ESE 531: Digital Signal Processing

Lec 20: April 9, 2020

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,

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

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

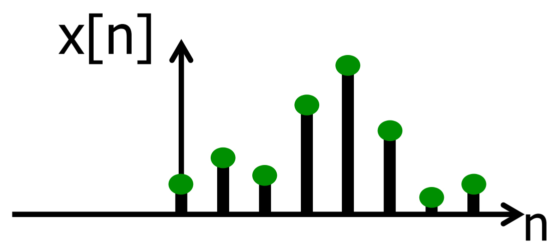
DFT vs. DTFT

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

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

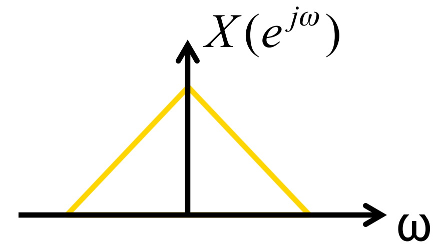
DFT Intuition

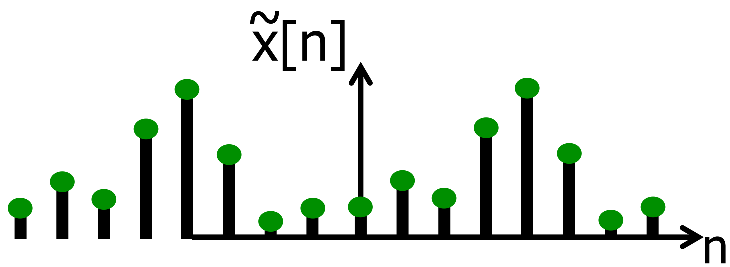
Time | **Transform** | **Frequency**



DTFT

$$x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}\,d\omega$$

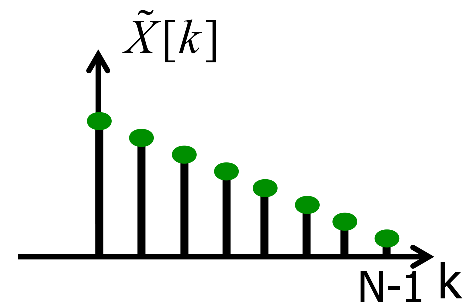


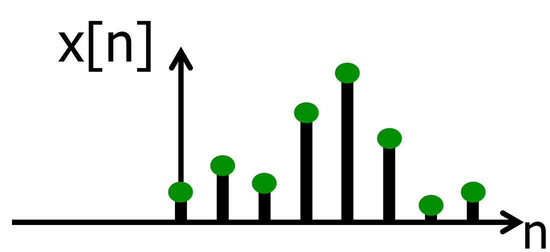


Periodic in N

DFS

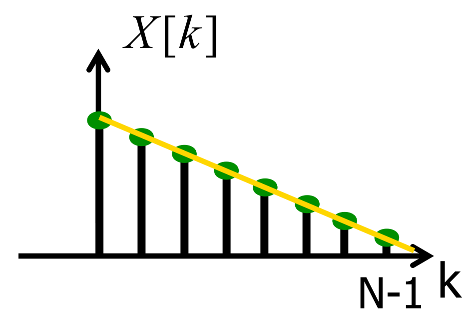
$$\tilde{x}[n] = \frac{1}{N}\sum_{k=0}^{N-1} \tilde{X}[k] W_N^{-kn}$$





DFT

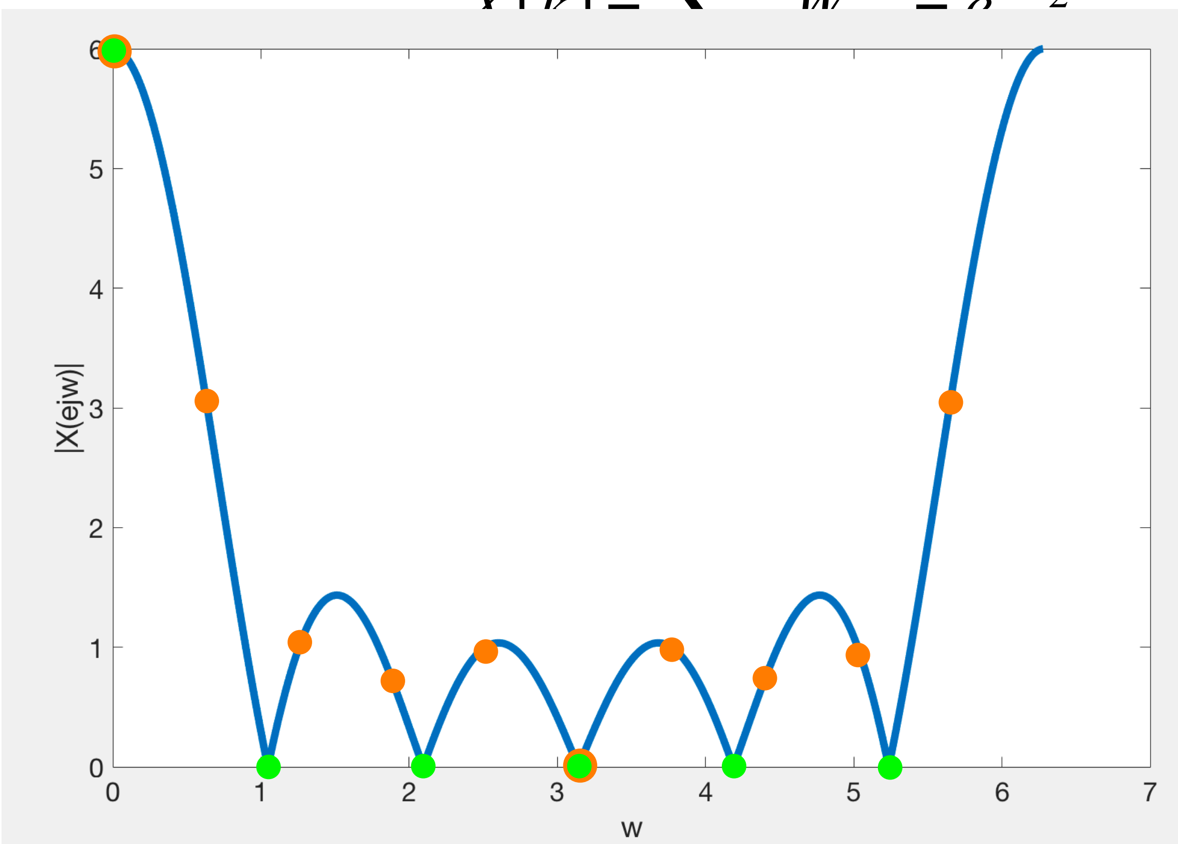
$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k] W_N^{-kn}$$



DFT vs DTFT

- Back to example

$$X[k] = \sum_{n=0}^{5} W^{nk} = e^{-j\frac{\pi}{2}k} \frac{\sin\left(\frac{3\pi}{5}k\right)}{\sin\left(\frac{\pi}{10}k\right)}$$



"6-point" DFT

"10-point" DFT

Use `fftshift` to center around dc

Fast Fourier Transform Algorithms

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

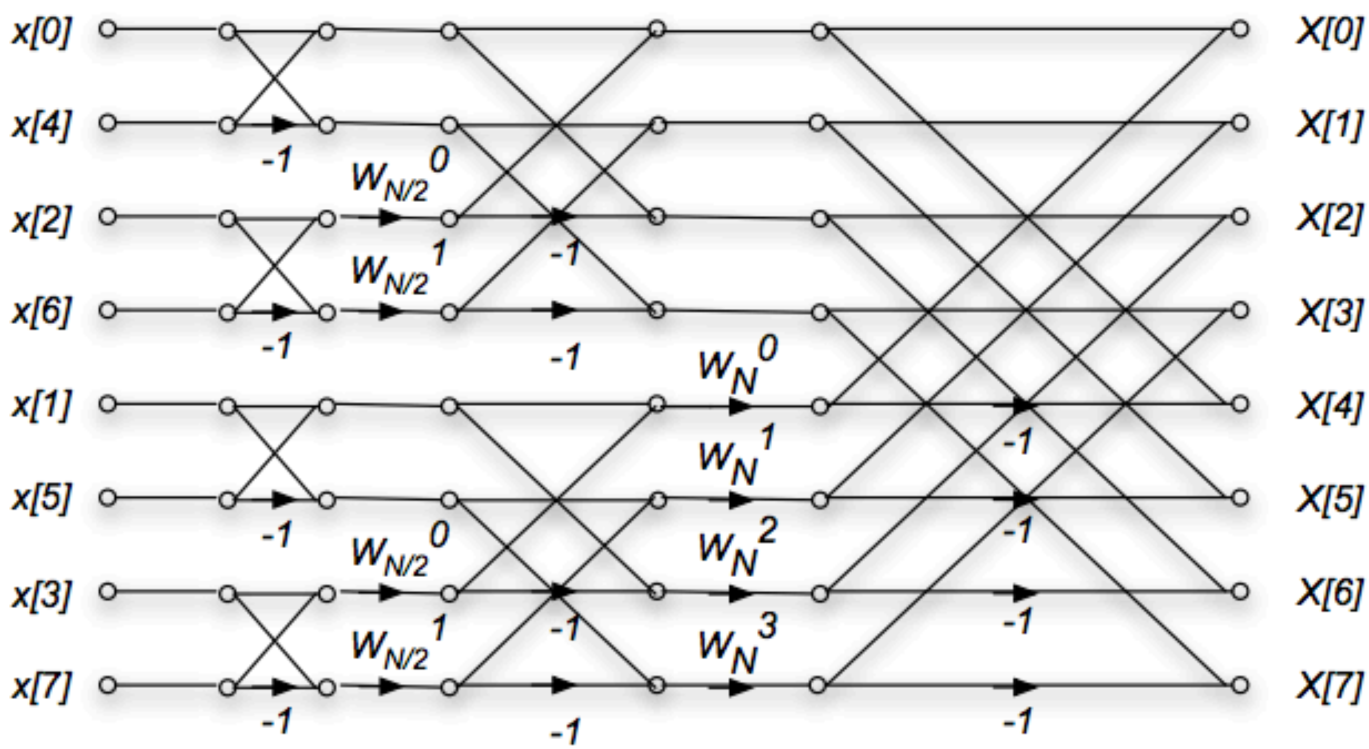
$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \quad k = 0, \ldots, N-1$$

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

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

Decimation-in-Time FFT

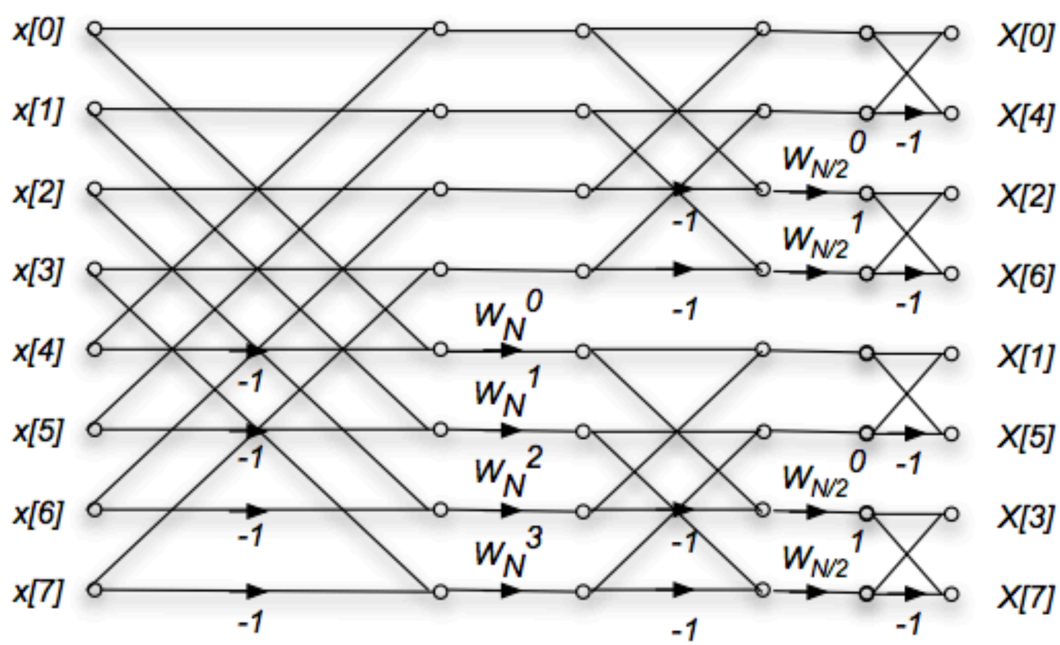
Combining all these stages, the diagram for the 8 sample DFT is:



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

Decimation-in-Frequency FFT

The diagram for and 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.

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?

What information do we know?

x[n]=x[n+N]

h[n] only non-zero for $0 \leq n < K$

$$y[n] = x[n] * h[n]$$

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?

What information do we know?

x[n]=x[n+N]

h[n] only non-zero for $0 \leq n < K$

$$y[n] = x[n] * h[n]$$

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

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?

What information do we know?

x[n]=x[n+N]

h[n] only non-zero for $0 \leq n < K$

What do we need to show?

y[n]=y[n+M]

$$y[n] = x[n] * h[n]$$

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

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?

What information do we know?

x[n]=x[n+N]

h[n] only non-zero for $0 \le n < K$

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

What do we need to show?

y[n]=y[n+M]

$$y[n+M] = \sum_{m=0}^{K-1} x[(n+M)-m]h[m]$$

M=?

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?

