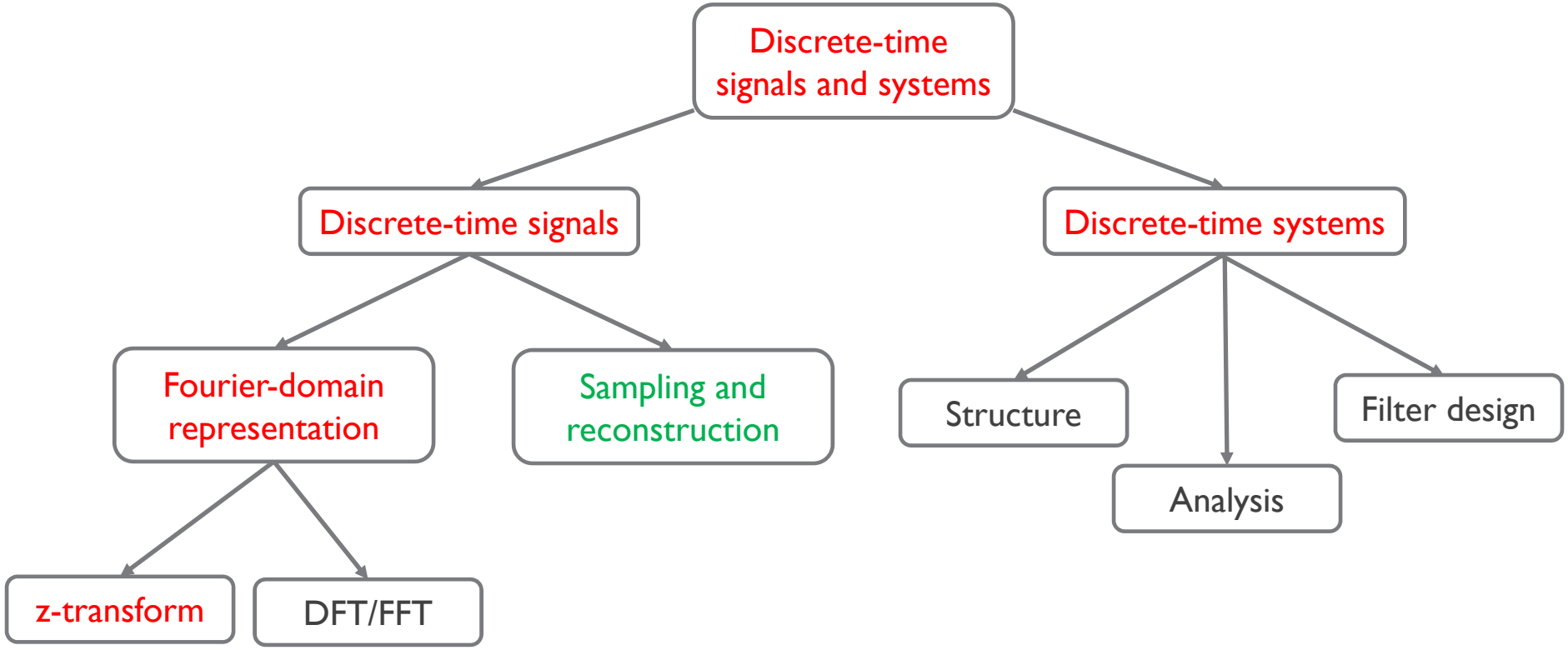
Digital Signal Processing

POSTECH

Department of Electrical Engineering

Junil Choi

Course at glance



Homework

- Due: 11/2 Friday midnight (11:59pm)
- Send one m file with proper (annotated) answers
- Implement ya using "for loop" without vectorization

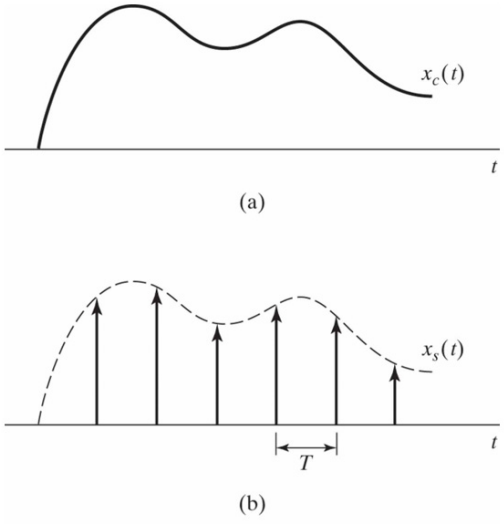
```
clear all; clf;

Ts=1/14;
f1=7; f2=3;            %(1) Why Ts should be smaller than 1/14 with these f1 and f2?

t=linspace(-0.5,1.5,500)';     %Time interval 2/500 to mimic continuous time. Try different numbers.
x_continuous=cos(2*pi*f1*t)+sin(2*pi*f2*t);

n=(-50:Ts:50)';            %(2) Why is n from -50 to 50, not from -0.5 to 1.5 as t?
% n=(-0.5:Ts:1.5)';

xs=cos(2*pi*f1*n)+sin(2*pi*f2*n);

ya=sinc((1/Ts)*t(:,ones(size(n)))-(1/Ts)*n(:,ones(size(t)))')*xs;
%(3) Explain the meaning of t(:,ones(size(n)))
%(4) Explain the meaning of n(:,ones(size(t)))'

plot(t,x_continuous,'k--');hold on;plot(n,xs,'o');plot(t,ya);grid;
xlabel('Time, msec'); ylabel('Amplitude');
title('Reconstructed continuous-time signal y_{a}(t)');
legend('Unsampled signal','Sampled sequence','Reconstructed signal')
axis([0 1 -2.5 2.5])
```

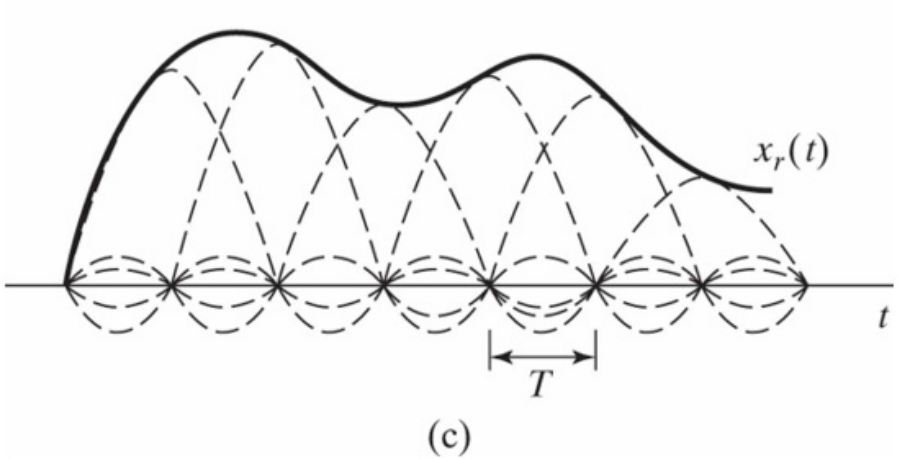
Mathematical expression of reconstruction

- Reconstructed signal becomes

$$x_r(t) = \sum_{n=-\infty}^{\infty} x[n]h(t-nT) = \sum_{n=-\infty}^{\infty} x[n]\frac{\sin[\pi(t-nT)/T]}{\pi(t-nT)/T}$$

➔ Is this the same as $x_c(t)$?





