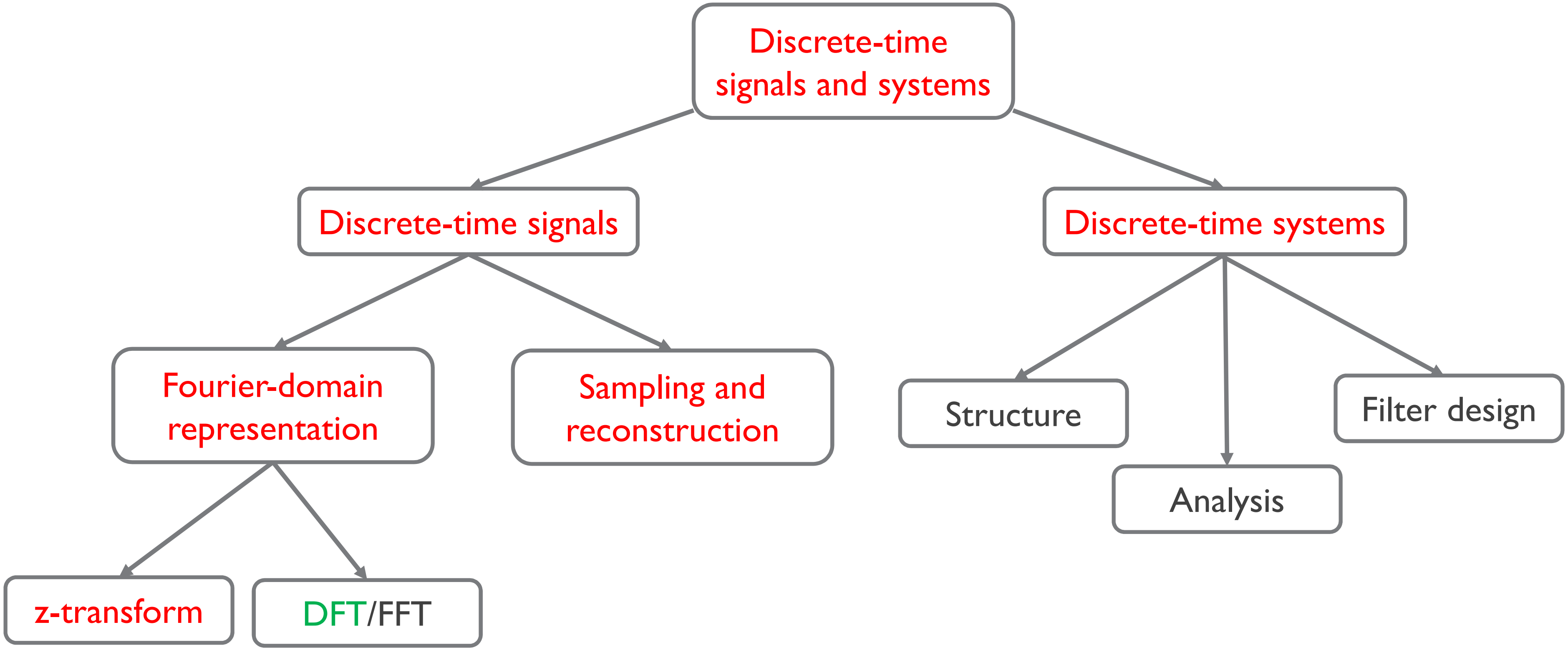
Digital Signal Processing

POSTECH
Department of Electrical Engineering
Junil Choi

Course at glance



Discrete Fourier Transform (DFT)

DFT vs. DFS pairs

- Analysis equations

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

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

- Synthesis equations

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

Zeros outside the range of [0,*N*]

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

Periodic with period *N*

- If we evaluate the values of DFT pairs outside of [0,*N*], they are not zeros, but a rather a periodic extension of *x*[*n*] and *X*[*k*]
  ➔ Assume they are zeros because…

Properties of the DFT

Difference of DFT properties

- Many properties similar to the properties of DTFT and z-transform

- Need careful derivations
  - Due to the finite-length assumption and implicit periodicity

Linearity

- With two finite-length sequences $x_1[n]$ and $x_2[n]$, if

$$x_3[n] = ax_1[n] + bx_2[n]$$

then $X_3[k] = aX_1[k] + bX_2[k]$

- The lengths of $x_1[n]$ and $x_2[n]$ may be different!

Length N_1 Length N_2

- The length of $x_3[n]$ should be $N_3 = \max(N_1, N_2)$
- DFTs $X_1[k]$ and $X_2[k]$ should be computed with the same length $N \geq N_3$
➔ Zero-padding for shorter sequence to have length *N* sequence

Circular shift

- For DTFT, if $x[n] \overset{\mathcal{DTFT}}{\longleftrightarrow} X(e^{j\omega})$, then $x[n-m] \overset{\mathcal{DTFT}}{\longleftrightarrow} e^{-j\omega m}X(e^{j\omega})$
  ➔ Delay in time corresponds to change in phase

- For DFT with finite-length sequence, if $x[n] \overset{\mathcal{DFT}}{\longleftrightarrow} X[k]$, then

$$X_1[k] = e^{-j(2\pi k/N)m}X[k] = W_N^{km}X[k] \overset{\mathcal{DFT}}{\longleftrightarrow} x_1[n] \ ???$$

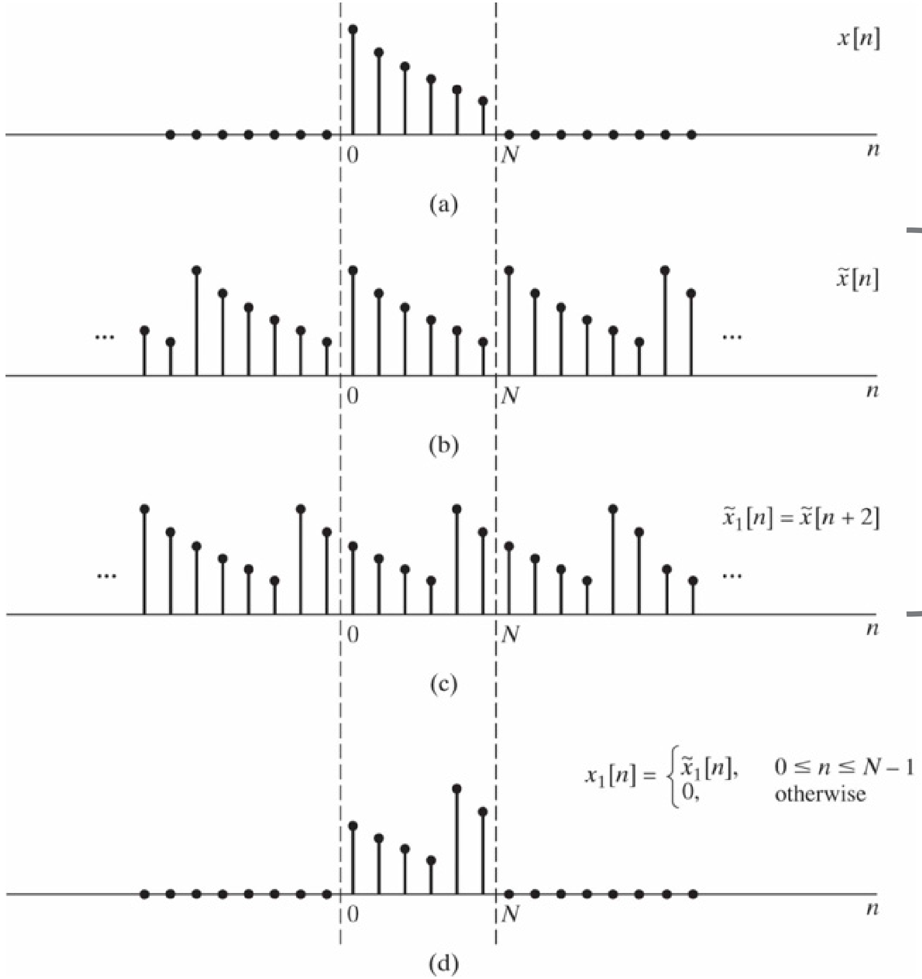
- $x_1[n]$ should be the length N sequence ➔ must be zero outside $0 \leq n \leq N-1$
  ➔ Cannot be a simple time shift of *x*[*n*]

