ESE 531: Digital Signal Processing

Lec 10: February 18, 2020

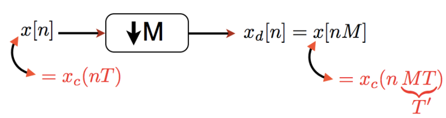
Non-Integer and Multi-rate Sampling

Lecture Outline

- Review: Downsampling/Upsampling
- Interpolation
- Non-integer Resampling
- Multi-Rate Processing
  - Interchanging Operations

Downsampling

- Definition: Reducing the sampling rate by an integer number

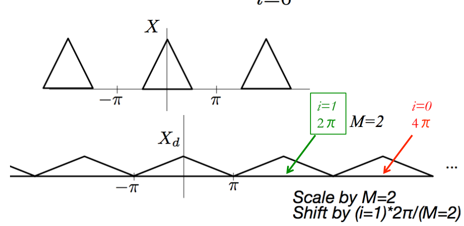


$$X_d(e^{j\omega}) = \frac{1}{M}\sum_{i=0}^{M-1} X(e^{j(\frac{\omega}{M} - \frac{2\pi}{M}i)})$$

stretch by M — replicate

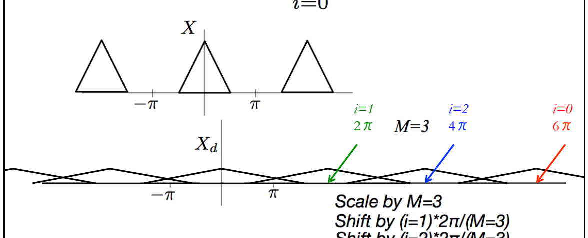
Example

$$X_d(e^{j\omega}) = \frac{1}{M}\sum_{i=0}^{M-1} X\left(e^{j(\frac{w}{M} - \frac{2\pi}{M}i)}\right)$$

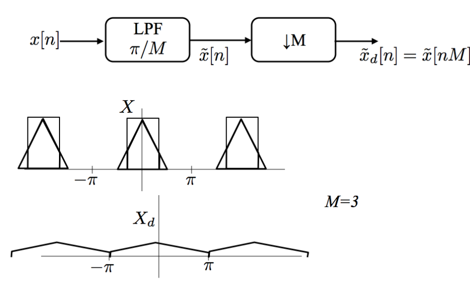


Example

$$X_d(e^{j\omega}) = \frac{1}{M}\sum_{i=0}^{M-1} X\left(e^{j(\frac{w}{M} - \frac{2\pi}{M}i)}\right)$$



Example



Upsampling

- Definition: Increasing the sampling rate by an integer number

$x[n] = x_c(nT)$

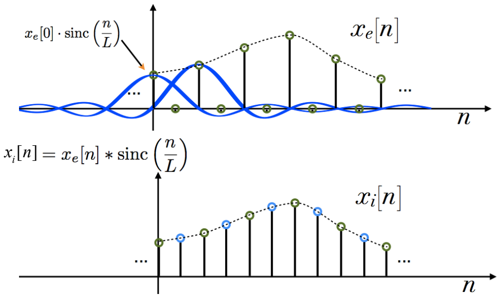
$x_i[n] = x_c(nT')$ where $T' = \frac{T}{L}$ L integer

Obtain $x_i[n]$ from $x[n]$ in two steps:

(1) Generate: $x_e[n] = \begin{cases} x[n/L] & n = 0, \pm L, \pm 2L, \cdots \\ 0 & \text{otherwise} \end{cases}$

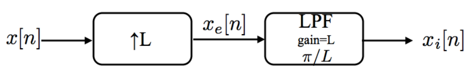
Upsampling

(2) Obtain $x_i[n]$ from $x_e[n]$ by bandlimited interpolation:

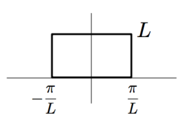


Frequency Domain Interpretation

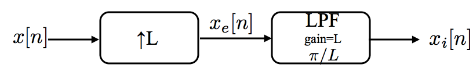
$$x_i[n] = x_e[n] * \text{sinc}(n/L)$$



$\text{sinc}(n/L)$ DTFT ⇒



Frequency Domain Interpretation

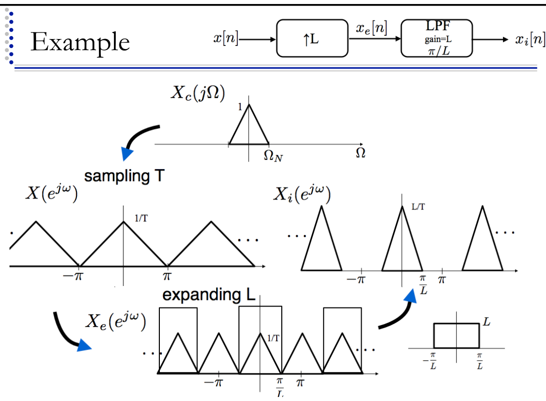


$$X_e(e^{j\omega}) = \sum_{n=-\infty}^{\infty} \underbrace{x_e[n]}_{\neq 0 \text{ only for } n=mL \text{ (integer m)}} e^{-j\omega n}$$

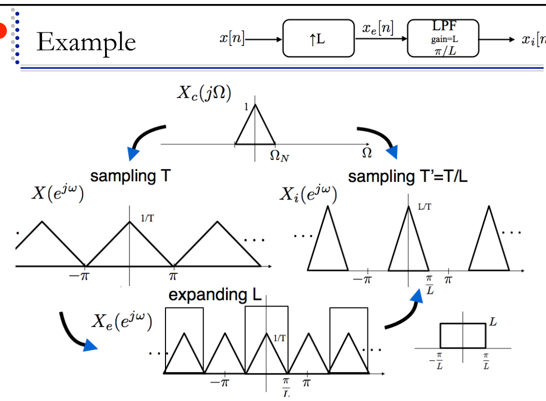
$$= \sum_{m=-\infty}^{\infty} \underbrace{x_e[mL]}_{=x[m]} e^{-j\omega mL} = X(e^{j\omega L})$$

Compress DTFT by a factor of L!

Example



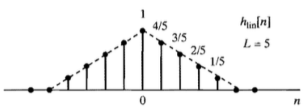
Example



Practical Interpolation

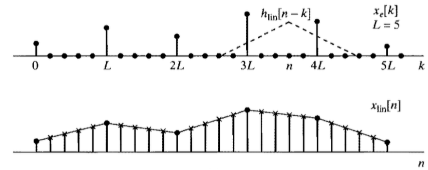
- Interpolate with simple, practical filters
  - Linear interpolation – samples between original samples fall on a straight line connecting the samples
    - Convolve with triangle instead of sinc

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



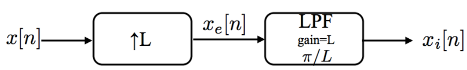
Practical Interpolation

- Interpolate with simple, practical filters
  - Linear interpolation – samples between original samples fall on a straight line connecting the samples
    - Convolve with triangle instead of sinc

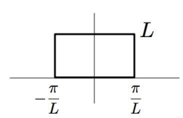


Frequency Domain Interpretation

$$x_i[n] = x_e[n] * \text{sinc}(n/L)$$

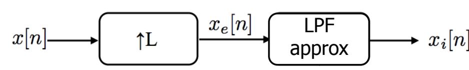


$\text{sinc}(n/L)$ DTFT ⇒

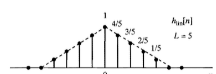


Linear Interpolation -- Frequency Domain

$$x_i[n] = x_e[n] * h_{lin}[n]$$

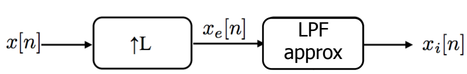


$$h_{\text{lin}}[n] = \begin{cases} 1 - |n|/L, & |n| \le L, \\ 0, & \text{otherwise}, \end{cases}$$

