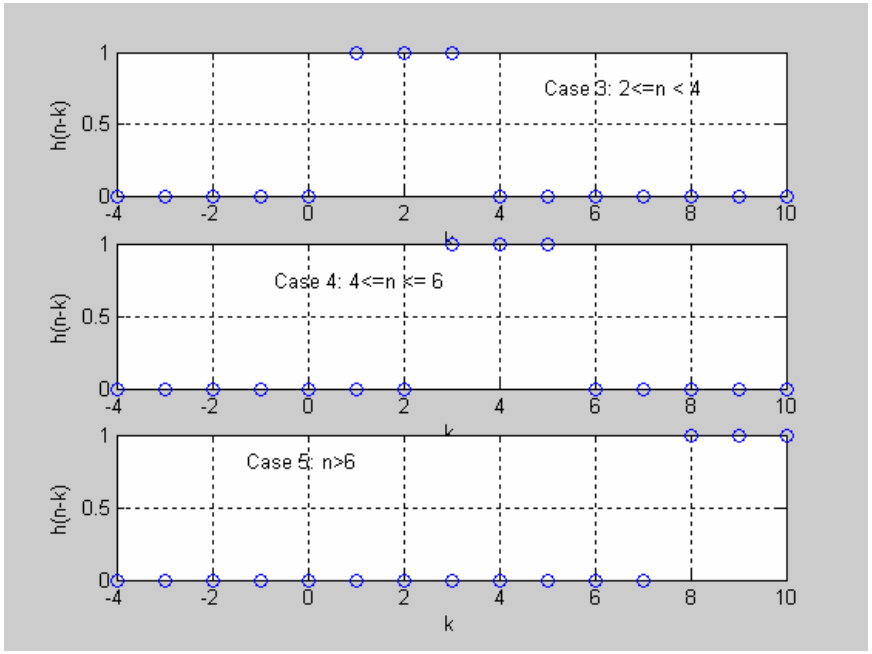
Linear Convolution Using DFT: As in all previous transforms, the response of an discrete LTI system with an impulse response $h[n]$, the output due to any input $x[n]$ is given by the linear convolution of the two sequences. However, the product: $X(k).H(k)$ corresponds to a periodic convolution.

Example 6.7: Perform linear convolution of two discrete functions as shown in the matlab text below.

```
%Example 6.7
% Linear convolution of two-discrete finite pulses
        n=-4:10;
        x=zeros(size(n));
        h=zeros(size(n));
        x(5:9)=ones(size(n(5:9)));
        h(5:7)=ones(size(n(5:7)));
        axis([-4,10,0,2])
        subplot(311),plot(n,x,'o');grid;
        title('Input Signal'); xlabel('n'); ylabel('x(n)');
        subplot(312),plot(n,h,'o');grid;
        title('Impulse Response'); xlabel('n'); ylabel('h(n)');
%convolution
        y=conv(h,x);
        subplot(313),plot(n,y(5:19),'*');grid;
        title('Output Signal'); xlabel('n'); ylabel('y(n)');
%Case: 1 n less than zero
        figure;
        subplot(311);plot(n,x,'o');grid;
        xlabel('k'); ylabel('x(k)');
        h1=zeros(size(n));
        h1(1:3)=ones(size(n(1:3)));
        subplot(312);plot(n,h1,'o');grid;
        xlabel('k'); ylabel('h(n-k)');
        gtext('Case 1: n < 0');
        pause;
%Case: 2  0<=n<2
        h2=zeros(size(n));
        h2(4:6)=ones(size(n(4:6)));
        subplot(313);plot(n,h2,'o');grid;
        xlabel('k'); ylabel('h(n-k)');
        gtext('Case 2: 0<=n < 2');
        pause
%Case: 3  2<=n<4
        h3=zeros(size(n));
        h3(6:8)=ones(size(n(6:8)));
        figure;
        subplot(311);plot(n,h3,'o');grid;
        xlabel('k'); ylabel('h(n-k)');
        gtext('Case 3: 2<=n < 4');
        pause
%Case: 4  4<=n<6
        h4=zeros(size(n));
        h4(8:10)=ones(size(n(8:10)));
        subplot(312);plot(n,h4,'o');grid;
        xlabel('k'); ylabel('h(n-k)');
        gtext('Case 4: 4<=n <= 6');
        pause;
%Case: 5  6<=n
        h5=zeros(size(n));
        h5(13:15)=ones(size(n(13:15)));
        subplot(313);plot(n,h5,'o');grid;
        xlabel('k'); ylabel('h(n-k)');
        gtext('Case 5: n>6');
```


Input Signal
x(n)
Impulse Response
h(n)
Output Signal
y(n)
n


x(k)
h(n-k)
Case 1: n < 0
Case 2: 0<=n < 2
k


h(n-k)
Case 3: 2<=n < 4
Case 4: 4<=n <= 6
Case 5: n>6
k

Now the question:

Can we use DFT to perform linear convolution, i.e. convolution of two different size functions?

1. Assume that $h[n]$ has length M and $x[n]$ has length N; $M < N$.
2. Zero-pad both to the length $K \geq Max\{M, N\}$ to form augmented sequences: $h_a[n]$; $x_a[n]$ and perform a K-point periodic convolution to yield: $y_P[n]$.
3. $y_P[n]$ is a K-point sequence whereas, the linear convolution $y_L[n]$ of these sequences would have been L-points long where $L = M + N - 1$.

    - For $K = L$, both $y_P[n]$ and $y_L[n]$ would be identical.
    - For $K < L$, $y_P[n]$ would correspond to the sequence obtained by adding in **(time-aliasing)** the last $L - K$ values of $y_L[n]$ to the first $L - K$ points. Thus, the first $L - K$ points $y_P[n]$ will NOT corresponds to $y_L[n]$, while the reaming $2K - L$ would be the same in both sequences. $y_L[n]$.

Example 6.8: Perform the circular convolution as linear convolution with aliasing (example from my NTU, Singapore Class notes):

Suppose $x_1[n]: \quad 0 \le n \le L-1$

$x_2[n]: \quad 0 \le n \le P-1$

When $N \ge L+P-1, \quad x_1[n] \textcircled{N} x_2[n] = x_1[n] * x_2[n]$

Prove: $x_1[n] \textcircled{N} x_2[n] = \sum_{m=0}^{N-1} x_1[m] x_2[((n-m))_N] \quad 0 \le n \le N-1$

$$= \sum_{m=0}^{N-1} x_1[m] \left[\sum_{r=-\infty}^{\infty} x_2[n+rN-m] \right]$$

$$= \sum_{r=-\infty}^{\infty} \left[\sum_{m=0}^{N-1} x_1[m] x_2[n+rN-m] \right]$$

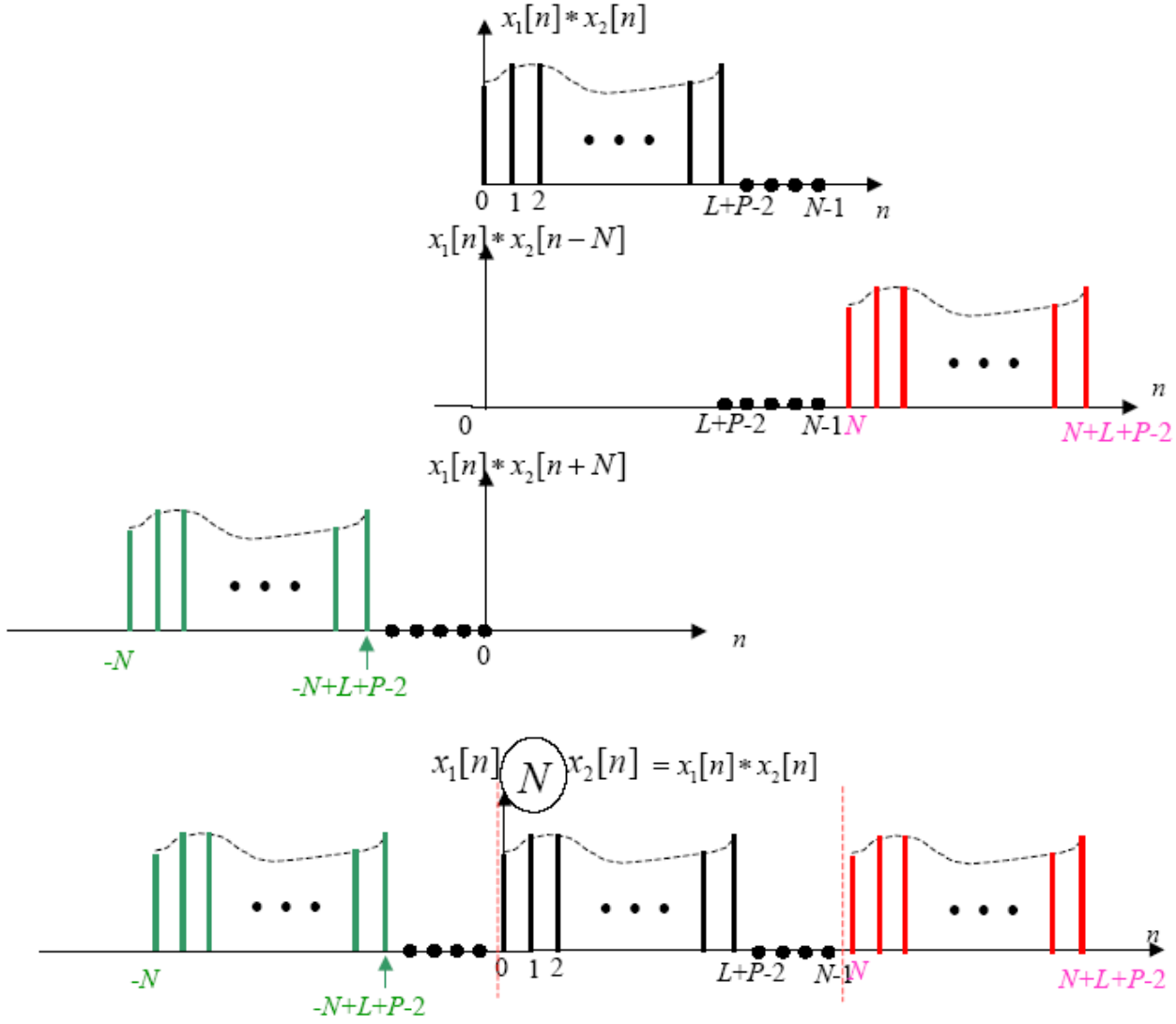
$$= \sum_{r=-\infty}^{\infty} [x_1[n] * x_2[n+rN]] \quad 0 \le n \le N-1 \qquad (\Delta)$$