Upsampling

Increasing sampling rate by integer factor

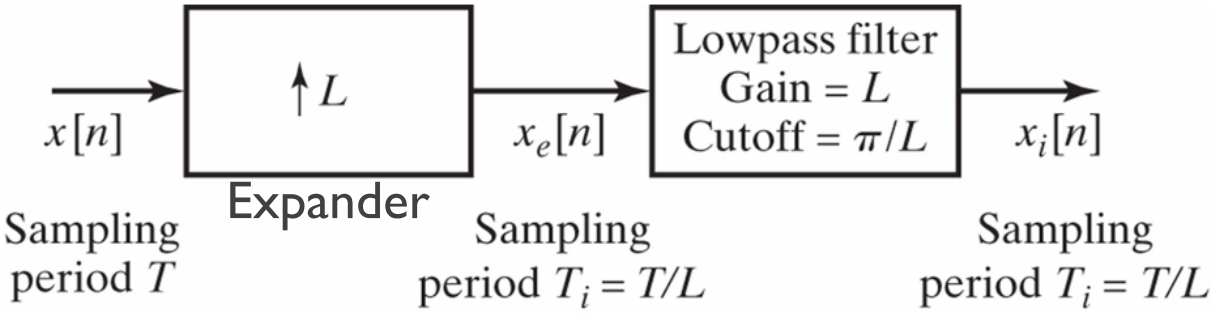
- Usually called "upsampling"
  - Downsampling ➔ analogous to sampling a CT signal
  - Upsampling ➔ analogous to D/C conversion

- Want to increase the sampling rate of x[n] by a factor of *L*

  - Obtain $x_i[n] = x_c(nT_i), \text{ where } T_i = T/L$

    from $x[n] = x_c(nT)$

Upsampling procedure



- It is obvious that $x_i[n] = x[n/L] = x_c(nT/L), \quad n = 0, \pm L, \pm 2L, \ldots$
- The output from the expander is

$$x_e[n] = \begin{cases} x[n/L], & n = 0, \pm L, \pm 2L, \ldots \\ 0, & \text{otherwise} \end{cases}$$

$$= \sum_{k=-\infty}^{\infty} x[k]\delta[n - kL]$$

How does $x_e[n]$ look like?

- The lowpass filter plays a role similar to the ideal D/C converter

Frequency-domain representation

- The Fourier transform of $x_e[n]$ is

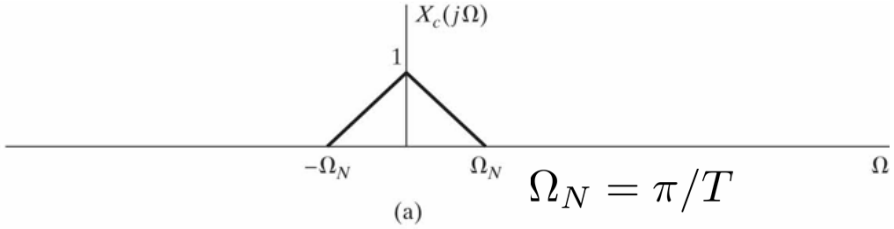
$$X_e(e^{j\omega}) = \sum_{n=-\infty}^{\infty} \left(\sum_{k=-\infty}^{\infty} x[k]\delta[n-kL] \right) e^{-j\omega n}$$

$$= \sum_{k=-\infty}^{\infty} x[k]e^{-j\omega Lk} = X(e^{j\omega L})$$

Frequency-scaled version of x[n]

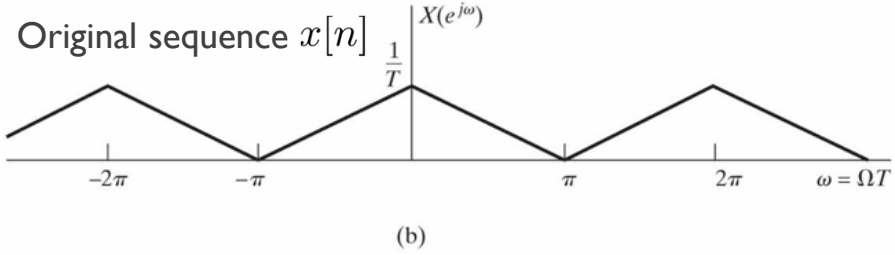
ω replaced by ωL

Frequency-domain representation

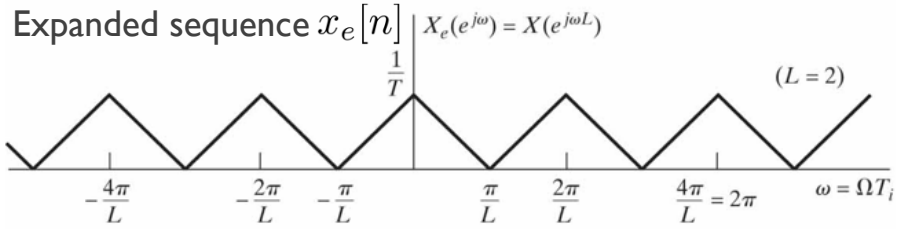


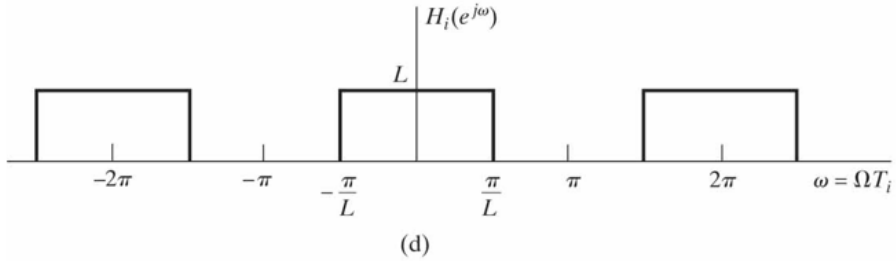
$\Omega_N = \pi/T$

Original sequence $x[n]$

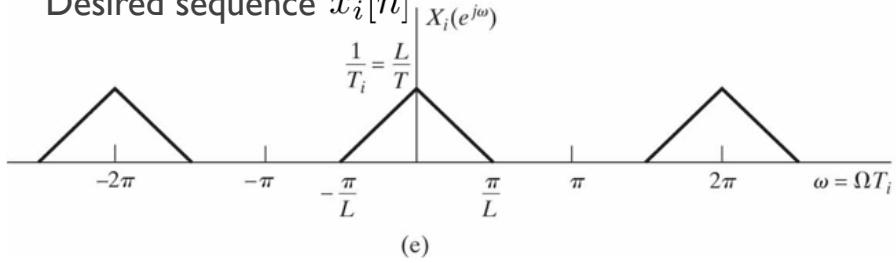


Expanded sequence $x_e[n]$





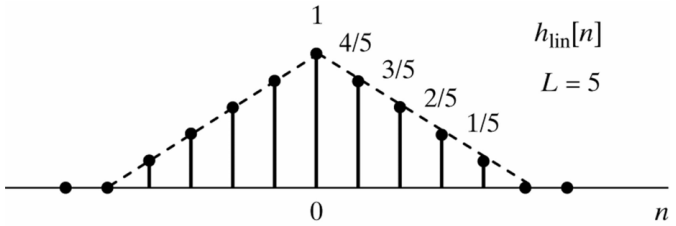
Desired sequence $x_i[n]$



- Lowpass filter removes *L* replicas
- Need to have a gain of *L*

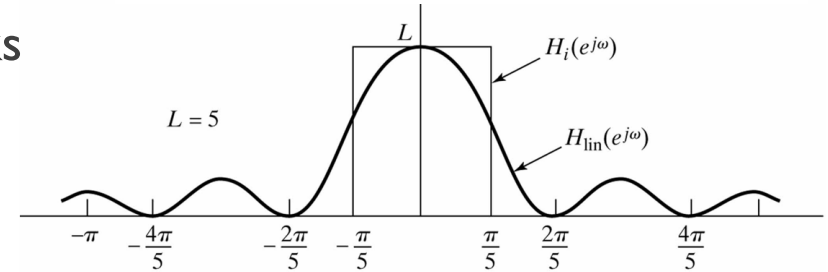
Practical linear interpolation

- Ideal lowpass filter is not possible in practice
  - Very good approximations are possible though

