

ESE 531: Digital Signal Processing

Lec 22: April 11, 2019

Fast Fourier Transform Pt 2



Lecture Outline

- ❑ DFT vs. DTFT
- ❑ FFT practice
- ❑ Chirp Transform Algorithm
- ❑ Circular convolution as linear convolution with aliasing



Discrete Fourier Transform

□ The DFT

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-kn} \quad \text{Inverse DFT, synthesis}$$

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

□ It is understood that,

$$x[n] = 0 \quad \text{outside } 0 \leq n \leq N - 1$$

$$X[k] = 0 \quad \text{outside } 0 \leq k \leq N - 1$$



DFT vs. DTFT

- The DFT are samples of the DTFT at N equally spaced frequencies


$$X[k] = X(e^{j\omega})|_{\omega=k\frac{2\pi}{N}} \quad 0 \leq k \leq N - 1$$

DFT vs. DTFT

□ Back to example

$$\begin{aligned} X[k] &= \sum_{n=0}^4 W_{10}^{nk} \\ &= e^{-j\frac{4\pi}{10}k} \frac{\sin(\frac{\pi}{2}k)}{\sin(\frac{\pi}{10}k)} \end{aligned}$$





Fast Fourier Transform Algorithms

- We are interested in efficient computing methods for the DFT and inverse DFT:

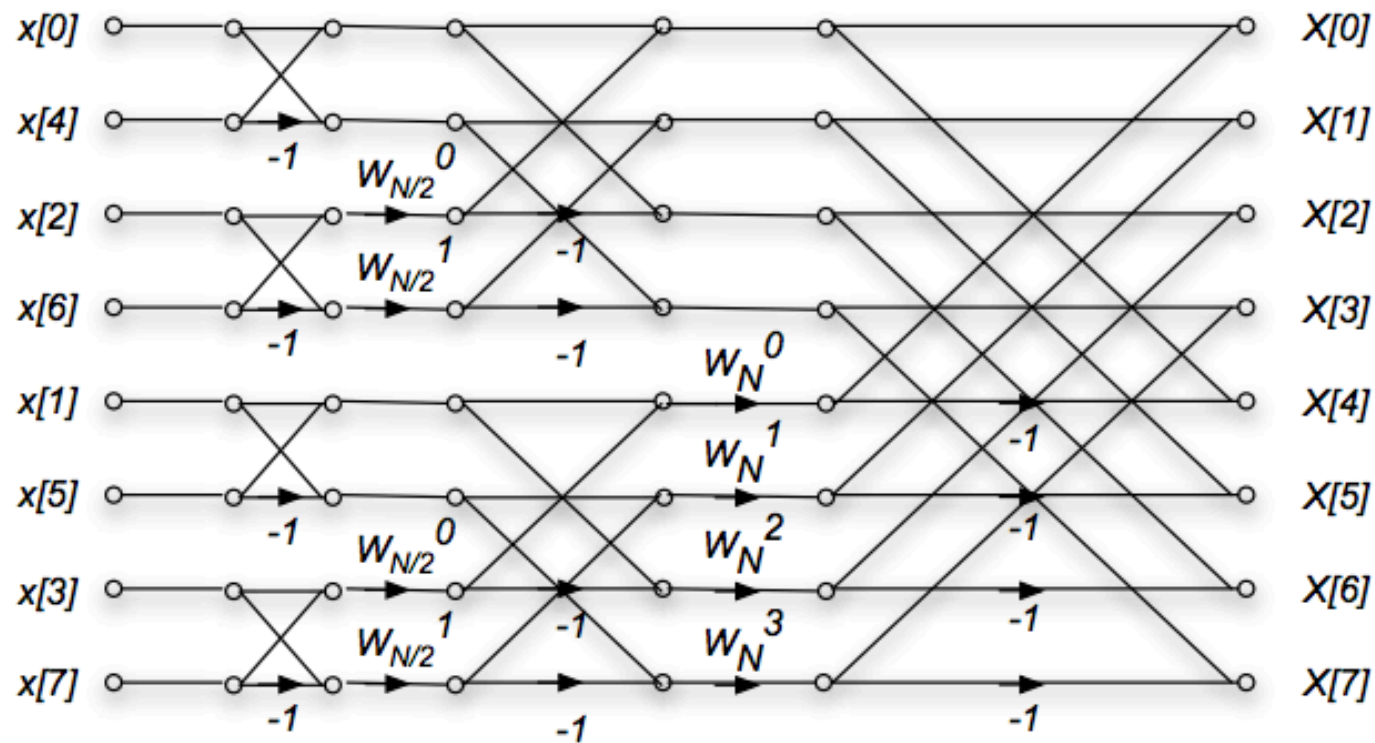
$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \quad k = 0, \dots, N-1$$

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


$$W_N = e^{-j\left(\frac{2\pi}{N}\right)}.$$

Decimation-in-Time FFT

Combining all these stages, the diagram for the 8 sample DFT is:



- $3 = \log_2(N) = \log_2(8)$ stages
- $4 = N/2 = 8/2$ multiplications in each stage
 - 1st stage has trivial multiplication

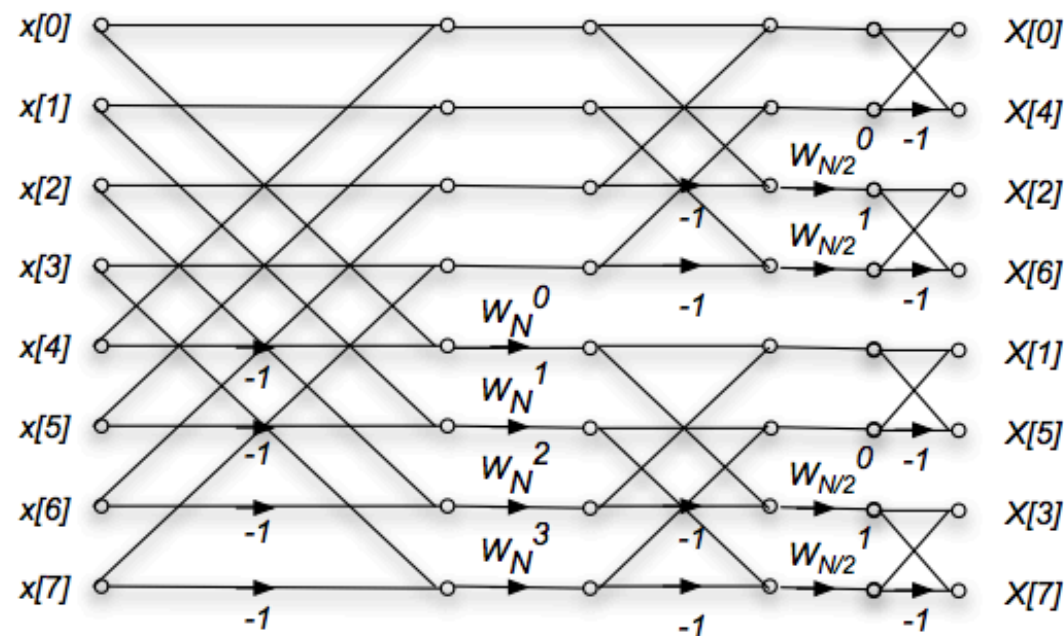


Decimation-in-Time FFT

- ❑ In general, there are $\log_2 N$ stages of decimation-in-time.
- ❑ Each stage requires $N/2$ complex multiplications, some of which are trivial.
- ❑ The total number of complex multiplications is $(N/2) \log_2 N$, or $O(N \log_2 N)$
- ❑ The order of the input to the decimation-in-time FFT algorithm must be permuted.
 - First stage: split into odd and even. Zero low-order bit (LSB) first
 - Next stage repeats with next zero-lower bit first.
 - Net effect is reversing the bit order of indexes

Decimation-in-Frequency FFT

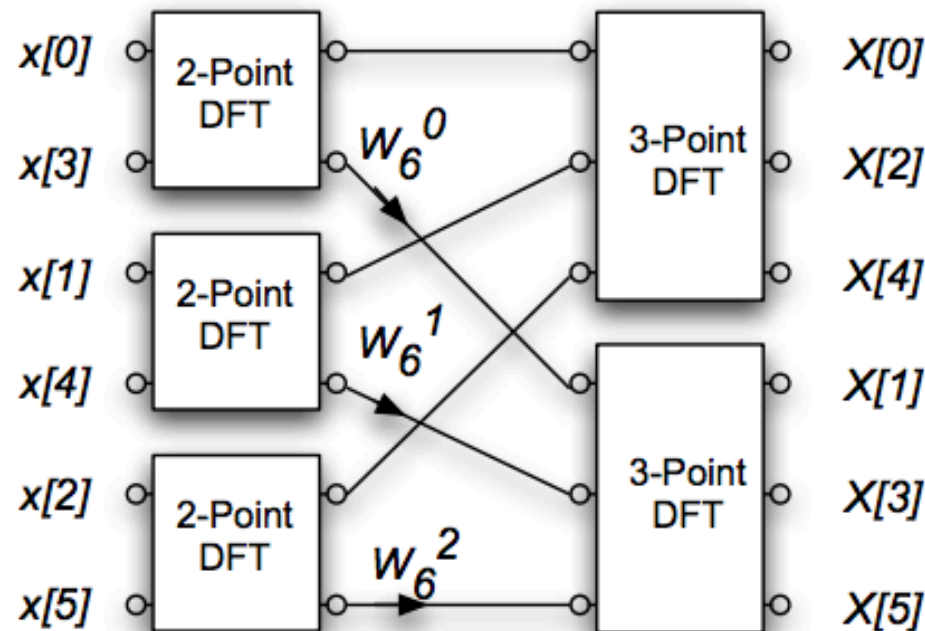
The diagram for an 8-point decimation-in-frequency DFT is as follows



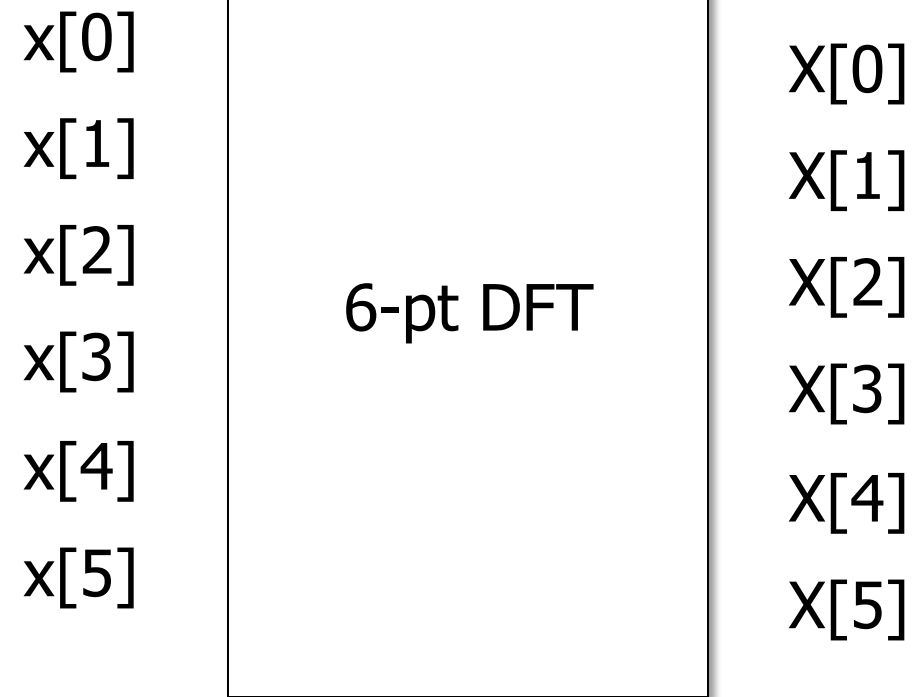
This is just the decimation-in-time algorithm reversed!
The inputs are in normal order, and the outputs are bit reversed.

Non-Power-of-2 FFTs

- A similar argument applies for any length DFT, where the length N is a composite number
- For example, if $N=6$, with decimation-in-frequency you could compute three 2-point DFTs followed by two 3-point DFTs



Example: Non-Power-of-2 FFTs



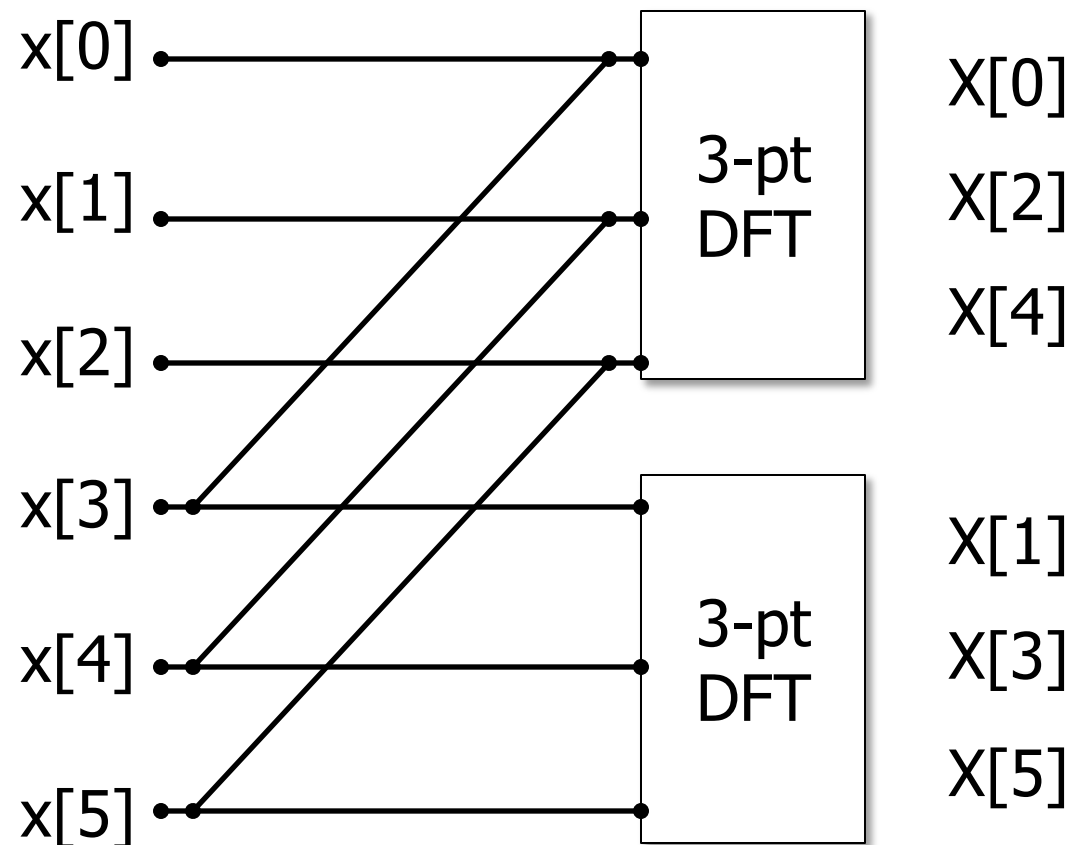


Decimation-in-Frequency FFT

$$\begin{aligned} X[2r] &= \text{DFT}_{\frac{N}{2}} \{(x[n] + x[n + N/2])\} \\ X[2r + 1] &= \text{DFT}_{\frac{N}{2}} \{(x[n] - x[n + N/2]) W_N^n\} \end{aligned}$$