Consider $x_1[n] * x_2[n+rN]$

When $r=0, \quad x_1[n] * x_2[n+rN] = x_1[n] * x_2[n]$

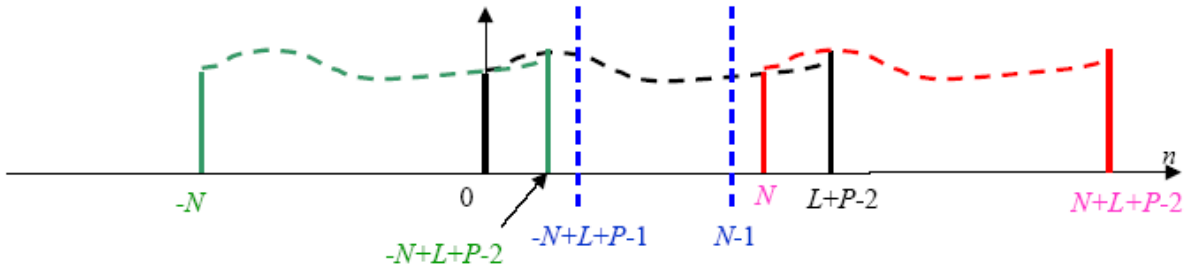
When $r=-1, \quad x_1[n] * x_2[n+rN] = x_1[n] * x_2[n-N]$

When $r=1, \quad x_1[n] * x_2[n+rN] = x_1[n] * x_2[n+N]$



$$x_1[n] \ \textcircled{N} \ x_2[n] = x_1[n] * x_2[n]$$

But, when $N < L+P-1$, $x_1[n] \textcircled{N} x_2[n] \neq x_1[n] * x_2[n]$



Duration of non-aliasing: $[L+P-N-1 \qquad N-1]$

Suppose $x_3[n] = x_1[n] * x_2[n]$

$$x_{3p}[n] = x_1[n] \textcircled{N} x_2[n]$$

then equation (Δ) become:

$$x_1[n] \textcircled{N} x_2[n] = \sum_{r=-\infty}^{\infty} [x_1[n] * x_2[n+rN]] \quad 0 \le n \le N-1 \qquad (\Delta)$$

$$x_{3p}[n] = \begin{cases} \sum_{r=-\infty}^{\infty} x_3[n+rN] & 0 \le n \le N-1 \\ 0 & \text{otherwise} \end{cases}$$

Thus the circular convolution of two finite-length sequences is Equivalent to linear convolution of the two sequences followed by time aliasing

Suppose $x_1[n]$-------length L,

$x_2[n]$-------length P

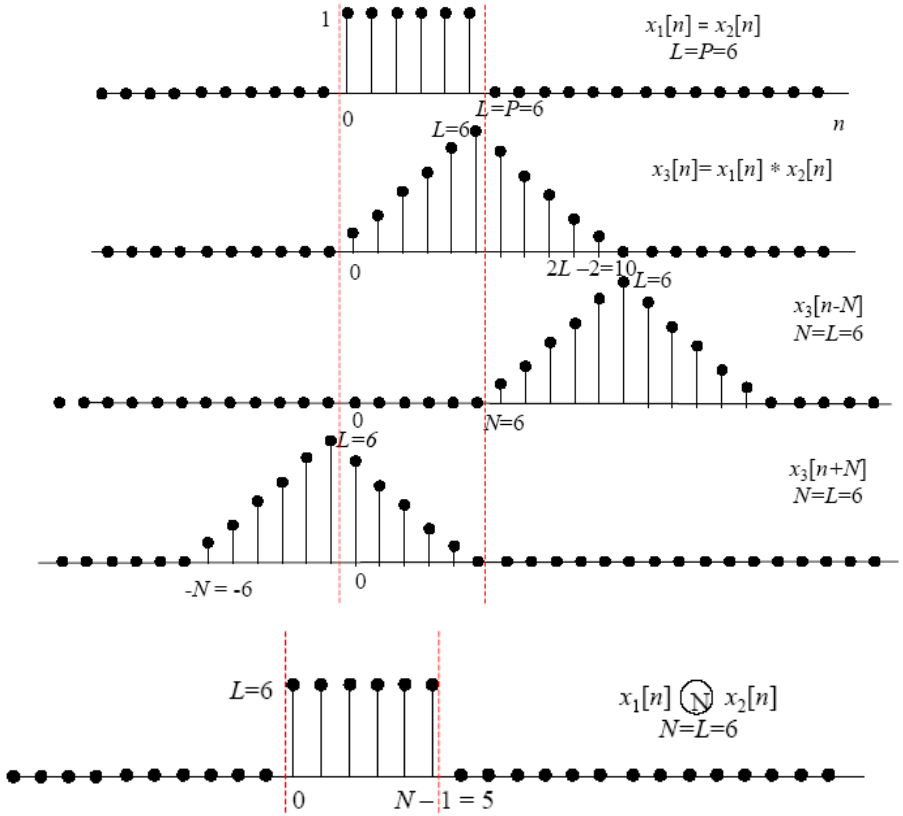
$x_3[n] = x_1[n] * x_2[n]$ -------length $N_1 = L+P-1$

Therefore: if $N \geqslant L+P-1$, then the circular convolution of $x_1[n]$ and $x_2[n]$ is equal to the linear convolution of the two sequences.

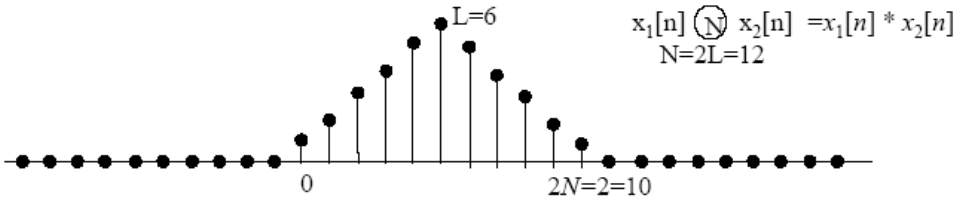
Example 6.9: Suppose $x_1[n] = x_2[n] = u[n] - u[n-6]$

(1) Compute $x_1[n]$ ⑥ $x_2[n]$

(2) Compute $x_1[n]$ (12) $x_2[n]$

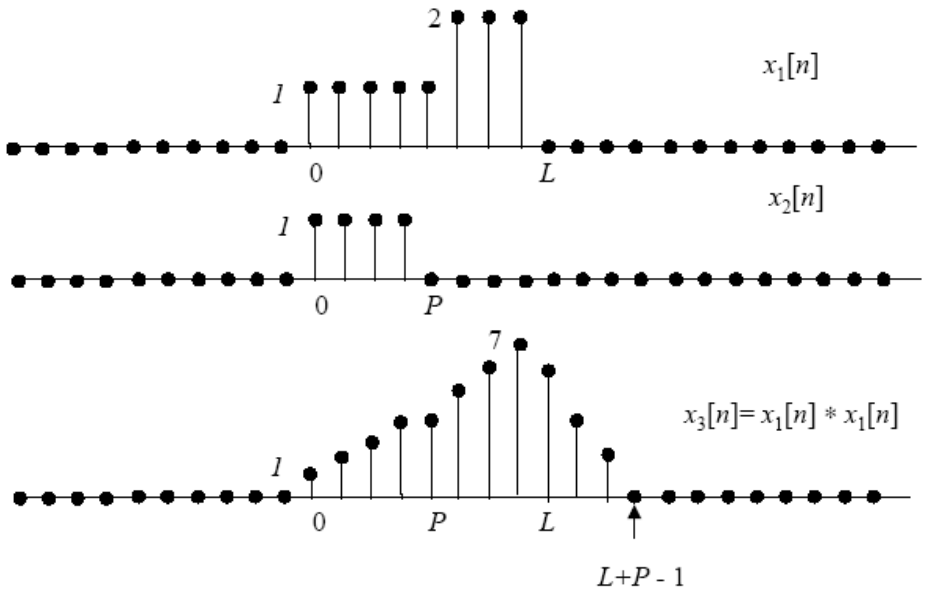


Duration of non-aliasing: $[L+P-N-1 \quad N-1]=[5 \quad 5]$



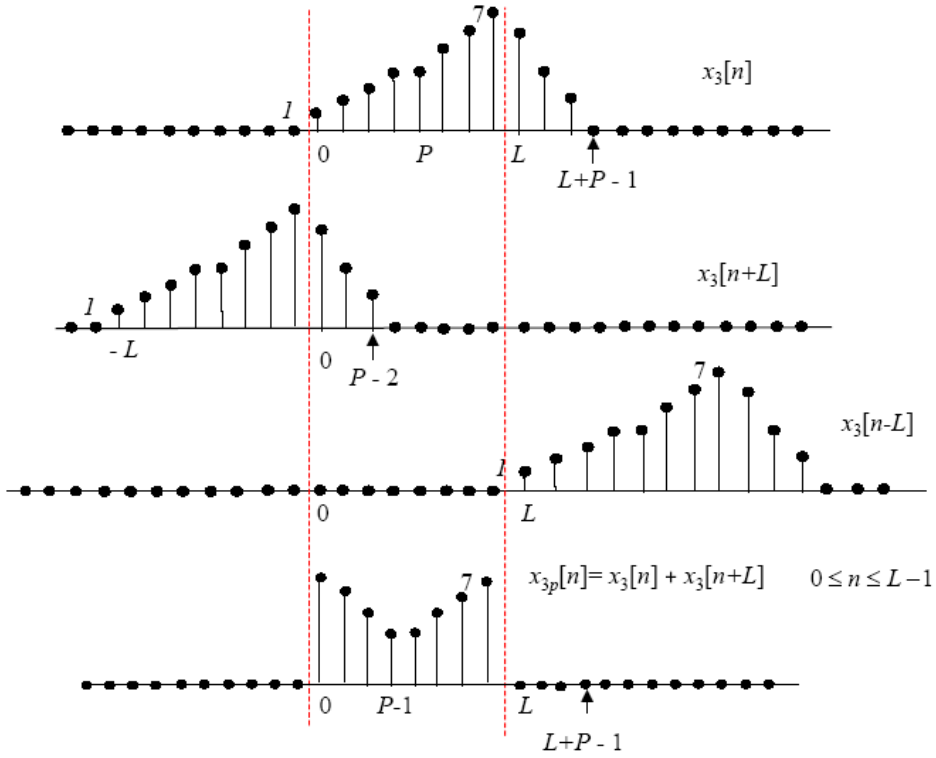
Conclusion: When $N \geq L+P-1$, time-aliasing in the periodic convolution of two-finite length discrete signals can be avoided.