- Correct result $x_1[n] = x[((n-m))_N], \quad 0 \leq n \leq N-1$

Circular shift example

$$N = 6$$

$$m = -2$$



DFS results in Section 8.2

Circular convolution

- If $x_1[n] \overset{\mathcal{DFT}}{\longleftrightarrow} X_1[k]$ and $x_2[n] \overset{\mathcal{DFT}}{\longleftrightarrow} X_2[k]$ both with length *N*

$$X_3[k] = X_1[k]X_2[k] \overset{\mathcal{DFT}}{\longleftrightarrow} x_3[n] = \sum_{m=0}^{N-1} x_1[((m))_N]x_2[((n-m))_N],\ 0 \le n \le N-1$$

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

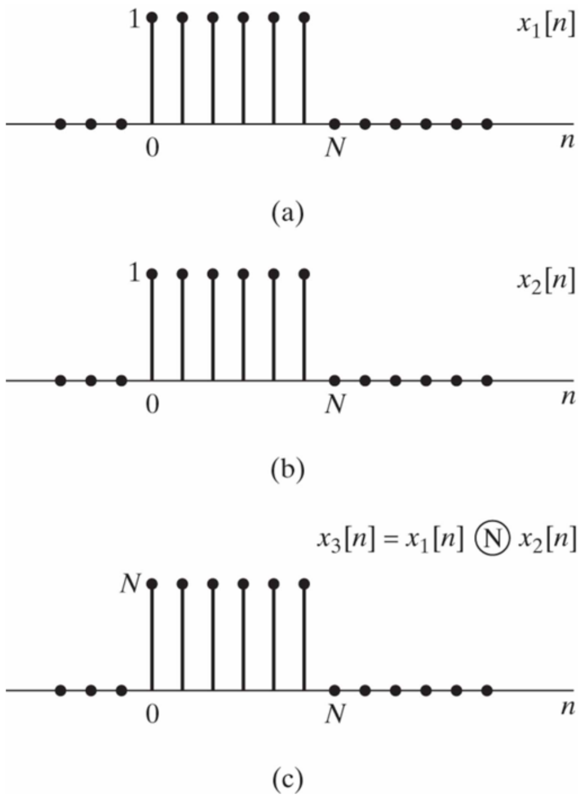
Circularly time reversed and shifted

➔ *N*-point circular convolution

- Define $x_3[n] = x_1[n] \circledN x_2[n]$
- Circular convolution is commutative as linear convolution
- Using duality: $x_1[n]x_2[n] \overset{\mathcal{DFT}}{\longleftrightarrow} \frac{1}{N} X_1[k] \circledN X_2[k]$

Circular convolution of two rectangular pulses

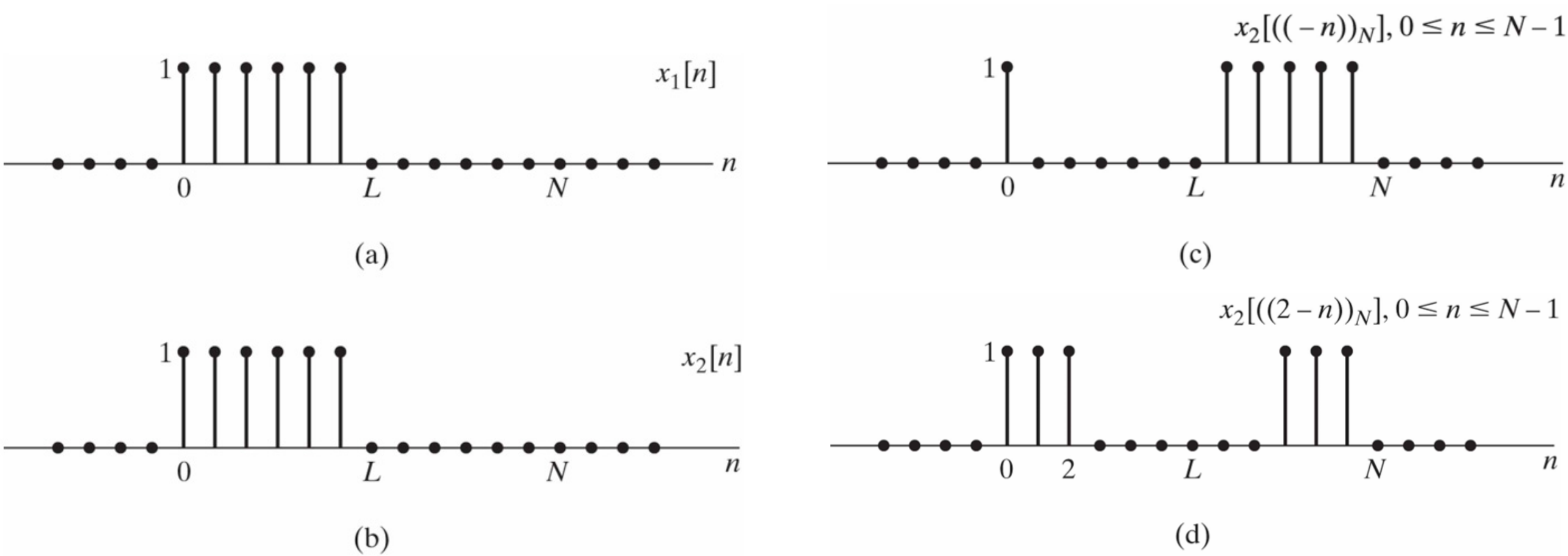
- N-point circular convolution of length N sequences