What information do we know?

x[n]=x[n+N]

h[n] only non-zero for $0 \le n < K$

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

What do we need to show?

y[n]=y[n+M]

$$y[n+N] = \sum_{m=0}^{K-1} x[(n+N)-m]h[m]$$
$$= \sum_{m=0}^{K-1} x[n-m]h[m]$$
$$= y[n]$$

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 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 $r = 10$, $m = 10$).

What information do we know?

$$x[n] = \{x[0], \ldots, x[N-1]\}$$

$$h[n] = \{h[0], \ldots, h[K-1]\}$$

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 $r = 10$, $m = 10$).

What information do we know?

$$x[n] = \{x[0], \ldots, x[N-1]\} \rightarrow \tilde{x}[n] = \sum_{r=-\infty}^{\infty} x[n - rN], \qquad \text{for all } 0 \le n < N$$

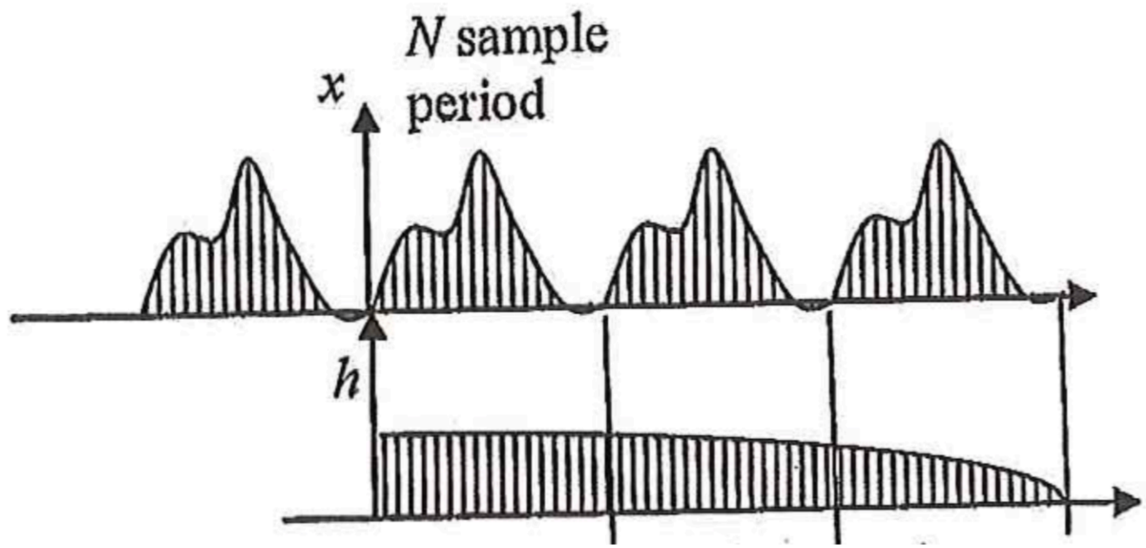
$$h[n] = \{h[0], \ldots, h[K-1]\} = \{h[0], \ldots, h[mN-1]\}$$

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 outp *periodic*. What is

b) Let $K = mN$ where you implement thi analysis clearly. (*multiplications* re direct implementa $r = 10$, $m = 10$).



What information do we know?

$$x[n] = \{x[0], \ldots, x[N-1]\} \rightarrow \tilde{x}[n] = \sum_{r=-\infty}^{\infty} x[n - rN], \qquad \text{for all } 0 \le n < N$$

$$h[n] = \{h[0], \ldots, h[K-1]\} = \{h[0], \ldots, h[mN-1]\}$$

Example 1:

What information do we know?

$$x[n] = \{x[0],\ldots,x[N-1]\} \rightarrow \tilde{x}[n] = \sum_{r=-\infty}^{\infty} x[n-rN], \qquad \text{for all } 0 \le n < N$$

$$h[n] = \{h[0],\ldots,h[K-1]\} = \{h[0],\ldots,h[mN-1]\}$$

$$y[n] = \{y[0],\ldots,y[N-1]\} \rightarrow \tilde{y}[n] = \sum_{r=-\infty}^{\infty} y[n-rN], \qquad \text{for all } 0 \le n < N$$

$$y[n] = \sum_{p=0}^{K-1} x[n-p]h[p]$$

Example 1:

$$\tilde{y}[n] = \sum_{p=0}^{K-1} \tilde{x}[n-p]h[p]$$

Example 1:

$$\tilde{y}[n] = \sum_{p=0}^{K-1} \tilde{x}[n-p]h[p]$$

$$= \sum_{p=0}^{mN-1} \tilde{x}[n-p]h[p]$$

Example 1:

$$\tilde{y}[n] = \sum_{p=0}^{K-1} \tilde{x}[n-p]h[p]$$

$$= \sum_{p=0}^{mN-1} \tilde{x}[n-p]h[p]$$

$$= \sum_{p=0}^{N-1} \tilde{x}[n-p]h[p] + \sum_{p=N}^{2N-1} \tilde{x}[n-p]h[p] + \sum_{p=2N}^{3N-1} \tilde{x}[n-p]h[p] + \ldots + \sum_{p=(m-1)N}^{mN-1} \tilde{x}[n-p]h[p]$$

Example 1:

$$\tilde{y}[n] = \sum_{p=0}^{K-1} \tilde{x}[n-p]h[p]$$

$$= \sum_{p=0}^{mN-1} \tilde{x}[n-p]h[p]$$

$$= \sum_{p=0}^{N-1} \tilde{x}[n-p]h[p] + \sum_{p=N}^{2N-1} \tilde{x}[n-p]h[p] + \sum_{p=2N}^{3N-1} \tilde{x}[n-p]h[p] + \ldots + \sum_{p=(m-1)N}^{mN-1} \tilde{x}[n-p]h[p]$$

$$= \sum_{p=0}^{N-1} \tilde{x}[n-p]h[p] + \sum_{p=0}^{N-1} \tilde{x}[n-(p+N)]h[p+N] + \sum_{p=0}^{N-1} \tilde{x}[n-(p+2N)]h[p+2N]$$

$$+ \ldots + \sum_{p=0}^{N-1} \tilde{x}[n-(p+(m-1)N)]h[p+(m-1)N]$$

Example 1:

$$\tilde{y}[n] = \sum_{p=0}^{N-1} \tilde{x}[n-p]h[p] + \sum_{p=0}^{N-1} \tilde{x}[n-(p+N)]h[p+N] + \sum_{p=0}^{N-1} \tilde{x}[n-(p+2N)]h[p+2N]$$

$$+\ldots+\sum_{p=0}^{N-1} \tilde{x}[n-(p+(m-1)N)]h[p+(m-1)N]$$

$$= \sum_{p=0}^{N-1} \tilde{x}[n-p]h[p] + \sum_{p=0}^{N-1} \tilde{x}[n-p]h[p+N] + \sum_{p=0}^{N-1} \tilde{x}[n-p]h[p+2N]$$

$$+\ldots+\sum_{p=0}^{N-1} \tilde{x}[n-p]h[p+(m-1)N]$$

$$= \tilde{x}[n] * \left\{h[n]\right\}_0^{N-1} + \tilde{x}[n] * \left\{h[n+N]\right\}_0^{N-1} + \tilde{x}[n] * \left\{h[n+2N]\right\}_0^{N-1}$$

$$+\ldots+\tilde{x}[n] * \left\{h[n+mN]\right\}_0^{N-1}$$

$$= \tilde{x}[n] * \left(\sum_{q=0}^{m-1} h[p+qN]\right)$$

Example 1:

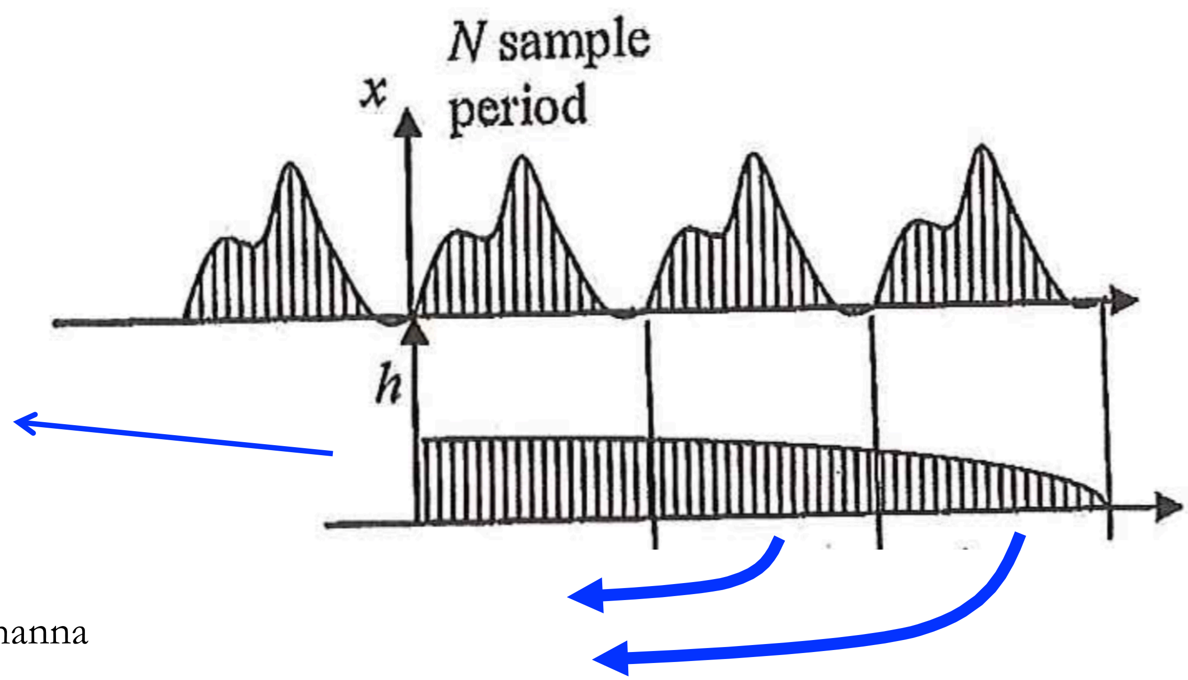
$$\tilde{y}[n] = \tilde{x}[n] * \left(\sum_{q=0}^{m-1} h[p+qN] \right) \rightarrow y[n] = x[n] * \left(\sum_{q=0}^{m-1} h[p+qN] \right)$$

Example 1:

$$\tilde{y}[n] = \tilde{x}[n] * \left(\sum_{q=0}^{m-1} h[p+qN] \right) \rightarrow y[n] = x[n] * \left(\sum_{q=0}^{m-1} h[p+qN] \right)$$

We see that the result of the convolution block of length N is a circular convolution of one period of the input with the N-block sum of the impulse response

$$\left(\sum_{q=0}^{m-1} h[p+qN] \right)$$



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.

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 $r = 10$, $m = 10$).

$$y[n] = x[n] * \left(\sum_{q=0}^{m-1} h[p + qN] \right)$$

What about the computation?

$N=2^{10}$, $K=2^{11}$

Example 2:

A sequence $x = \{x[n], n = 0,1,\ldots, N - 1\}$ 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 $x = \{x[n], n = 0,1,..., N - 1\}$ 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.)*

$$e^{j2\pi\cdot\frac{7}{12}} = e^{j2\pi\cdot\frac{14}{24}}$$

$$e^{j2\pi\cdot\frac{3}{8}} = e^{j2\pi\cdot\frac{9}{24}}$$

24-point DFT

Example 2:

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

b) Suppose N is large. You want to obtain $X(e^{j\omega})$ at the following $2M$ frequencies:

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

Here $M = 2^{\mu} << N = 2^{v}$, (Eg. μ=8, v=14)

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?

Example 2:

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

Here $M = 2^{\mu} << N = 2^{v}$, (Eg. μ=8, v=14)

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.

$$X\left(e^{j\left(\frac{2\pi}{M}m+\frac{2\pi}{N}\right)}\right) = \sum_{n=0}^{N-1} x[n]e^{-j\left(\frac{2\pi}{M}m+\frac{2\pi}{N}\right)n}$$

Example 2:

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

Here $M = 2^{\mu} << N = 2^{v}$, (Eg. μ=8, v=14)

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.

$$X\left(e^{j\left(\frac{2\pi}{M}m+\frac{2\pi}{N}\right)}\right) = \sum_{n=0}^{N-1} x[n]e^{-j\left(\frac{2\pi}{M}m+\frac{2\pi}{N}\right)n} = \sum_{n=0}^{N-1} \boxed{x[n]e^{-j\frac{2\pi}{M}n}}\, e^{-j\frac{2\pi}{M}nm}$$

=y[n]

Example 2:

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

Here $M = 2^{\mu} << N = 2^{v}$, (Eg. μ=8, v=14)

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.

$$X\left(e^{j\left(\frac{2\pi}{M}m+\frac{2\pi}{N}\right)}\right) = \sum_{n=0}^{N-1} x[n]e^{-j\left(\frac{2\pi}{M}m+\frac{2\pi}{N}\right)n} = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{M}n}e^{-j\frac{2\pi}{M}nm}$$

$$= \sum_{n=0}^{N-1} y[n]e^{-j\frac{2\pi}{M}nm} = Y\left(e^{j\frac{2\pi}{M}m}\right)$$

Example 2:

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

Here $M = 2^{\mu} << N = 2^{\nu}$, (Eg. μ=8, ν=14)

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?