Example 6.10: Perform linear convolution for $P<L$ of two sequences shown in the picture.



Let us now calculate $L-point$ periodic convolution:

$$x_{3p}[n]=\begin{cases} x_1[n] \textcircled{L} x_2[n]=\sum_{r=-\infty}^{\infty} x_3[n+rL] & 0\le n\le L-1 \\ 0 & \text{otherwise} \end{cases}$$



6.5 Fast Fourier Transform (FFT)

FFT since its introduction by Cooley-Tukey almost a half century ago has been playing historically sustained significant role in the development of DSP since it is the most widely used "fast algorithm" in solving many engineering challenges, designing filters, performing spectral analysis, estimation, noise cancellation and benchmark testing devices and systems, etc. Also, it is also very readily useable for computing the inverse transforms. Consider the definition of DFT in (6.1)

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2p}{N}nk} = \sum_{n=0}^{N-1} x[n]\{Cos(\frac{2p}{N}nk) - jSin(\frac{2p}{N}nk)\} \quad for \quad k = 0,1,2,\cdots,N-1$$

To appreciate its computational efficiency let us define:

$$1\,OP = 1\,Complex\ Multiply + 1\,Complex\ Add$$
$$= 1\,Cos\ Multiply + 1\,Sin\ Multiply + 1\,Complex\ Add + 1\,Complex\ Accumulate \qquad (6.20)$$

- To compute N-Point DFT we need $N^2\ OPS$.
- Cooley-Tukey basic FFT algorithm requires $N.\log_2 N\ OPS$.

m	$N = 2^m$	DFT OPs	Cooley-Tukey FFT OPs	FFT Savings
1	2	4	2	50%
2	4	16	8	50%
3	8	64	24	62.5%
4	16	256	64	75%
5	32	1024	160	84.4%
6	64	4096	384	90.6%
7	128	16384	896	94.5%
8	256	65536	2048	96.9%
9	512	262144	4608	98.2%
10	1024	1048576	10240	99.0%

- 99% savings in computational complexity is unheard of in any other fast algorithm in science. Even for reasonable and frequently observed FTT sizes of 128 or 256-points FFT we are in the 90% savings range.

Decimation-in-Time Algorithm (DIT): Let us divide $x[n]$ into two subsequences, each with length $N/2$, by grouping the even indexed samples and the odd-indexed samples. Then (6.7) can be written as:

$$X(k) = \sum_{n=0}^{N-1} x[n].W_N^{kn} = \sum_{n=odd} x[n].W_N^{kn} + \sum_{n=even} x[n].W_N^{kn} \qquad (6.21)$$

Let $n=2r$ in the first and $n=2r+1$ in the second sum:

$$X(k) = \sum_{r=0}^{N/2-1} x[2r].W_N^{2kr} + \sum_{r=0}^{N/2-1} x[2r+1].W_N^{(2r+1)k} \qquad with \quad g[r] = x[2r]$$
$$= \sum_{r=0}^{N/2-1} g[r].W_N^{2kr} + W_N^k . \sum_{r=0}^{N/2-1} h[r].W_N^{2kr} \qquad with \quad h[r] = x[2r+1] \qquad (6.22)$$

These $N/2$ point terms uses the fact that

$$W_{N/2} = e^{-j\frac{2p}{N/2}} = e^{-j\frac{2p.2}{N}} = (e^{-j\frac{2p}{N}})^2 = W_N^2 \qquad (6.23)$$

with this we can write (6.22) as:

$$X(k) = G(k) + W_N^k . H(k) \qquad for\ k = 0,1,2,\cdots, N-1 \qquad (6.24)$$

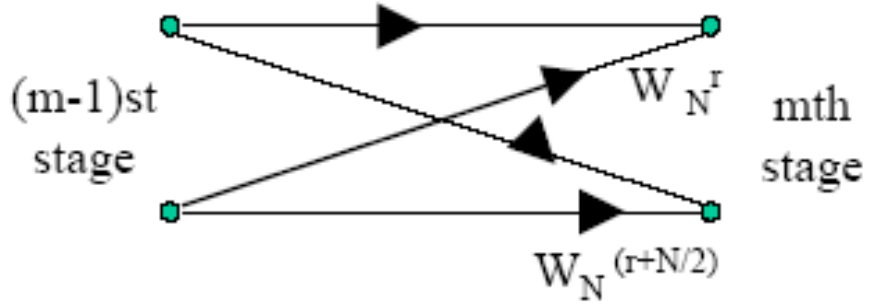
where $G(k)$ and $H(k)$ represent *N/2* point DFTs of the corresponding sequences in the time-domain, respectively.. Since $G(k)$ and $H(k)$ are periodic with a period *N/2*, we can write the last equation as:

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

$$and \qquad for\ k = 0,1,2,\cdots,\frac{N}{2}-1 \qquad (6.25)$$

$$X(k+\frac{N}{2}) = G(k) + W_N^{k+N/2} . H(k)$$

This last expression can be illustrated by a signal-flow graph called “Butterfly.”



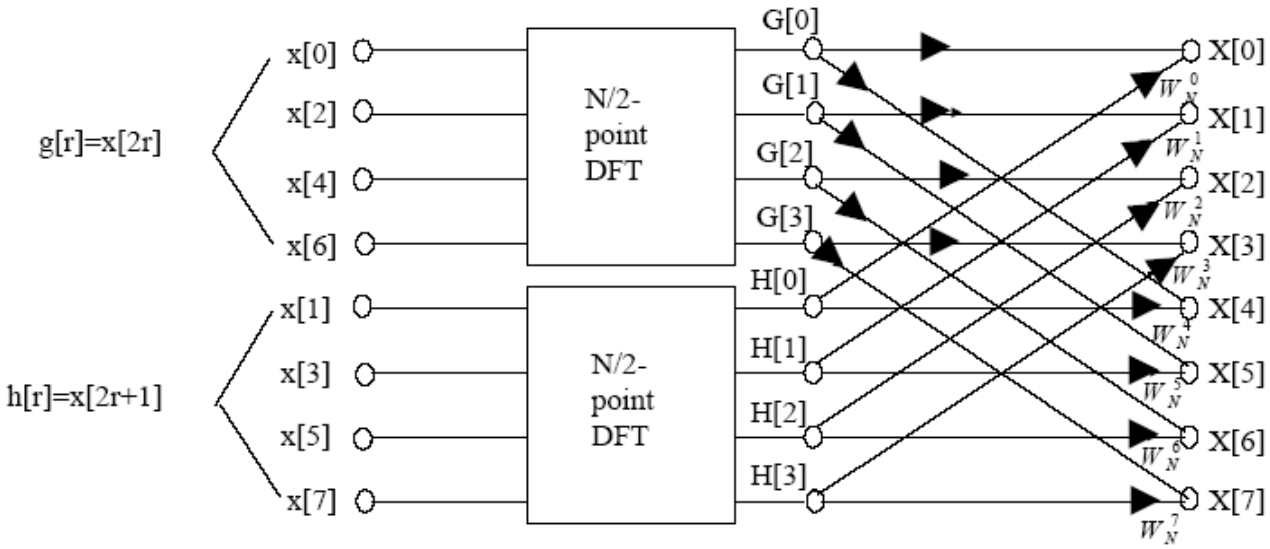
where $W_N^r = W_N^{n.k}$. It is clear from above that the signal at any node is the sum of all branches entering the node. This butterfly is true for any particular value of k in a given range. It can be applied repeatedly to compute the larger order FFT tasks as it is done below for an 8-point FFT.

$$X(0) = G(0) + W_8^0 . H(0) \qquad X(1) = G(1) + W_8^1 . H(1) \qquad X(2) = G(2) + W_8^2 . H(2)$$

$$X(3) = G(3) + W_8^3 . H(3) \qquad \text{But: } G(0) = G(4);\ G(1) = G(5);\ G(2) = G(6);\ G(3) = G(7)$$

$$X(4) = G(4) + W_8^4 . H(4) = G(0) + W_8^4 . H(0) \qquad X(5) = G(1) + W_8^5 . H(1)$$

$$X(6) = G(2) + W_8^6 . H(2) \qquad X(7) = G(3) + W_8^7 . H(3)$$



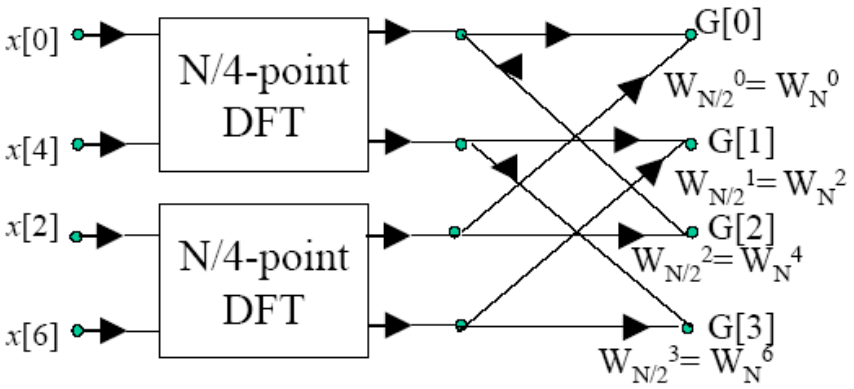
Since $N/4 = 2$ is even, we can re-iterate the process again:

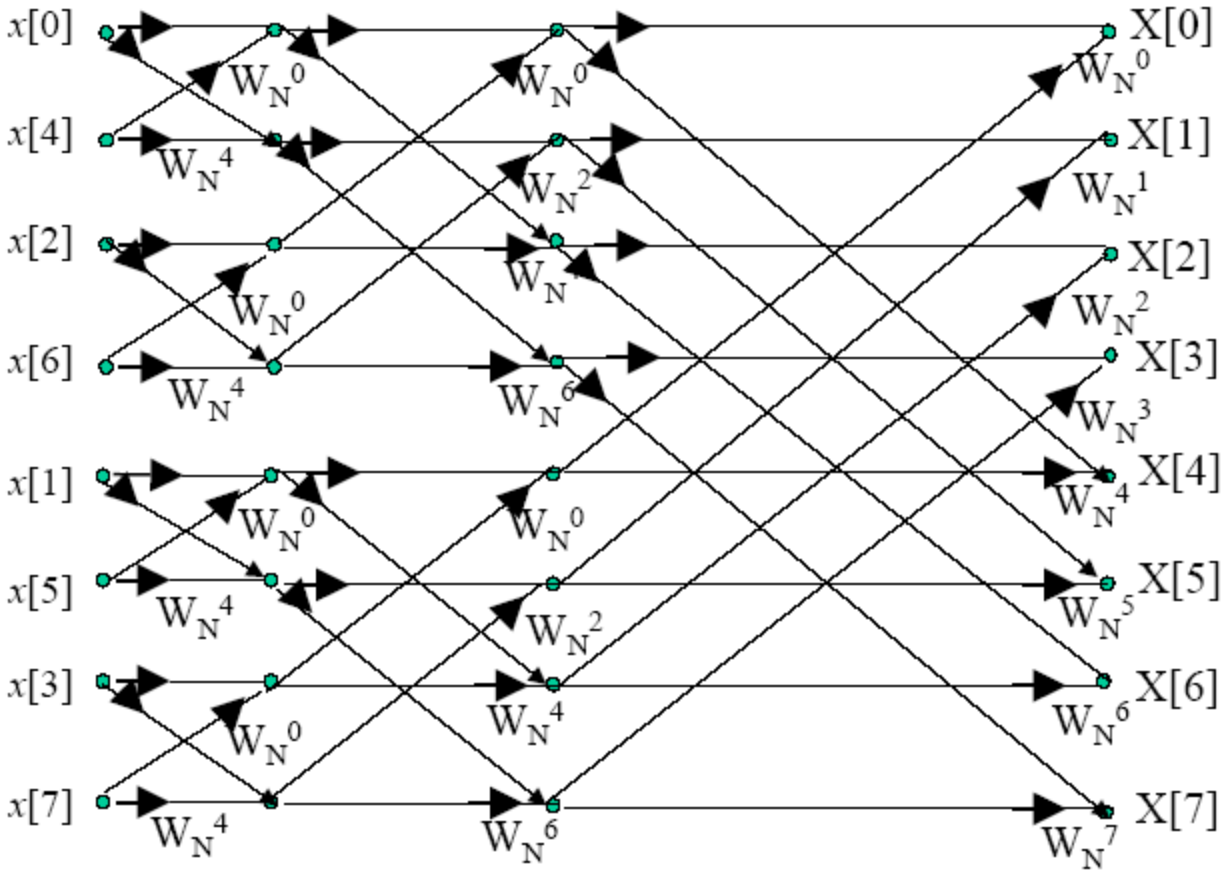
$$G(k) = \sum_{r=0}^{N/2-1} g[r].W_{N/2}^{rk} = \sum_{l=0}^{N/4-1} g[2l].W_{N/2}^{2lk} + \sum_{l=0}^{N/4-1} g[2l+1].W_{N/2}^{(2l+1)k}$$

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

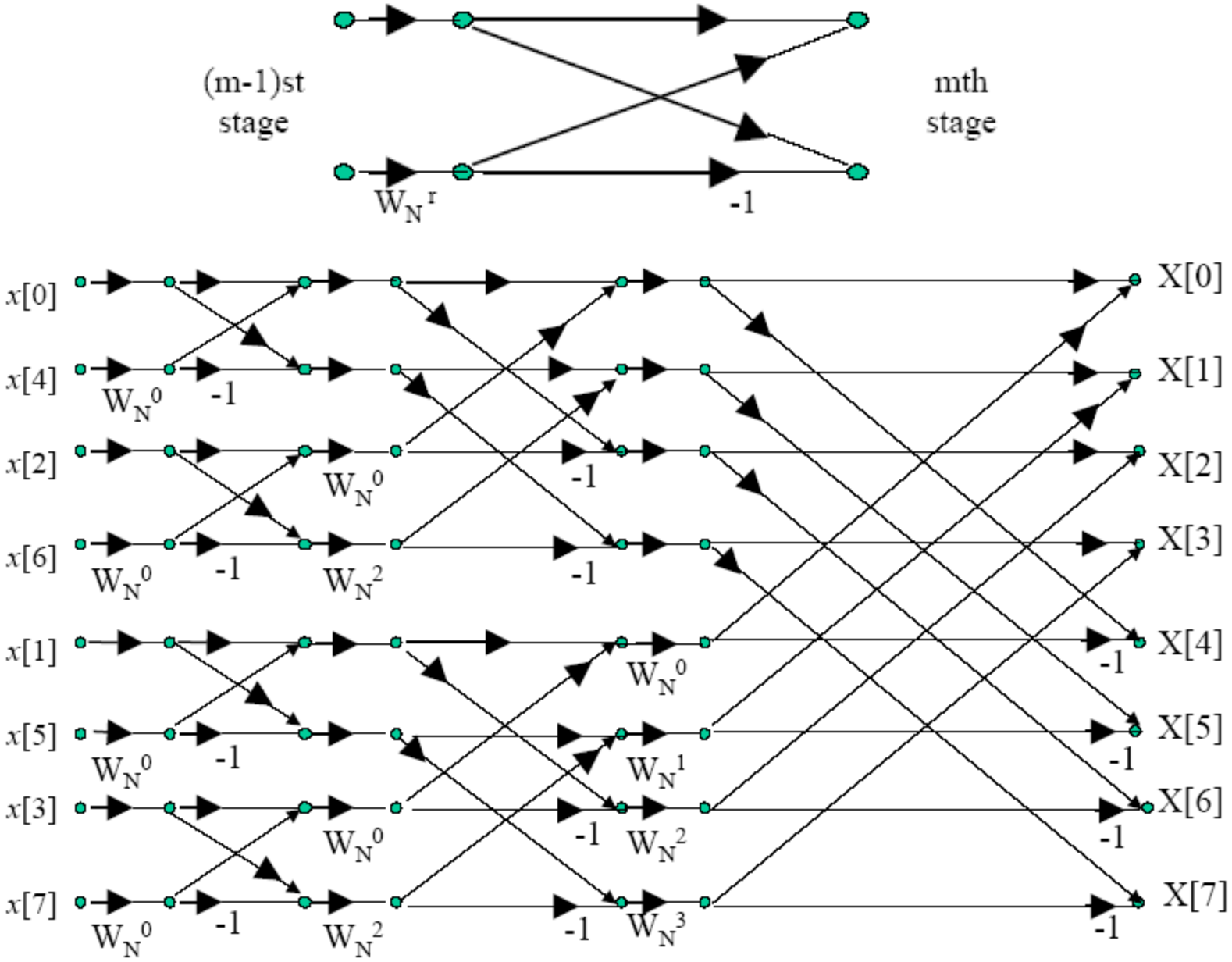
and (6.26)

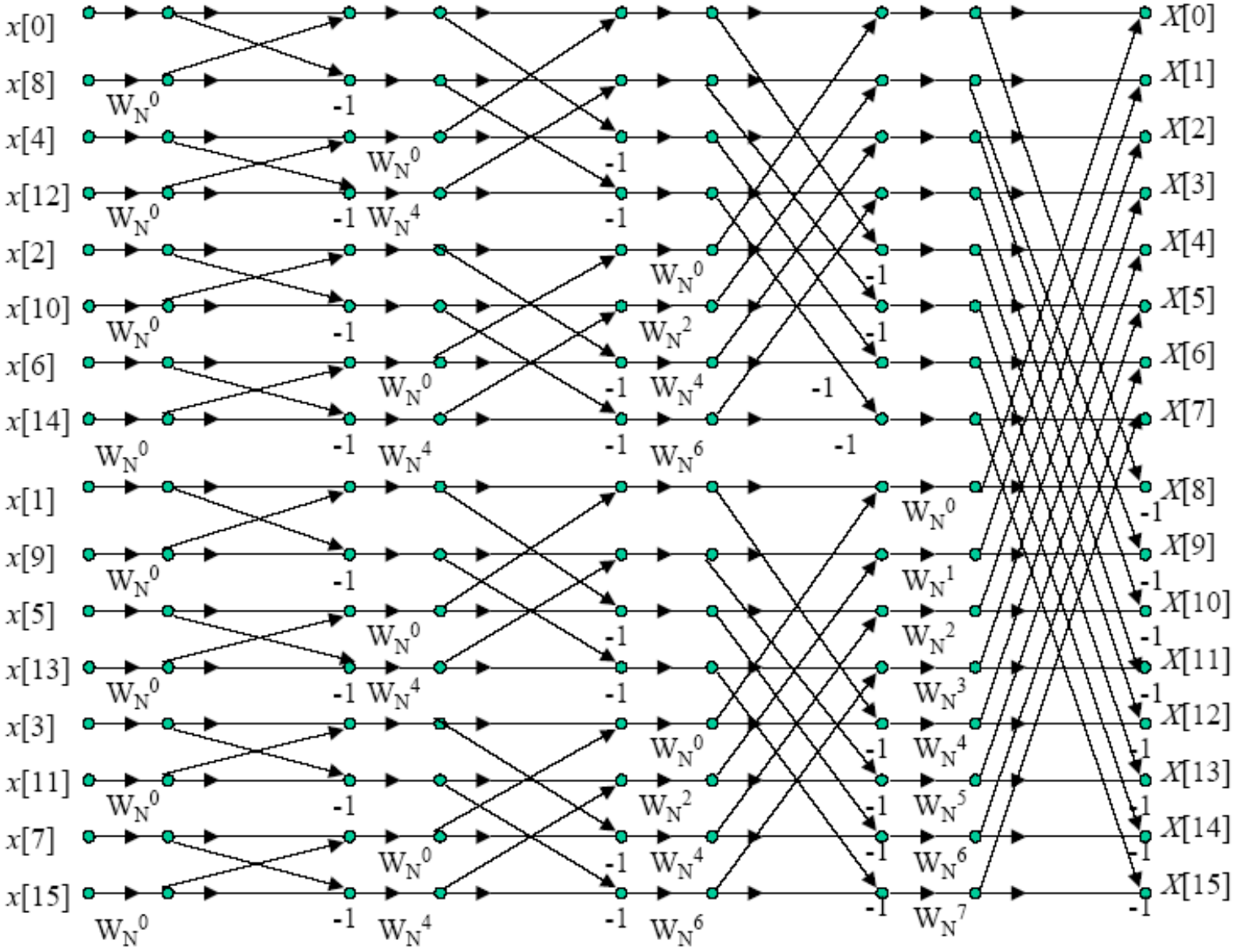
$$H(k) = \sum_{l=0}^{N/4-1} h[2l].W_{N/2}^{2lk} + W_{N/2}^{lk} \sum_{l=0}^{N/4-1} h[2l+1].W_{N/2}^{2lk}$$