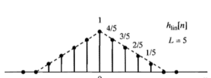


Linear Interpolation -- Frequency Domain

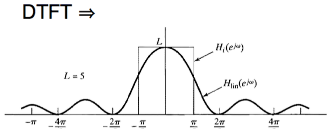
$$x_i[n] = x_e[n] * h_{lin}[n]$$



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

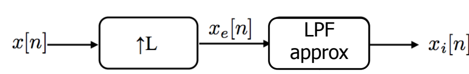


DTFT ⇒

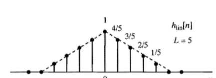


Linear Interpolation -- Frequency Domain

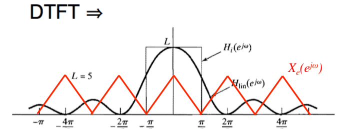
$$x_i[n] = x_e[n] * h_{lin}[n]$$



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

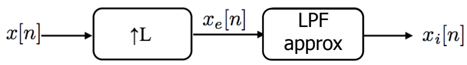


DTFT ⇒

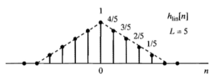


Linear Interpolation -- Frequency Domain

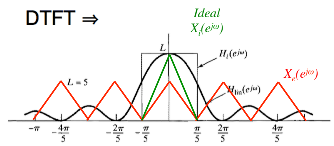
$$x_i[n] = x_e[n] * h_{lin}[n]$$



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

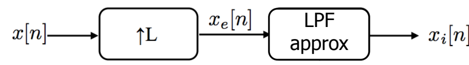


DTFT ⇒

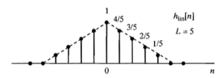


Linear Interpolation -- Frequency Domain

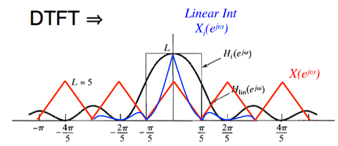
$$x_i[n] = x_e[n] * h_{lin}[n]$$



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

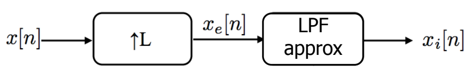


DTFT ⇒

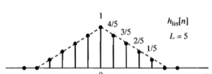


Linear Interpolation -- Frequency Domain

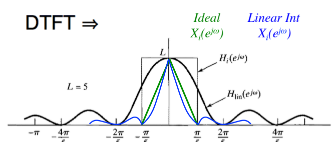
$$x_i[n] = x_e[n] * h_{lin}[n]$$



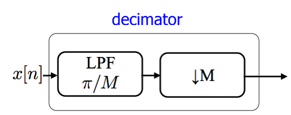
$$h_{\text{lin}}[n] = \begin{cases} 1 - |n|/L, & |n| \le L, \\ 0, & \text{otherwise}, \end{cases}$$

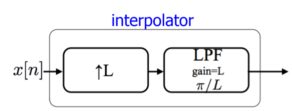


DTFT ⇒



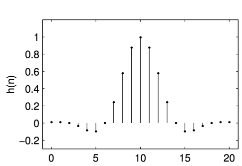
Interpolation and Decimation

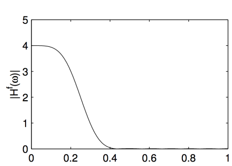




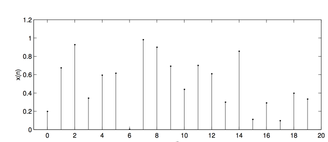
Interpolation Filter Example 1

- In this example, we interpolate a signal x(n) by a factor of 4.
- We use a linear phase Type I FIR lowpass filter of length 21.