Circular shift same as original sequence

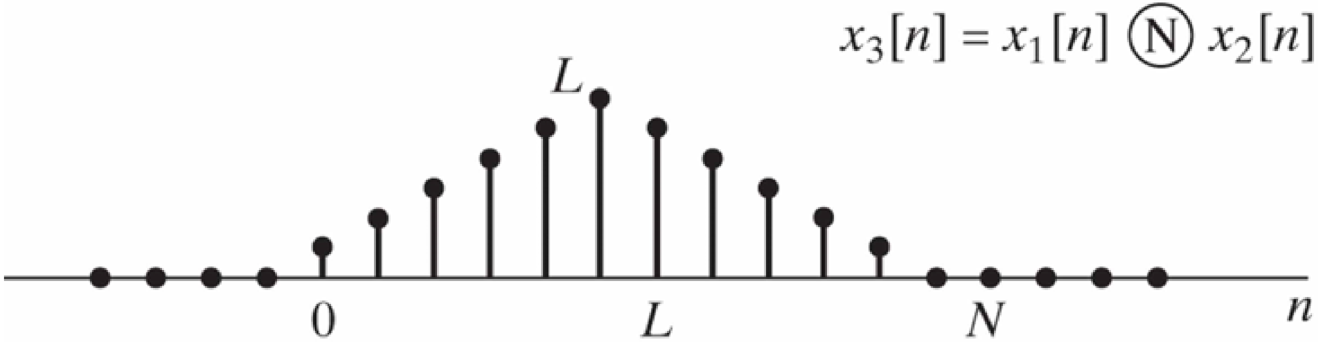
Circular convolution of two rectangular pulses

- N=2L-point circular convolution of length L sequences



Circular convolution of two rectangular pulses

- N=2L-point circular convolution of length L sequences



(e)

➔ Same as linear convolution!

Computing Linear Convolution Using the DFT

Importance of linear convolution

- In many DSP applications, we want linear convolution
  ➔ LTI systems represented with linear convolution
  - Filtering
  - Auto/cross-correlations

- DFT can be efficiently computed using Fast Fourier Transform (FFT)
  - Results in circular convolution, not linear convolution
  - Can we use DFT operations to get linear convolution?
    ➔ Yes!

From DTFT to DFT

- Let $x_3[n] = x_1[n] * x_2[n]$ and $X_3(e^{j\omega}) = X_1(e^{j\omega})X_2(e^{j\omega})$

  Linear convolution

- If we define DFT $X_3[k] = X_3(e^{j(2\pi k/N)})$

$$
\begin{aligned}
X_3[k] &= X_3(e^{j(2\pi k/N)}) \\
&= X_1(e^{j(2\pi k/N)})X_2(e^{j(2\pi k/N)}) \\
&= X_1[k]X_2[k], \quad 0 \le k \le N-1
\end{aligned}
$$

- Inverse DFT of

$$
X_3[k] \overset{\mathcal{DFT}}{\longleftrightarrow} x_1[n] \circledN x_2[n] = x_{3p}[n] = \begin{cases} \sum_{r=-\infty}^{\infty} x_3[n-rN], & 0 \le n \le N-1 \\ 0, & \text{otherwise} \end{cases}
$$

➔ Circular convolution = linear convolution followed by time aliasing!!!

When circular convolution = linear convolution?

- Consider length *L* sequence $x_1[n]$ and length *P* sequence $x_2[n]$

- Linear convolution of the two sequences

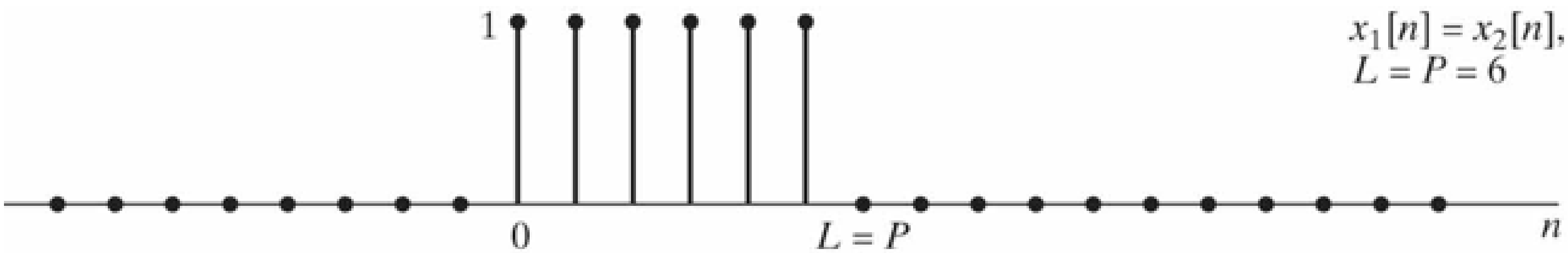
$$x_3[n] = \sum_{m=-\infty}^{\infty} x_1[m]x_2[n-m]$$

  is length *L*+*P*-1 sequence

- With *N*-point DFT where $N \geq L + P - 1$
  ➔ Circular convolution = linear convolution