x[0]
x[4]
x[2]
x[6]
N/4-point DFT
N/4-point DFT
G[0]
G[1]
G[2]
G[3]
$W_{N/2}^{0} = W_N^{0}$
$W_{N/2}^{1} = W_N^{2}$
$W_{N/2}^{2} = W_N^{4}$
$W_{N/2}^{3} = W_N^{6}$


x[0]
x[4]
x[2]
x[6]
x[1]
x[5]
x[3]
x[7]
X[0]
X[1]
X[2]
X[3]
X[4]
X[5]
X[6]
X[7]
W_N^0
W_N^4
W_N^2
W_N
W_N^6
W_N^1
W_N^3
W_N^5
W_N^7

Using the special case for a two-point butterfly, we obtain the full 8-point FFT process:

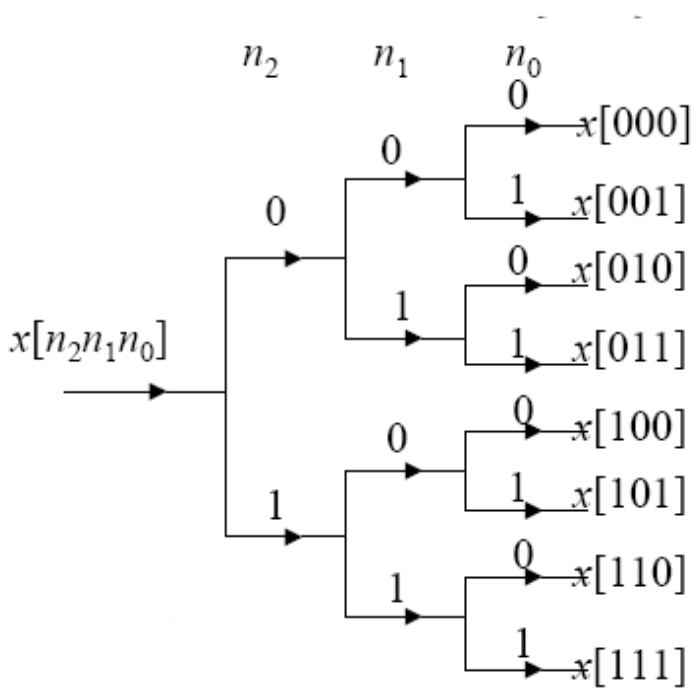


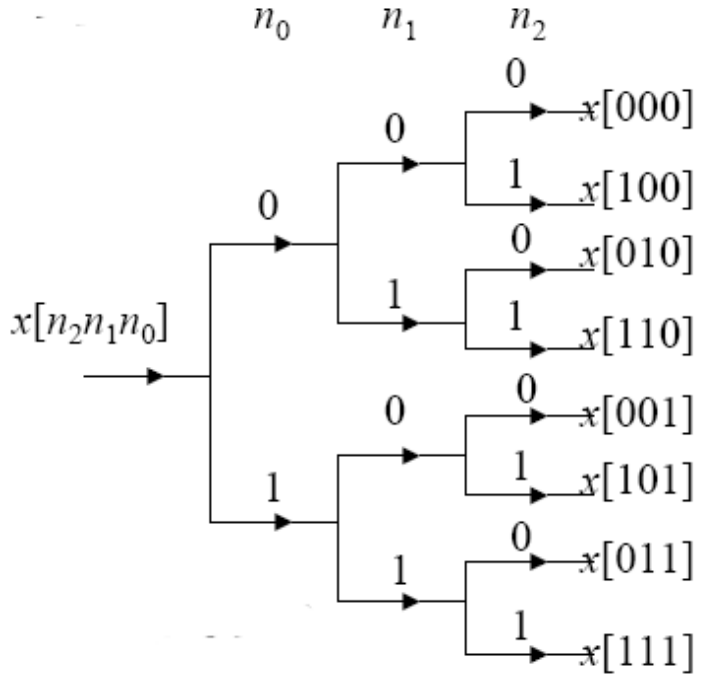


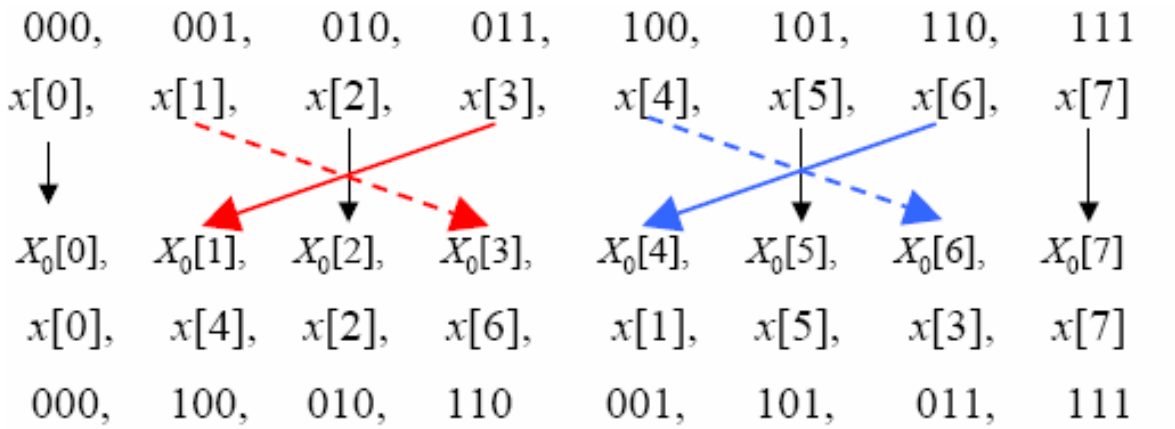
It is clear from above examples that the input sequence has to be organized in such a way that the output frequency terms appear in a proper sequential manner. The technique to achieve this systematically is called **In-Place FFT** or **bit-shuffling**.

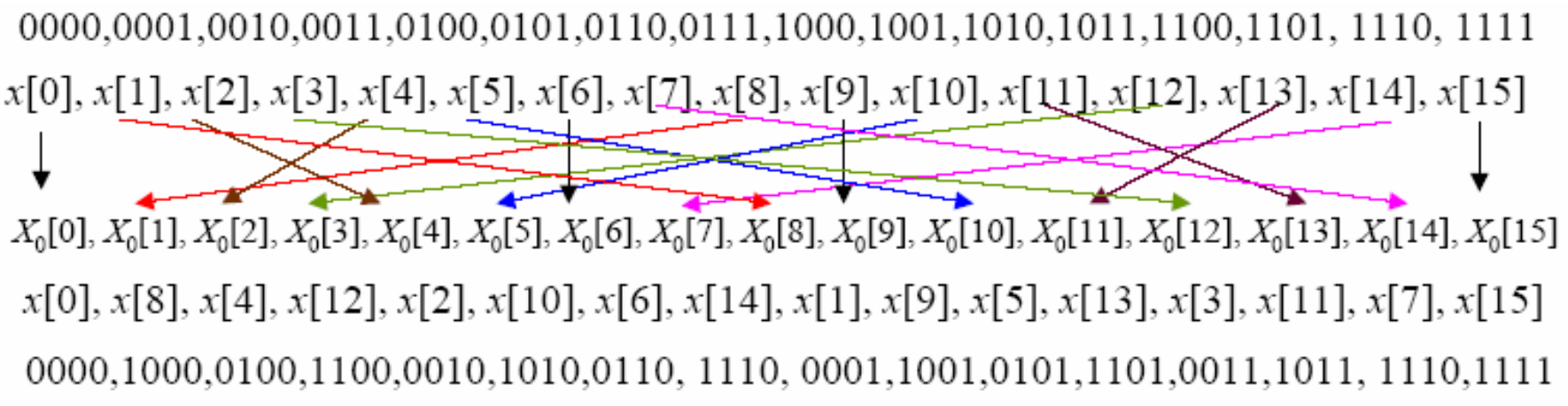
$x[000] = x_0[000]$ $x[001] = x_0[100]$ $x[010] = x_0[010]$ $x[011] = x_0[110]$

$x[100] = x_0[001]$ $x[101] = x_0[101]$ $x[110] = x_0[011]$ $x[111] = x_0[111]$









Decimation-in-Frequency Algorithm (DIF): It is obtained by dividing the output sequence $X(k)$, each with length $N/2$, by grouping the even indexed samples and the odd-indexed samples. Then (6.7) can be re-written this time:

$$X(k) = \sum_{n=0}^{N/2-1} x[n].W_N^{kn} + \sum_{n=N/2}^{N-1} x[n].W_N^{kn} = \sum_{n=0}^{N/2-1} x[n].W_N^{kn} + W_N^{k.N/2}.\sum_{n=0}^{N/2-1} x[n+\frac{N}{2}].W_N^{kn} \qquad (6.27)$$

Since $W_N^{k.N/2} = (-1)^k$ we can write the last expression as:

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

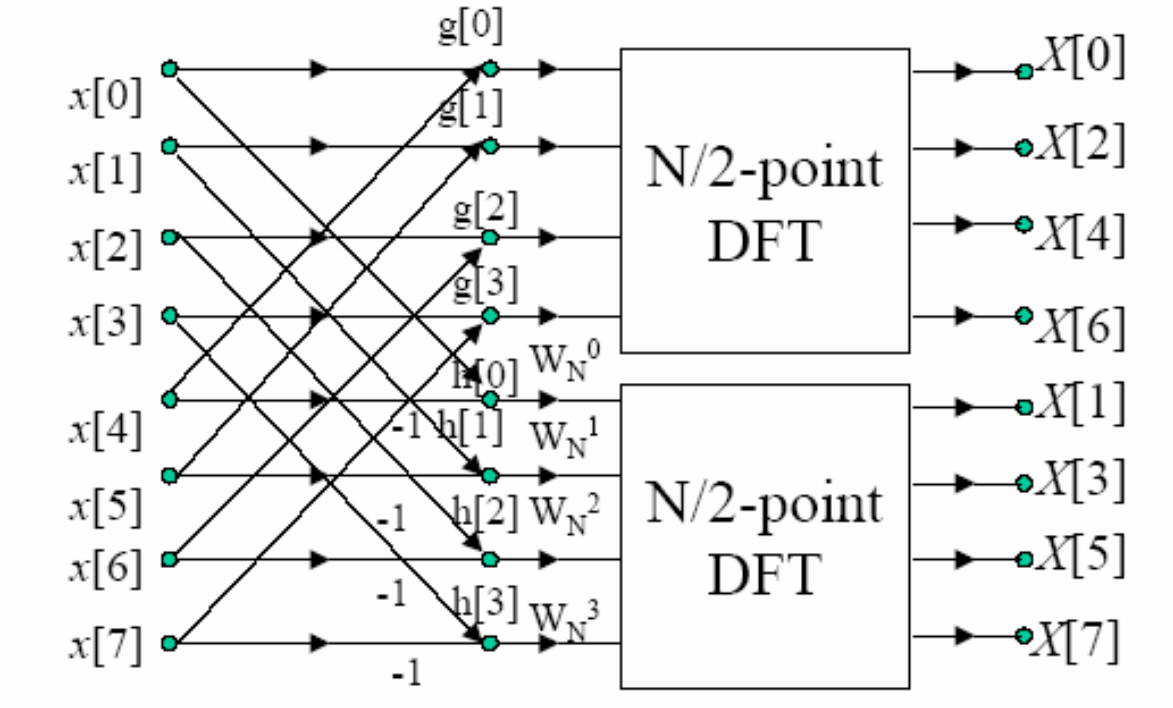
Now let us use the similar formulation as in DIT case:

$$g[n] = x[n] + x[n+\frac{N}{2}] \quad and \quad h[n] = (x[n] - x[n+\frac{N}{2}]).W_N^n \quad where \quad 0 \le n \le \frac{N}{2} - 1 \qquad (6.28)$$

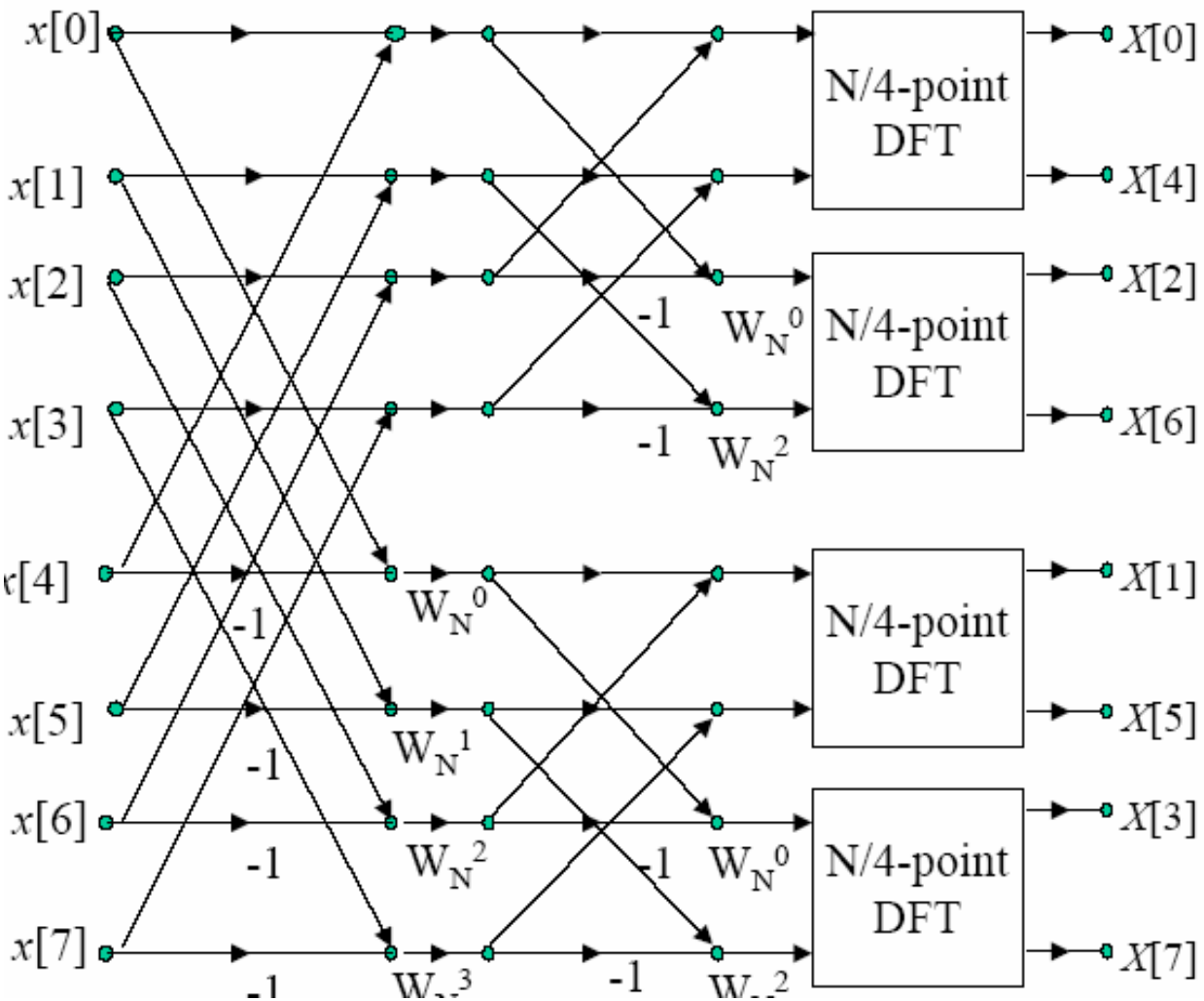
As we did in the previous case, we determine the two N/2-point DFTs $\{G(k), H(k)\}$ by computing the even and odd values separately by forming:

$$g_1[n] = g[n] + g[n + \frac{N}{4}] \quad \text{and} \quad g_2[n] = (g[n] - g[n + \frac{N}{4}]).W_{N/2}^{n} \tag{6.29a}$$

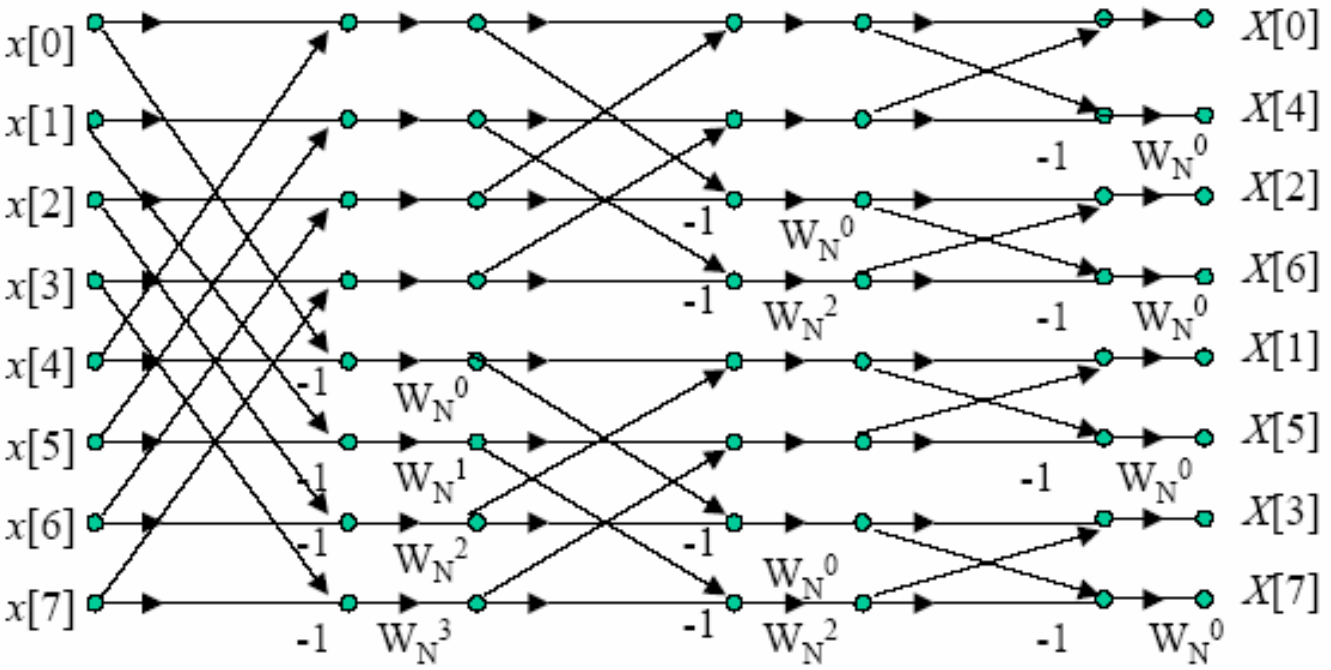
$$h_1[n] = h[n] + h[n + \frac{N}{4}] \quad \text{and} \quad h_2[n] = (h[n] - h[n + \frac{N}{4}]).W_{N/2}^{n} \tag{6.29b}$$



and



Replacing the last stage with butterflies we have the final 8-point FFT-DIF.



This time the output sequence is NOT in proper order. As in the DIT case, we need to perform bit-shuffling or in-place FFT at the output stage.

6.6 Spectrum Estimation and Windowing Using FFT

A large majority of real-life signals are analog signals and the question becomes how can we benefit from the computational efficiency of FFT for these signals.