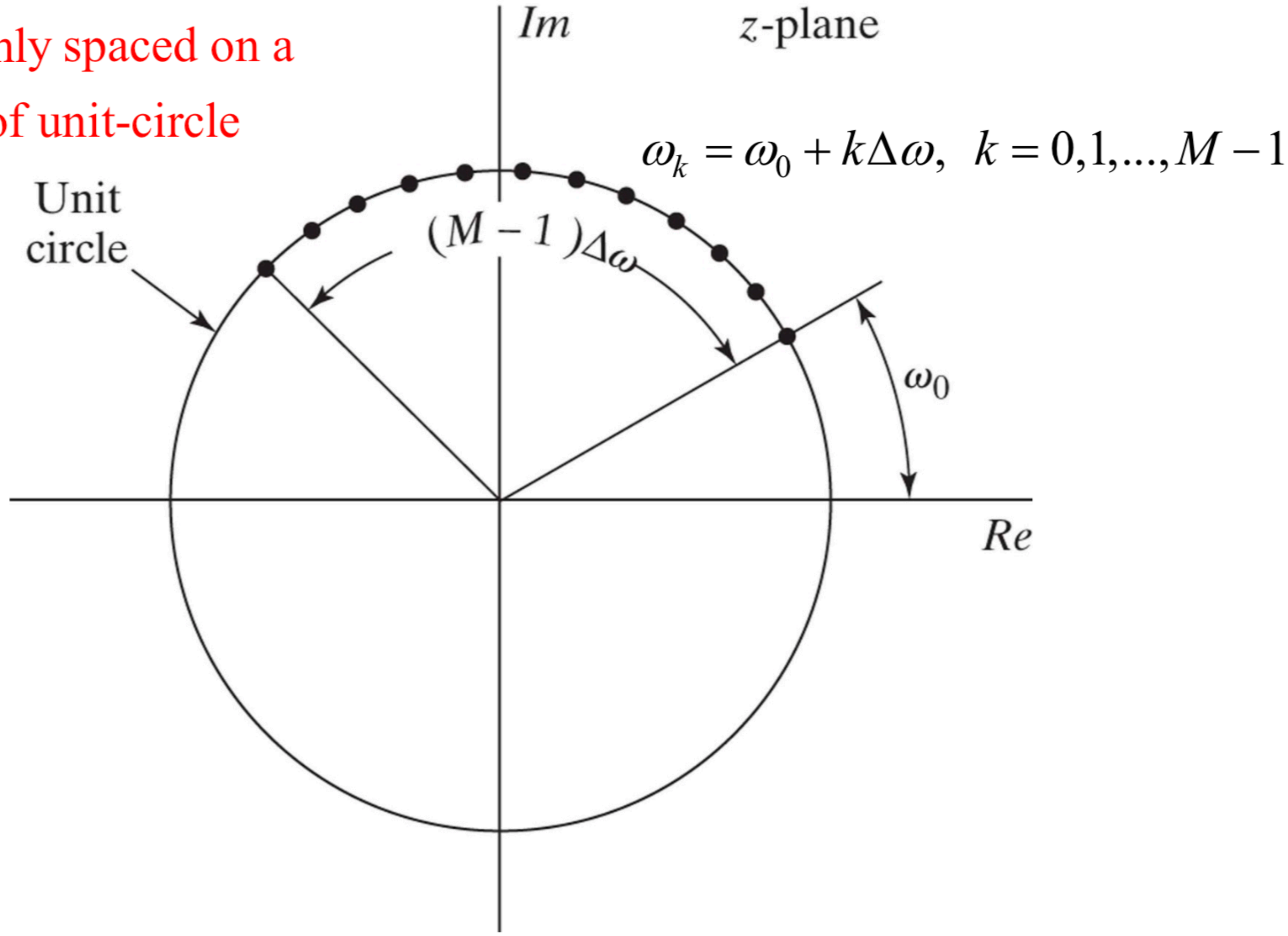
Chirp Transform Algorithm

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.

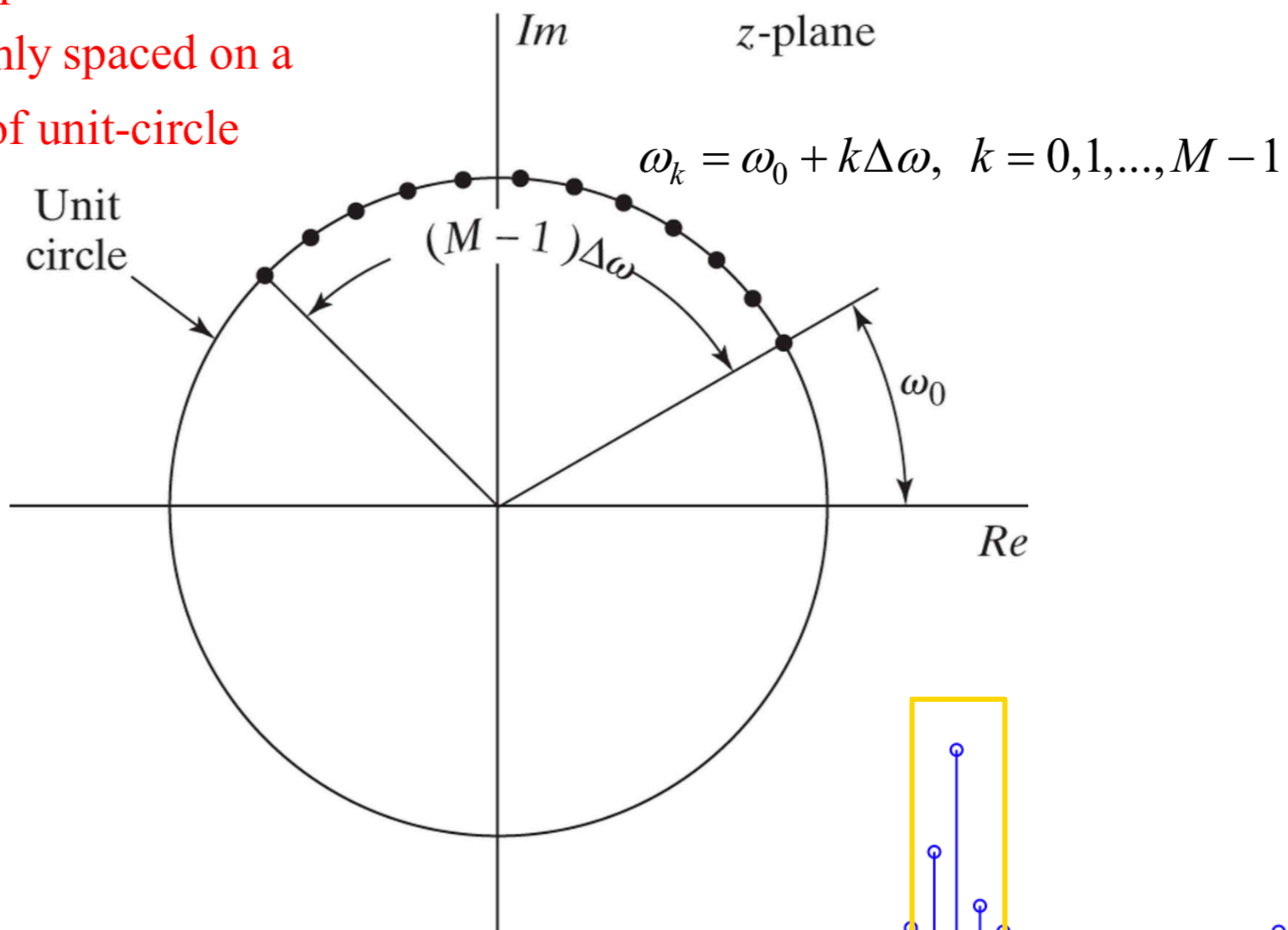
Chirp Transform Algorithm

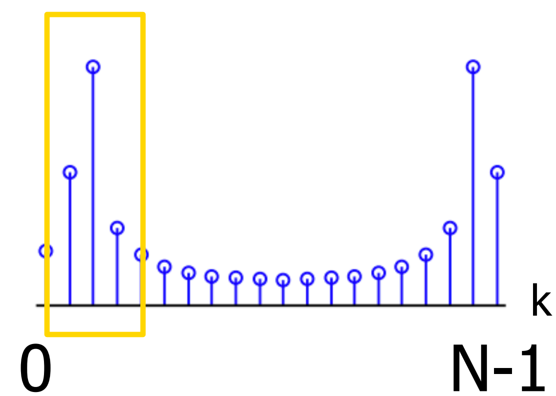
For M points of DTFT uniformly spaced on a sector of unit-circle



Chirp Transform Algorithm

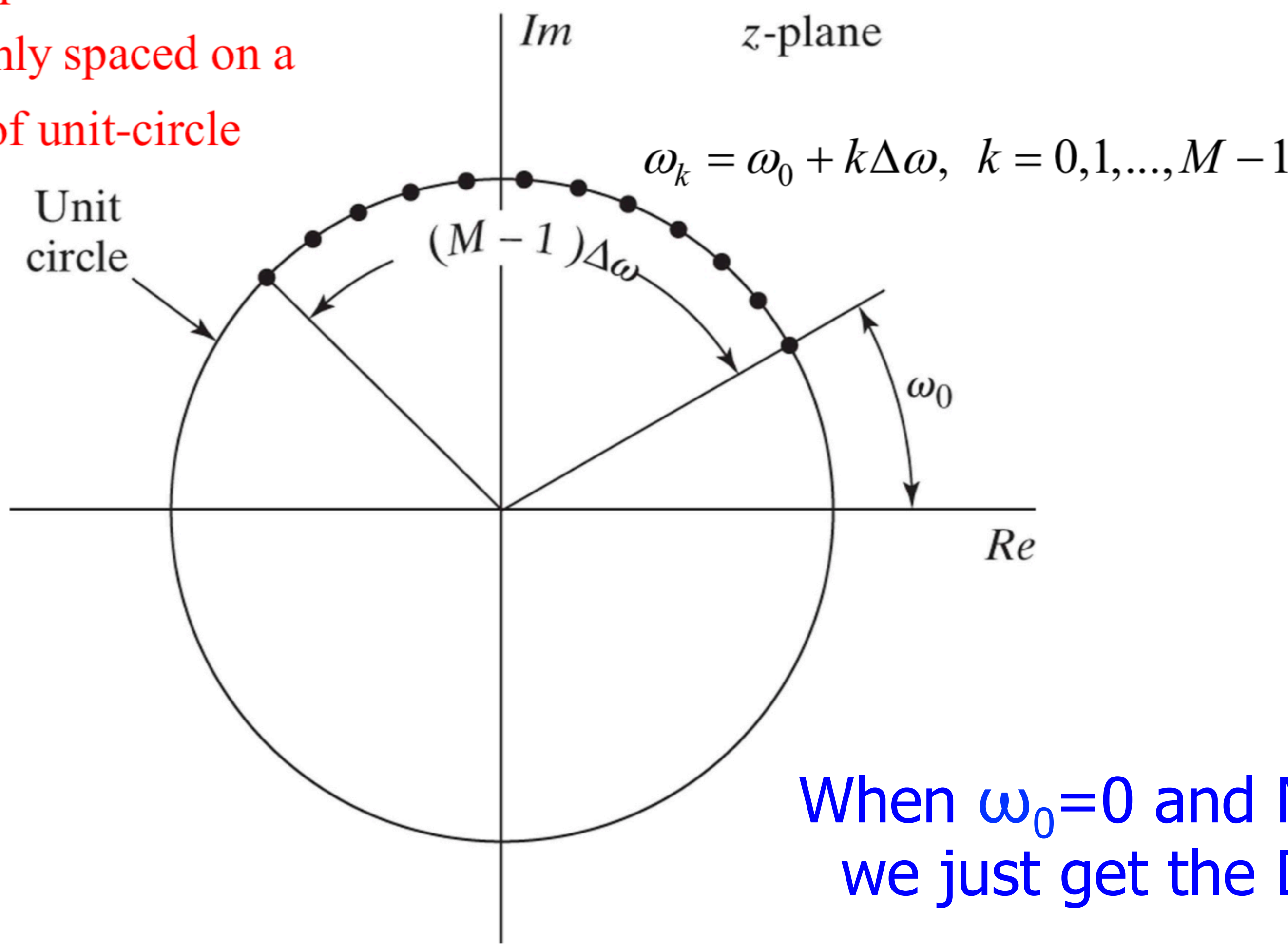
For M points of DTFT uniformly spaced on a sector of unit-circle





Chirp Transform Algorithm

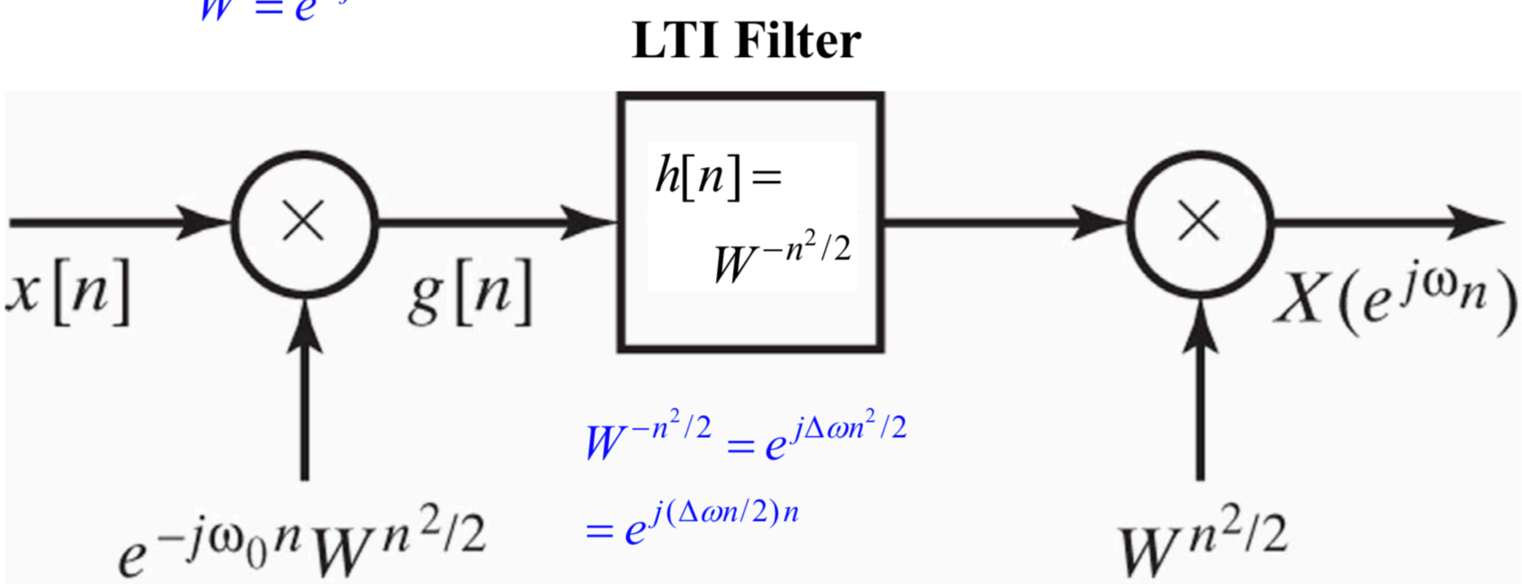
For M points of DTFT uniformly spaced on a sector of unit-circle