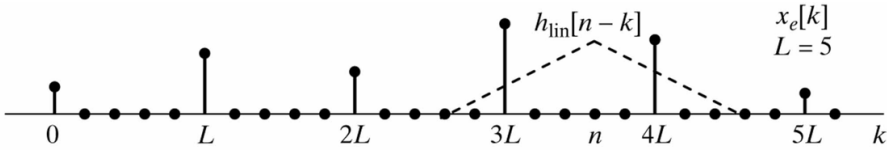


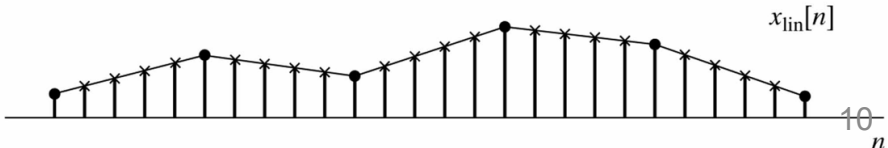
- Very simple linear interpolation also works

$$h_{\text{lin}}[n] \begin{cases} 1 - |n|/L, & |n| \le L \\ 0, & \text{otherwise} \end{cases}$$



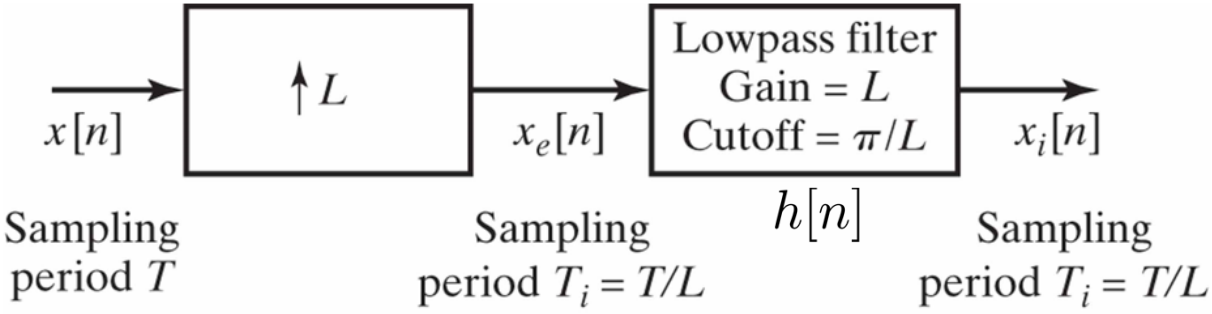
$$H_{\text{lin}}(e^{j\omega}) = \frac{1}{L}\left[\frac{\sin(\omega L/2)}{\sin(\omega/2)}\right]^2$$





Efficient implementation of upsampling

- Recall



$$x_e[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-kL] \qquad x_i[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-kL] * h[n]$$

$$= \sum_{k=-\infty}^{\infty} x[k]h[n-kL]$$

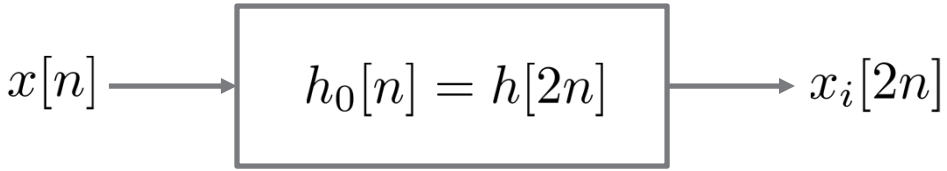
- Assume L=2 for simply illustration

$$x_i[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-2k]$$

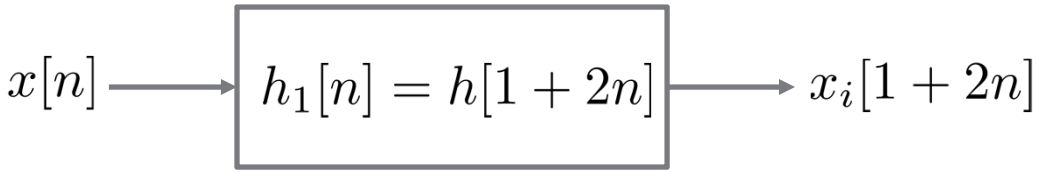
Efficient implementation of upsampling

- Consider $x_i[2n]$ and $x_i[2n+1]$ separately

$$x_i[2n] = \sum_{k=-\infty}^{\infty} x[k]h[2n-2k] = x[n] * h_0[n], \text{ with } h_0[n] = h[2n]$$

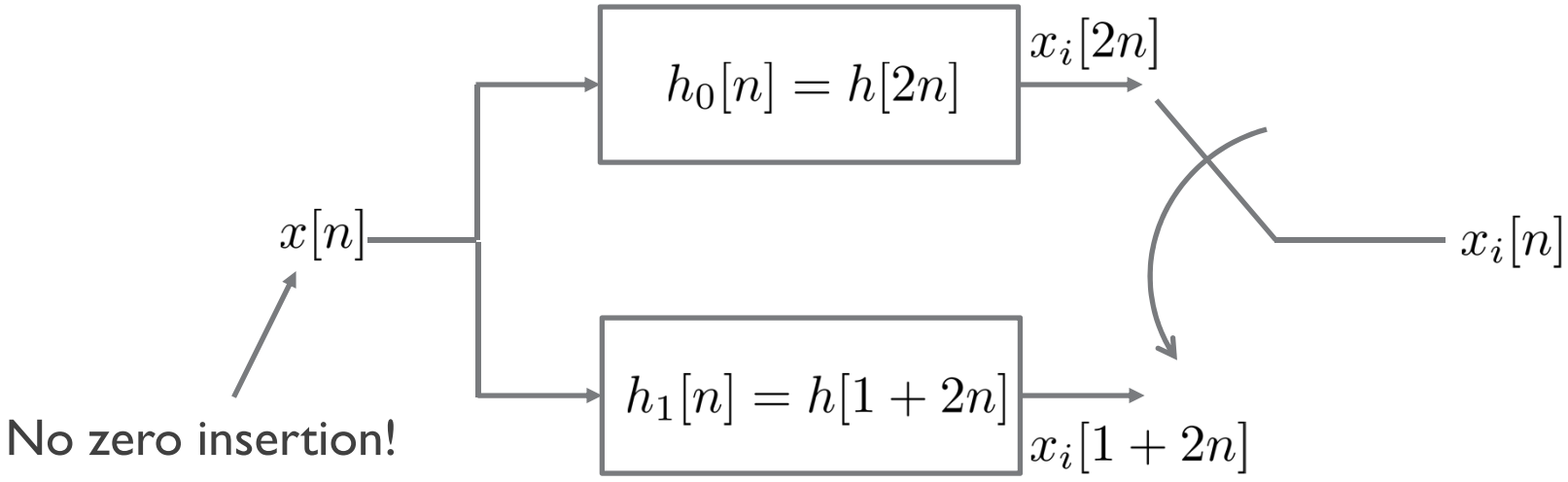


$$x_i[1+2n] = \sum_{k=-\infty}^{\infty} x[k]h[1+2n-2k] = x[n] * h_1[n], \text{ with } h_1[n] = h[1+2n]$$