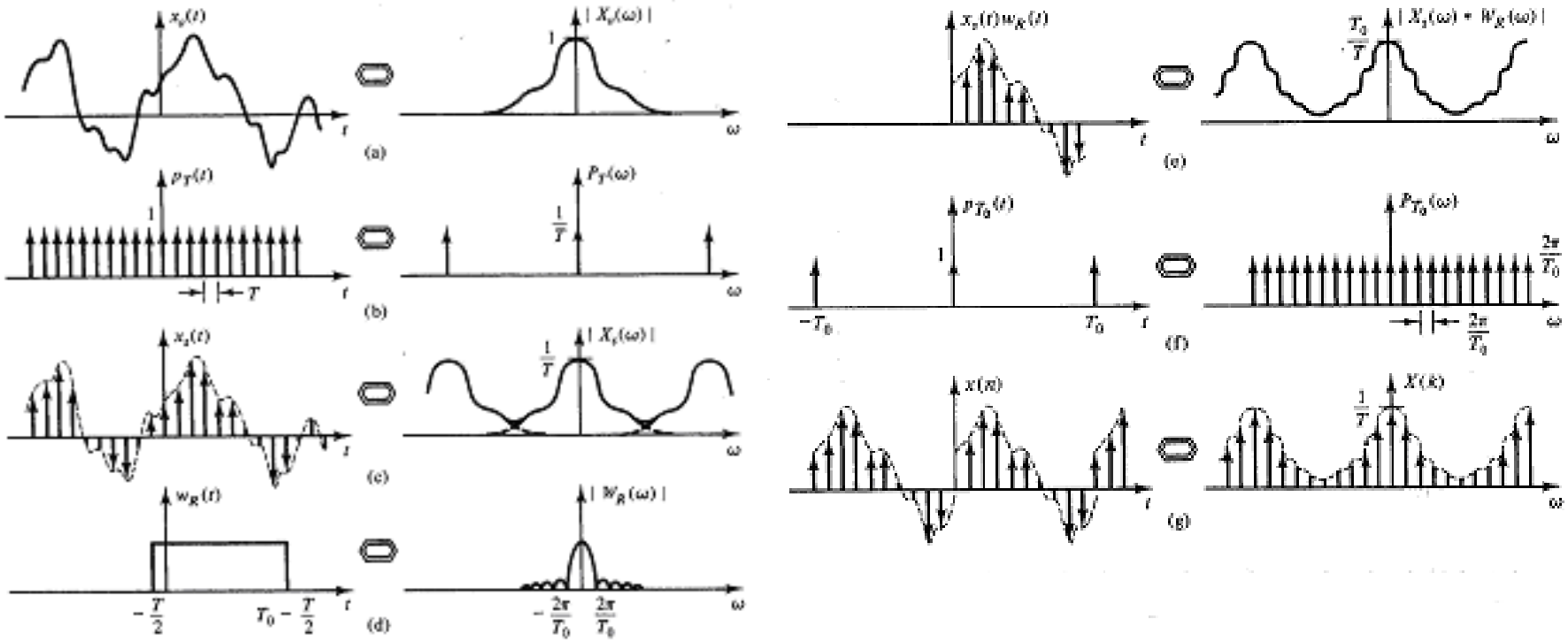
1. The first step in the process is to sample an analog signal $x_a(t)$ at a uniform rate, hopefully faster than the Nyquist rate in order to avoid aliasing. Let the sampling pulse be an ideal impulse train:

$$p_T(t) = \sum_{n=-\infty}^{\infty} d(t - nT) \qquad (6.30)$$

and the resulting sampled sequence and the term-by-term Fourier transform become:

$$x_S(t) = x_a(t).p_T(t) \qquad \text{and} \qquad X_S(w) = \frac{1}{T}\sum_{m=-\infty}^{\infty} X_a(w + mw_S) \qquad (6.31)$$

The last expression has infinitely many terms. If the signal is band-limited then we can succeed.

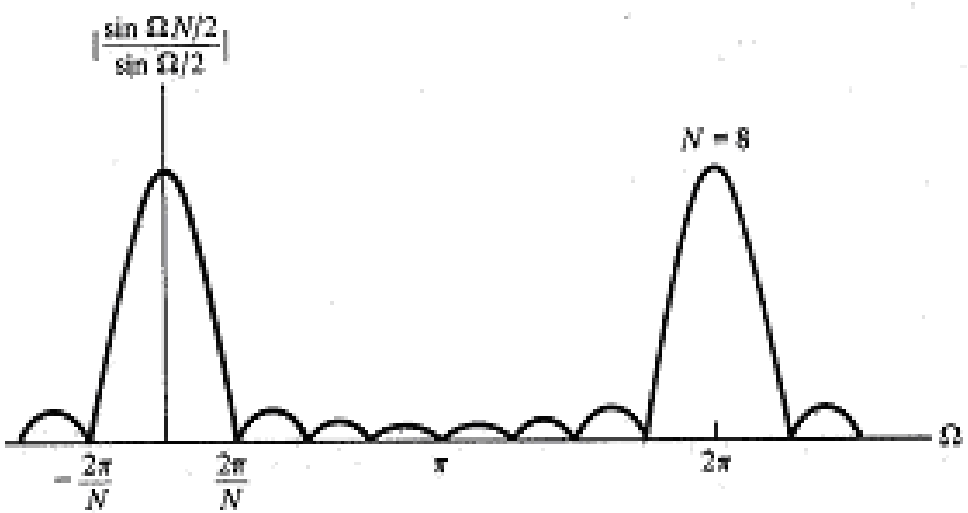


(Extracted with permission from *The Fourier Transform* by Omar Brigham, Prentice-Hall, 1987.)

2. The second step in the process is to isolate the sampled signal in a finite-window by multiplying with a window function $w(t)$. The length (size) of the window T_0 is related to the number of data points and the sampling interval via $T_0 = N.T$. The simplest window from is a rectangular one as shown in (c). The function and its Fourier transform are defined by:

$$w_R(t) = \begin{cases} 1 & if\ -\frac{T}{2} \le t < T_0 - \frac{T}{2} \\ 0 & Otherwise \end{cases} \quad \text{and} \quad W_R(w) = T_0 . \frac{Sin(wT_0/2)}{wT_0/2} e^{-jw\frac{T_0 - T}{2}} \qquad (6.32)$$

The shift of $T/2$ from the origin is introduced to avoid having data samples at the edges of the window. The resulting Fourier transform is shown in (d) or as in next picture.



Magnitude spectrum of a rectangular window.

3. The final step is to sample the number of frequency points at equal intervals in the frequency-domain. The spacing between samples is $w_S / N = 2p / T_0$.

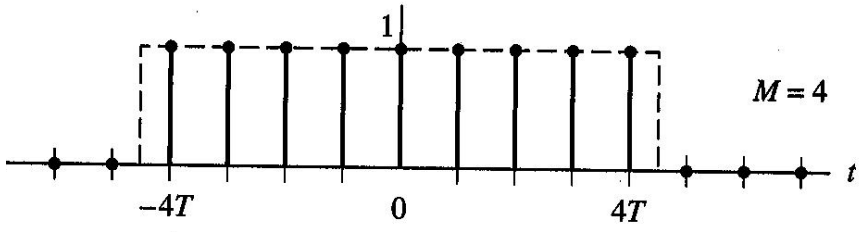
- Since the sampled signal in time-domain is obtained by multiplication then the sampling in frequency-domain will be modeled as the multiplication of: $X_S(w) * W_R(w)$ by the impulse train in the frequency-domain: $P_{T_0}(w) = \frac{2p}{T_0} \sum_{m=-\infty}^{\infty} d(w - m\frac{2p}{T_0})$, which has an inverse FT: $p_{T_0}(t) = \sum_{m=-\infty}^{\infty} d(t - mT_0)$. The result will be the periodic extension of the signal $x_S(t).w_R(t)$ with period T_0. (figure g).
- So, the original signal is replaced by its periodic extension.
- For a general analog signal the spectrum obtained by DFT is somewhat different from the true spectrum.

There are two errors in the process:

1. The aliasing error due to sampling if not properly pre-filtered before sampling.

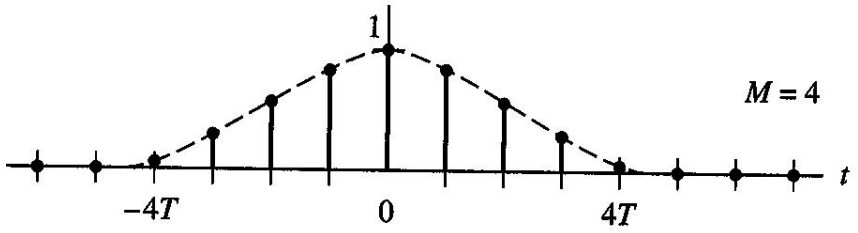
2. The windowing introduces ripples into the spectrum due to the convolution operation causing the signal component to spread over or leak into other frequencies. For no leakage, the FT of the window function must be a delta function, which is not. So, windowing causes leakage.

- To minimize spectral leakage, we need to choose a window which is close to a delta function as possible. The rectangular function is not.