(By a similar argument that gives the odd samples)

Example: Non-Power-of-2 FFTs



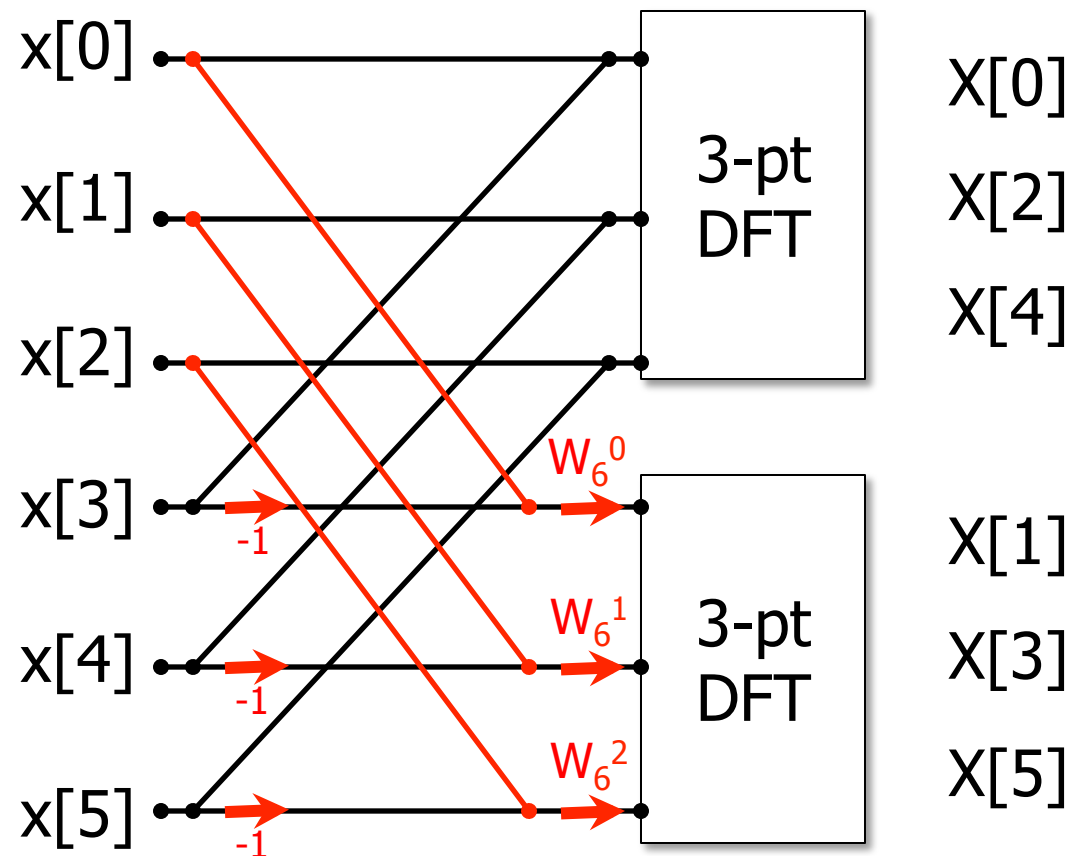


Decimation-in-Frequency FFT

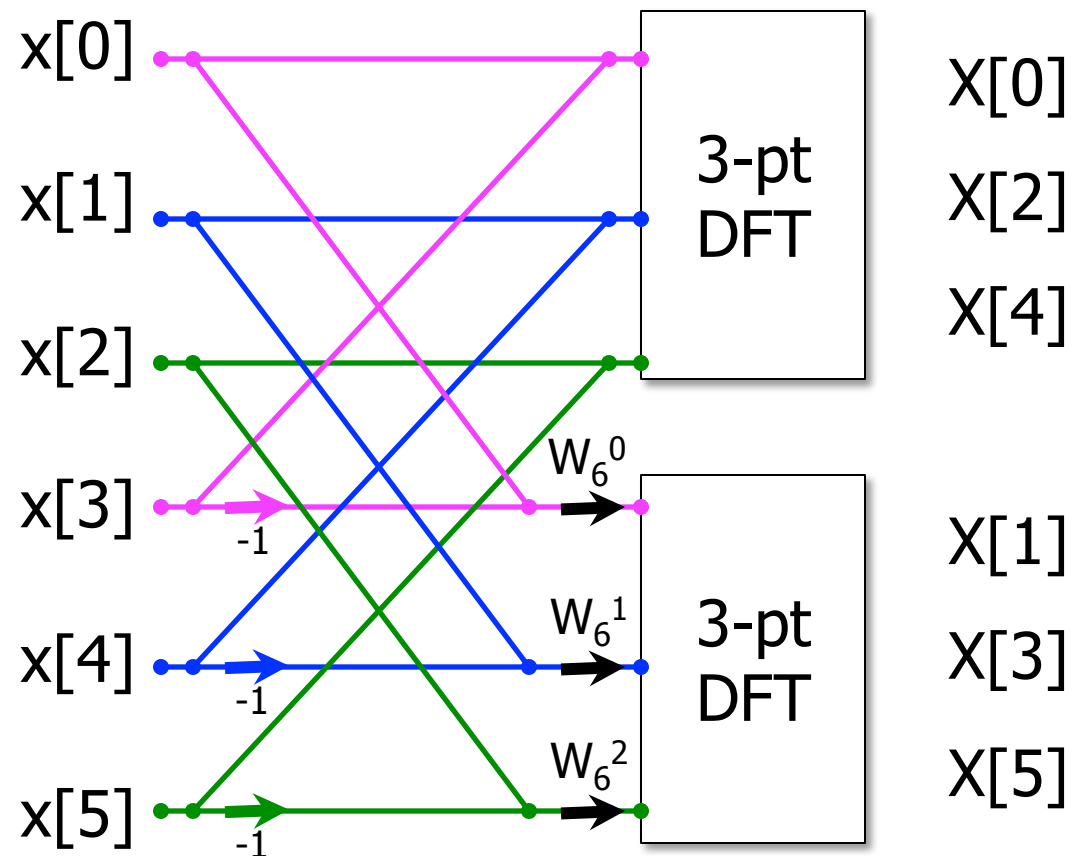
$$\begin{aligned} X[2r] &= \text{DFT}_{\frac{N}{2}} \{(x[n] + x[n + N/2])\} \\ X[2r + 1] &= \text{DFT}_{\frac{N}{2}} \{(x[n] - x[n + N/2]) W_N^n\} \end{aligned}$$

(By a similar argument that gives the odd samples)

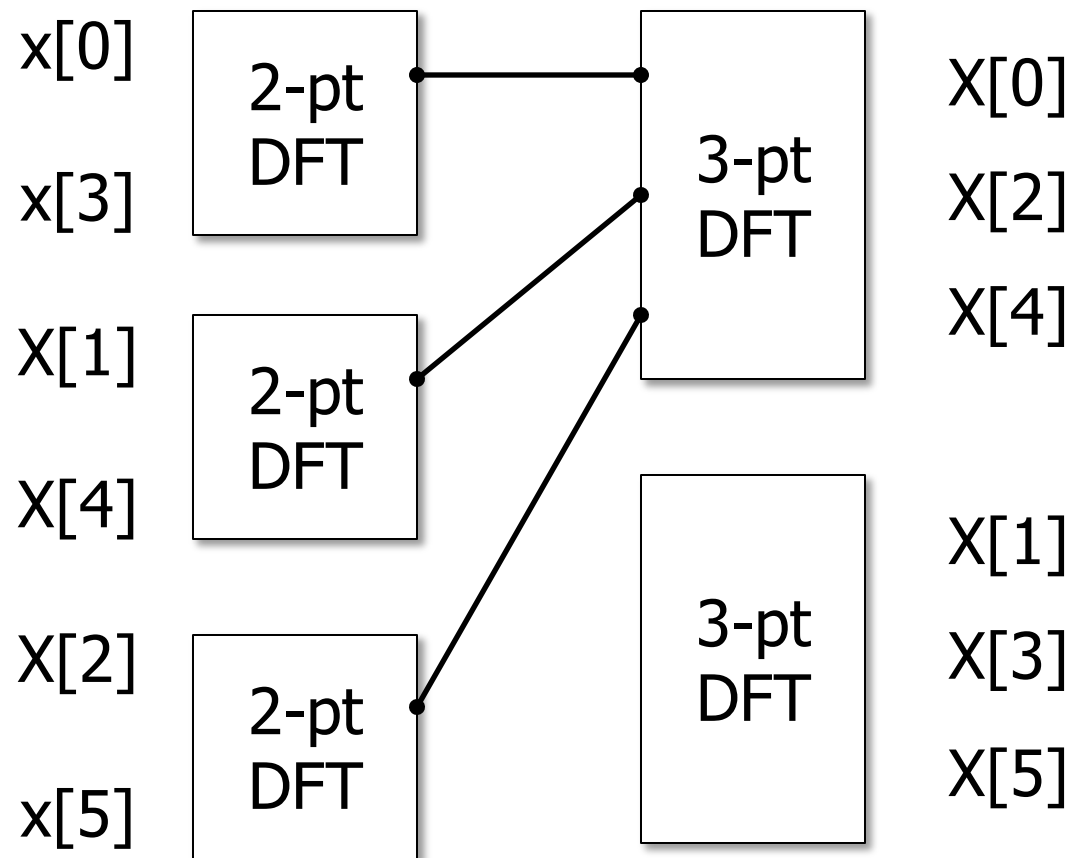
Example: Non-Power-of-2 FFTs



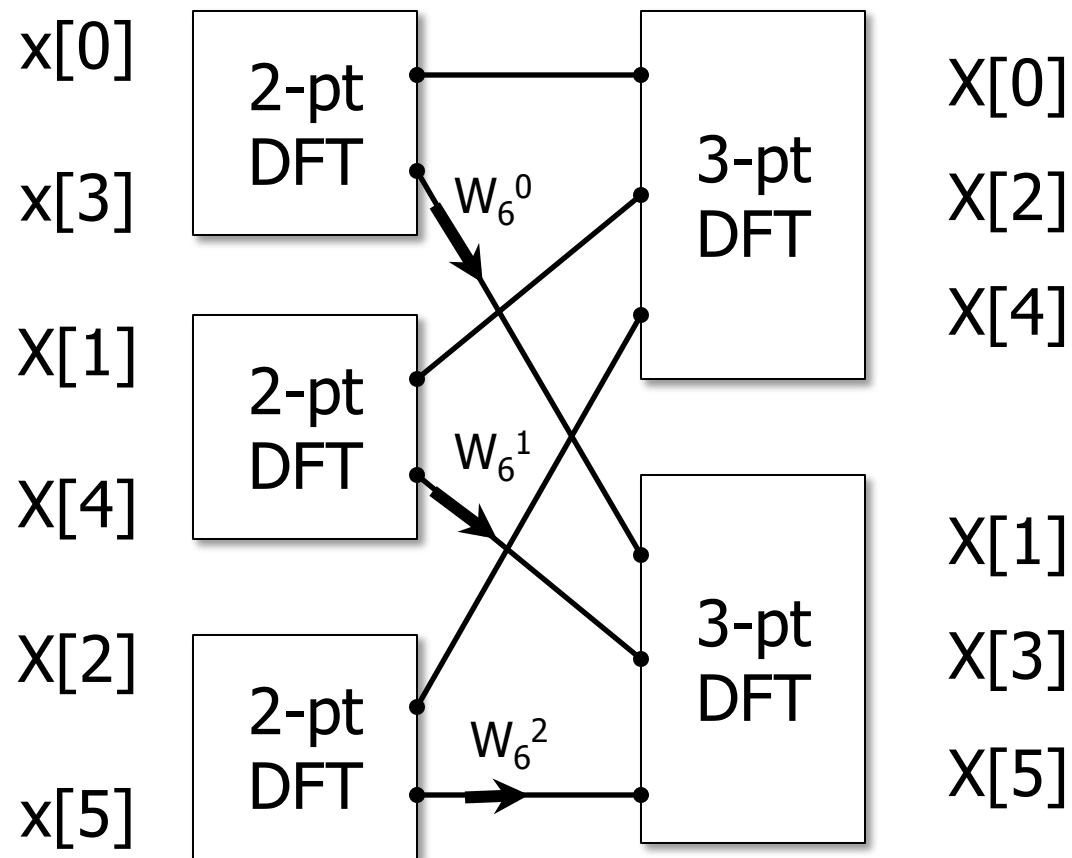
Example: Non-Power-of-2 FFTs



Example: Non-Power-of-2 FFTs



Example: Non-Power-of-2 FFTs





Example 1:

A long *periodic* sequence x of period $N = 2^r$ (r is an integer) is to be convolved with a finite-length sequence h of length K .

(a) *Show* that the output y of this convolution (filtering) is *periodic*; what is its period?



Example 1:

A long *periodic* sequence x of period $N = 2^r$ (r is an integer) is to be convolved with a finite-length sequence h of length K .