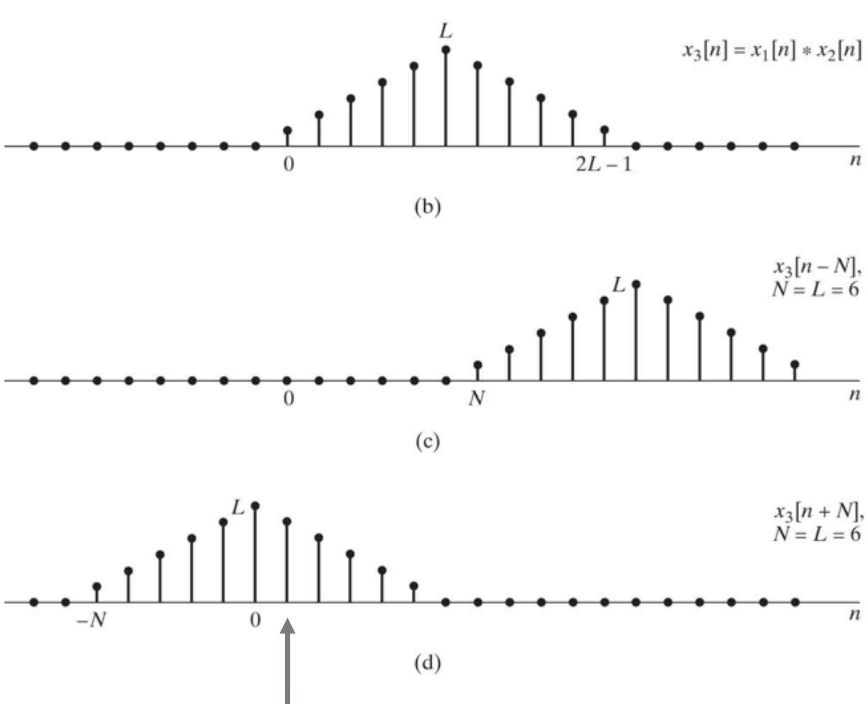
Previous examples

- Consider two sequences

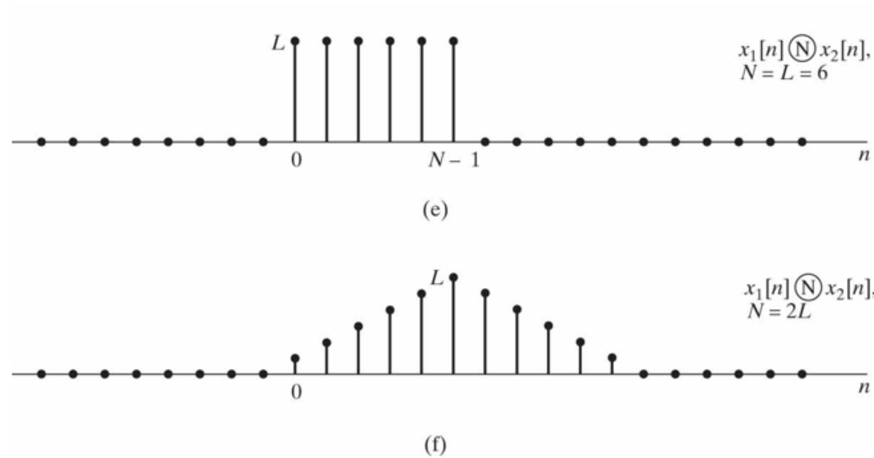


(a)

Previous examples



Correct 0



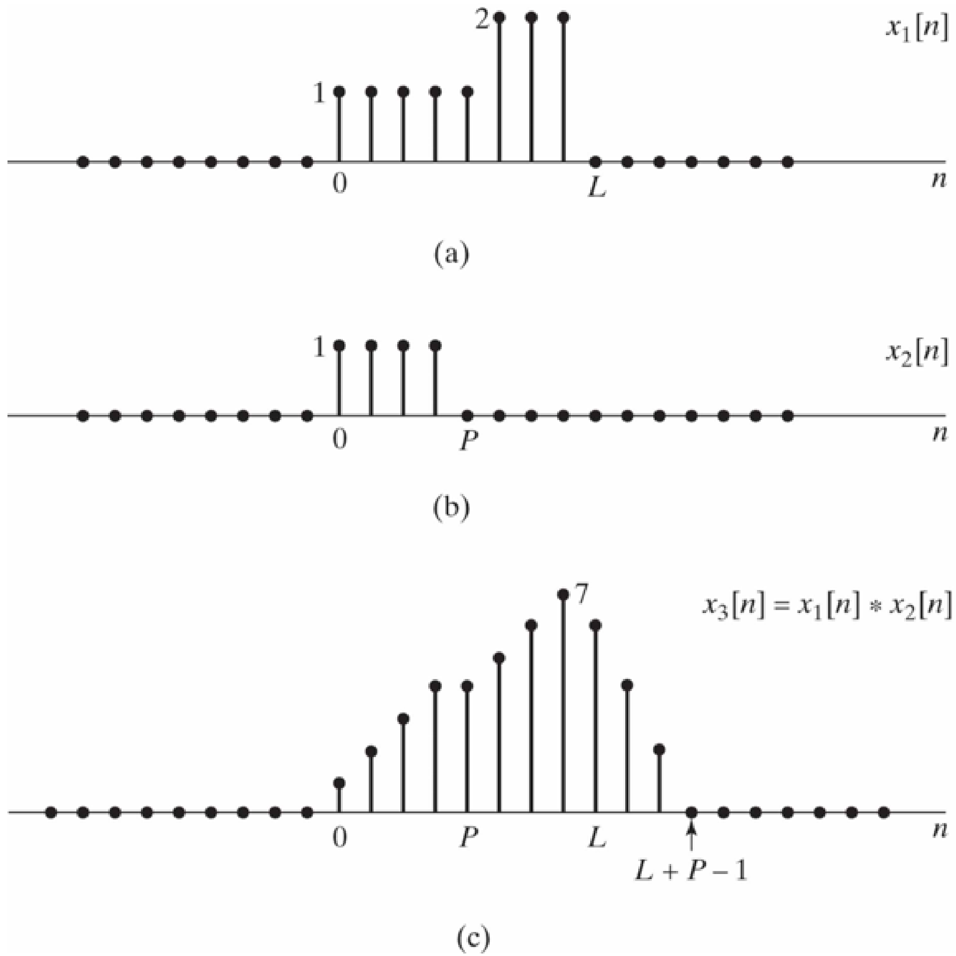
Partial time domain aliasing

- With *L*-point DFT (instead of *N*)

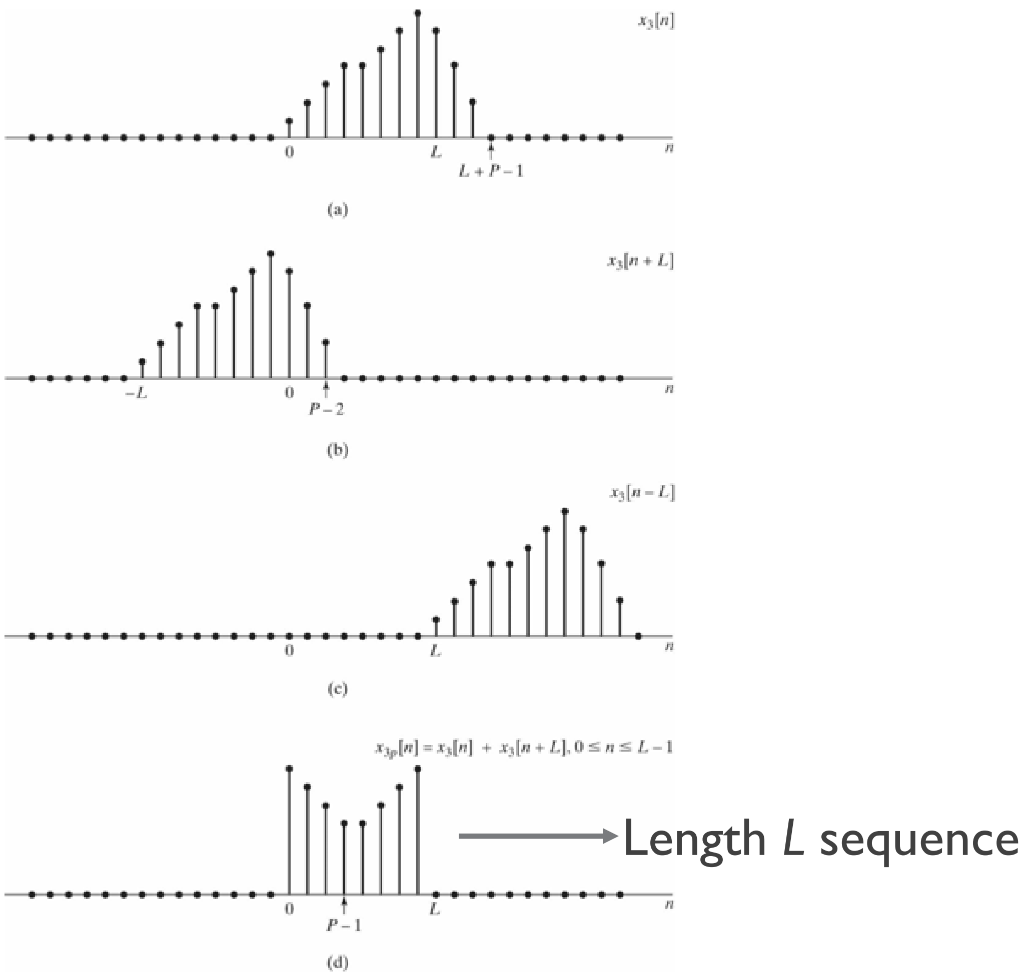
$$x_{3p}[n] = \begin{cases} x_1[n] \text{Ⓛ} x_2[n] = \sum_{r=-\infty}^{\infty} x_3[n - rL], & 0 \leq n \leq L-1 \\ 0, & \text{otherwise} \end{cases}$$