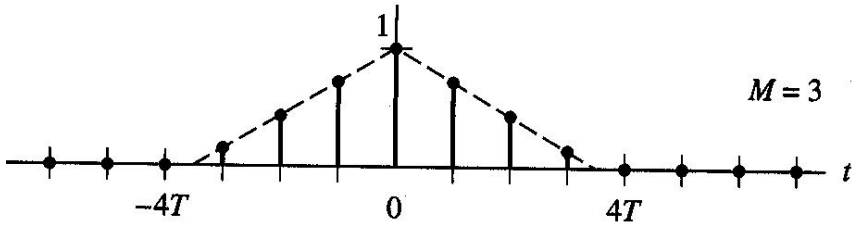
$$w[nT] = \begin{cases} 1 & |n| \le M \\ 0 & |n| > M \end{cases}$$

(a) Rectangular Window



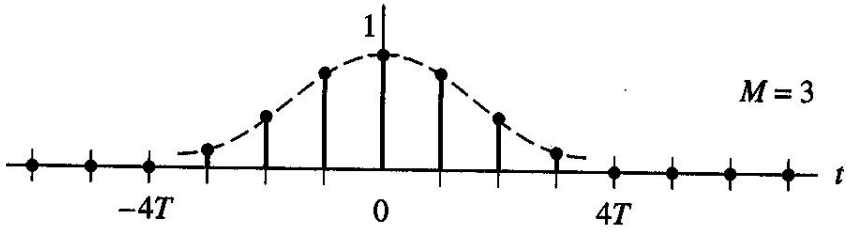
$$w[nT] = \begin{cases} 0.5(1 + \cos[\pi n/(M + 0.5)]) & |n| \le M \\ 0 & |n| > M \end{cases}$$

(c) Raised Cosine (Hanning) Window



$$w[nT] = \begin{cases} 1 - |n|/(M + 0.5) & |n| \le M \\ 0 & |n| > M \end{cases}$$

(b) Triangular (Bartlett) Window



$$w[nT] = \begin{cases} 0.54 + 0.46\cos[\pi n/(M + 0.5)] & |n| \le M \\ 0 & |n| > M \end{cases}$$

(d) Hamming Window

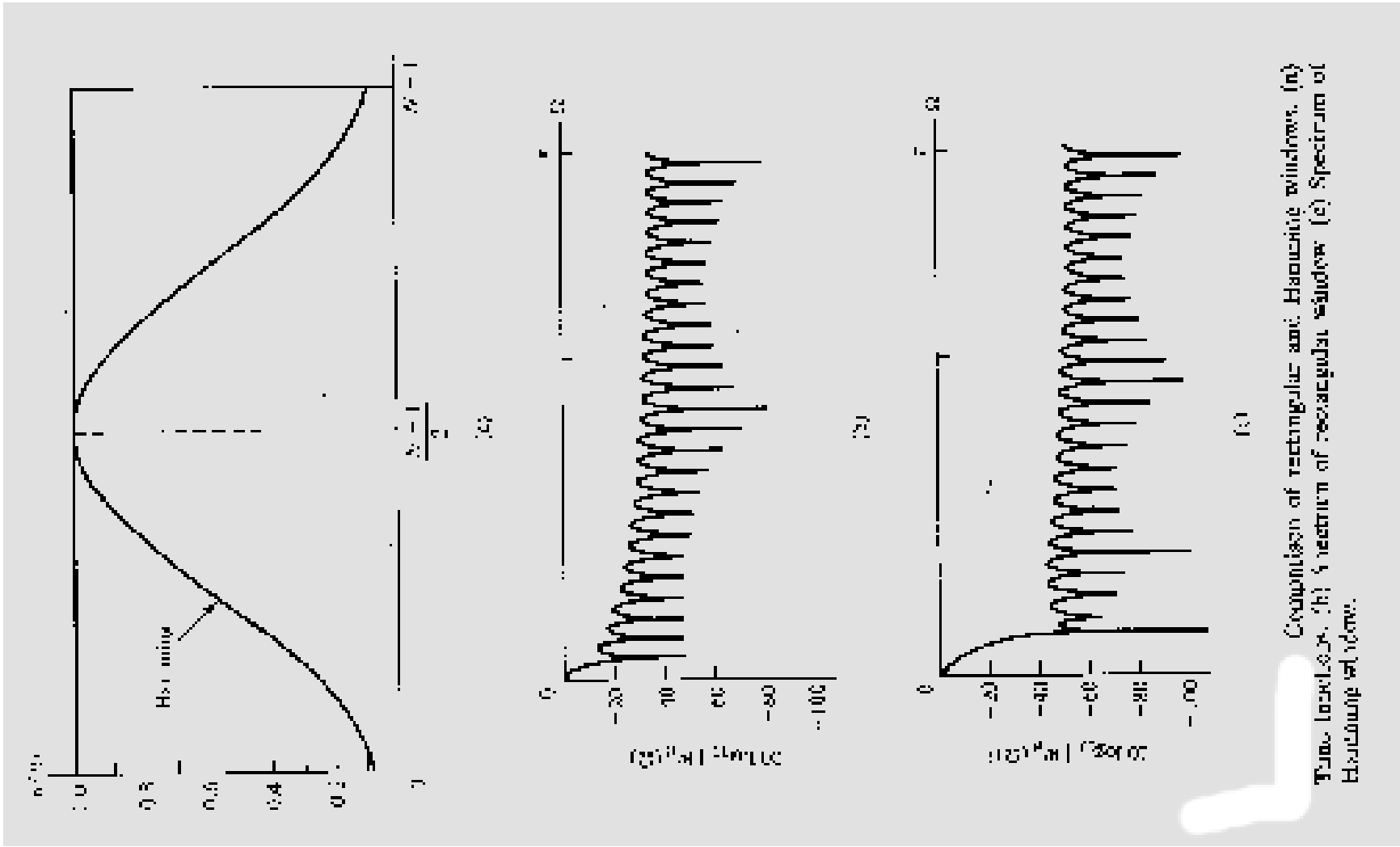
- The discrete version of the rectangular window is expressed by:

$$w_R[n] = \begin{cases} 1 & 0 \le n < N-1 \\ 0 & Otherwise \end{cases}$$ and the frequency response: $$W_R(\Omega) = \frac{Sin(\Omega N/2)}{Sin(\Omega/2)} e^{-j\Omega\frac{N-1}{2}}$$ (6.33)

- Let us compare the performance of a popular (significantly efficient) window (Hamming) against a rectangular window of (6.33). The time-domain behavior of an Hamming window is given by:

$$W_H[n] = 0.54 - 0.46.Cos\frac{2p}{N-1}n \qquad (6.34)$$

The windows and their spectra are shown below:



Observations:

- Since the tails of the window tapers off to zero in a smooth fashion, the end effects are avoided (no Gibbs phenomenon.)
- Both magnitude spectra exhibit LP characteristics.
- Rectangular window has approximately -13 dB for the first side lobe and settles down to about -30 dB in the long run.
- Hamming window has the first side lobe at -50 dB and it settles down to -48 dB in the long run. So, drastically superior to the rectangular window.
- Hamming window as the spectrum indicates can be used as a LOW-PASS filter by itself. In other words, take a sequence and subject it to a Hamming window and the net result is a low-pass filtering action! So, we have designed a LPF without going through a digital filter design process!

The most critical factor to be watched for is the *frequency resolution*, which stands for the spacing between frequency samples in the window. If the resolution is too low, we may miss critical information due to scarcity If the resolution is too high then the computational cost increases to make it impractical in many applications.

The frequency resolution is defined by:

$$\Delta w = \frac{w_S}{N} = \frac{2p}{NT} = \frac{2p}{T_0} \tag{6.35}$$

To improve the resolution, we need to use a longer data sequence if possible, otherwise we pad it with zeros.

Example 6.11: Compare the DFT (DFT) spectra of Hamming window for $N = 32,64,128$.

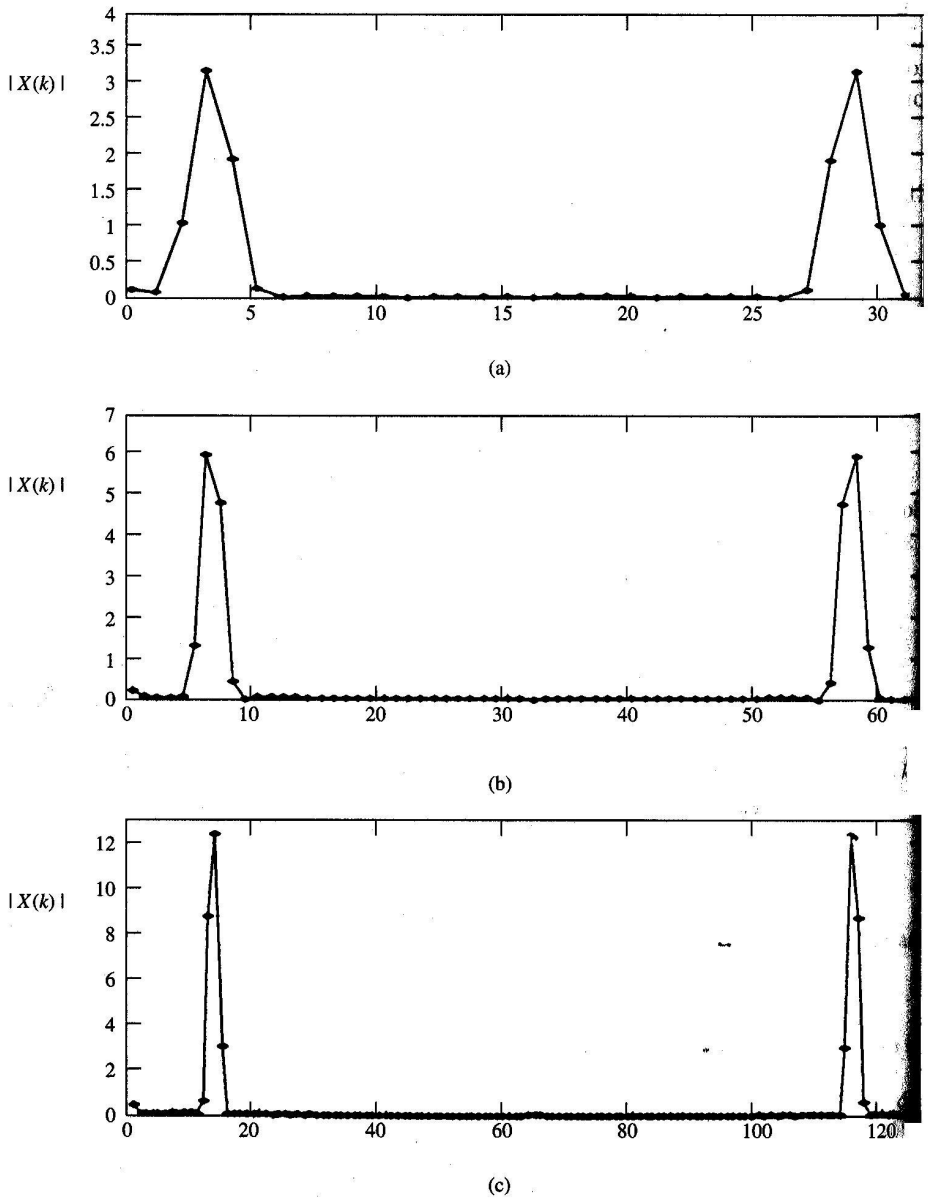


Figure 6.11 **DFT spectrum of analog signal $x_a(t)$ using Hamming window. (a) $N = 32$. (b)$N = 64$. (c) $N = 128$.**