- (a) *Show* that the output y of this convolution (filtering) is *periodic*; what is its period?
- (b) Let $K = mN$ where m is an integer; N is large. How would you implement this convolution *efficiently*? Explain your analysis clearly.

Compare the *total number of multiplications* required in your scheme to that in the direct implementation of FIR filtering. (Consider the case $r = 10$, $m = 10$).



Example 2:

A sequence $x = \{x[n], n = 0, 1, \dots, N-1\}$ is given; let $X(e^{j\omega})$ be its DTFT.

- (a) Suppose $N = 10$. You want to evaluate both $X(e^{j2\pi 7/12})$ and $X(e^{j2\pi 3/8})$. The only computation you can perform is one DFT, on any one input sequence of your choice. Can you find the desired DTFT values? (*Show your analysis and explain clearly.*)

Example 2:

A sequence $x = \{x[n], n = 0, 1, \dots, N-1\}$ is given; let $X(e^{j\omega})$ be its DTFT.

- (a) Suppose $N = 10$. You want to evaluate both $X(e^{j2\pi 7/12})$ and $X(e^{j2\pi 3/8})$. The only computation you can perform is one DFT, on any one input sequence of your choice. Can you find the desired DTFT values? (*Show your analysis and explain clearly.*)
- (b) Suppose N is large. You want to obtain $X(e^{j\omega})$ at the following $2M$ frequencies:

$$\omega = \frac{2\pi}{M}m, \quad m = 0, 1, \dots, M-1 \quad \text{and} \quad \omega = \frac{2\pi}{M}m + \frac{2\pi}{N}, \quad m = 0, 1, \dots, M-1.$$

Here $M = 2^\mu \ll N = 2^\nu$

A standard radix-2 FFT algorithm is available. You may execute the FFT algorithm *once or more than once*, and *multiplications* and *additions* outside of the FFT are *allowed*, if necessary.

- (i) You want to get the $2M$ DTFT values with as few *total multiplications* as possible (*including those in the FFT*). Give explicitly the best method you can find for this, with an estimate of the *total number of multiplications* needed in terms of M and N .

Example 2:

A sequence $x = \{x[n], n = 0, 1, \dots, N-1\}$ is given; let $X(e^{j\omega})$ be its DTFT.

- (a) Suppose $N = 10$. You want to evaluate both $X(e^{j2\pi 7/12})$ and $X(e^{j2\pi 3/8})$. The only computation you can perform is one DFT, on any one input sequence of your choice. Can you find the desired DTFT values? (*Show your analysis and explain clearly.*)
- (b) Suppose N is large. You want to obtain $X(e^{j\omega})$ at the following $2M$ frequencies:

$$\omega = \frac{2\pi}{M}m, \quad m = 0, 1, \dots, M-1 \quad \text{and} \quad \omega = \frac{2\pi}{M}m + \frac{2\pi}{N}, \quad m = 0, 1, \dots, M-1.$$

Here $M = 2^\mu \ll N = 2^\nu$

A standard radix-2 FFT algorithm is available. You may execute the FFT algorithm *once or more than once*, and *multiplications* and *additions* outside of the FFT are *allowed*, if necessary.

- (i) You want to get the $2M$ DTFT values with as few *total multiplications* as possible (*including those in the FFT*). Give explicitly the best method you can find for this, with an estimate of the *total number of multiplications* needed in terms of M and N .
- (ii) Does your result change if extra multiplications outside of FFTs are *not* allowed?

Chirp Transfer Algorithm

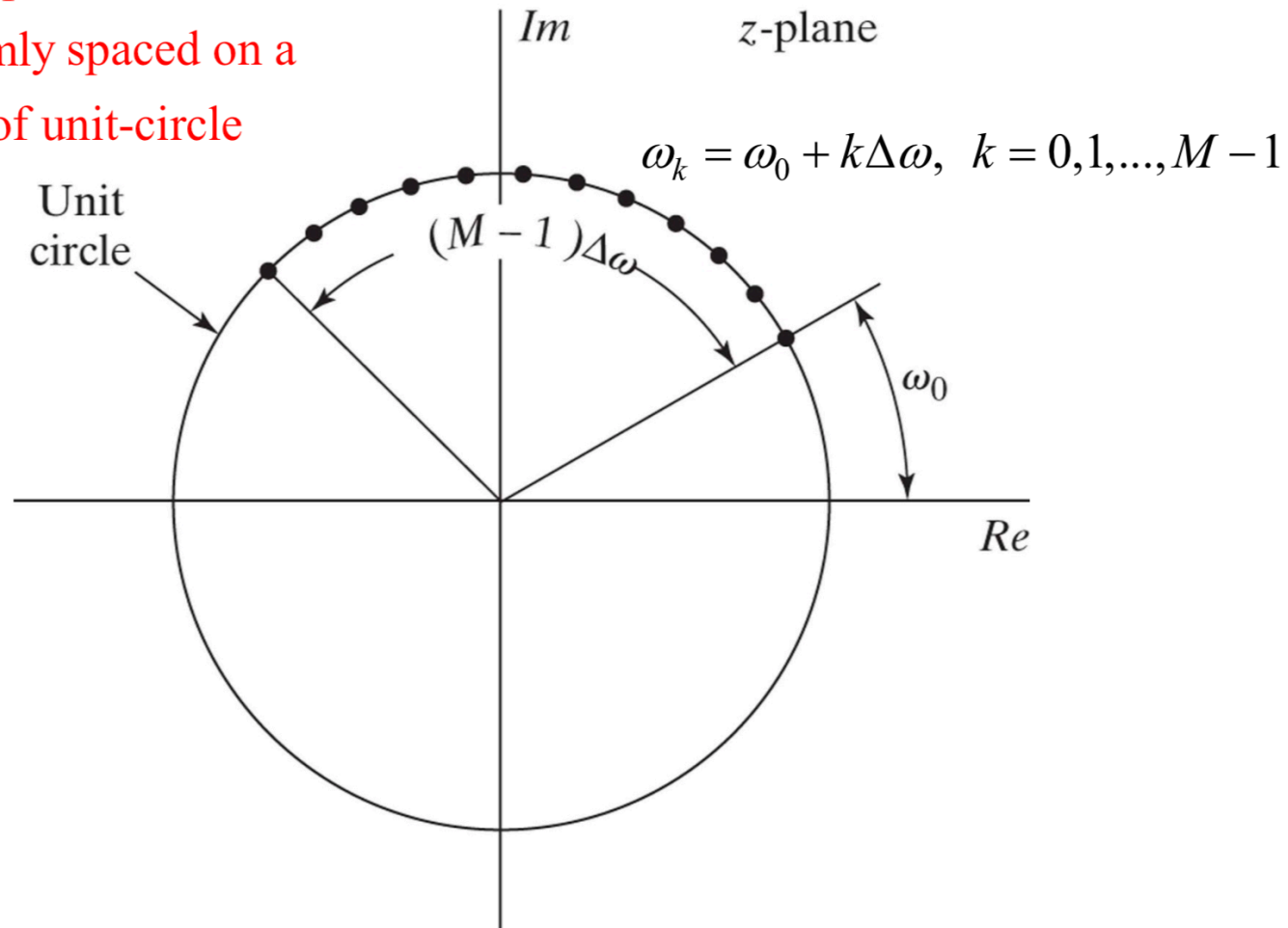


Chirp Transform Algorithm

- ❑ Uses convolution to evaluate the DFT
- ❑ This algorithm is not optimal in minimizing any measure of computational complexity, but it has been useful in a variety of applications, particularly when implemented in technologies that are well suited to doing convolution with a fixed, pre-specified impulse response.
- ❑ The CTA is also more flexible than the FFT, since it can be used to compute *any* set of equally spaced samples of the Fourier transform on the unit circle.