- How does it look like?

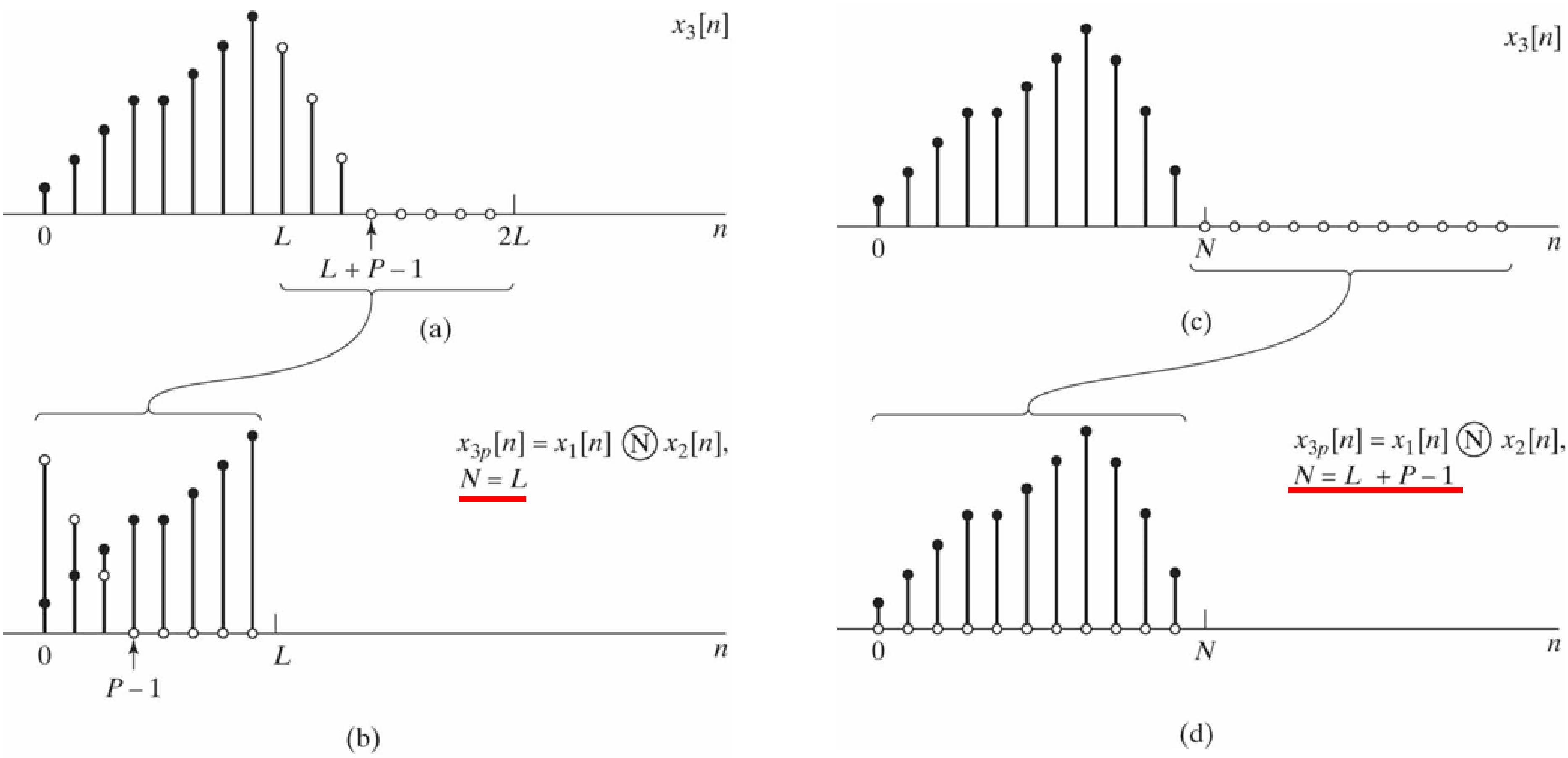
Partial time domain aliasing



Partial time domain aliasing



Partial time domain aliasing – systematic approach

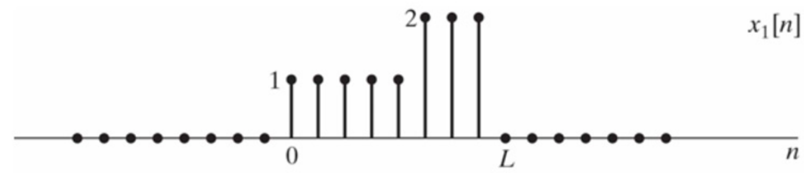


Efficient way to calculate circular convolution

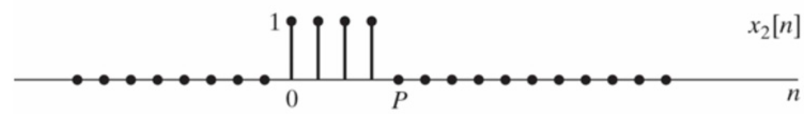
- Consider the two sequences
- Want to have *L*-point circular convolution
- Possible to use circulant matrix

$$x_1[n] = \{1, 1, 1, 1, 1, 2, 2, 2\}$$

$$x_2[n] = \{1, 1, 1, 1, 0, 0, 0, 0\}$$

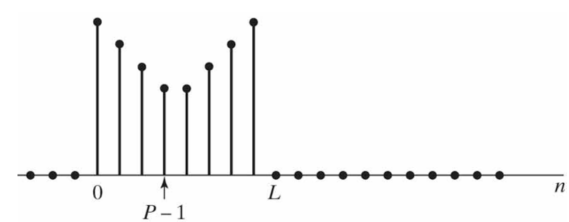


(a)



(b)

$$x_3[n] = x_1[n] \textcircled{L} x_2[n] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 7 \\ 6 \\ 5 \\ 4 \\ 4 \\ 5 \\ 6 \\ 7 \end{bmatrix}$$