Efficient implementation of upsampling

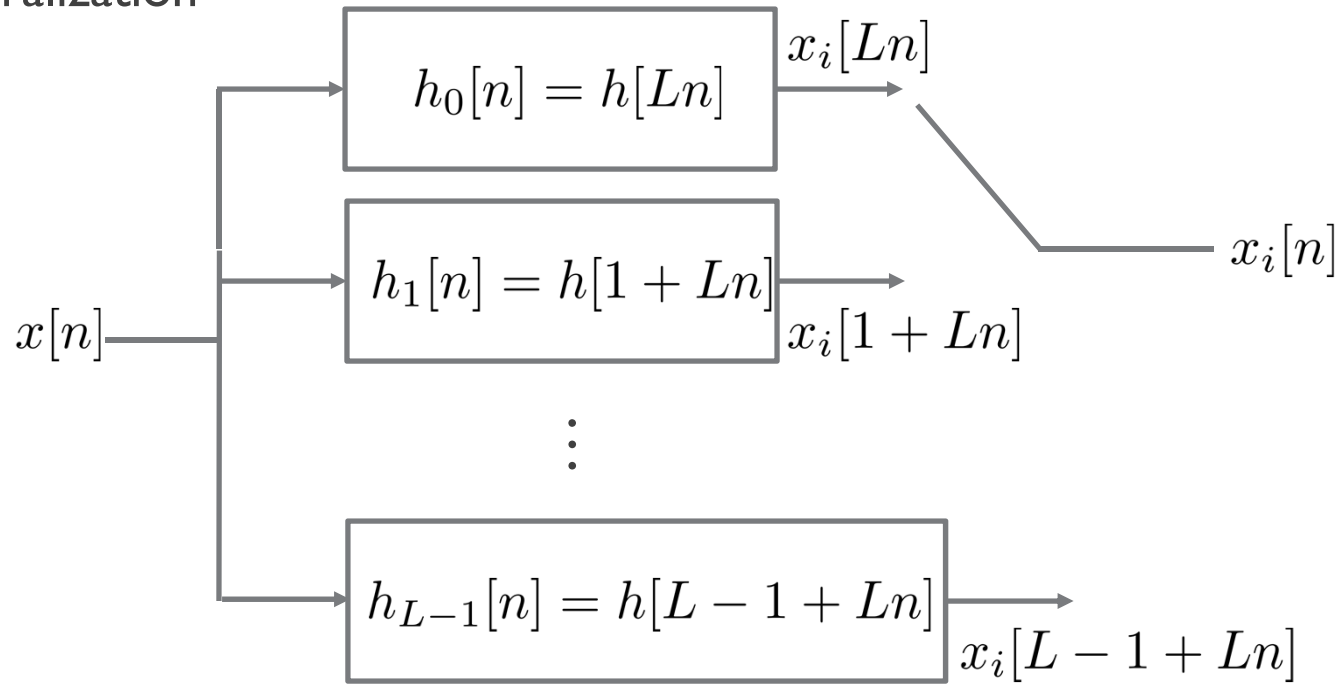
- Overall system



- "Commutator" operates at twice original sampling rate

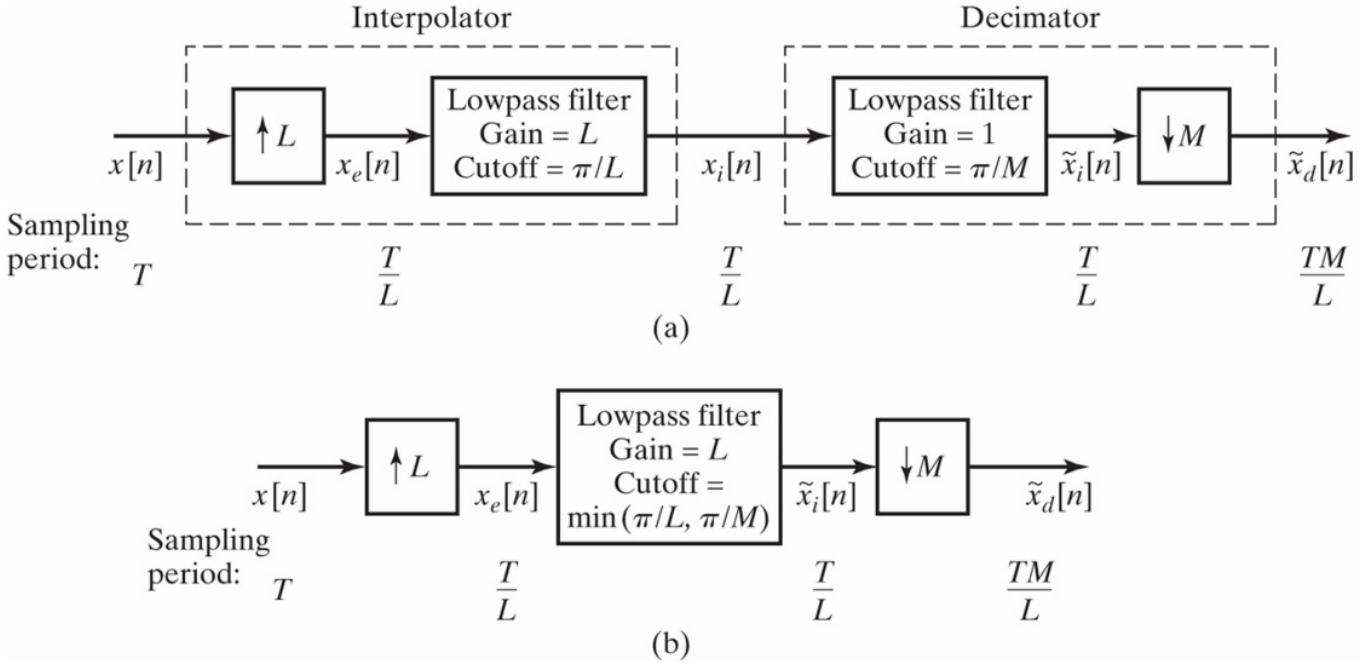
Efficient implementation of upsampling

- Generalization



- "Commutator" operates at $L \cdot F_s$

Changing sampling rate by a noninteger factor

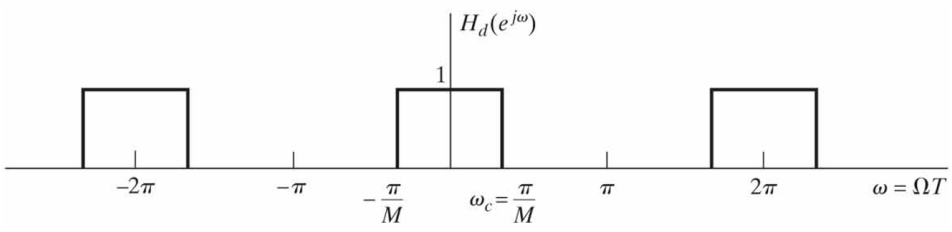


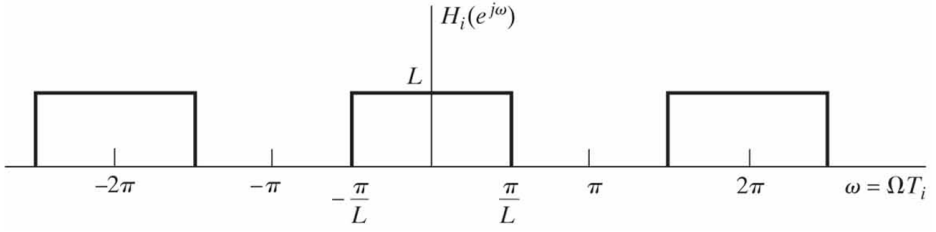
- Combine interpolator and decimator
  - The order of two systems is important!
- Change sampling rate by a rational factor *L*/*M*
  = Change sampling period by a rational factor *M*/*L*

Multirate Signal Processing

Changing sampling rate possible

- Use downsampling/upsampling to reduce/increase the sampling rate