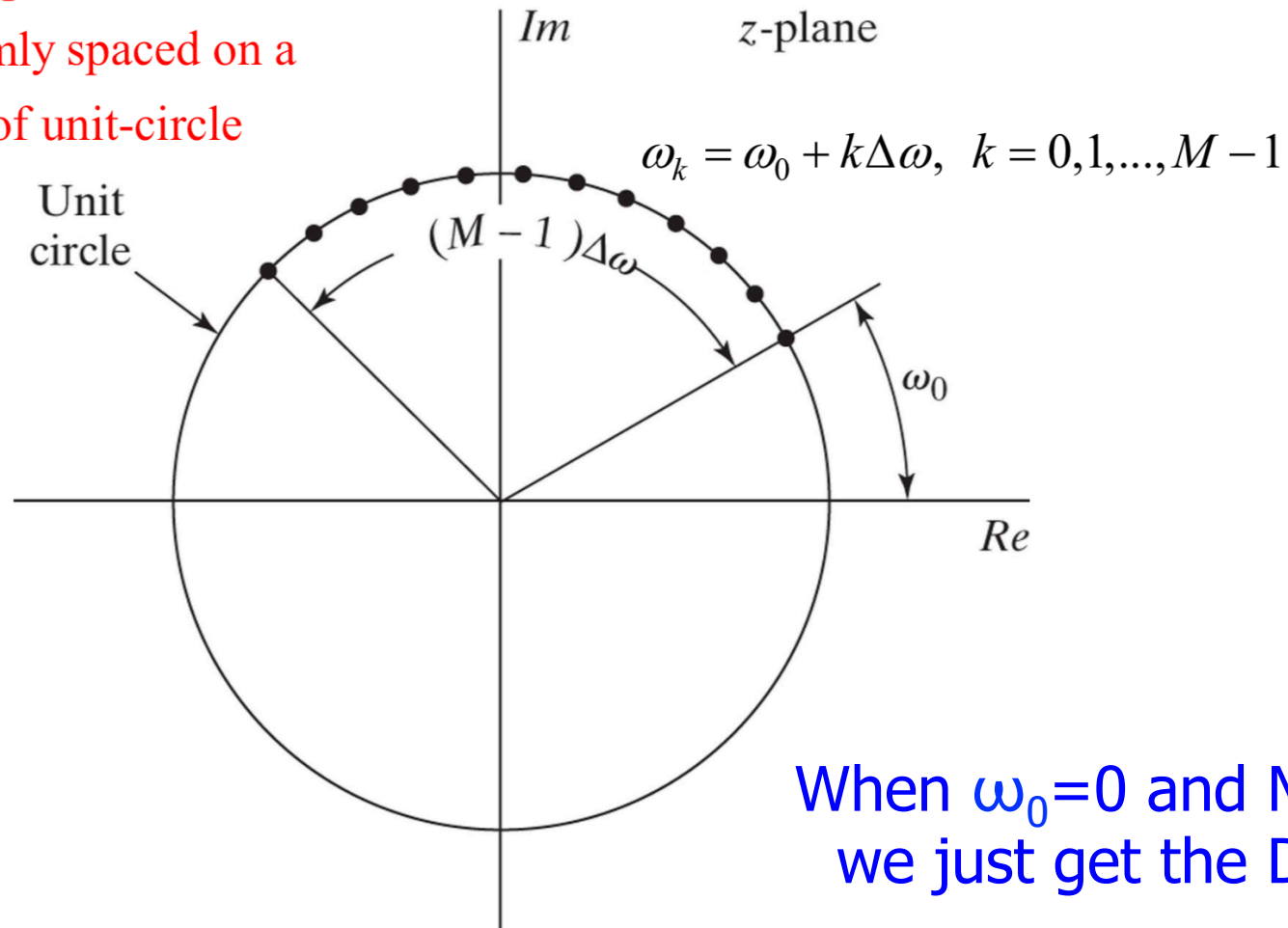
Chirp Transform Algorithm

For M points of DTFT
uniformly spaced on a
sector of unit-circle



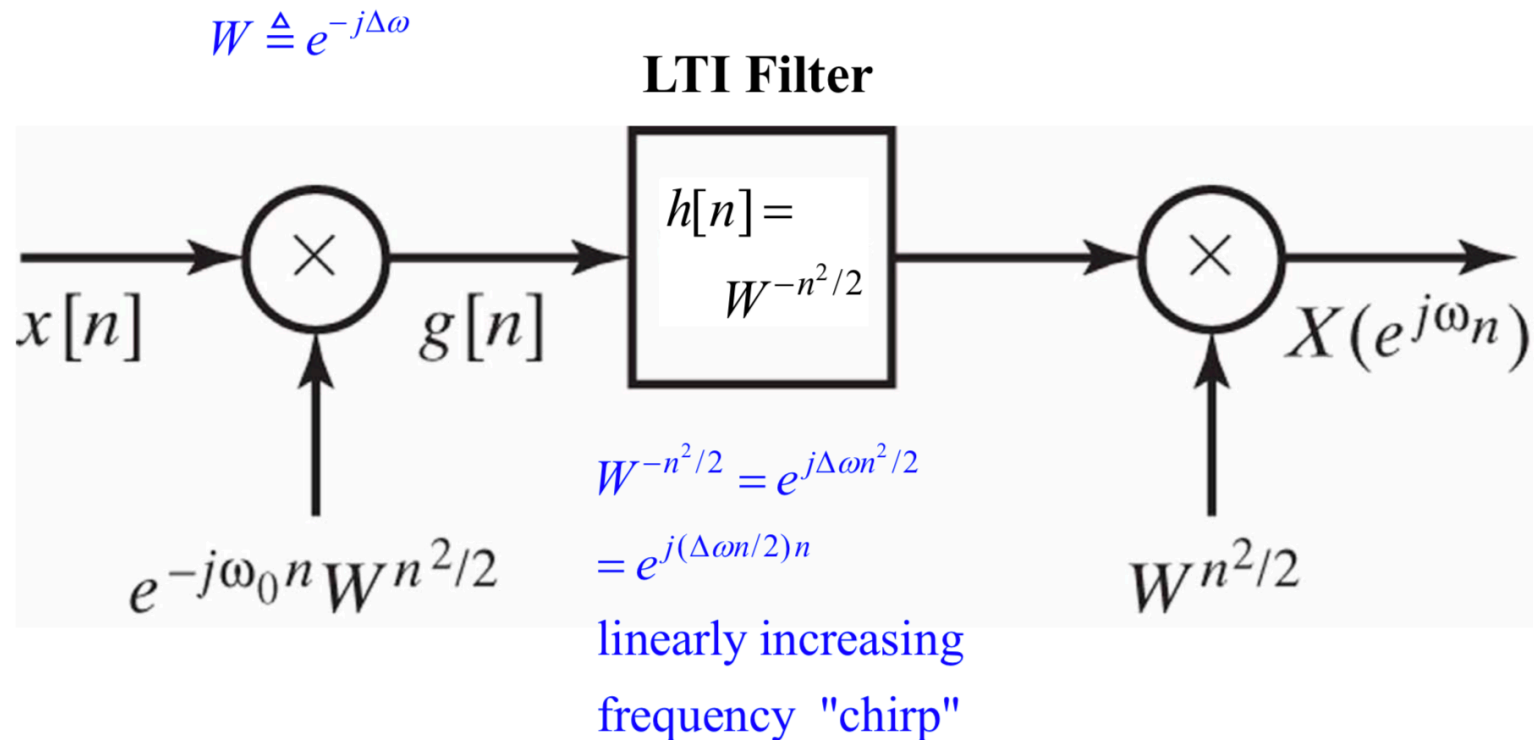
Chirp Transform Algorithm

For M points of DTFT
uniformly spaced on a
sector of unit-circle



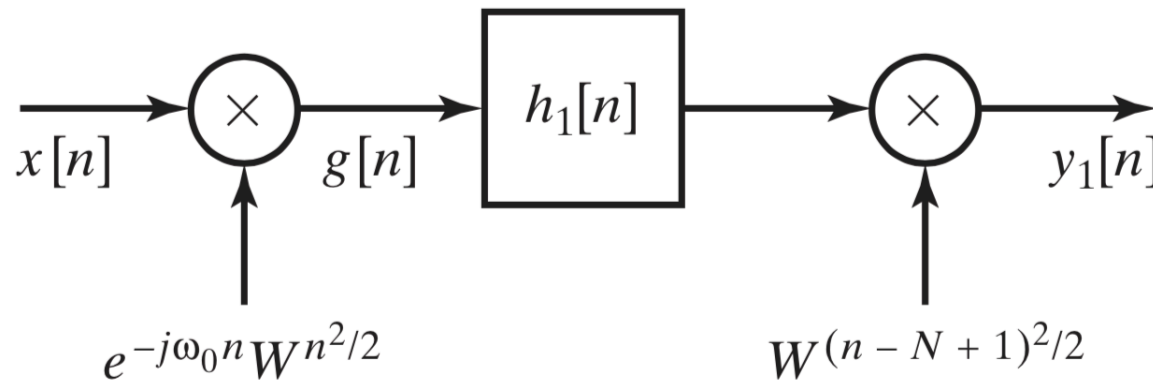
When $\omega_0=0$ and $M=N$,
we just get the DFT

Chirp Transform Algorithm



Causal FIR CTA

$$h_1[n] = \begin{cases} W^{-(n-N+1)^2/2}, & n = 0, 1, \dots, M + N - 2, \\ 0, & \text{otherwise.} \end{cases}$$

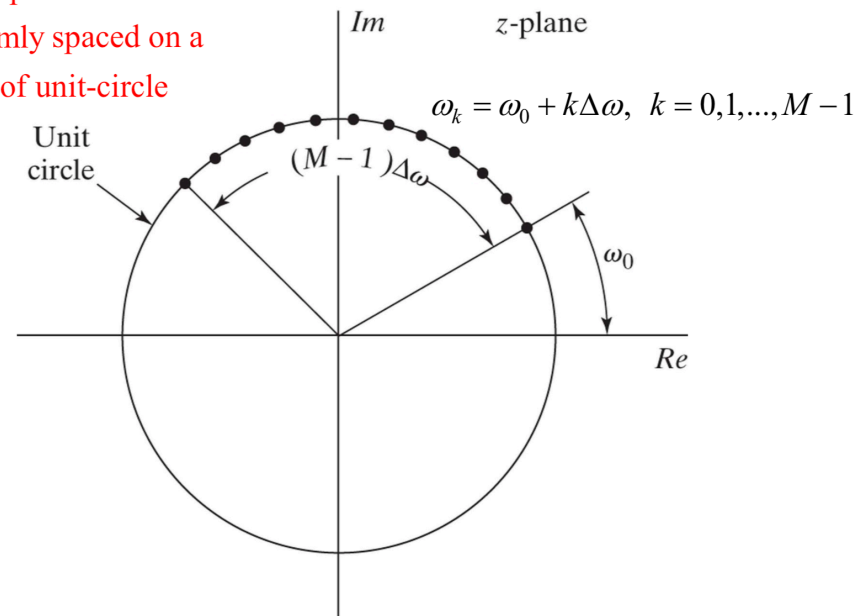


$$X(e^{j\omega_n}) = y_1[n + N - 1], \quad n = 0, 1, \dots, M - 1.$$

Example: Chirp Transform Parameters

- We have a finite-length sequence $x[n]$ that is nonzero only on the interval $n = 0, \dots, 25$, (Length $N=26$) and we wish to compute 16 samples of the DTFT $X(e^{j\omega})$ at the frequencies $\omega_k = 2\pi/27 + 2\pi k/1024$ for $k = 0, \dots, 15$.

For M points of DTFT
uniformly spaced on a
sector of unit-circle



Circular Convolution

Linear Convolution with aliasing!



Circular Convolution

□ Circular Convolution:

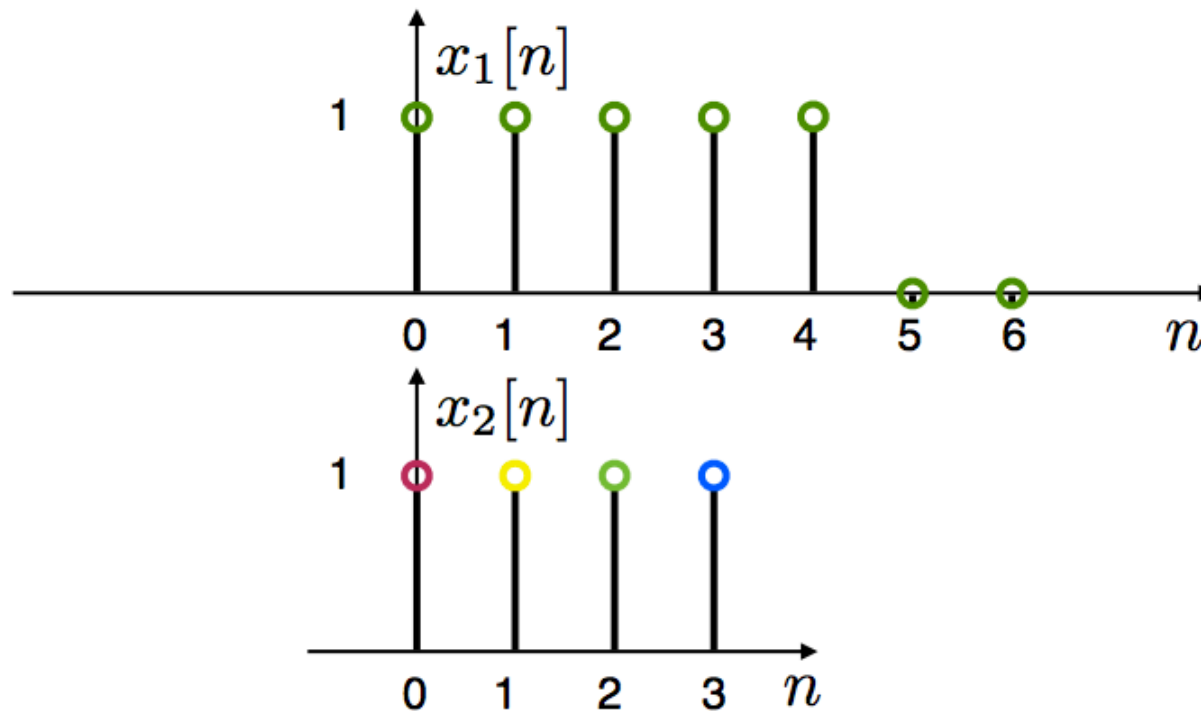
$$x_1[n] \textcircled{N} x_2[n] \triangleq \sum_{m=0}^{N-1} x_1[m] x_2[((n - m))_N]$$

For two signals of length N

Note: Circular convolution is commutative

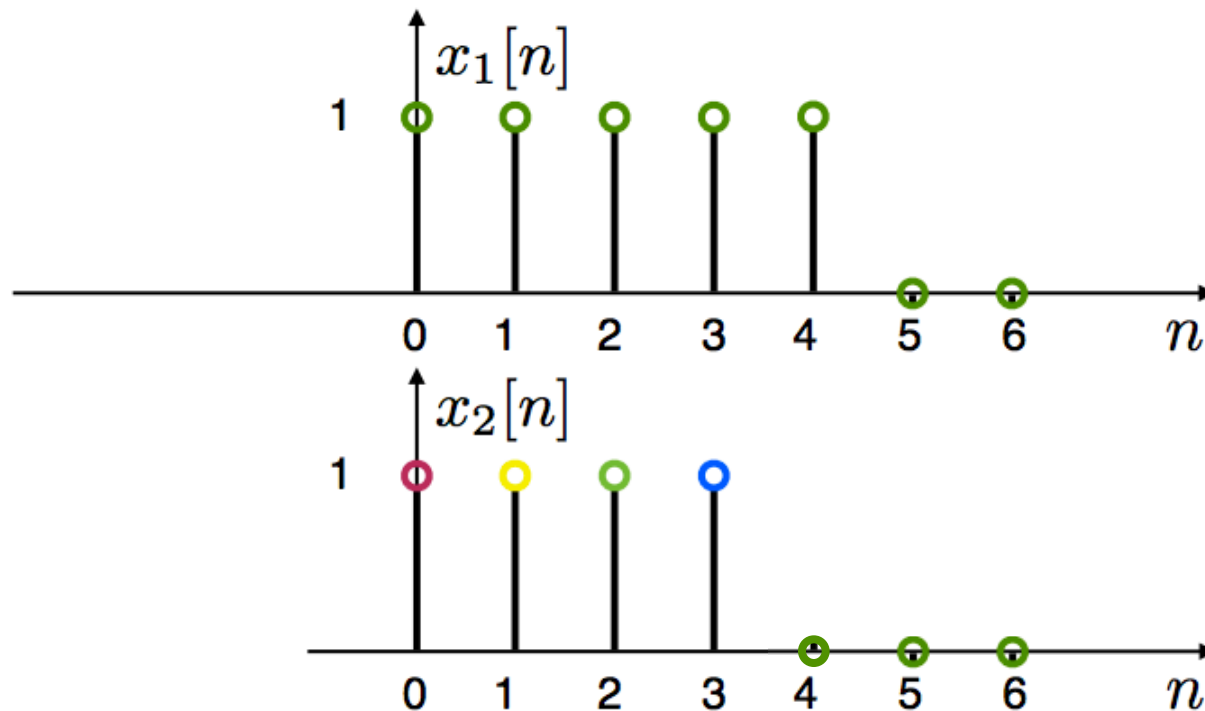
$$x_2[n] \textcircled{N} x_1[n] = x_1[n] \textcircled{N} x_2[n]$$

Compute Circular Convolution Sum



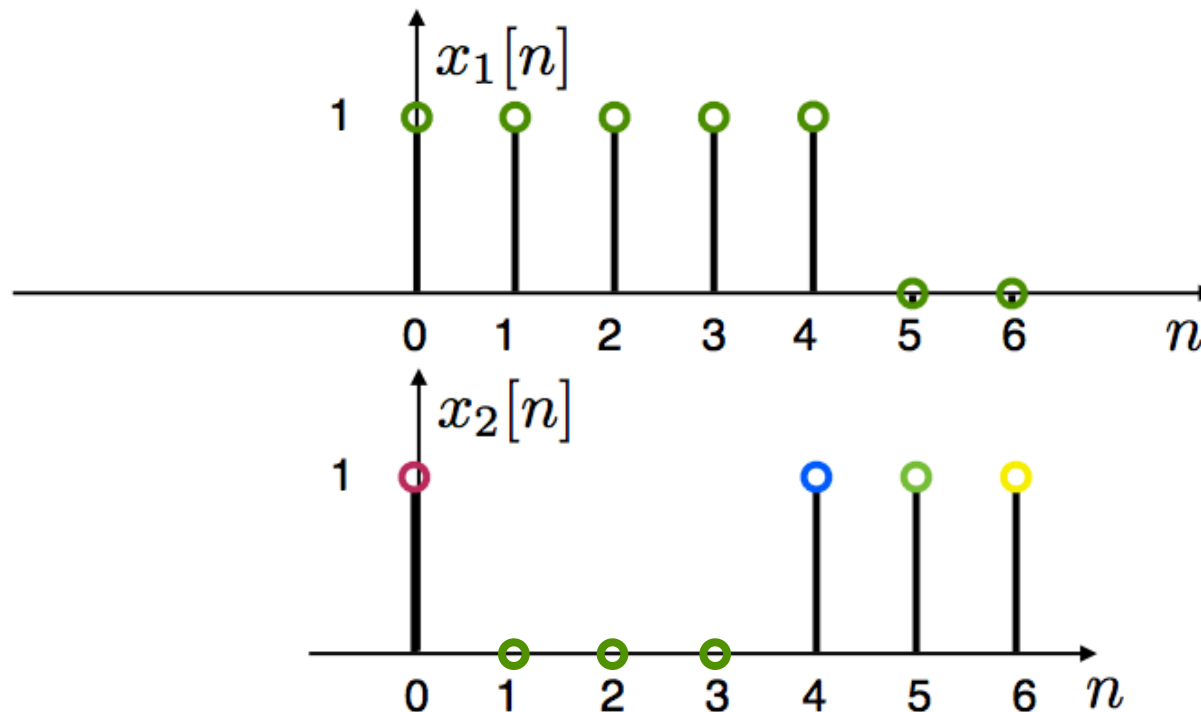
$$x_1[n] \circledN x_2[n] \triangleq \sum_{m=0}^{N-1} x_1[m] x_2[((n - m))_N]$$