When ω_0=0 and M=N, we just get the DFT

Chirp Transform Algorithm

$W \triangleq e^{-j\Delta\omega}$



$W^{-n^2/2} = e^{j\Delta\omega n^2/2}$

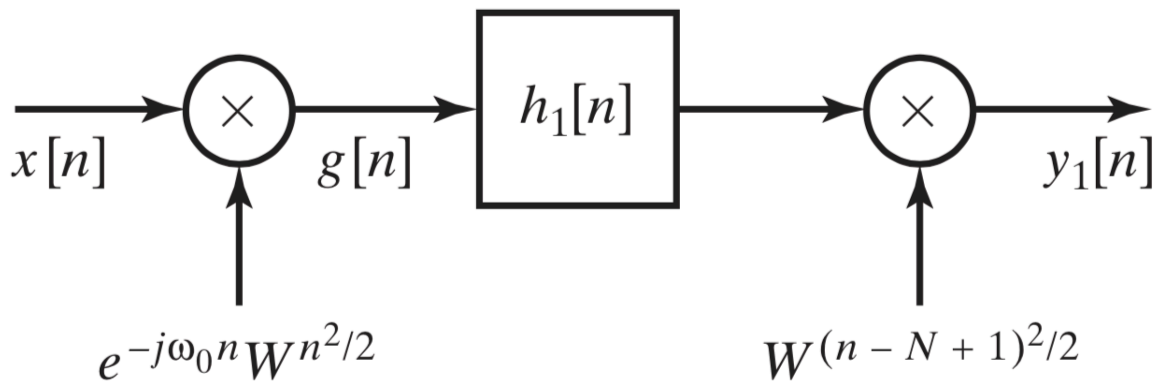
$= e^{j(\Delta\omega n/2)n}$

linearly increasing

frequency "chirp"

Causal FIR CTA

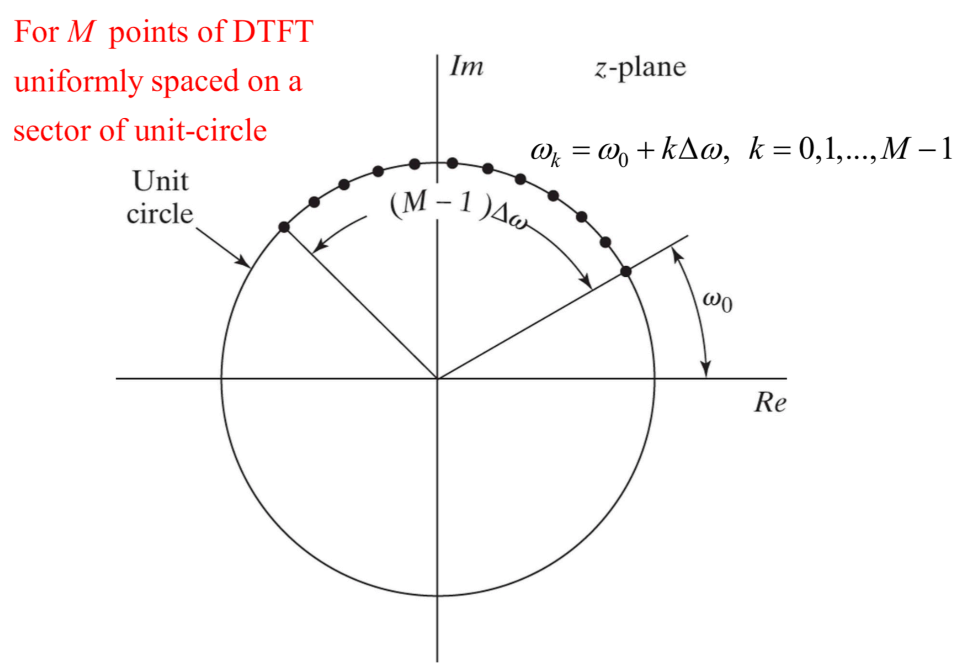
$$h_1[n] = \begin{cases} W^{-(n-N+1)^2/2}, & n = 0, 1, \ldots, M+N-2, \\ 0, & \text{otherwise.} \end{cases}$$



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

Example: Chirp Transform Parameters

- We have a finite-length sequence x[n] that is nonzero only on the interval n = 0, ..., 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, . . . , 15.



Circular Convolution

Linear Convolution with aliasing!

Circular Convolution

- Circular Convolution:

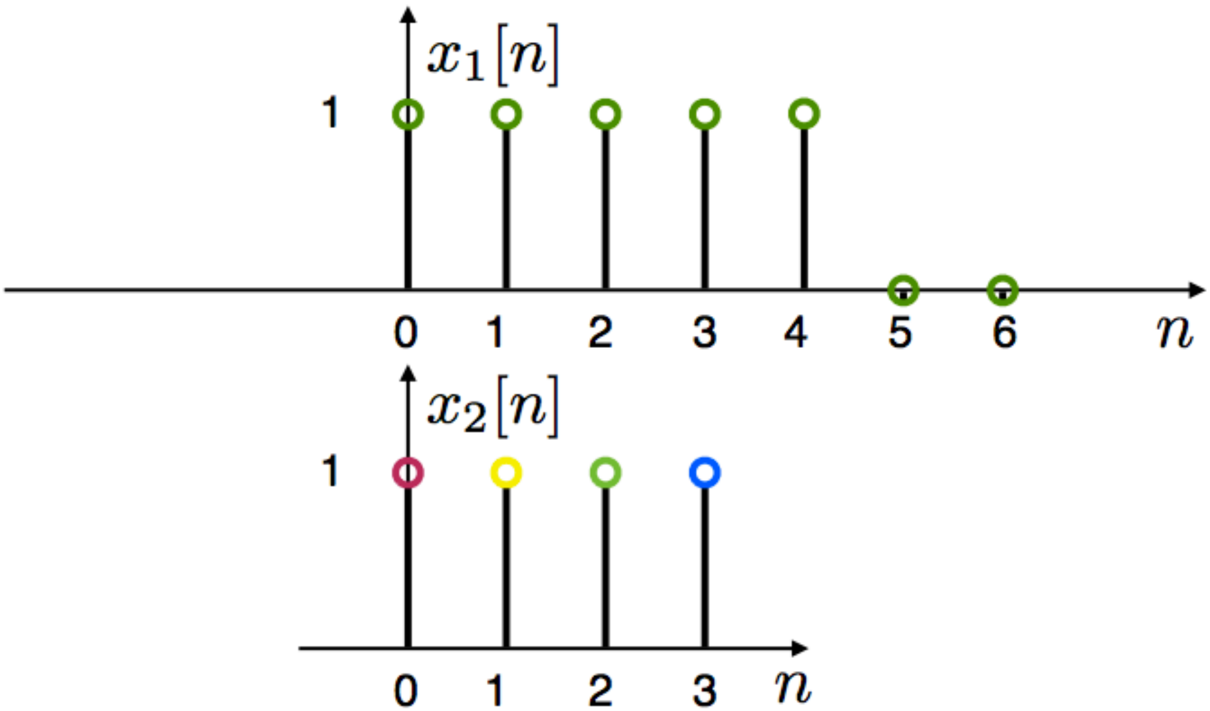
$$x_1[n] \circledN x_2[n] \triangleq \sum_{m=0}^{N-1} x_1[m] x_2[((n-m))_N]$$

For two signals of length N

Note: Circular convolution is commutative

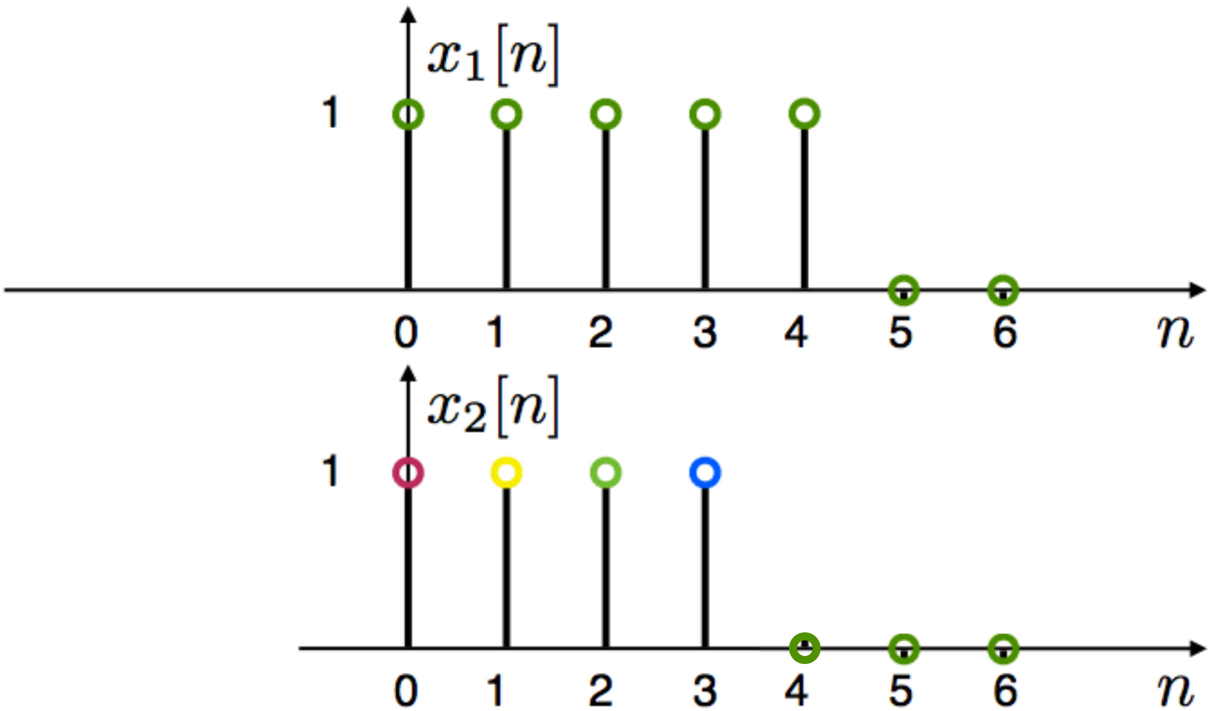
$$x_2[n] \circledN x_1[n] = x_1[n] \circledN x_2[n]$$

Compute Circular Convolution Sum



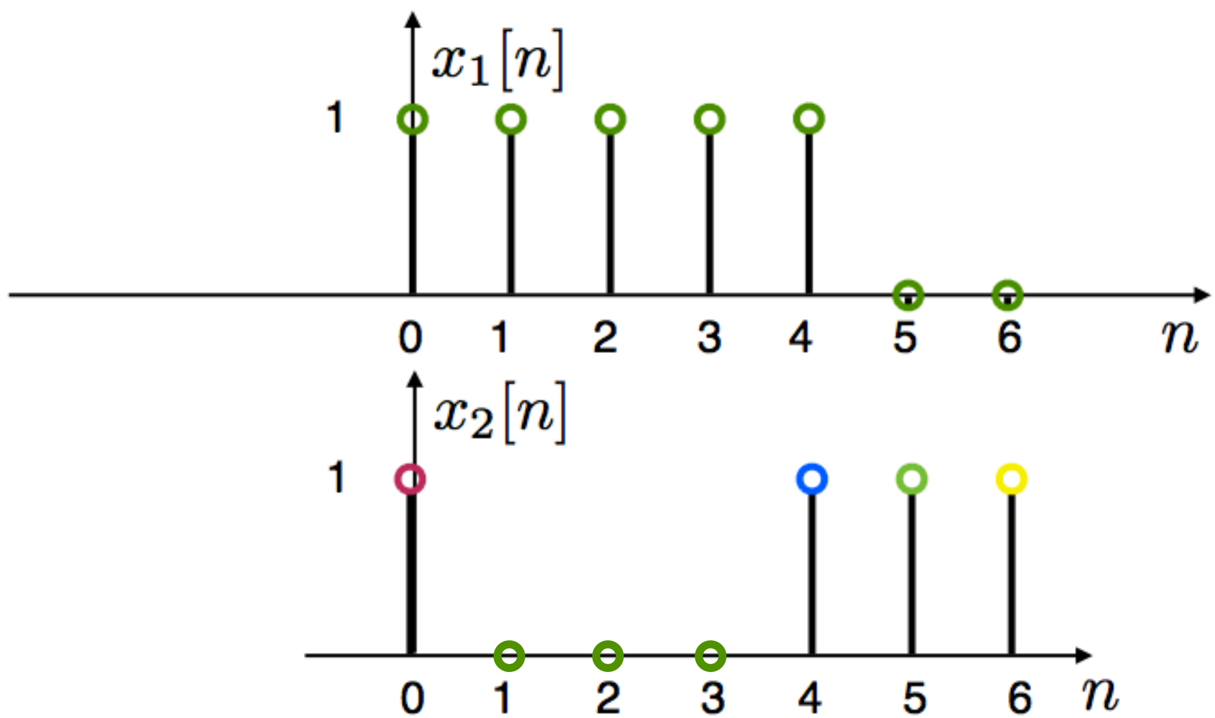
$$x_1[n] \circledN x_2[n] \triangleq \sum_{m=0}^{N-1} x_1[m] x_2[((n-m))_N]$$