POSTECH

Interesting feature of DFT

- Consider $x[n] = e^{j2\pi \frac{k_0}{N} n}\{u[n] - u[n-N]\}$

$$H_N[k_0] = H(e^{j\omega})\,|_{\omega = 2\pi\frac{k_0}{N}}$$

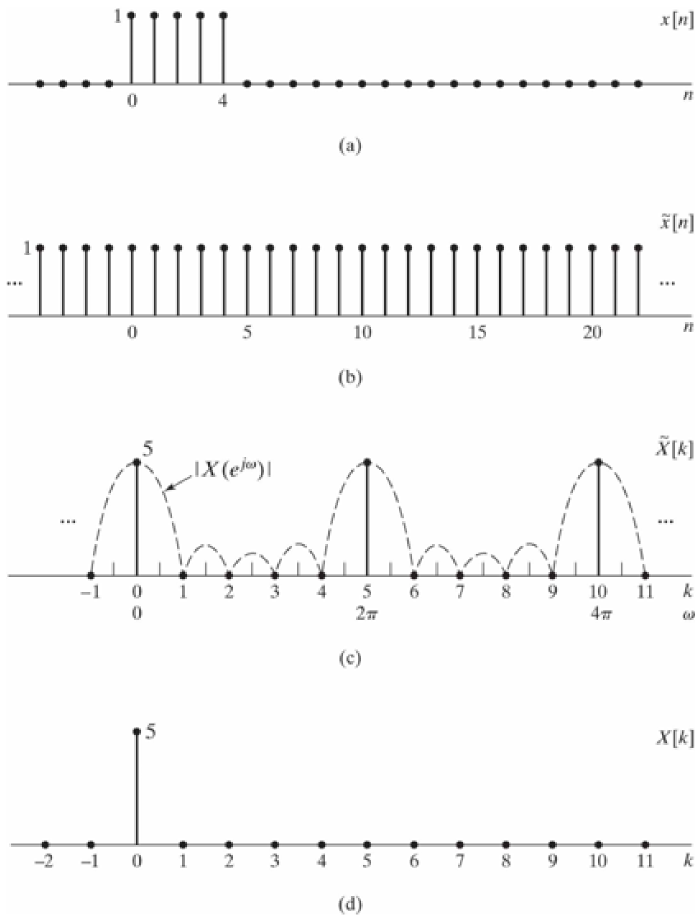
- N-point DFT gives $x[n] \overset{\mathcal{DFT}}{\longleftrightarrow} X_N[k] = N\delta[k-k_0]$
- Consider $h[n]$ of length $M \leq N$ and its N-point DFT $H_N[k]$
- Note $Y_N[k] = X_N[k]H_N[k] = NH_N[k_0]\delta[k-k_0] \overset{\mathcal{DFT}}{\longleftrightarrow} y_p[n] = H_N[k_0]x[n]$
  - In other words

$$y_p[n] = \left(e^{j2\pi\frac{k_0}{N}n}\{u[n]-u[n-N]\}\right) Ⓝ h[n] = H_N[k_0]\left(e^{j2\pi\frac{k_0}{N}n}\{u[n]-u[n-N]\}\right)$$

which mimics $y[n] = e^{j\omega_0 n} * h[n] = H(e^{j\omega_0})e^{j\omega_0 n}$

Linear convolution

Example (8.7 in the textbook)



Interesting feature of DFT

- Because $x[n]$ is length N and $h[n]$ is length $M<N$, the linear convolution

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

is length $N+M-1$ sequence
➔ Time-domain aliasing occurs in $y_p[n] = h[n] \circledN x[n]$

- But these aliased M-1 points amazingly yield "good" points to have

$$y_p[n] = \left(e^{j2\pi\frac{k_0}{N}n}\{u[n]-u[n-N]\}\right) \circledN h[n] = H_N[k_0]\left(e^{j2\pi\frac{k_0}{N}n}\{u[n]-u[n-N]\}\right)$$

Interesting feature of DFT

- Because DFT operation is linear (all DFT operations below are *N*-point DFT)

$$x[n] = \sum_{\ell=0}^{N-1} \alpha_\ell e^{j2\pi \frac{\ell}{N} n} \{u[n] - u[n-N]\} \overset{\mathcal{DFT}}{\longleftrightarrow} X_N[k]$$

$$h[n] \overset{\mathcal{DFT}}{\longleftrightarrow} H_N[k]$$

Length $M \leq N$

$$Y_N[k] = H_N[k] X_N[k] \overset{\mathcal{DFT}}{\longleftrightarrow} y_N[n] = \sum_{\ell=0}^{N-1} \alpha_\ell H_N[\ell] e^{j2\pi \frac{\ell}{N} n} \{u[n] - u[n-N]\}$$

Implementing LTI Systems Using the DFT

LTI systems and DFT

- LTI systems characterized by impulse response and linear convolution

- DFT can be
  - efficiently implemented using FFT
  - used to implement linear convolution with N-point DFT with $N \geq L + P - 1$
    ➔ Both $x[n]$ and $h[n]$ must be zero-padded to become length N sequences

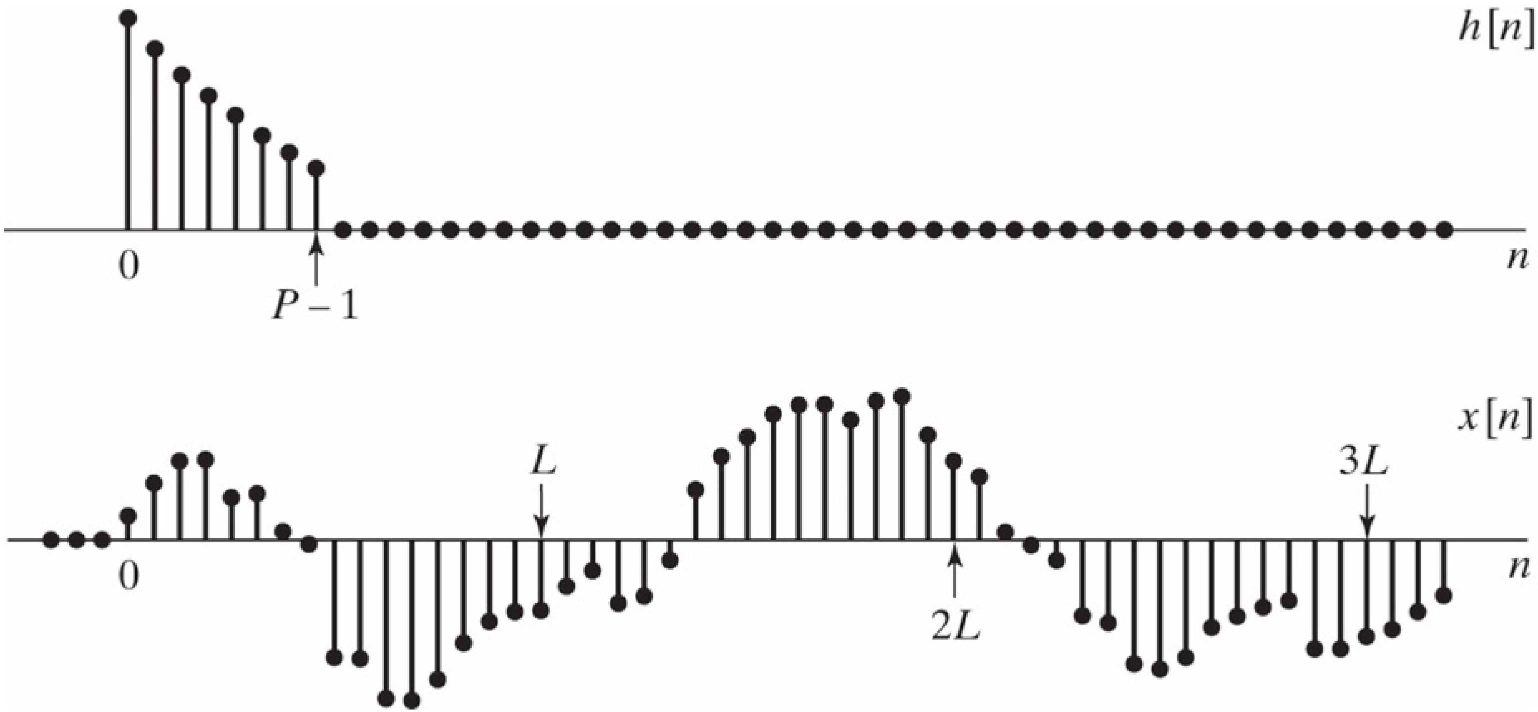
- Input sequence $x[n]$ can have very large number of samples
  - Impractical to compute DFT when N too large or
  - All the samples should be collected to computed N-point DFT
    ➔ Cause large delay