Compute Circular Convolution Sum



$$x_1[n] \circledcirc x_2[n] \triangleq \sum_{m=0}^{N-1} x_1[m] x_2[((n - m))_N]$$

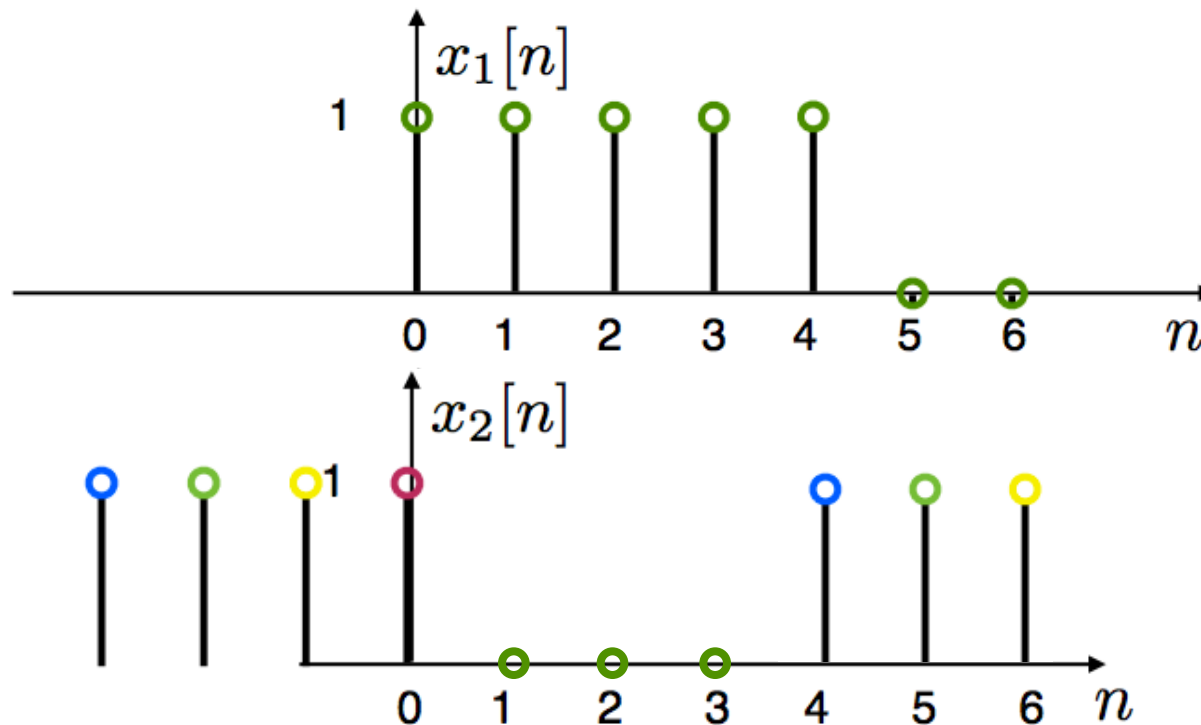
Compute Circular Convolution Sum



$$x_1[n] \circledcirc x_2[n] \triangleq \sum_{m=0}^{N-1} x_1[m] x_2[((n - m))_N]$$

Compute Circular Convolution Sum

$$y[0]=2$$

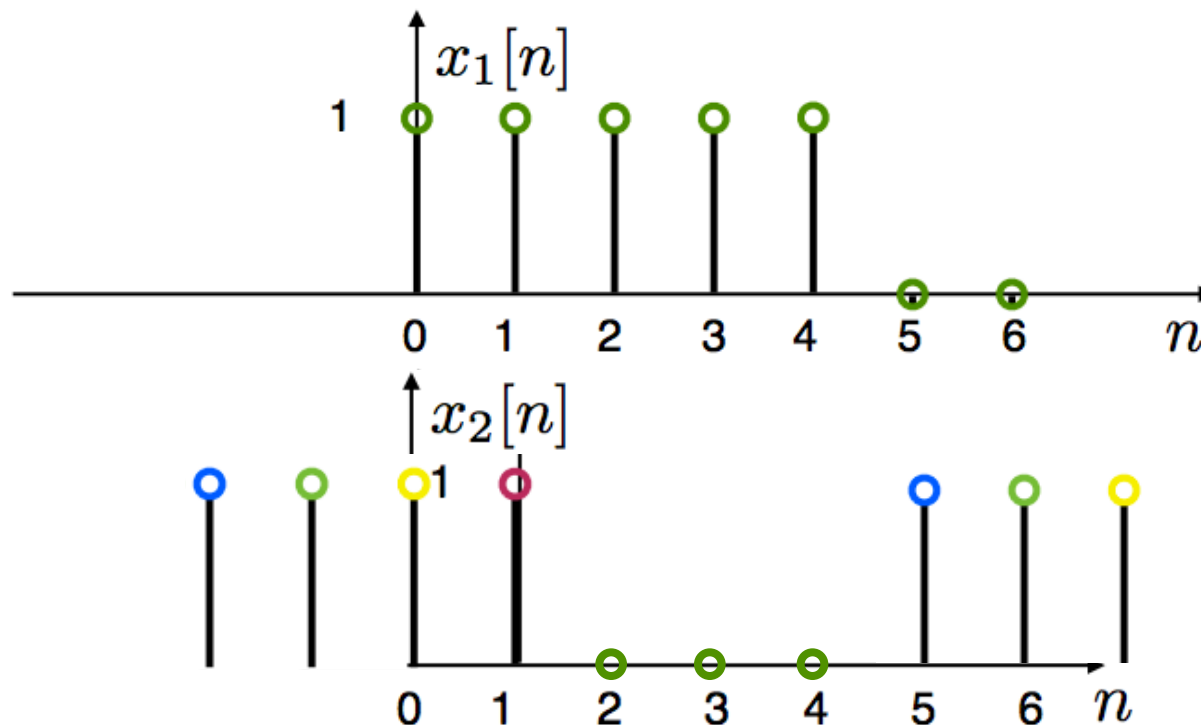


$$x_1[n] \circledN x_2[n] \triangleq \sum_{m=0}^{N-1} x_1[m] x_2[((n - m))_N]$$

Compute Circular Convolution Sum

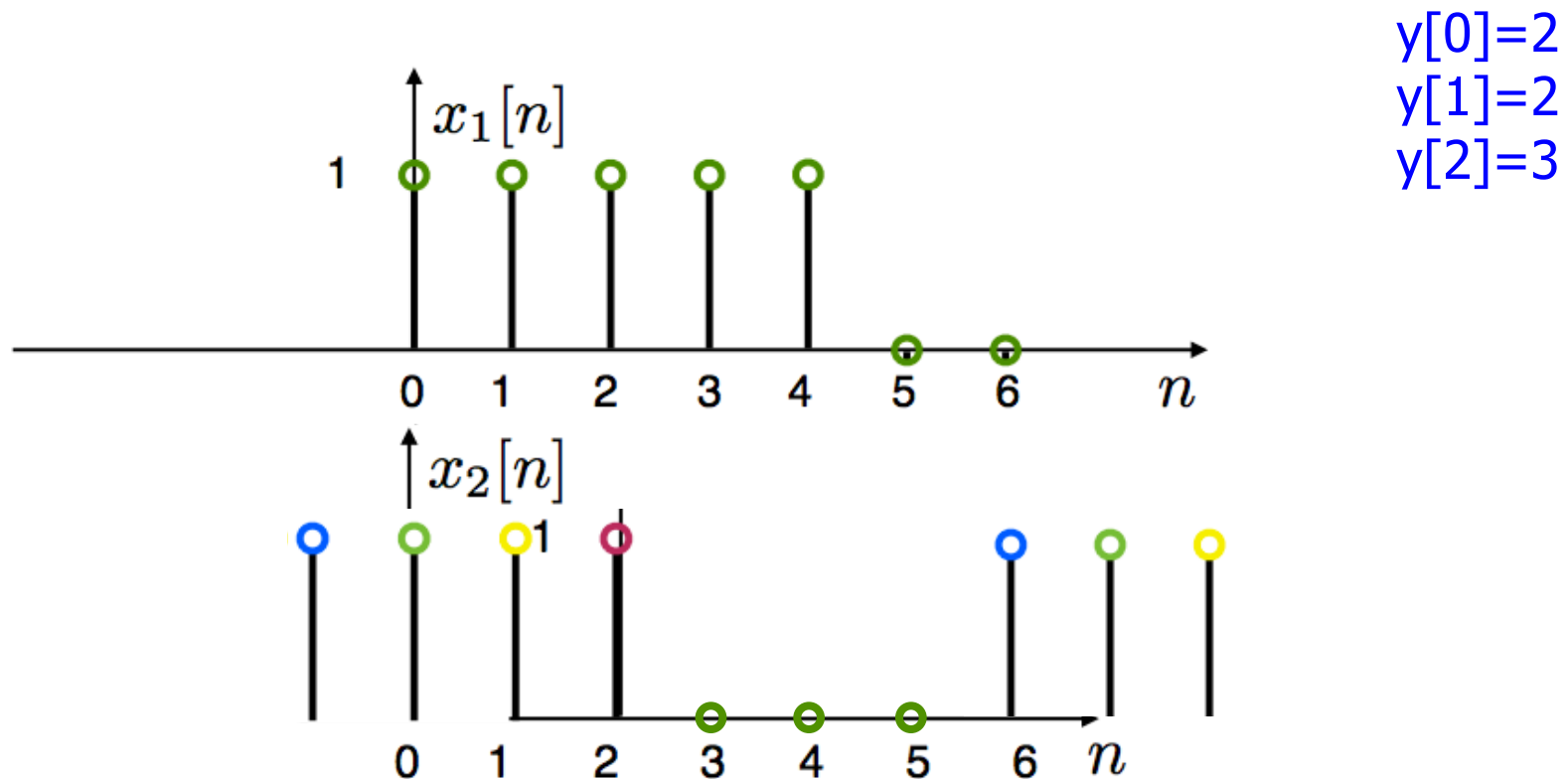
$$y[0]=2$$

$$y[1]=2$$



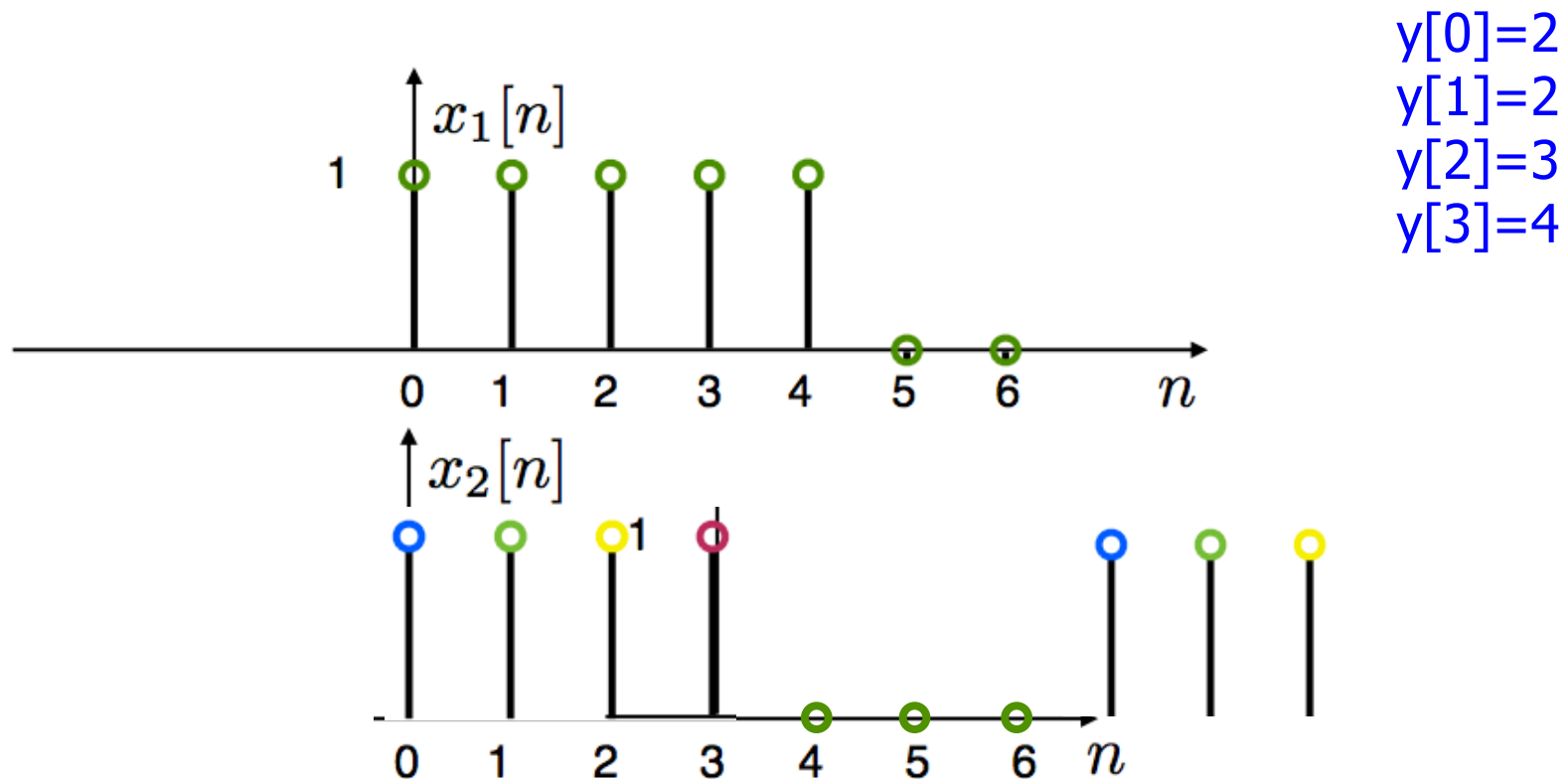
$$x_1[n] \circledast x_2[n] \triangleq \sum_{m=0}^{N-1} x_1[m] x_2[((n - m))_N]$$