  - Both systems need lowpass filters
- With large changing factors *M* and *L*, we need narrowband lowpass filters

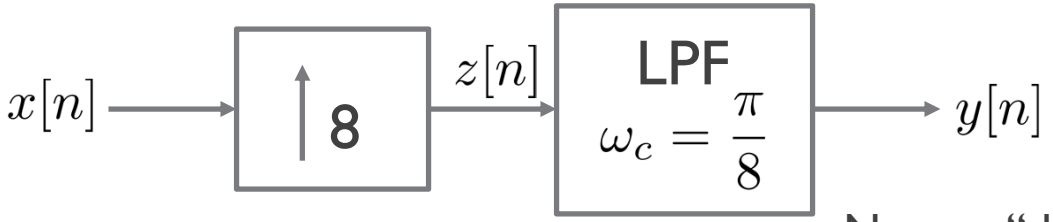


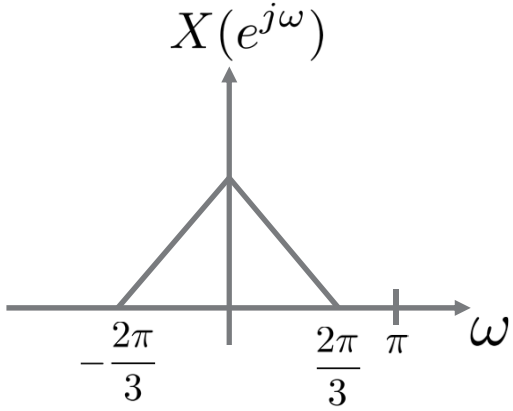


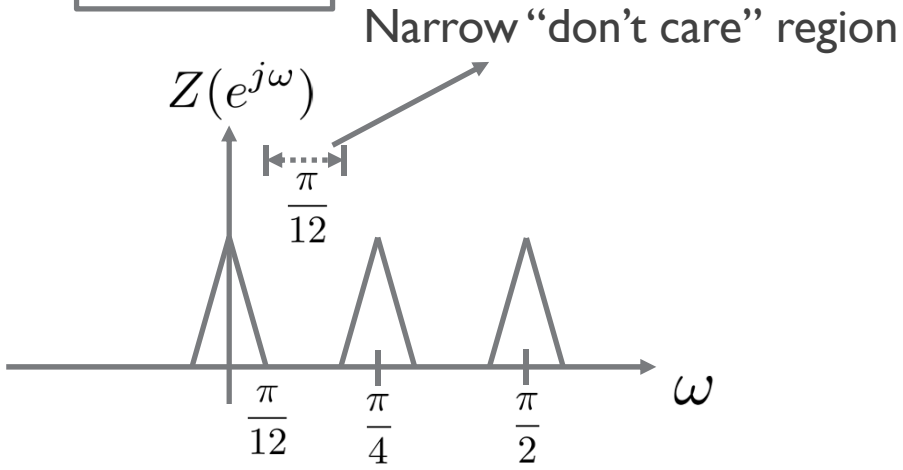
- Hard to implement narrowband lowpass filters with sharp cutoff frequency
  - Requires "long" FIR filter
  - Compare ideal lowpass filter and linear interpolation filter
- How to reduce the complexity of resampling for e.g., *M*=101 and *L*=100?

Example of multistage interpolation

- Consider 4kHz bandwidth speech sampled at 12kHz
- Want to implement a system

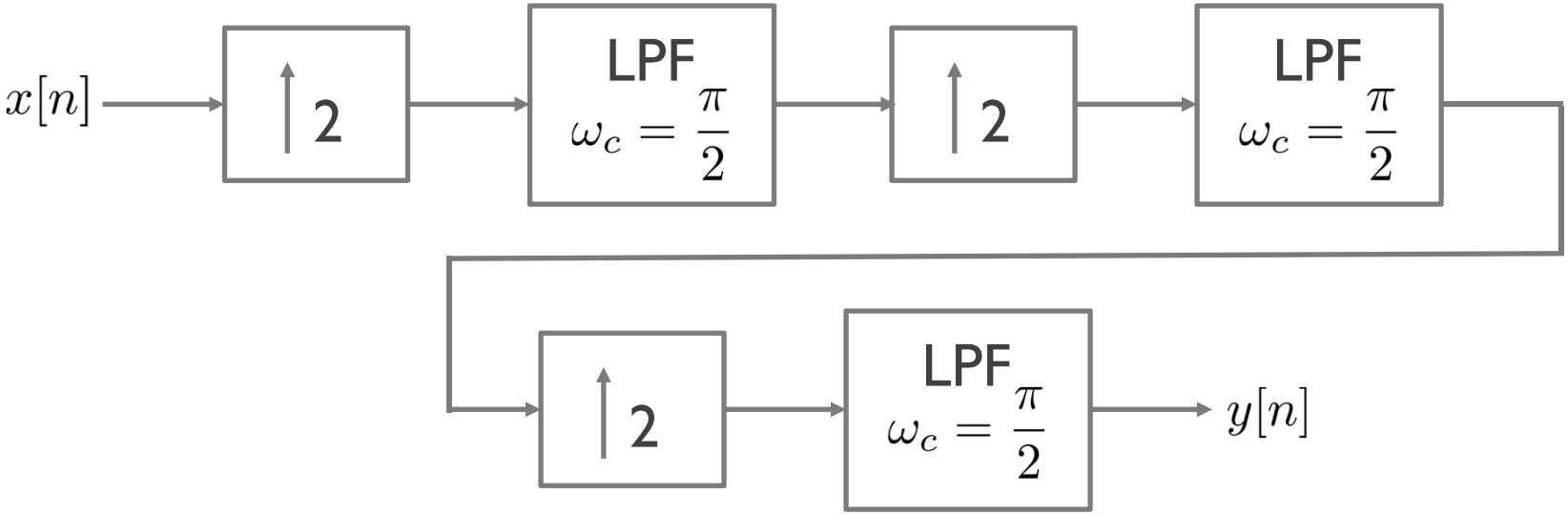






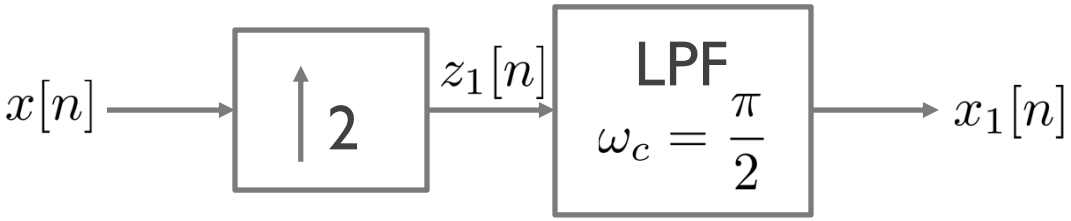
Practical example of multistage interpolation