Block convolution

- Solution for both problems
- The sequence $x[n]$ to be filtered is segmented into sections of length L
- Each section can be convolved with finite-length impulse response
- Filtered section then can be combined in a proper way
- Linear filtering of each block implemented using DFT
- **Overlap-add method** vs. overlap-save method

Overlap-add method

- Consider two sequences

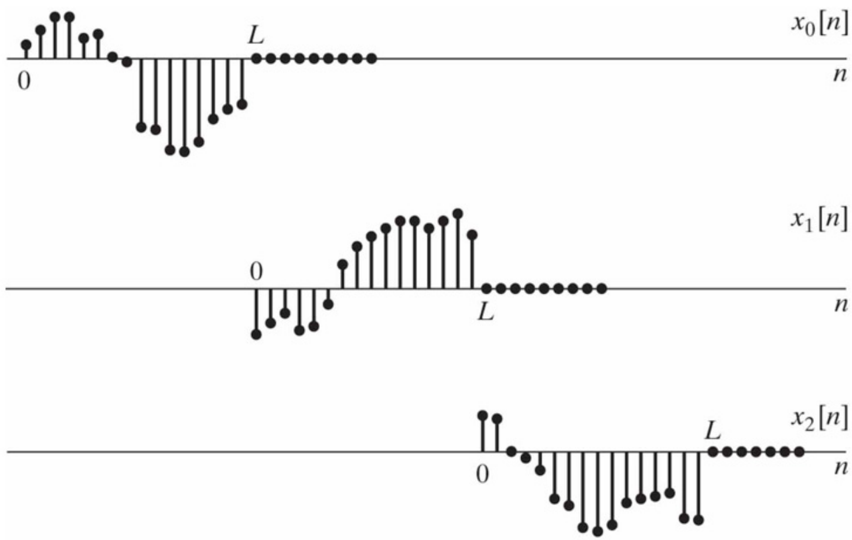


Overlap-add method

- Represent x[n] as a sum of shifted nonoverlapping finite-length segments of length L

$$x[n] = \sum_{r=0}^{\infty} x_r[n - rL] \quad \text{where} \quad x_r[n] = \begin{cases} x[n+rL], & 0 \le n \le L-1 \\ 0 & \text{otherwise} \end{cases}$$

Redefining time origin



(a)

Overlap-add method

- After filling

$$y[n] = x[n] * h[n] = \sum_{r=0}^{\infty} y_r[n - rL] \quad \text{where} \quad y_r[n] = x_r[n] * h[n]$$