Compute Circular Convolution Sum



$$x_1[n] \circledN x_2[n] \triangleq \sum_{m=0}^{N-1} x_1[m] x_2[((n - m))_N]$$

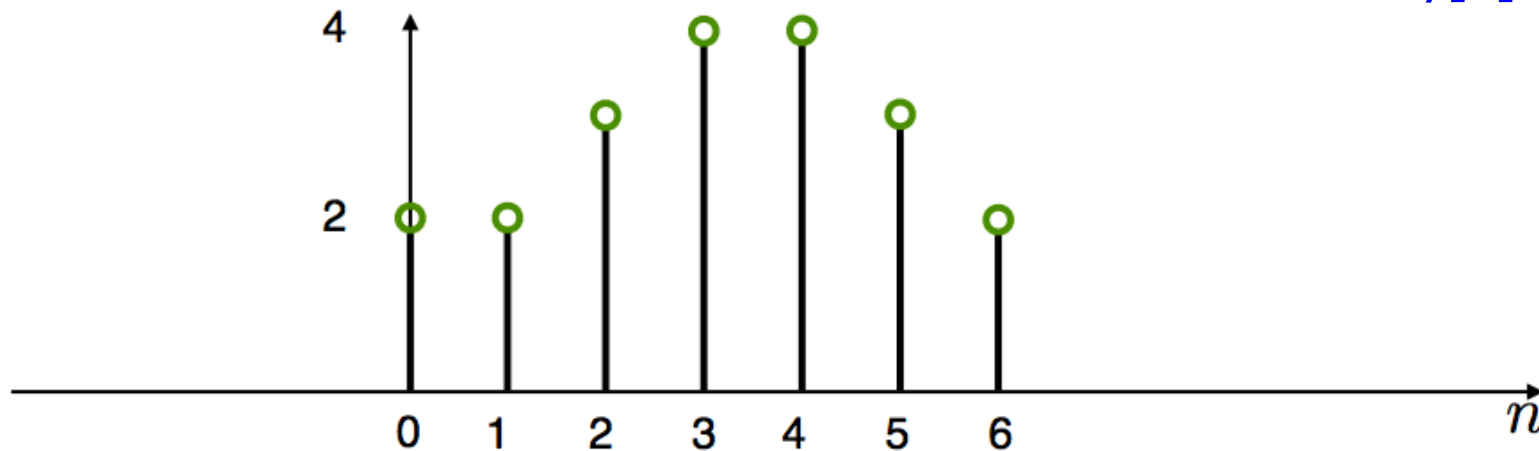
Compute Circular Convolution Sum



$$x_1[n] \circledN x_2[n] \triangleq \sum_{m=0}^{N-1} x_1[m] x_2[((n - m))_N]$$

Result

$y[0]=2$
 $y[1]=2$
 $y[2]=3$
 $y[3]=4$



$$x_1[n] \circledast x_2[n] \triangleq \sum_{m=0}^{N-1} x_1[m]x_2[((n-m))_N]$$



Linear Convolution

- We start with two non-periodic sequences:

$$\begin{aligned}x[n] & 0 \leq n \leq L - 1 \\h[n] & 0 \leq n \leq P - 1\end{aligned}$$

- E.g. $x[n]$ is a signal and $h[n]$ a filter's impulse response
- We want to compute the linear convolution:

$$y[n] = x[n] * h[n] = \sum_{m=0}^{L-1} x[m]h[n - m]$$

- $y[n]$ is nonzero for $0 \leq n \leq L+P-2$ with length $M=L+P-1$

Requires LP multiplications



Linear Convolution via Circular Convolution

- ❑ Zero-pad $x[n]$ by $P-1$ zeros

$$x_{zp}[n] = \begin{cases} x[n] & 0 \leq n \leq L-1 \\ 0 & L \leq n \leq L+P-2 \end{cases}$$

- ❑ Zero-pad $h[n]$ by $L-1$ zeros

$$h_{zp}[n] = \begin{cases} h[n] & 0 \leq n \leq P-1 \\ 0 & P \leq n \leq L+P-2 \end{cases}$$

- ❑ Now, both sequences are length $M=L+P-1$

Circular Conv. via Linear Conv. w/ Aliasing

- If the DTFT $X(e^{j\omega})$ of a sequence $x[n]$ is sampled at N frequencies $\omega_k = 2\pi k/N$, then the resulting sequence $X[k]$ corresponds to the periodic sequence

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

- And $X[k] = \begin{cases} X(e^{j(2\pi k/N)}), & 0 \leq k \leq N-1, \\ 0, & \text{otherwise,} \end{cases}$ is the DFT of one period given as

$$x_p[n] = \begin{cases} \tilde{x}[n], & 0 \leq n \leq N-1, \\ 0, & \text{otherwise.} \end{cases}$$



Circular Conv. via Linear Conv. w/ Aliasing

$$x_p[n] = \begin{cases} \tilde{x}[n], & 0 \leq n \leq N - 1, \\ 0, & \text{otherwise.} \end{cases}$$

- If $x[n]$ has length less than or equal to N , then $x_p[n] = x[n]$
- However if the length of $x[n]$ is greater than N , this might not be true and we get aliasing in time
 - N -point convolution results in N -point sequence



Circular Conv. via Linear Conv. w/ Aliasing

- ❑ Given two N-point sequences ($x_1[n]$ and $x_2[n]$) and their N-point DFTs ($X_1[k]$ and $X_2[k]$)
- ❑ The N-point DFT of $x_3[n] = x_1[n] * x_2[n]$ is defined as

$$X_3[k] = X_3(e^{j(2\pi k/N)})$$

Circular Conv. via Linear Conv. w/ Aliasing

- Given two N -point sequences ($x_1[n]$ and $x_2[n]$) and their N -point DFTs ($X_1[k]$ and $X_2[k]$)
- The N -point DFT of $x_3[n]=x_1[n]*x_2[n]$ is defined as

$$X_3[k] = X_3(e^{j(2\pi k/N)})$$

- And $X_3[k]=X_1[k]X_2[k]$, where the inverse DFT of $X_3[k]$ is

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

Circular Conv. as Linear Conv. w/ Aliasing

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

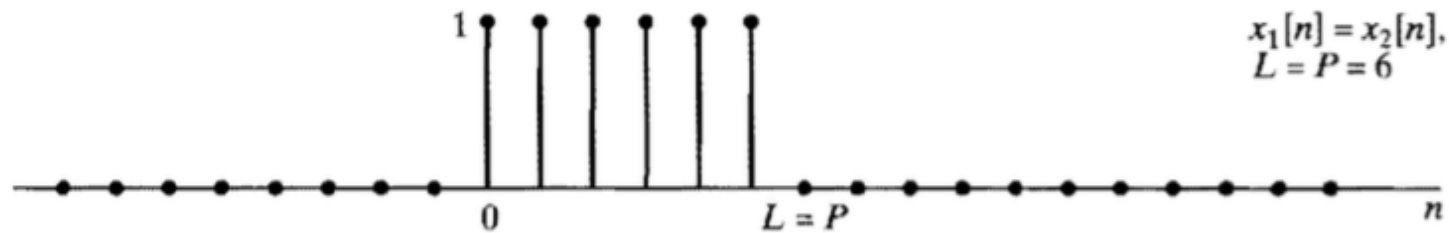
□ Thus

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

□ The N-point circular convolution is the sum of linear convolutions shifted in time by N

Example 1:

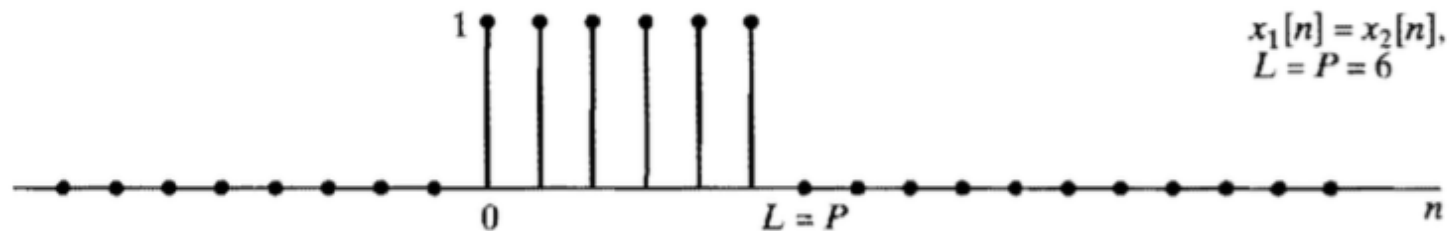
□ Let



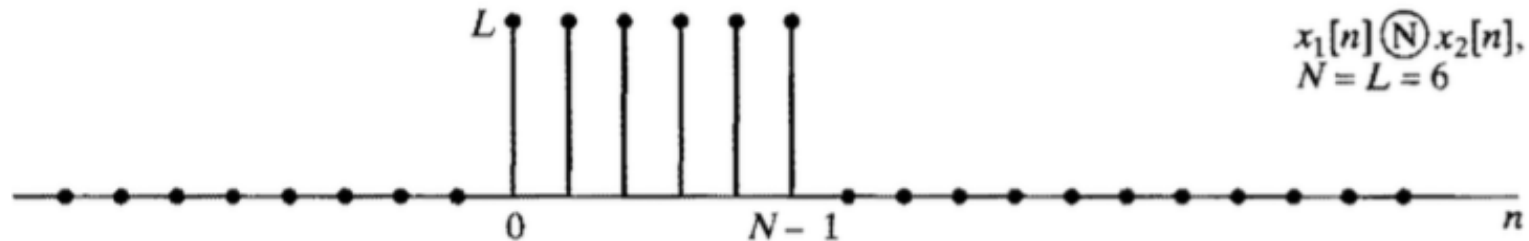
□ The $N=L=6$ -point circular convolution results in

Example 1:

□ Let

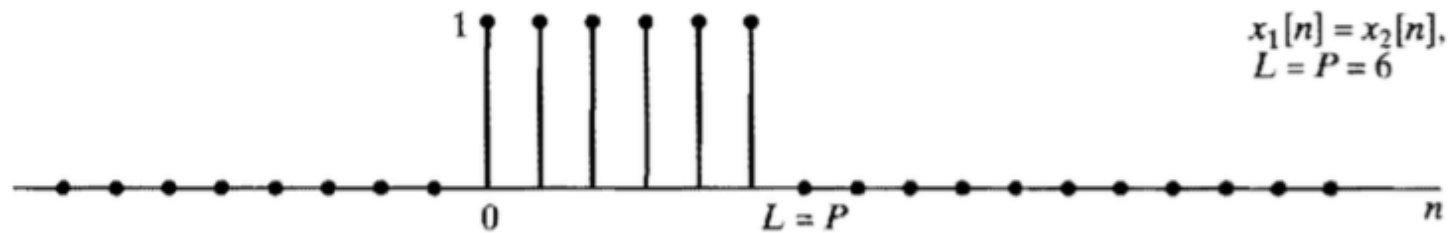


□ The $N=L=6$ -point circular convolution results in



Example 1:

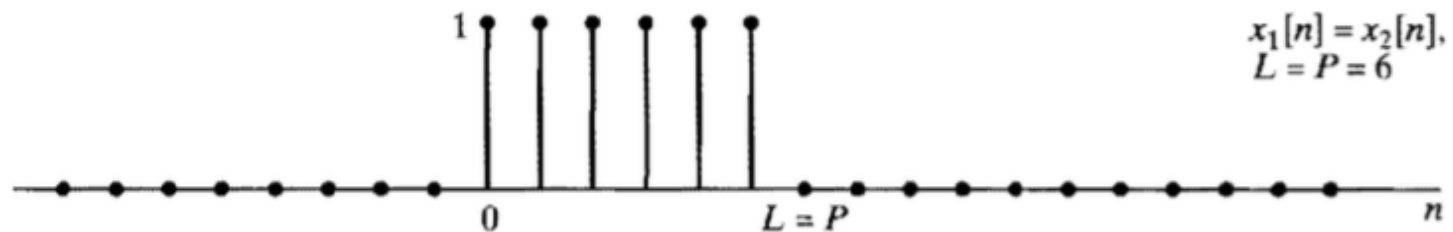
□ Let



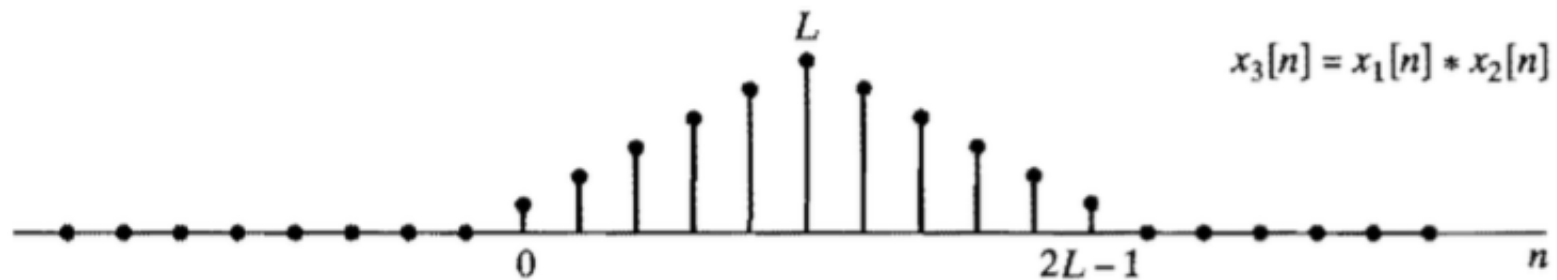
□ The linear convolution results in

Example 1:

□ Let

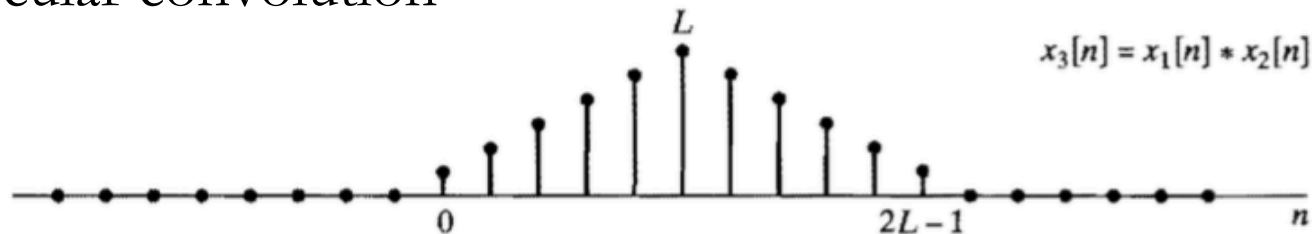


□ The linear convolution results in

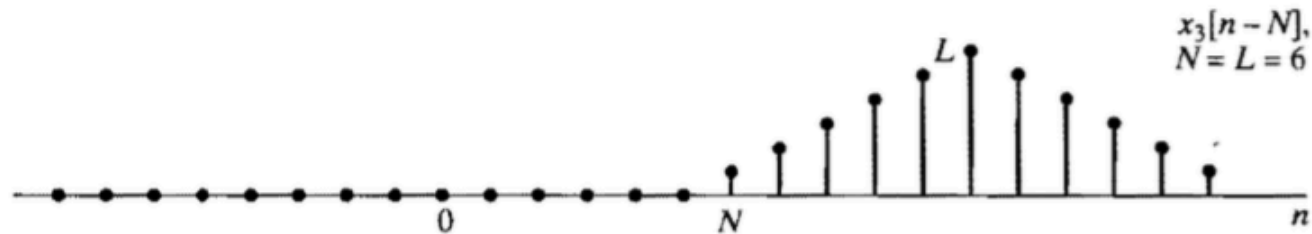


Example 1:

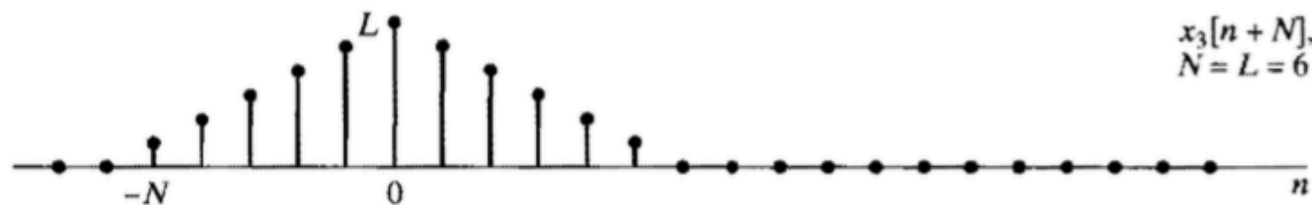
- The sum of N -shifted linear convolutions equals the N -point circular convolution



(b)

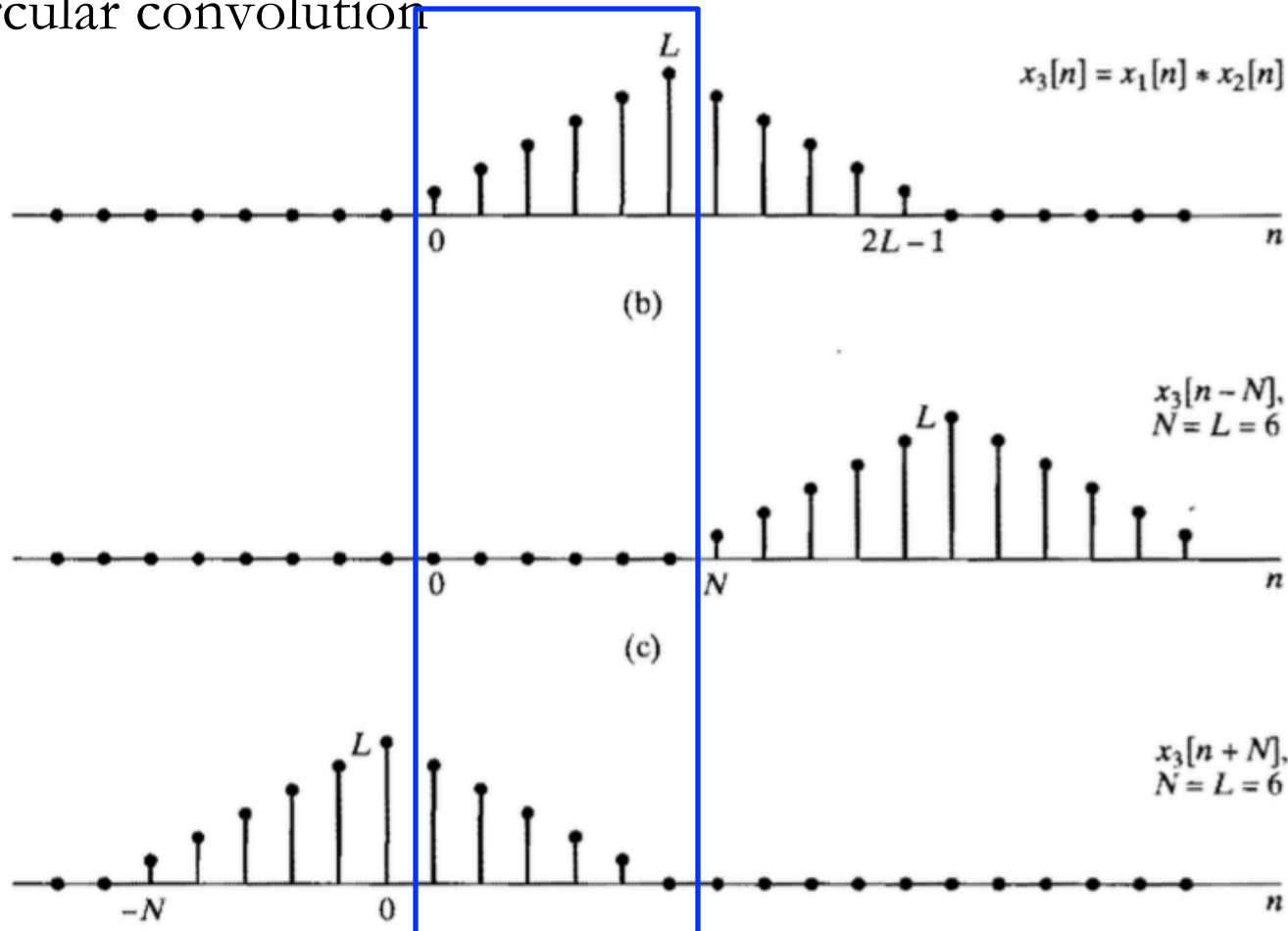


(c)



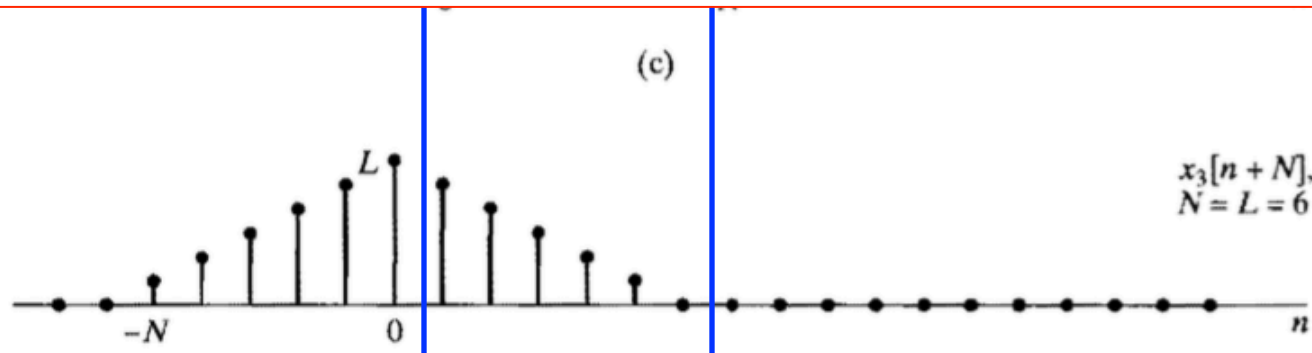
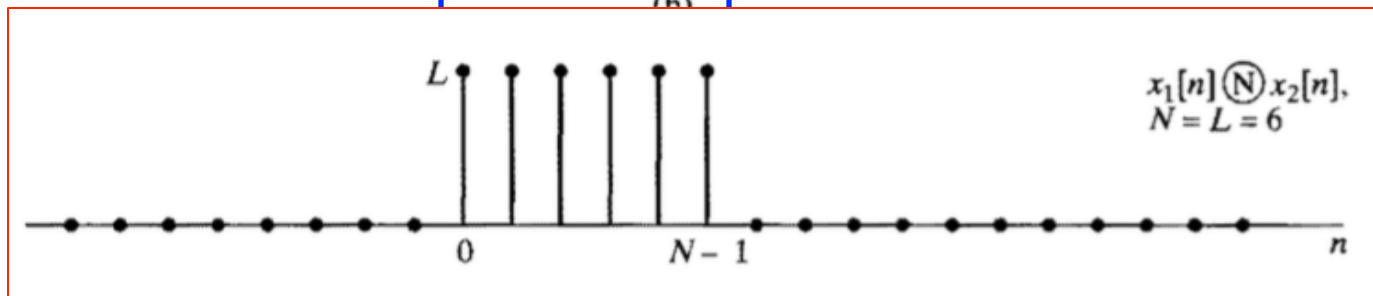
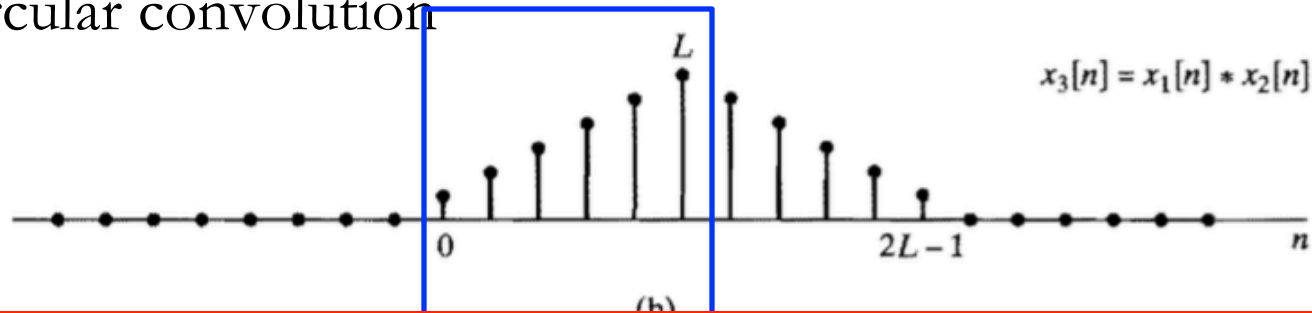
Example 1:

- The sum of N -shifted linear convolutions equals the N -point circular convolution



Example 1:

- The sum of N -shifted linear convolutions equals the N -point circular convolution



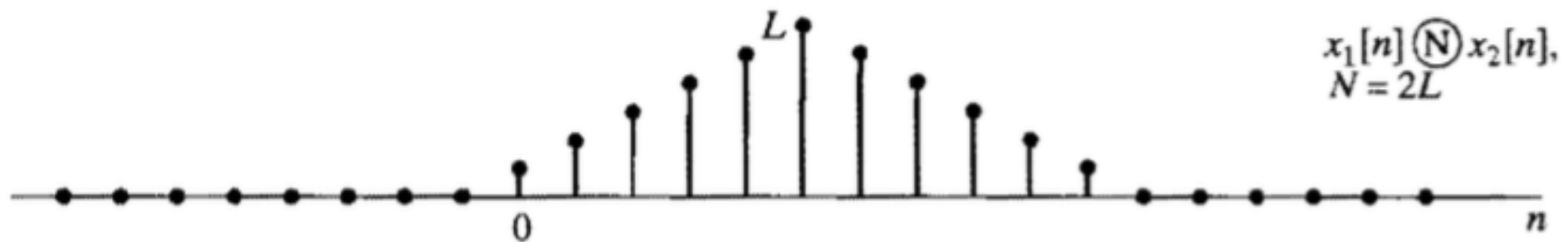


Example 1:

- If I want the circular convolution and linear convolution to be the same, what do I do?