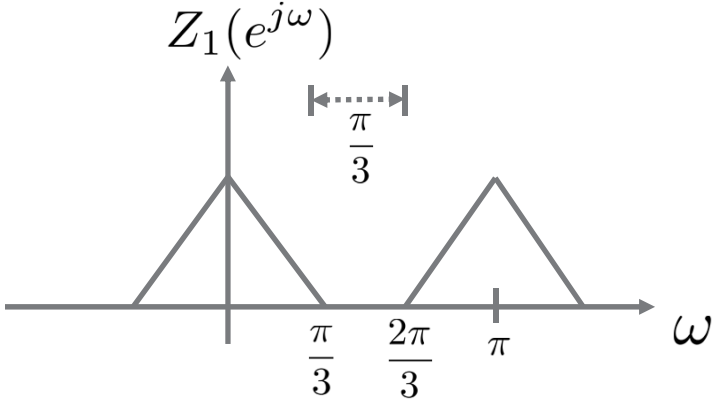
- Instead, consider implementing in 3 stages



Practical example of multistage interpolation

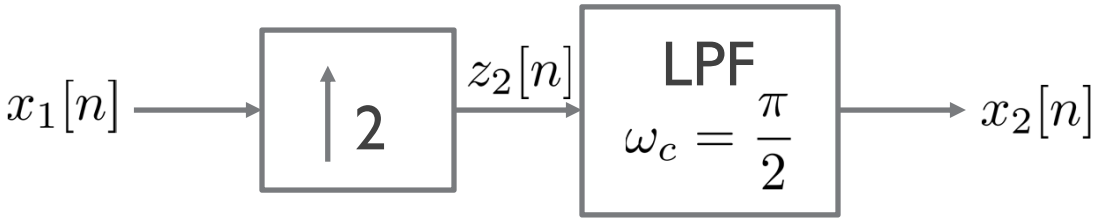
- Stage 1

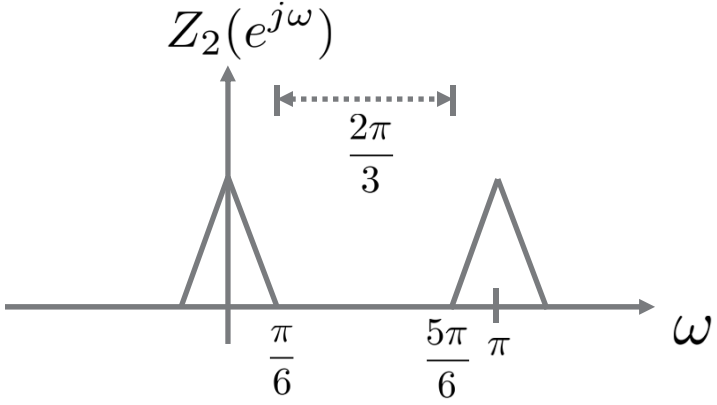




Practical example of multistage interpolation

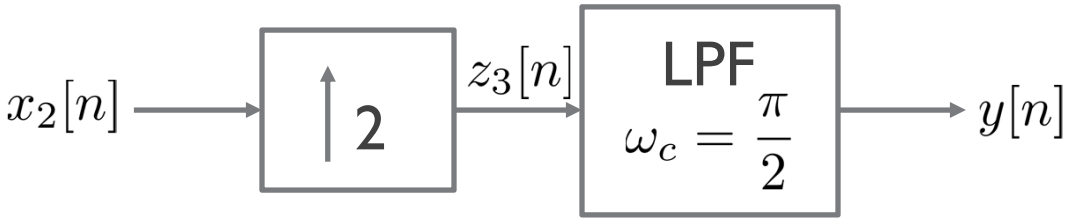
- Stage 2

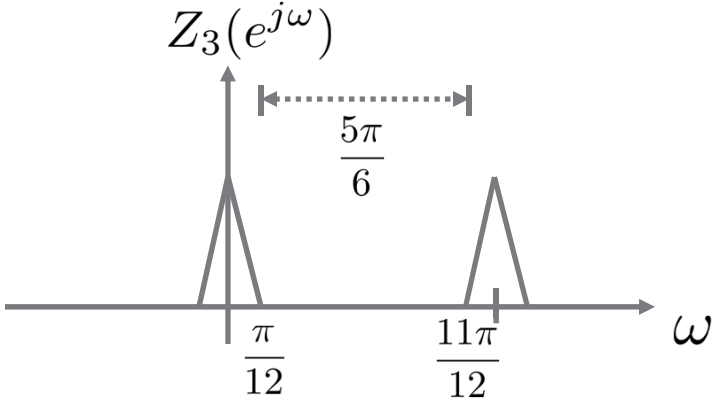




Practical example of multistage interpolation

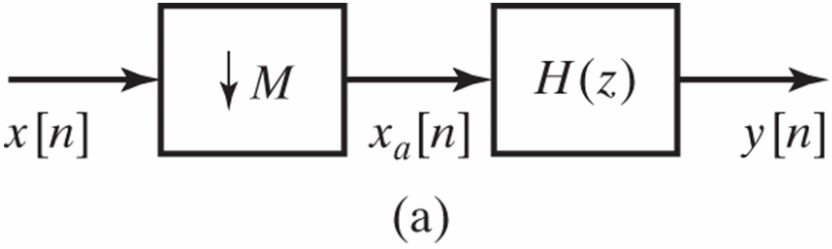
- Stage 3

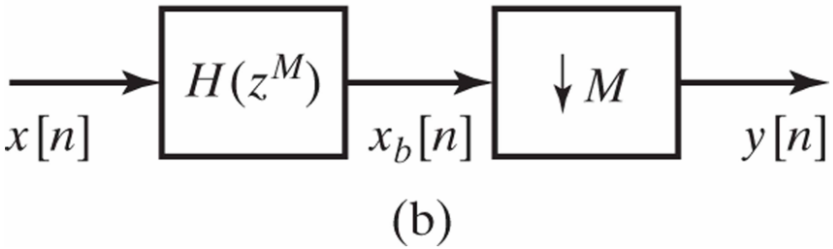




Multistage decimation analysis – preliminary

- The two systems

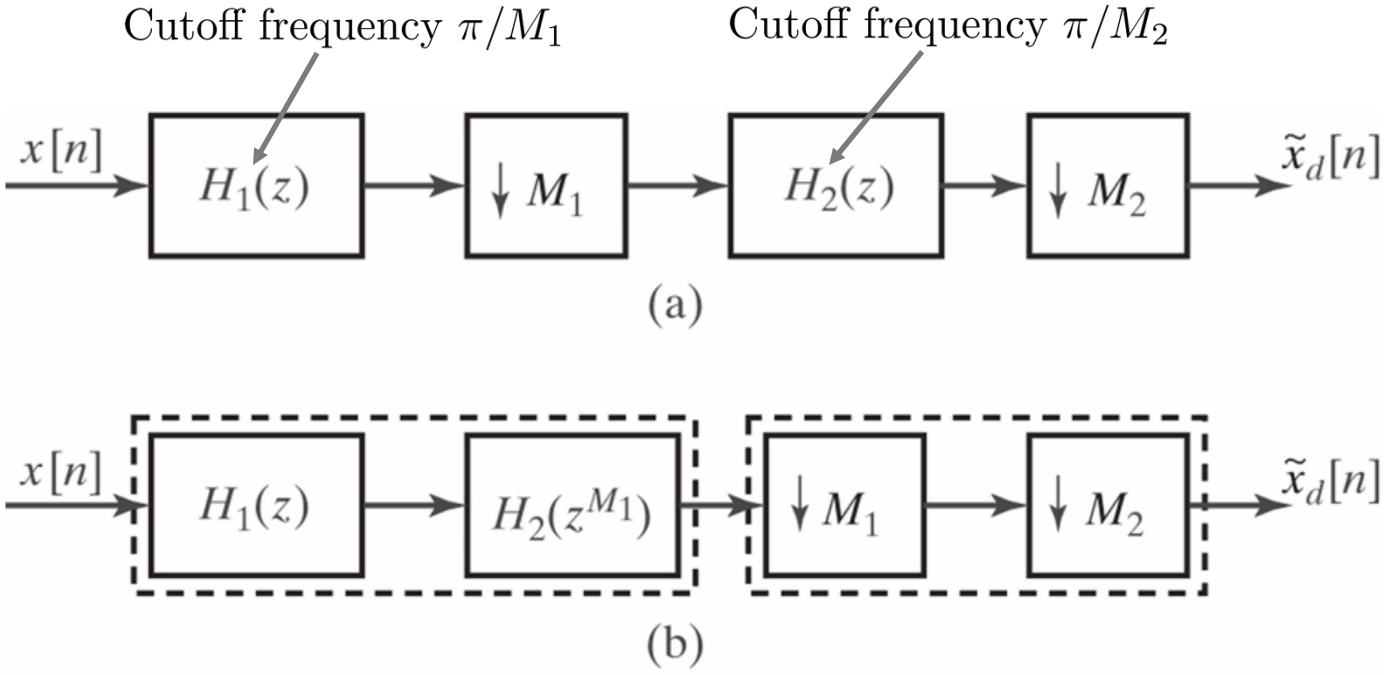