Compute Circular Convolution Sum



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

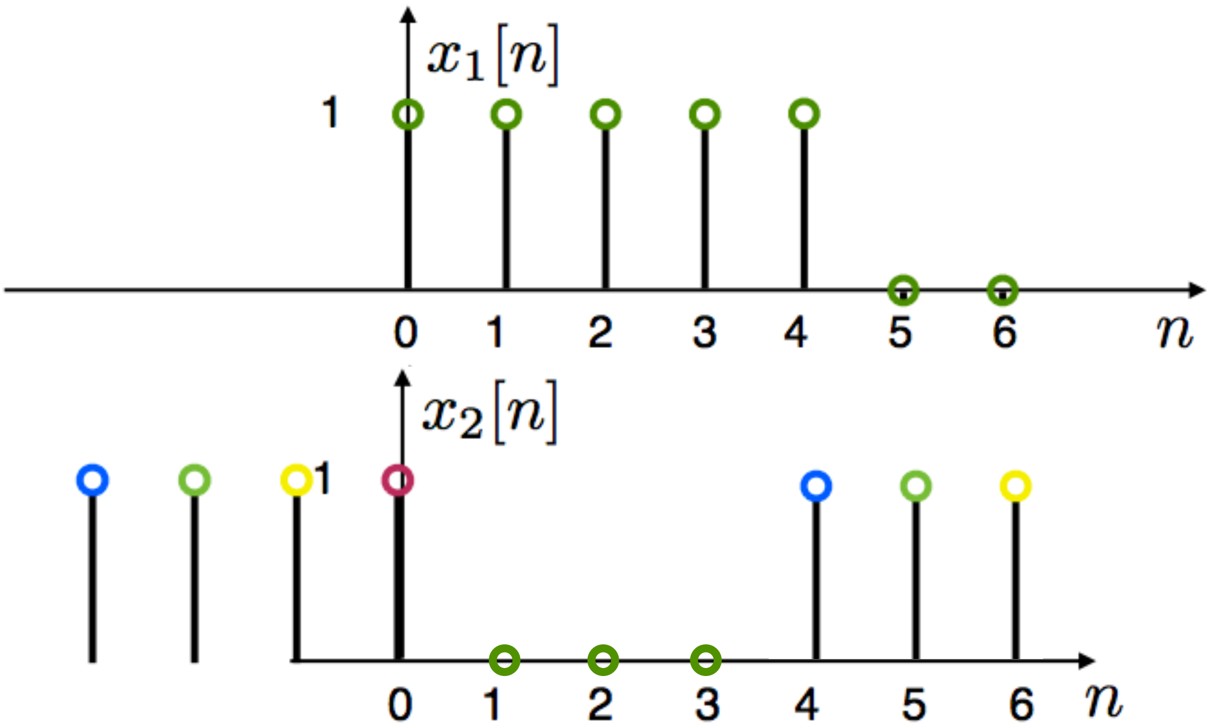
Compute Circular Convolution Sum



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

Compute Circular Convolution Sum

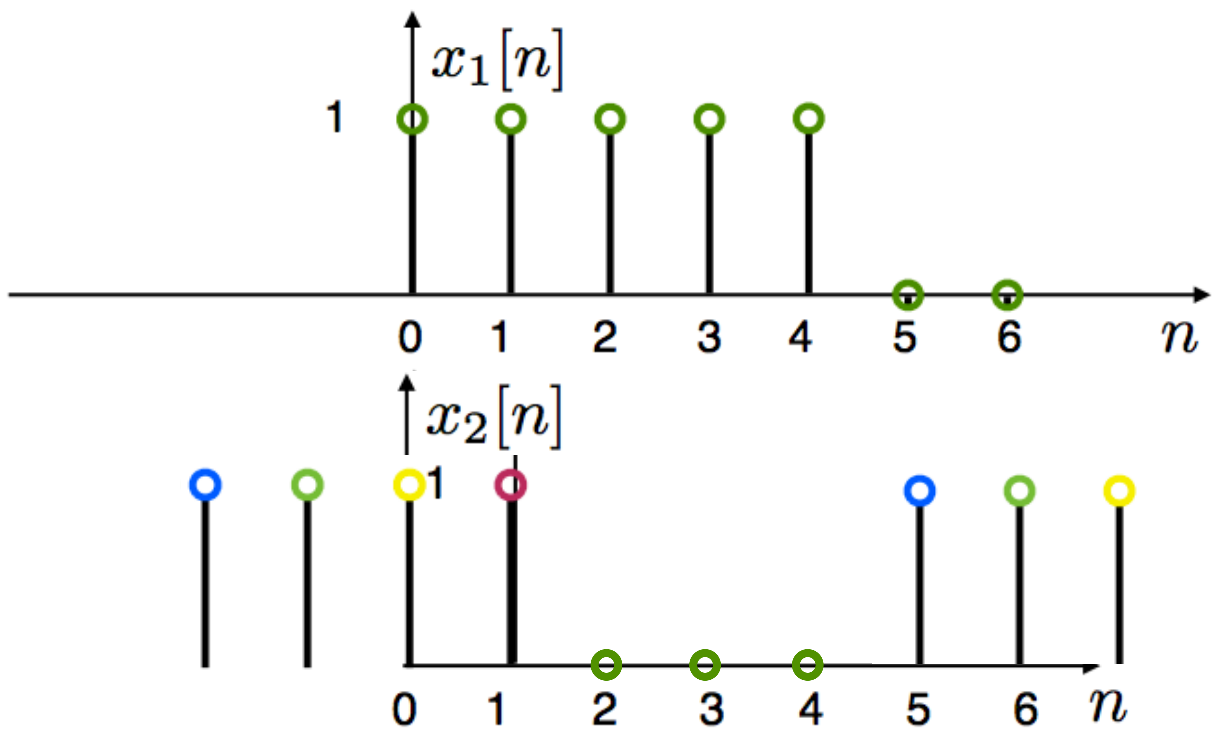
y[0]=2



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

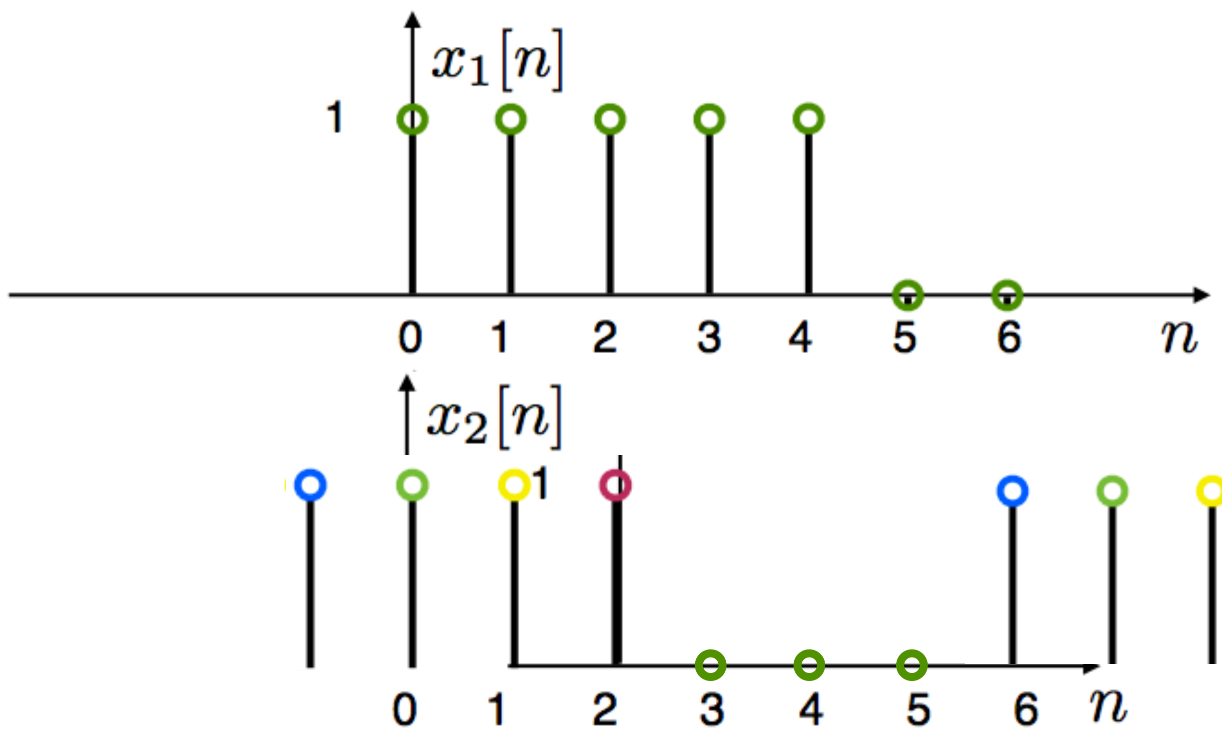
Compute Circular Convolution Sum

y[0]=2
y[1]=2



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

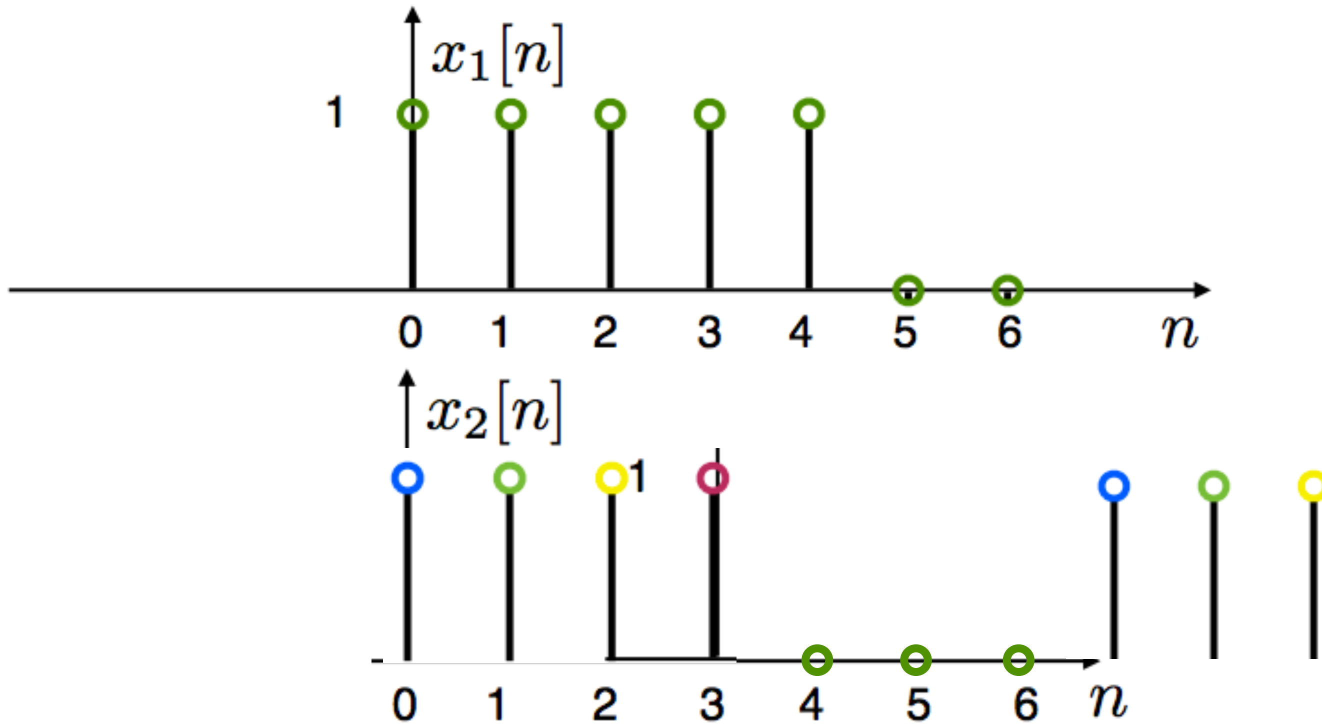
Compute Circular Convolution Sum



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

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

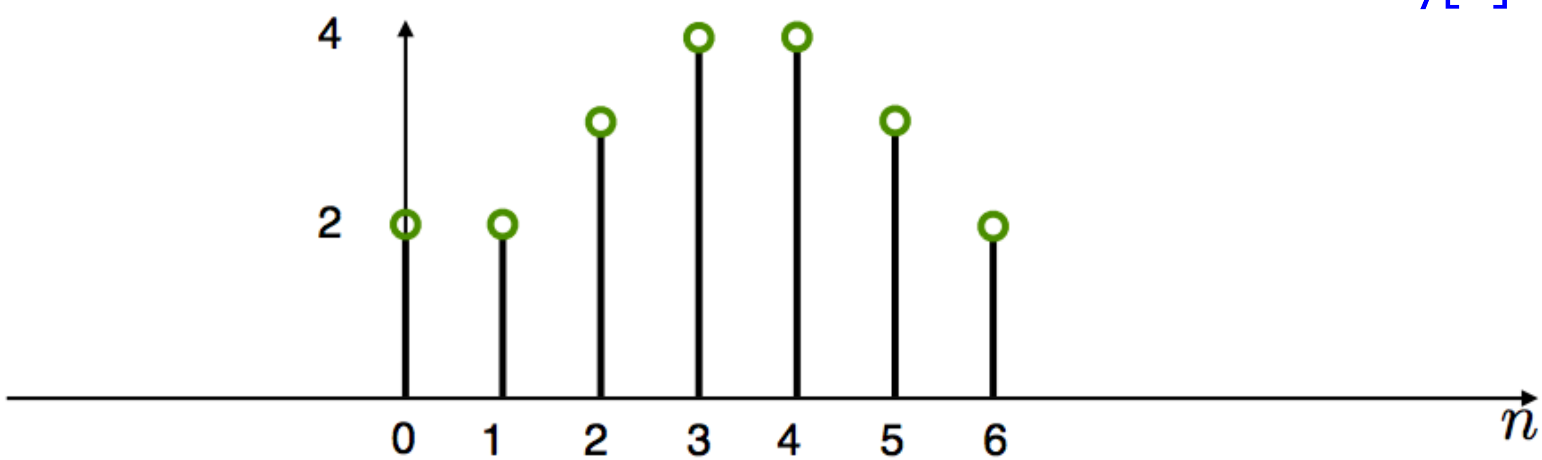
Compute Circular Convolution Sum



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

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

Result



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

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

Linear Convolution

- We start with two non-periodic sequences:

$$x[n] \quad 0 \leq n \leq L-1$$
$$h[n] \quad 0 \leq n \leq P-1$$

  - 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

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

- Zero-pad h[n] by L-1 zeros

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

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

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

Circular Conv. via Linear Conv. w/ Aliasing

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

- 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

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

Circular Conv. as Linear Conv. w/ Aliasing

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

Circular Conv. as Linear Conv. w/ Aliasing

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

- Thus

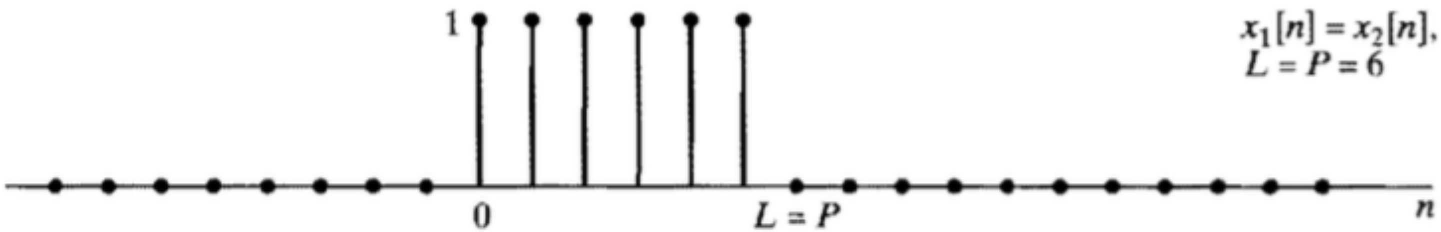
$$x_{3p}[n] = \begin{cases} \sum_{r=-\infty}^{\infty} x_1[n-rN] * x_2[n-rN] & 0 \le n \le N-1 \\ 0 & \text{else} \end{cases}$$

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

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

Example 1:

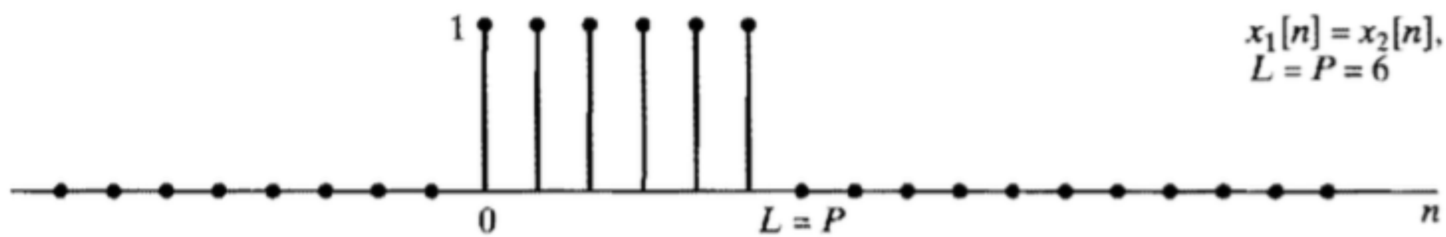
- Let



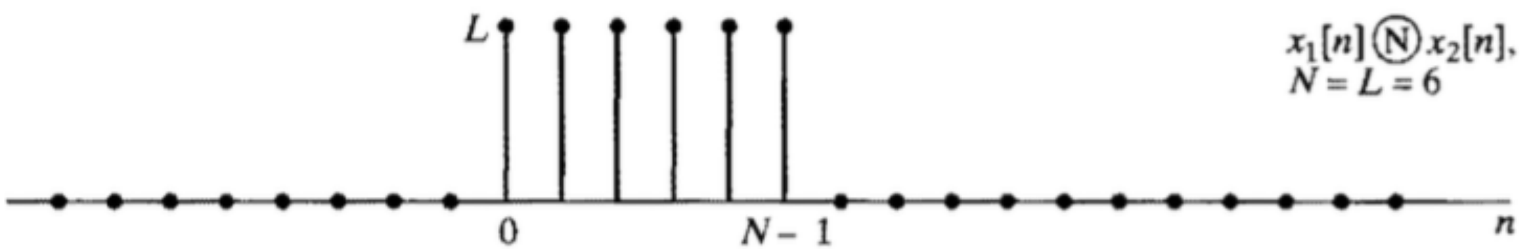
- The N=L=6-point circular convolution results in

Example 1:

- Let

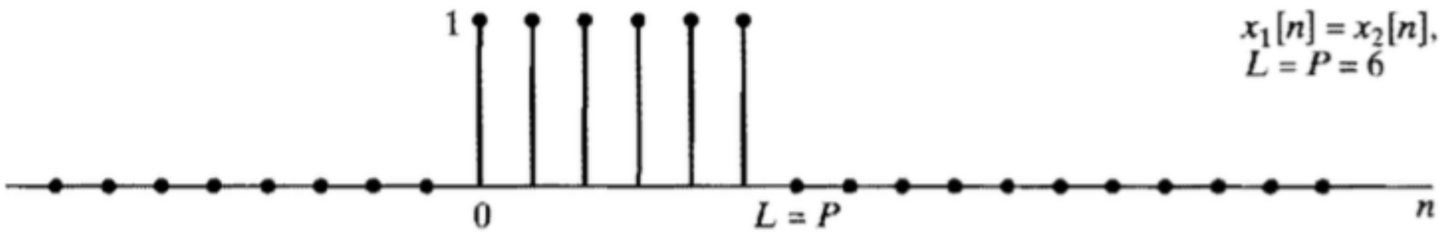


- The N=L=6-point circular convolution results in



Example 1:

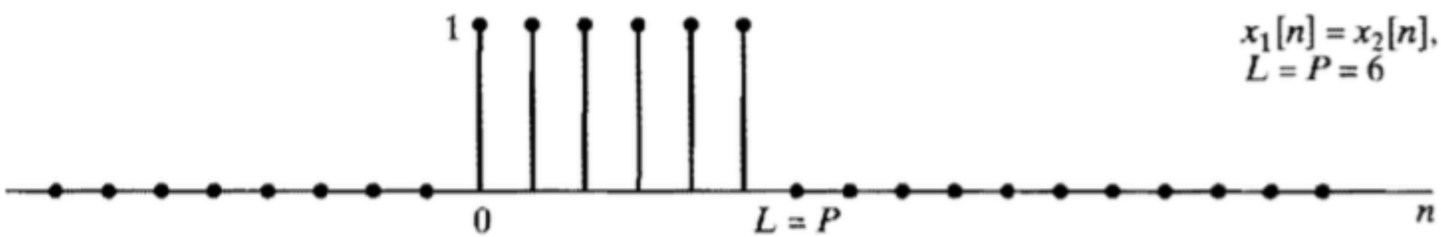
- Let



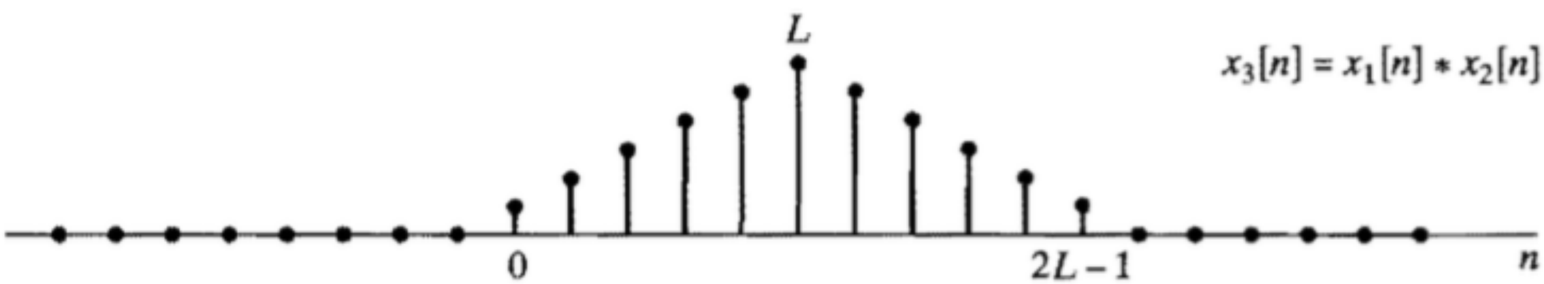
- The linear convolution results in

Example 1:

- Let

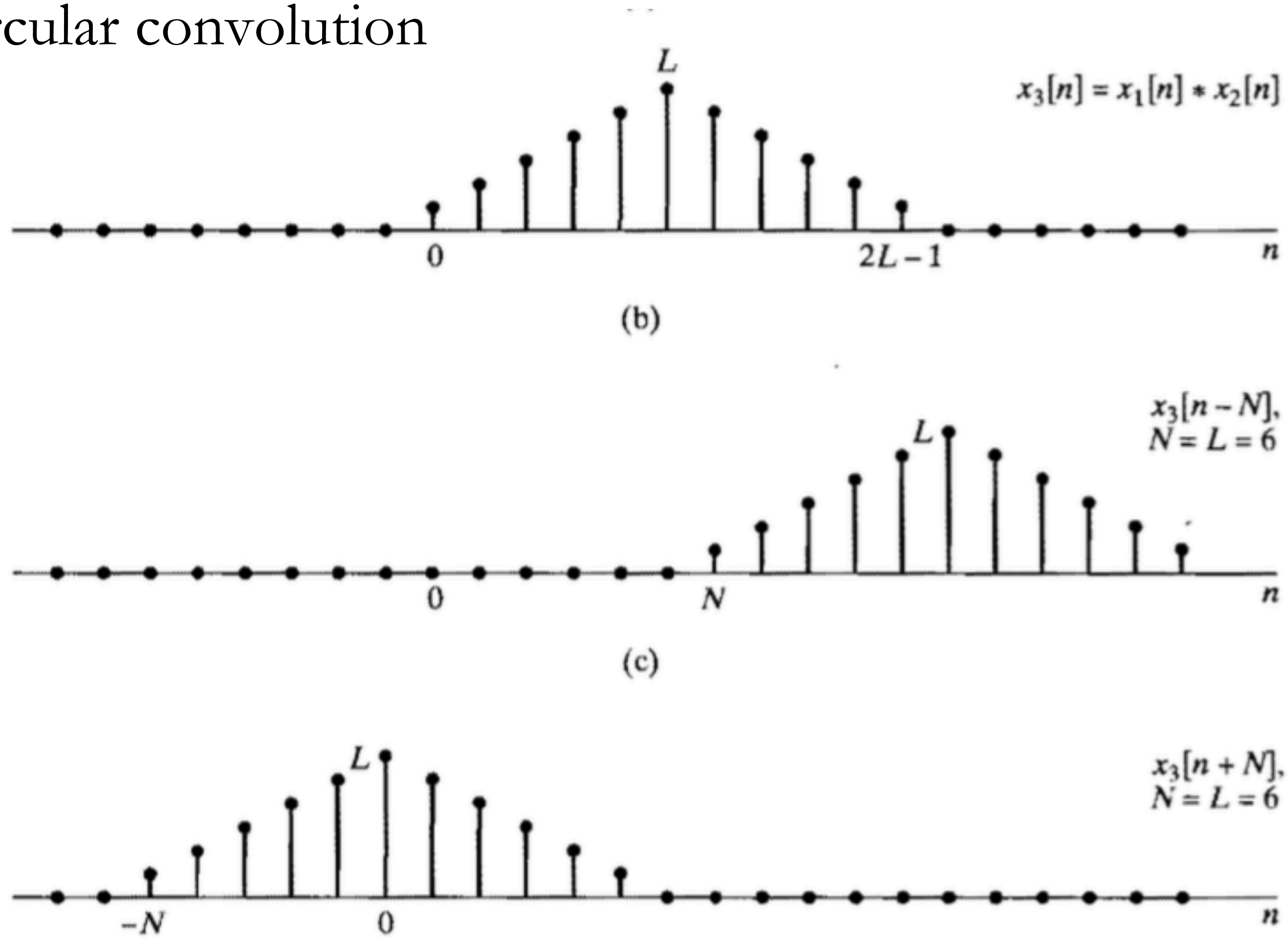


- The linear convolution results in



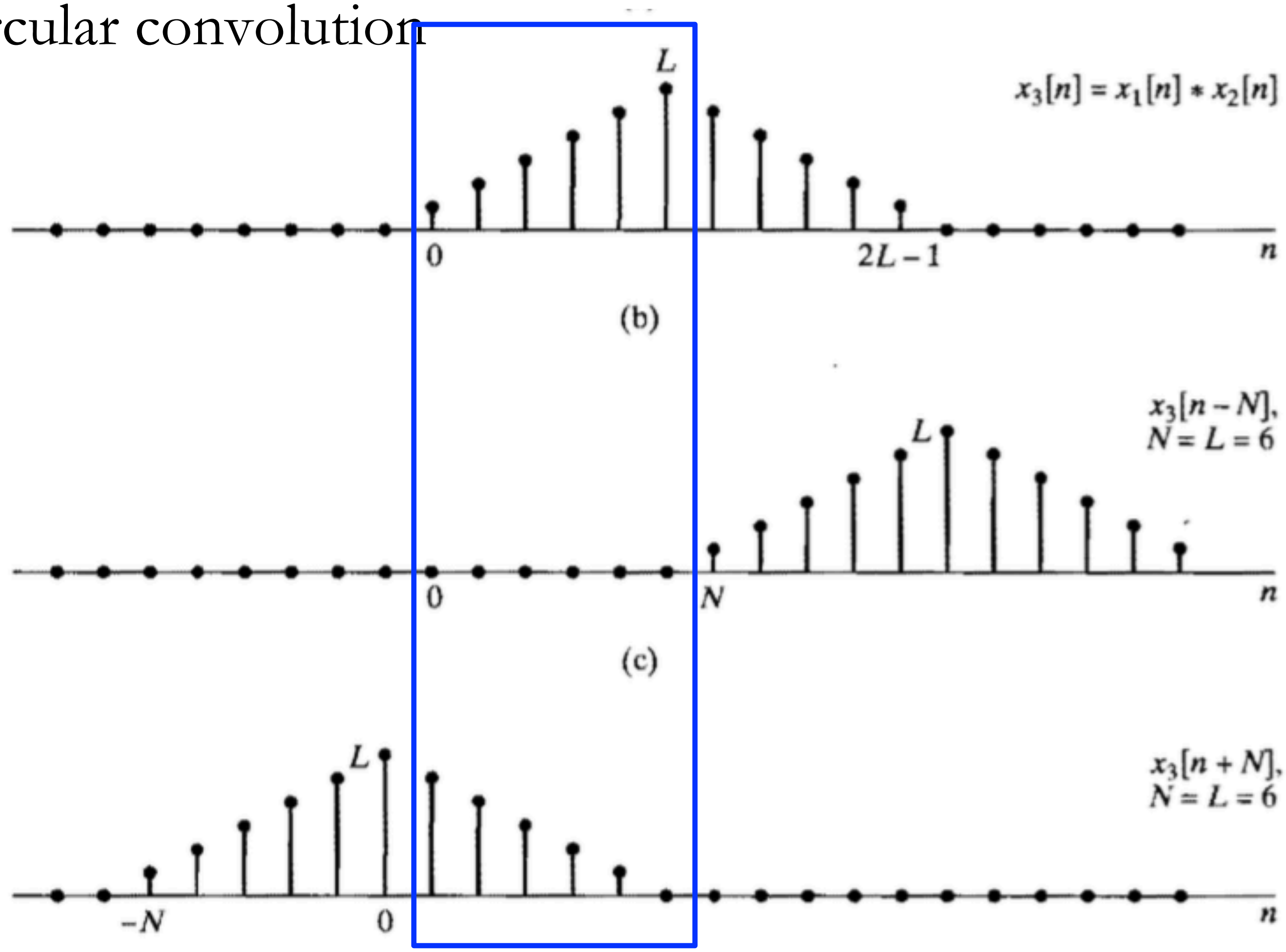
Example 1:

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



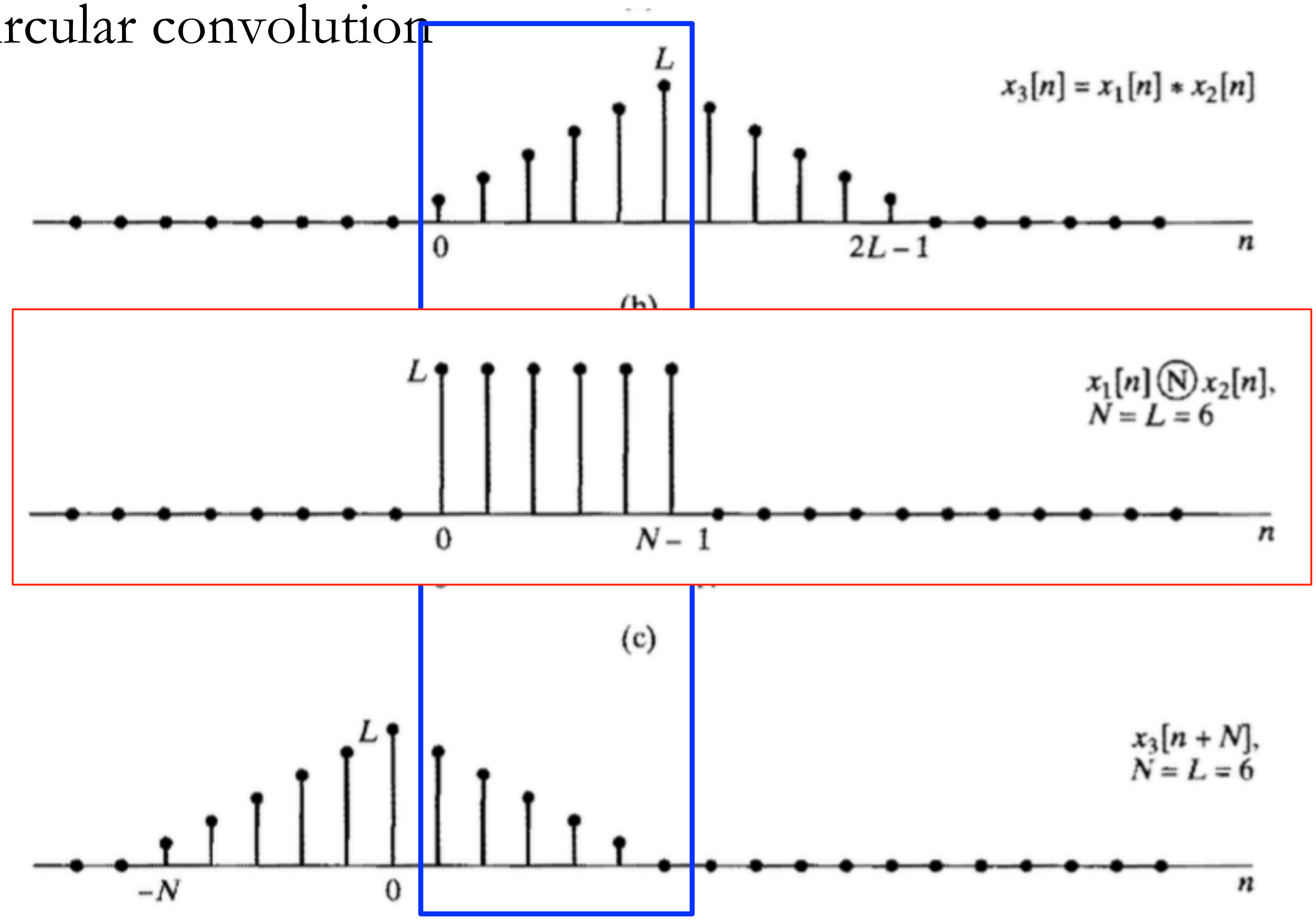
Example 1:

- 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

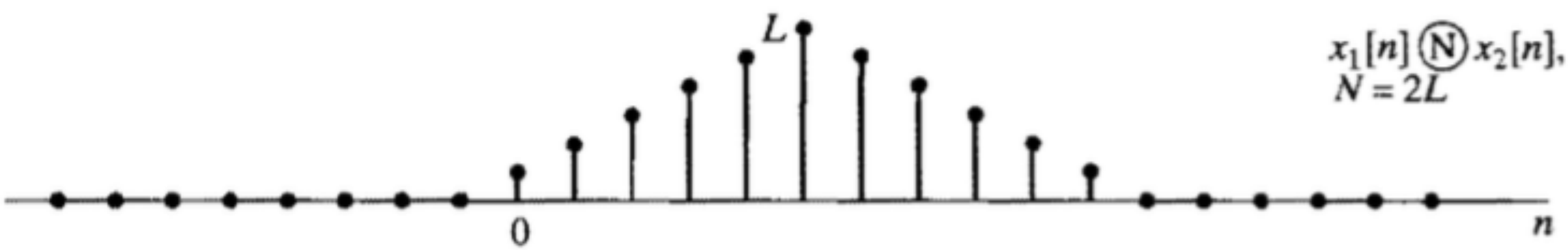


Example 1:

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

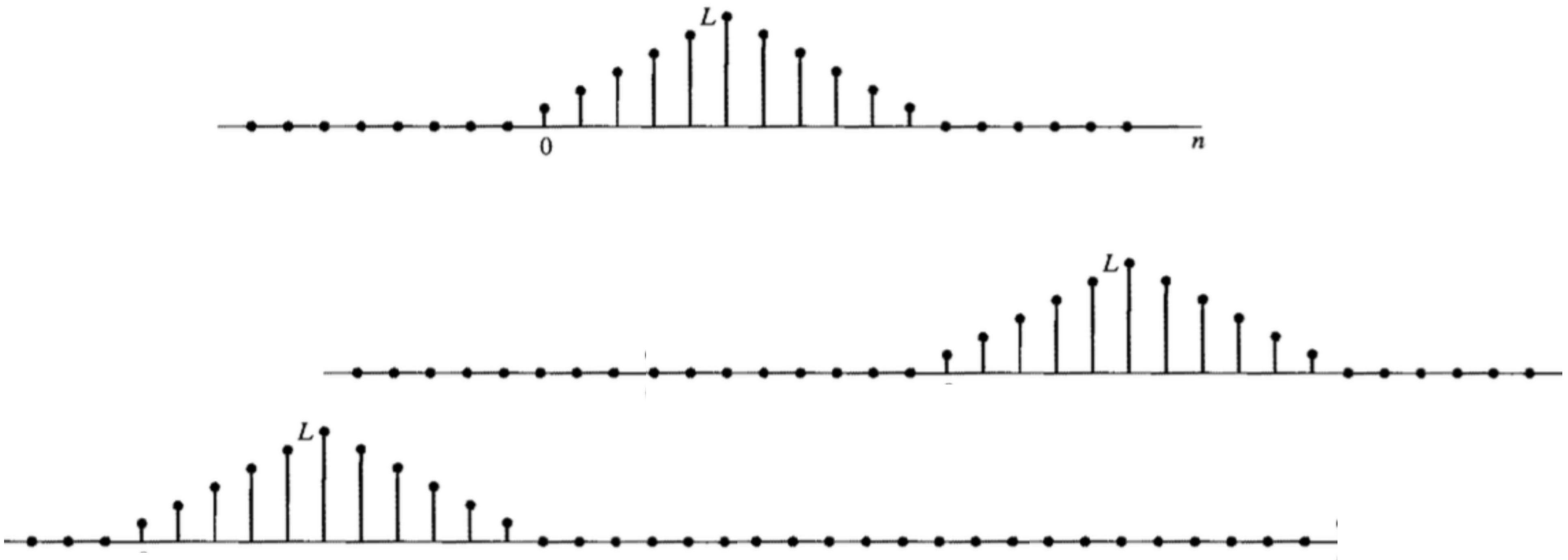
Example 1:

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



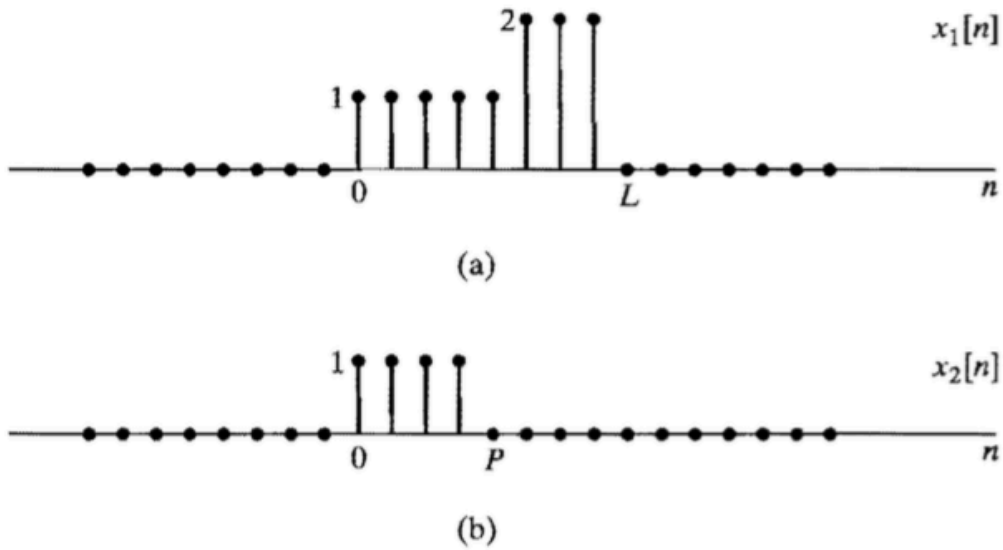
Example 1:

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



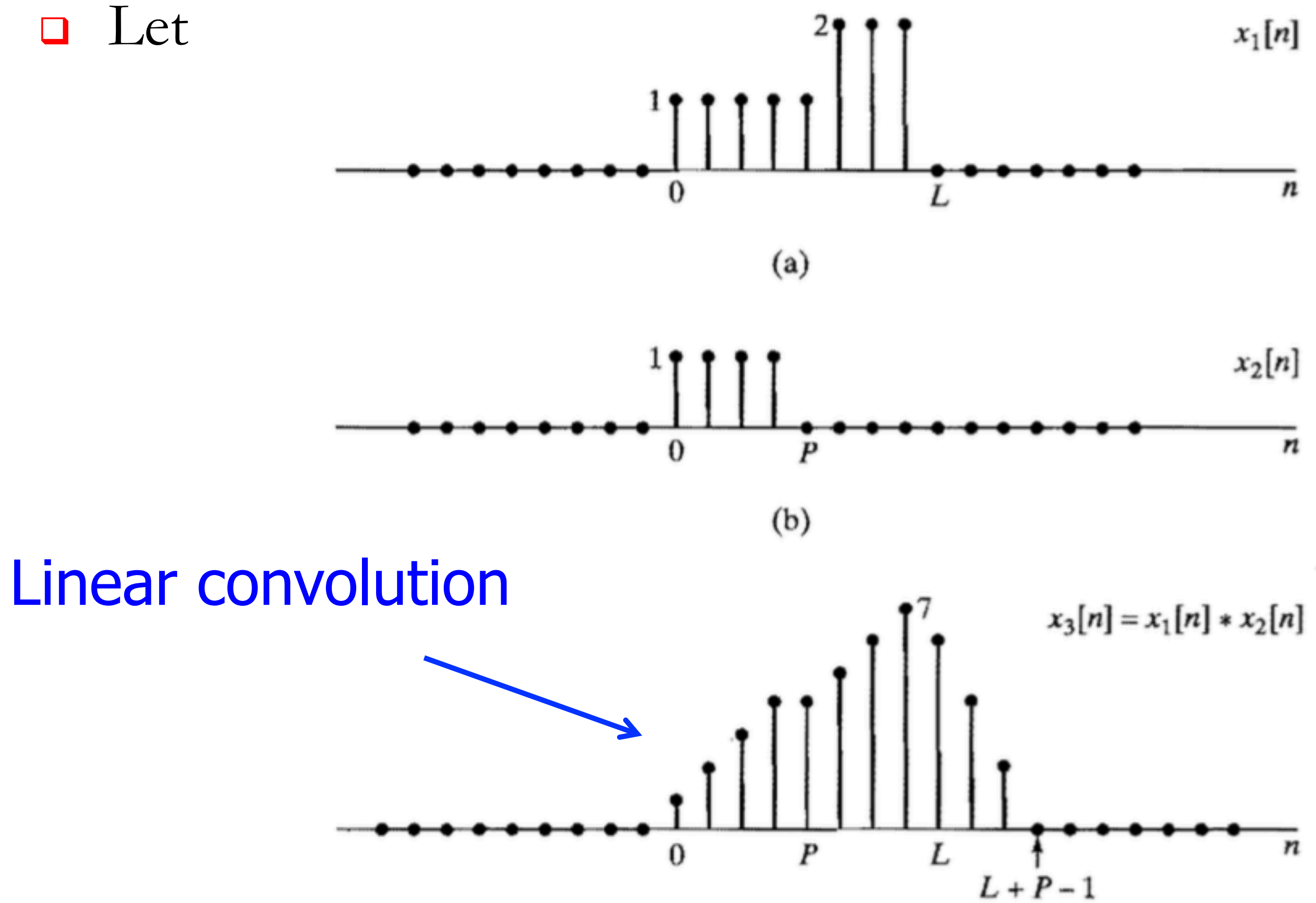
Example 2:

- Let



Example 2:

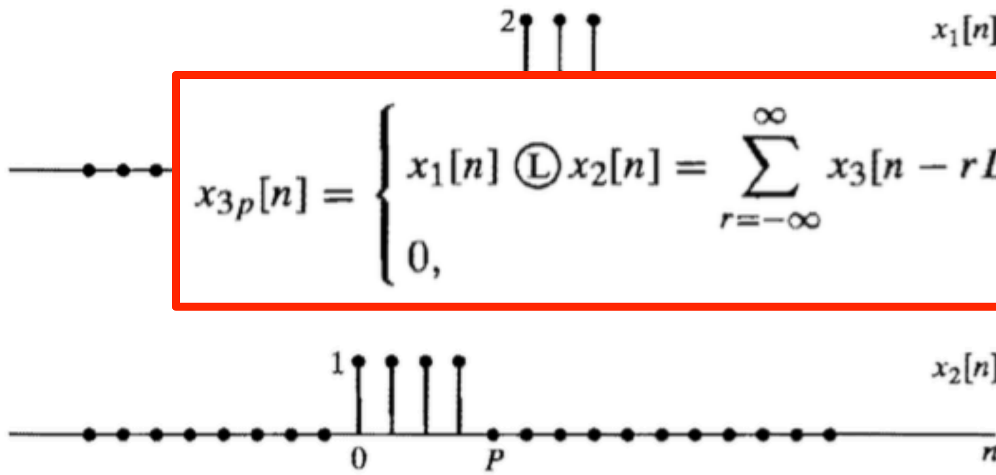
- Let



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

Example 2:

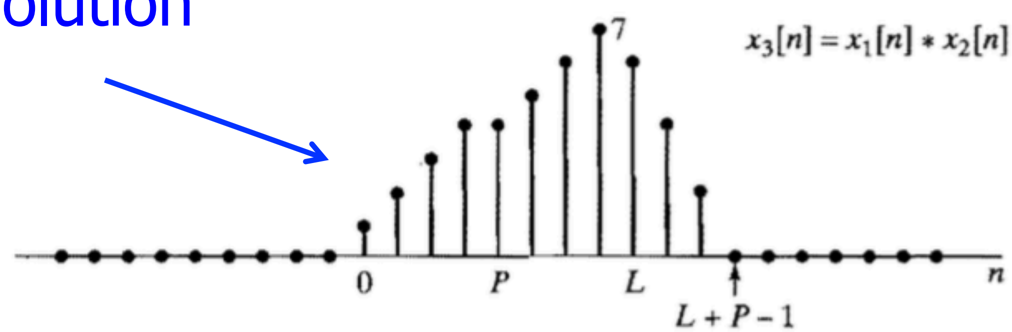
- Let



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

(b)

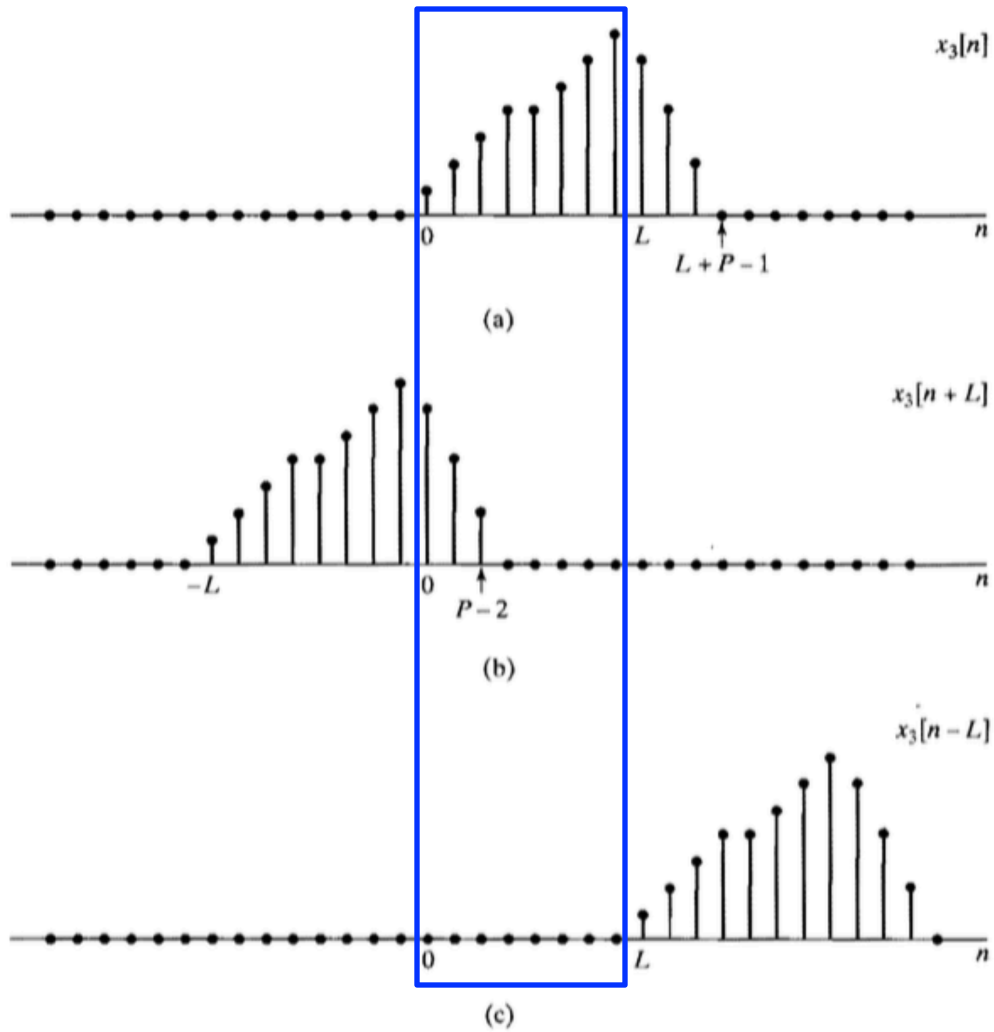
Linear convolution



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

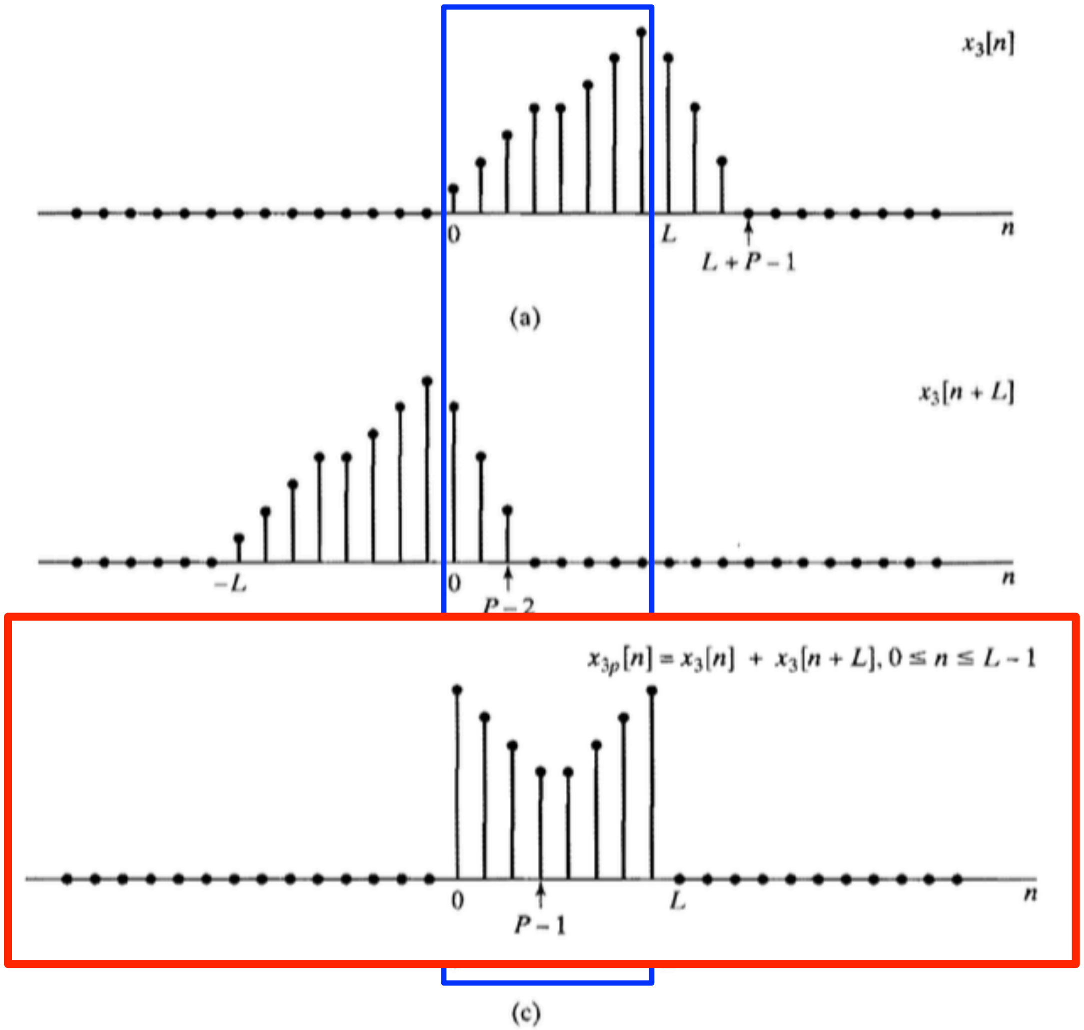
Example 2:

- 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
  - Find on course calendar

- Project: Adaptive Filtering
  - Handout posted over the weekend
  - 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 4/28

ESE GRADUATE ASSOCIATION

WHERE IN THE WORLD IS ESE?

MASTER'S MIXER

with ESE PROFESSORS
TANIA KHANNA
RAKESH VOHRA
& GEORGE PAPPAS

Penn Engineering

THURSDAY, APRIL 9
7 PM-8 PM (EST)
ZOOM ROOM