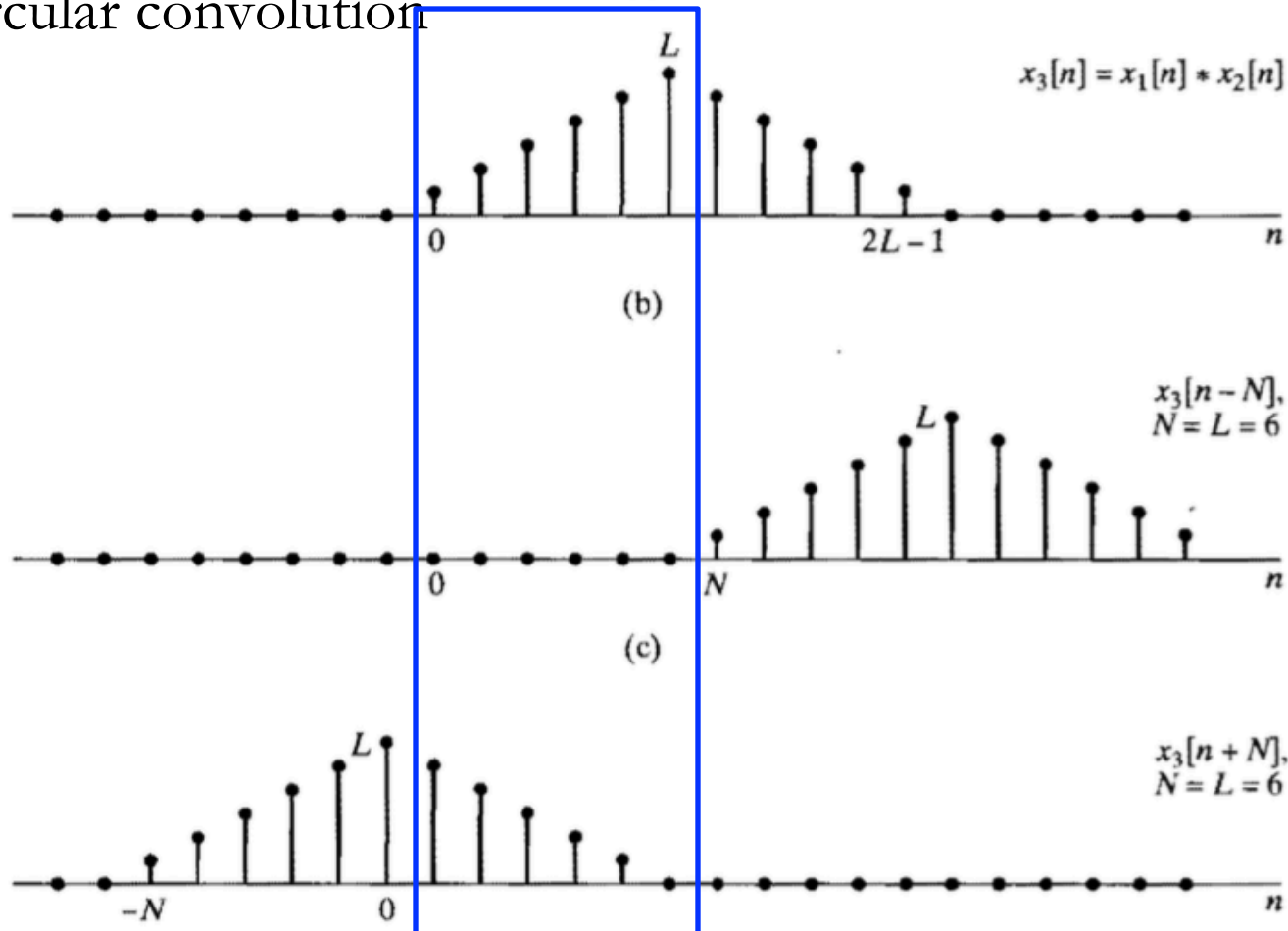
Example 1:

- If I want the circular convolution and linear convolution to be the same, what do I do?
 - Take the $N=2L$ -point circular convolution



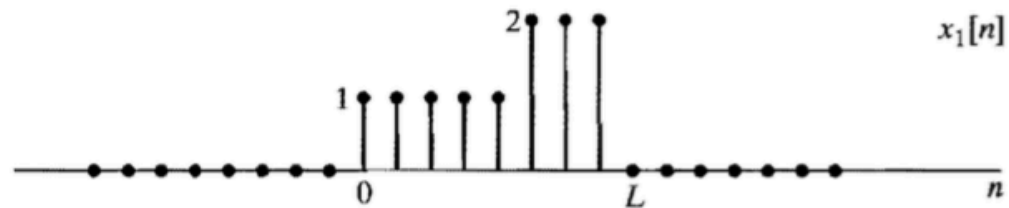
Example 1:

- The sum of N -shifted linear convolutions equals the N -point circular convolution

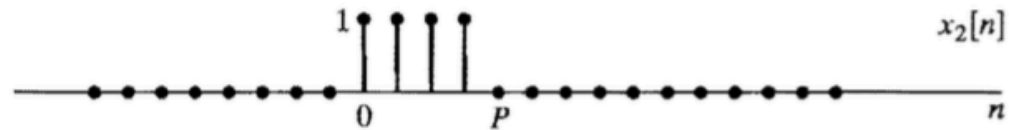


Example 2:

□ Let



(a)



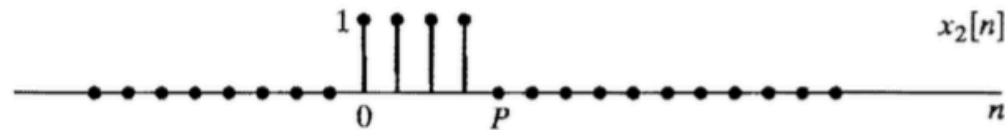
(b)

Example 2:

□ Let

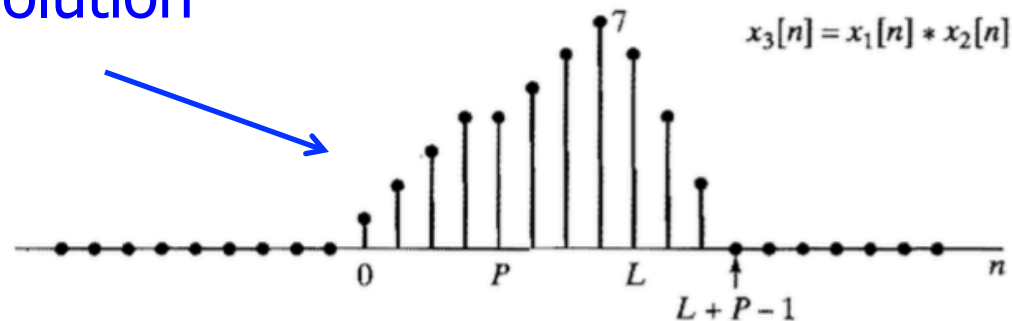


(a)



(b)

Linear convolution



□ What does the L-point circular convolution look like?

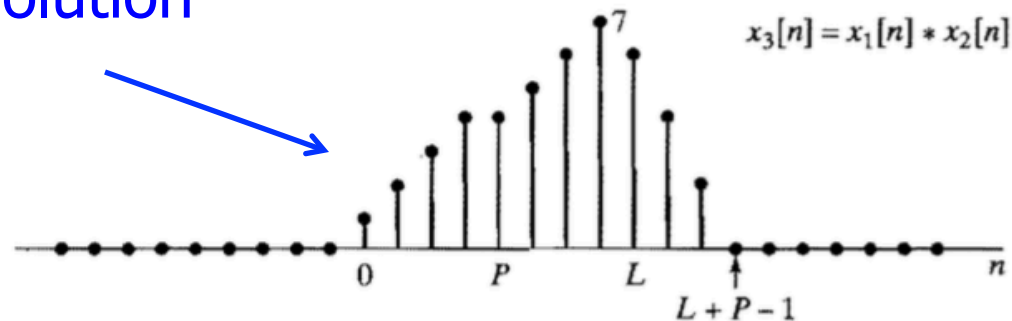
Example 2:

□ Let

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

(b)

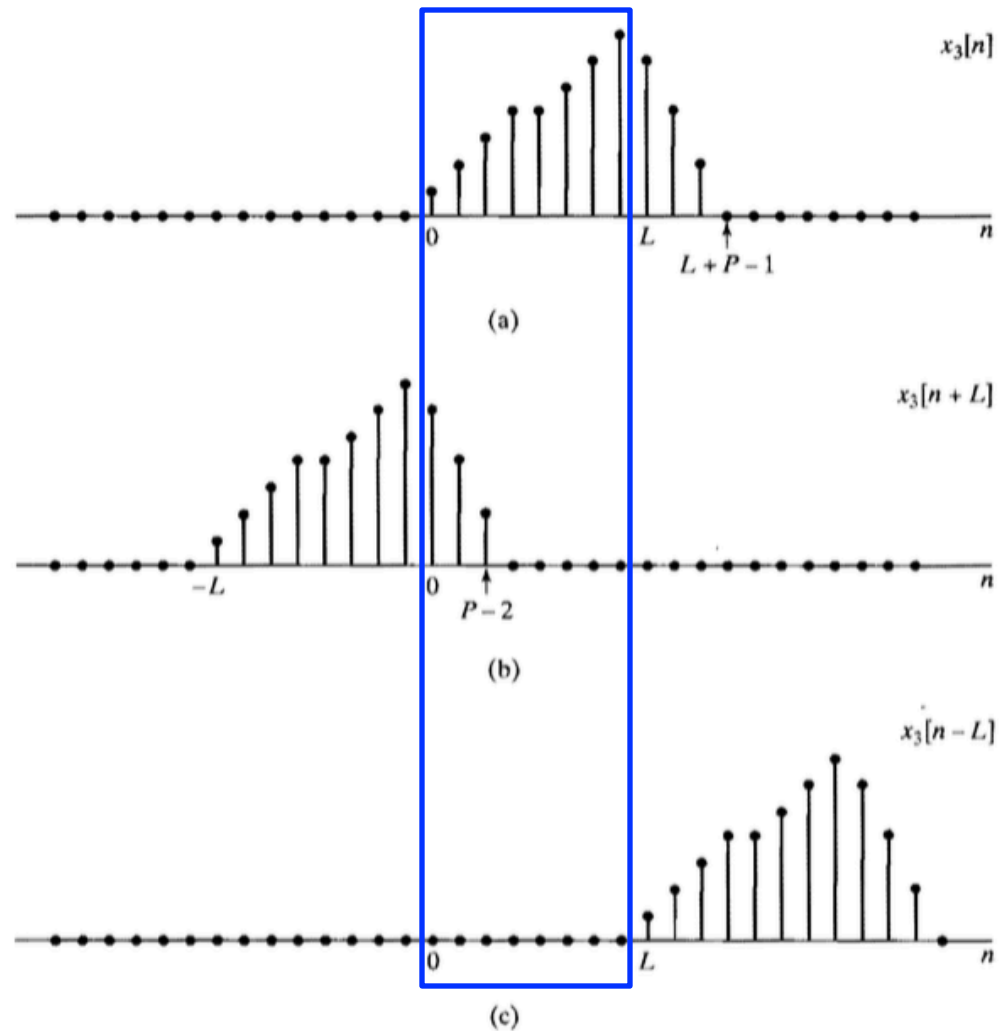
Linear convolution



□ What does the L-point circular convolution look like?

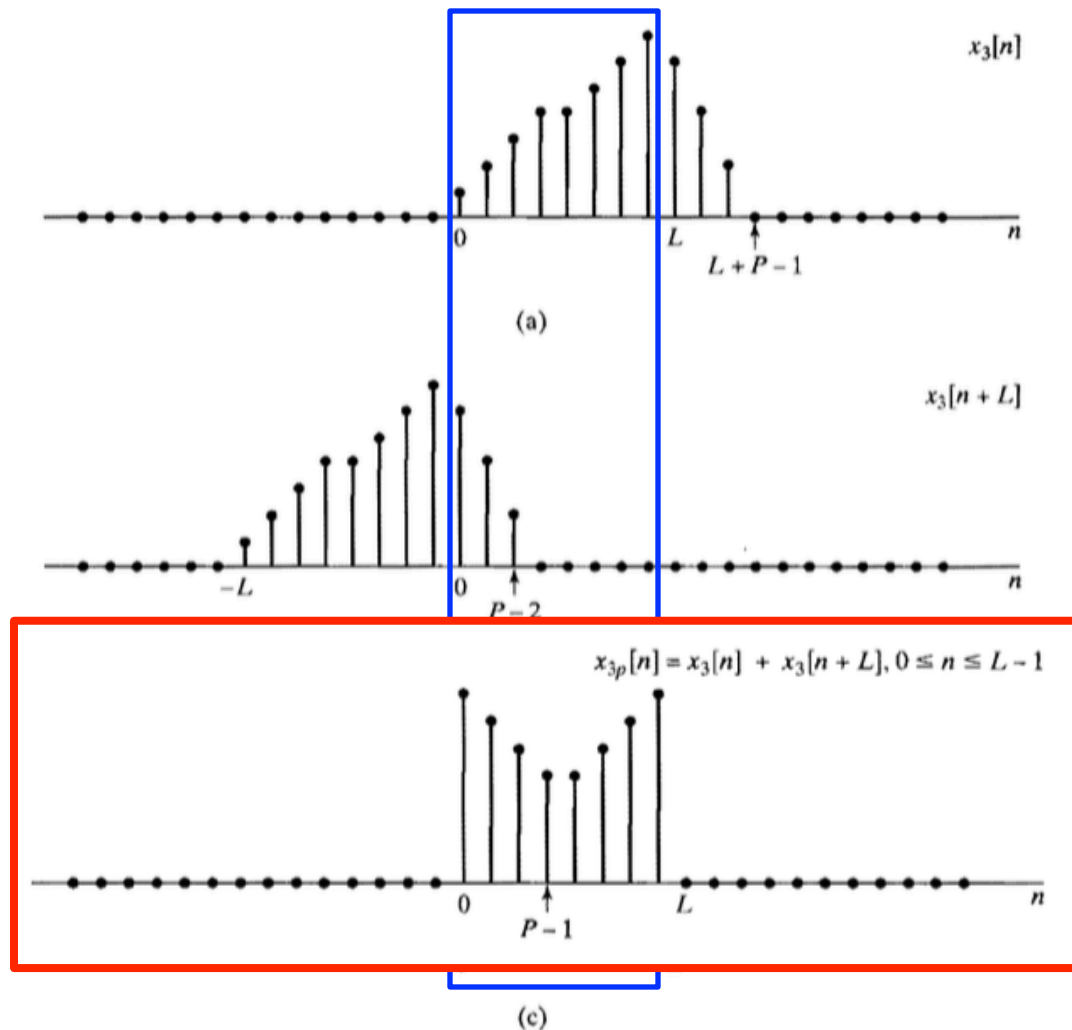
Example 2:

- The L-shifted linear convolutions



Example 2:

- The L-shifted linear convolutions





Big Ideas

- ❑ Discrete Fourier Transform (DFT)
 - For finite signals assumed to be zero outside of defined length
 - N-point DFT is sampled DTFT at N points
 - Useful properties allow easier linear convolution
- ❑ Fast Fourier Transform
 - Enable computation of an N-point DFT (or DFT^{-1}) with the order of just $N \cdot \log_2 N$ complex multiplications.
- ❑ Fast Convolution Methods
 - Use circular convolution (i.e DFT) to perform fast linear convolution
 - Overlap-Add, Overlap-Save
 - Circular convolution is linear convolution with aliasing
- ❑ Design DSP methods to minimize computations!



Admin

- ❑ Read adaptive filter reference for next lecture

- ❑ Project: Adaptive Filtering
 - Handout posted now
 - Work in pairs
 - Use Piazza to find partners
 - Will discuss next lecture
 - Read handout linked on calendar before class!
 - Additional resources in Canvas Files
 - Due 4/30