are identical (check Section 4.7.1.)

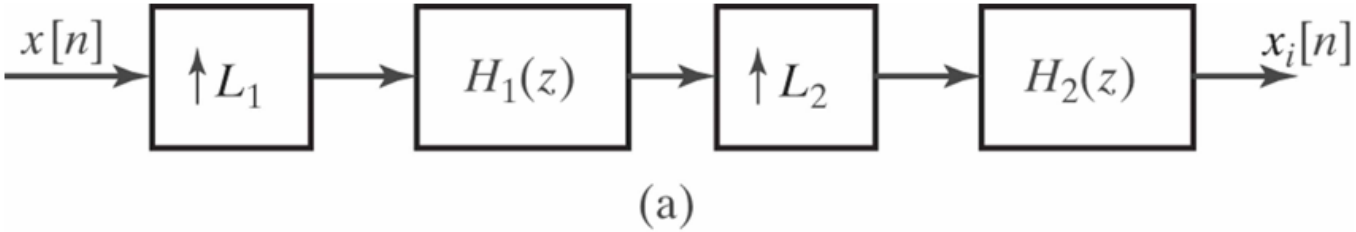
Multistage decimation

- To obtain the decimation ratio $M = M_1 M_2$

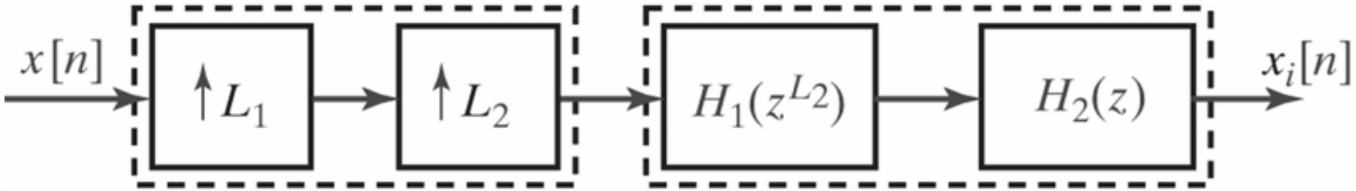


Multistage interpolation

- To obtain the upsampling ratio $M = M_1 M_2$



(a)



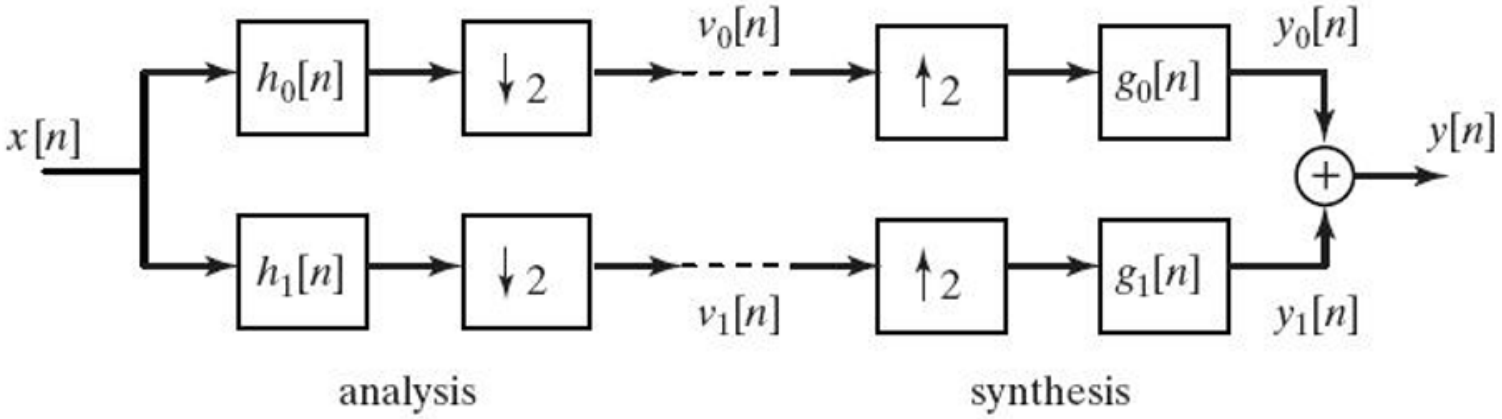
(b)

Multirate filter bank

- Split audio/speech signals into different frequency components
  - Low, medium, high frequency components
  - Process (quantize and/or storage) each frequency component separately
  - Reconstruct the signal by synthesizing frequency components