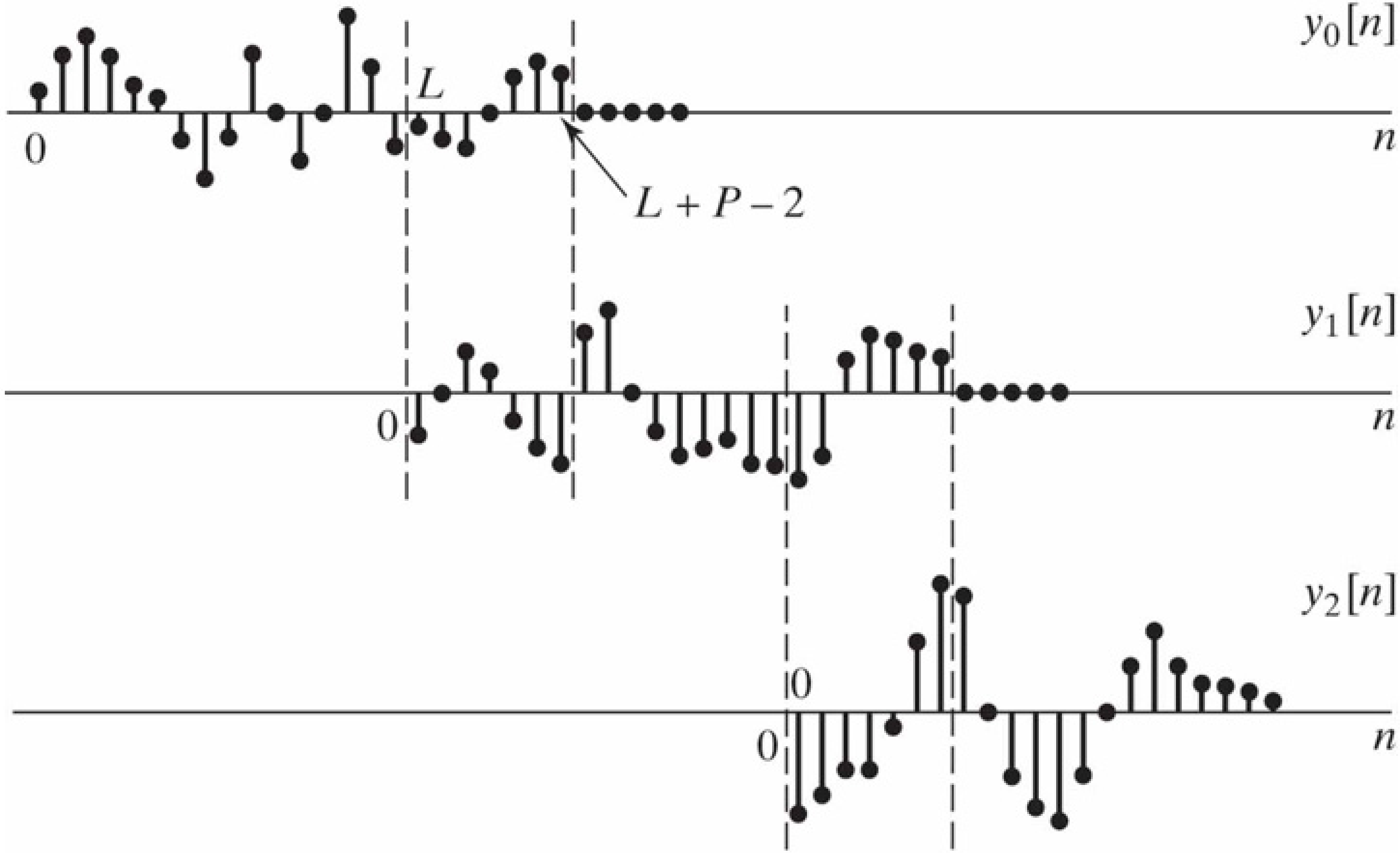
Length $L+P-1$

- Each filtered segment obtained with $N \geq L + P - 1$-point DFT

- Each filtered segment then time-shifted and added to obtain $y[n]$

POSTECH

Overlap-add method



Matlab Programming

Circular time-shifting property of DFT

```
% Circular time-shifting property of DFT
clc;clf;

x=0:2:16;
N=length(x);
n=0:N-1;

y=circshift(x,5,2);
x
y
XF=fft(x);
YF=fft(y);

subplot(2,2,1);stem(n,abs(XF));grid;
title('Magnitude of DFT of original sequence');
subplot(2,2,2);stem(n,abs(YF));grid;
title('Magnitude of DFT of circularly shifted sequence');
subplot(2,2,3);stem(n,angle(XF));grid;
title('Phase of DFT of original sequence');
subplot(2,2,4);stem(n,angle(YF));grid;
title('Phase of DFT of circularly shifted sequence');
```

Circular convolution property of DFT

```
%Circular convoultion property of DFT
clc;

g1=1:6;
g2=[1 -2 3 3 -2 1];
ylin=conv(g1,g2)
ycir=cconv(g1,g2)

G1=fft(g1);
G2=fft(g2);
yc=ifft(G1.*G2)
```

Textbook homework

- Problems in textbook: 8.29 (typo in (b): seven-point DFT => five-point DFT), 8.30, 8.34, 8.44 (total 4 problems)

MATLAB homework 1

- Due 12/04 (Tuesday) by 1:50pm
- Implement circular convolution
  - Implement a function with 3 inputs (have proper annotations!!!)

    function y=circonv(x1,x2,N) ➔ Use the name "circonv" !!! (unless 50% loss)
    - x1, x2: **arbitrary** length real sequences
    - N: length of output y
  - Implement circular convolution WITHOUT using fft, ifft, conv. (no point with these)
    - You many use circshift if needed
  - The output y should be the same with the result from cconv(x1,x2,N)
    - Should work for all lengths of x1 and x2 and arbitrary N!!!
  - Use main.m file in the next slide for evaluation

MATLAB homework 1

- Have main.m file as follows. We will change N value randomly

```
clear all

N = 300;

x1=randn(1,58); x2=randn(1,198);

y_circonv=circonv(x1,x2,N);
y_cconv=cconv(x1,x2,N);

figure(1)
plot(1:N,y_circonv,'b-o')
hold on
plot(1:N,y_cconv,'r-x')
%==========================
x1=randn(1,117); x2=randn(1,33);

y_circonv=circonv(x1,x2,N);
y_cconv=cconv(x1,x2,N);

figure(2)
plot(1:N,y_circonv,'b-o')
hold on
plot(1:N,y_cconv,'r-x')
%==========================
x1=randn(1,11); x2=randn(1,237);

y_circonv=circonv(x1,x2,N);
y_cconv=cconv(x1,x2,N);

figure(3)
plot(1:N,y_circonv,'b-o')
hold on
plot(1:N,y_cconv,'r-x')
```

MATLAB homework 2

- Implement overlap-add method
- Generate two sequences x=randn(1,10), y=randn(1,10000)
- Compare
  - Linear convolution
  - Use (10+10000-1)-point DFT/IDFT to obtain linear convolution
  - Use 1024-point DFT/IDFT with overlap-add method using proper segment length to obtain linear convolution
- Compare plots of three results
- All results must be the same

DFT (FFT) Applications

DFT applications

- Short list from Wikipedia
  - Spectral analysis
  - Filter bank
  - Data compression
  - Partial differential equations
  - Multiplication of large integers
  - Convolution
  - ...

- We will briefly discuss 'spectral analysis' and 'digital subbanding'

Notch filters (bandstop filter with narrow stopband)

- Want to get rid of frequency component at ω_0

- z-transform representation of general notch filters

$$H_{\text{notch}}(z) = \frac{G(z - e^{j\omega_0})(z - e^{-j\omega_0})}{(z - re^{j\omega_0})(z - re^{-j\omega_0})}$$

- Clearly, $H_{\text{notch}}(e^{j\omega_0}) = H_{\text{notch}}(e^{-j\omega_0}) = 0$

- The difference equation for notch filter

$$y[n] = 2r\cos(\omega_0)y[n-1] - r^2 y[n-2] + Gx[n] - G2\cos(\omega_0)x[n-1] + Gx[n-2]$$

Matlab example of notch filter

```
%Shows how simple difference equation can remove a tone
%that's corrupting a speech utterance.
clf
clear all
%these commands read in the speech file: need getspeech.m
datar=getspeech('woman_voice.wav');
Fs=12500;
%plot data to cut off silence
plot(datar)
[d,dsize]=size(datar);
input('play back utterance at 12.5 KHz sampling rate');
soundsc(datar,Fs)
input('add tone at 3.125 KHz to utterance and play back');
omega_noise=pi/2;
nc=1:dsize;
x=datar+500*cos(omega_noise*nc);
Figure(2)
plot(x)
soundsc(x,Fs)
%define coefficients for second-order notch filter
r=0.95;
omega0=pi/2;
input('run tone corrupted speech through simple second order difference equation');
y(1)=0; y(2)=0;
for n=3:dsize
y(n)=2*r*cos(omega0)*y(n-1)-r^2*y(n-2)+x(n)-2*cos(omega0)*x(n-1)+x(n-2);
end
input('play back output of difference eqn.');
soundsc(y,Fs)
```

Spectral analysis

- Check out notcheg_r2.m file

- Questions
  - What is the difference between the DFT plots?
  - What is the right value for the DFT size N?
    - Notch filter is IIR! Cannot use $N>L+P-1$ argument.

- https://stackoverflow.com/questions/20108462/matlab-filter-in-the-frequency-domain-using-fft-ifft-with-an-iir-filter/40833174#40833174