- We can exploit downsampling/upsampling for multirate filter bank

Two-channel perfect reconstruction filter bank

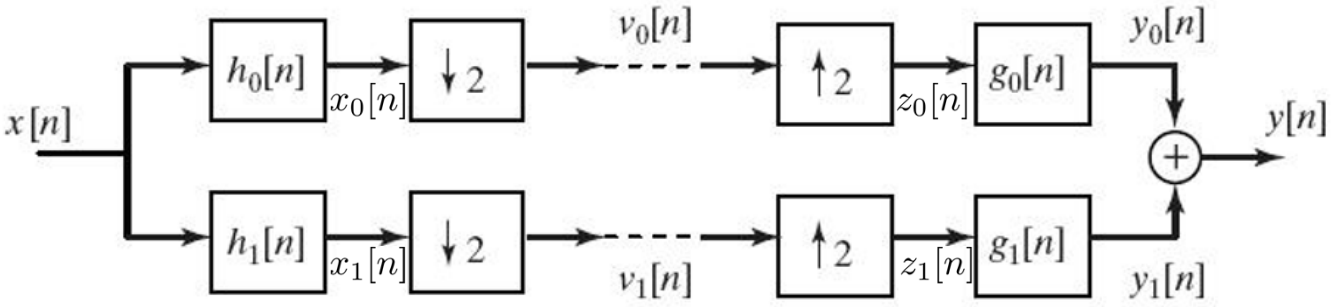


- Analysis part by downsampling / synthesis part by upsampling
- We want to have $x[n] = y[n]$ without further processing on ----
- What are relationships among $h_0[n], h_1[n], g_0[n], g_1[n]$?
  - Decomposition requires $h_0[n], h_1[n]$ to be lowpass and highpass filters

$$H_0(e^{j\omega}): \text{ lowpass filter with passband } 0 \le |\omega| \le \pi/2$$

$$h_1[n] = e^{j\pi n} h_0[n] \overset{\mathcal{F}}{\longleftrightarrow} H_1(e^{j\omega}) = H_0(e^{j(\omega-\pi)})$$

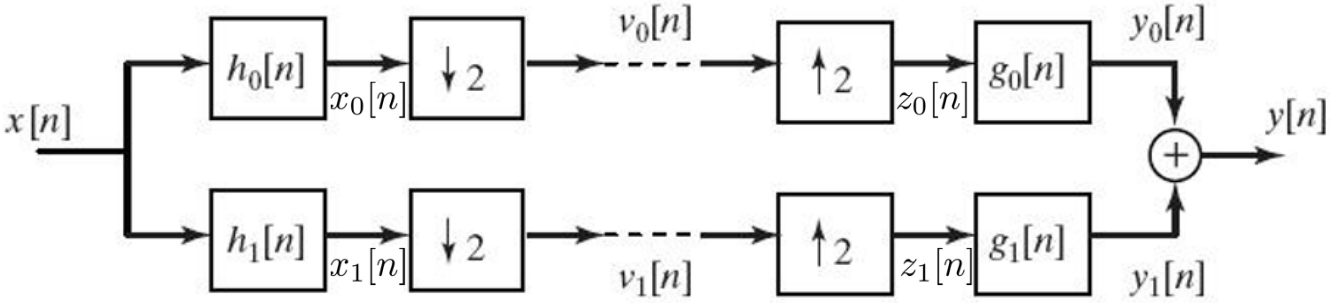
Frequency-domain relationships



- $X_i(e^{j\omega}) = H_i(e^{j\omega})X(e^{j\omega})$

$$V_i(e^{j\omega}) = \frac{1}{2}\left(X_i(e^{j\omega/2}) + X_i(e^{j(\omega-2\pi)/2})\right)$$
$$= \frac{1}{2}\left(H_i(e^{j\omega/2})X(e^{j\omega/2}) + H_i(e^{j(\omega-2\pi)/2})X(e^{j(\omega-2\pi)/2})\right)$$

Frequency-domain relationships



- $Z_i(e^{j\omega}) = V_i(e^{2j\omega})$

$$= \frac{1}{2}\left(H_i(e^{j\omega})X(e^{j\omega}) + H_i(e^{j(\omega-\pi)})X(e^{j(\omega-\pi)})\right)$$

- $Y(e^{j\omega}) = G_0(e^{j\omega})Z_0(e^{j\omega}) + G_1(e^{j\omega})Z_1(e^{j\omega})$

$$= \frac{1}{2}\left(H_0(e^{j\omega})G_0(e^{j\omega}) + H_1(e^{j\omega})G_1(e^{j\omega})\right)X(e^{j\omega})$$

$$+ \frac{1}{2}\left(H_0(e^{j(\omega-\pi)})G_0(e^{j\omega}) + H_1(e^{j(\omega-\pi)})G_1(e^{j\omega})\right)X(e^{j(\omega-\pi)})$$

Potential aliasing distortion

Ideal case

- With ideal filters that exactly split the band $0 \leq |\omega| \leq \pi$ without overlapping, it is easy to show that

$$\left(H_0(e^{j\omega})G_0(e^{j\omega}) + H_1(e^{j\omega})G_1(e^{j\omega}))\right) = 2$$

$$\left(H_0(e^{j(\omega-\pi)})G_0(e^{j\omega}) + H_1(e^{j(\omega-\pi)})G_1(e^{j\omega})\right) = 0$$

- What about with non-ideal, practical filters?

Using practical filters

- Alias cancellation condition

$$g_0[n] = 2h_0[n] \xleftrightarrow{\mathcal{F}} G_0(e^{j\omega}) = 2H_0(e^{j\omega})$$

$$g_1[n] = -2h_1[n] \xleftrightarrow{\mathcal{F}} G_1(e^{j\omega}) = -2H_0(e^{j(\omega-\pi)})$$

- Combined with $h_1[n] = e^{j\pi n}h_0[n] \xleftrightarrow{\mathcal{F}} H_1(e^{j\omega}) = H_0(e^{j(\omega-\pi)})$

$$Y(e^{j\omega}) = \left[H_0^2(e^{j\omega}) - H_0^2(e^{j(\omega-\pi)})\right] X(e^{j\omega})$$

- Thus, perfect reconstruction (with possible delay of *M* samples) requires

$$H_0^2(e^{j\omega}) - H_0^2(e^{j(\omega-\pi)}) = e^{-j\omega M}$$

- $h_0[n] = c_0\delta[n-2n_0] + c_1\delta[n-2n_1-1]$ satisfies this condition

$c_0c_1 = 1/4$ Arbitrary integer

Simple example

- Let $h_0[n] = \frac{1}{2}(\delta[n] + \delta[n-1]) \overset{\mathcal{F}}{\longleftrightarrow} H_0(e^{j\omega}) = \cos(\omega/2)e^{-j\omega/2}$

- $$\begin{aligned} H_0^2(e^{j\omega}) - H_0^2(e^{j(\omega-\pi)}) &= \cos^2(\omega/2)e^{-j\omega} - \cos^2((\omega-\pi)/2)e^{-j(\omega-\pi)} \\ &= \frac{1+\cos\omega}{2}e^{-j\omega} + \frac{1+\cos(\omega-\pi)}{2}e^{-j\omega} \\ &= \frac{1+\cos\omega}{2}e^{-j\omega} + \frac{1-\cos\omega}{2}e^{-j\omega} \\ &= e^{-j\omega} \end{aligned}$$

  $$\cos^2\left(\frac{\omega}{2}\right) = \frac{1+\cos\omega}{2}$$

- Therefore, $y[n] = x[n-1]$

- $h_0[n] = c_0\delta[n-2n_0] + c_1\delta[n-2n_1-1]$ is very crucial lowpass filter though