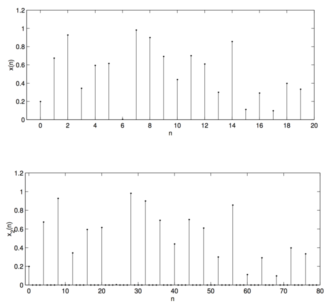




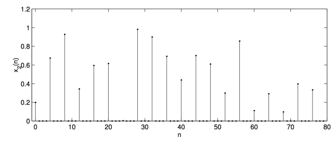
Interpolation Filter Example 1



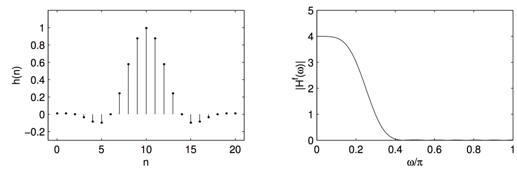
Interpolation Filter Example 1



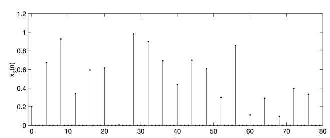
Interpolation Filter Example 1



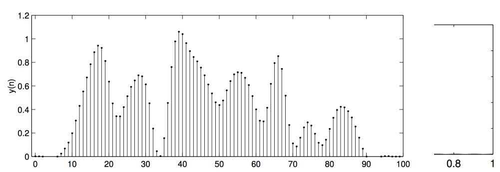
∗
(convolve)



Interpolation Filter Example 1

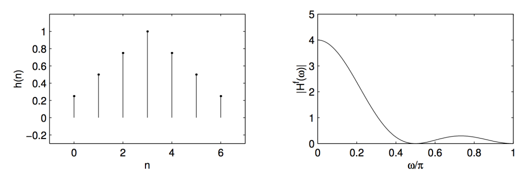


∗
(convolve)

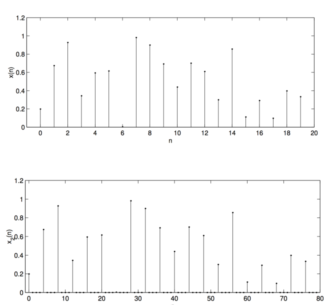


Interpolation Filter Example 2

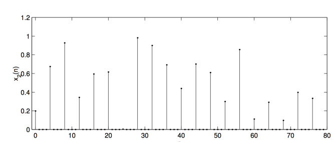
- This time we use a filter of length 7 with the effect of linear interpolation



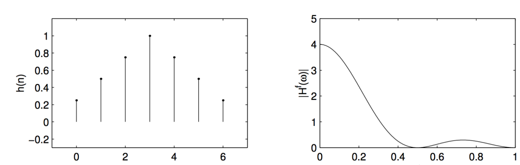
Interpolation Filter Example 2



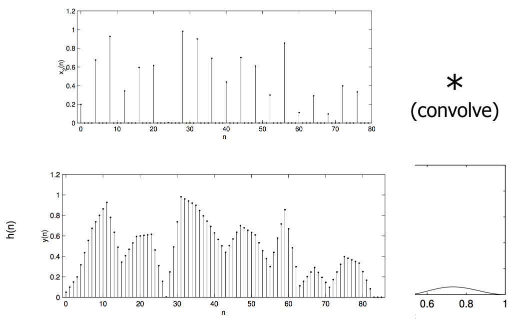
Interpolation Filter Example 2



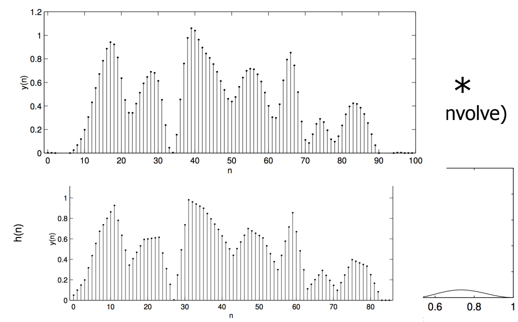
∗
(convolve)



Interpolation Filter Example 2



Interpolation Filter Example 2



Interpolation Filter Example 3

- When interpolating a signal x(n), the interpolation filter h(n) will in general change the samples of x(n) in addition to filling in the zeros.
- Can a filter be designed so as to preserve the original samples x(n)?

Interpolation Filter Example 3

- When interpolating a signal x(n), the interpolation filter h(n) will in general change the samples of x(n) in addition to filling in the zeros.
- Can a filter be designed so as to preserve the original samples x(n)?
- To be precise, if $y(n) = h(n) * [\uparrow 2]\, x(n)$ then can we design h(n) so that $y(2n) = x(n)$?

Interpolation Filter Example 3

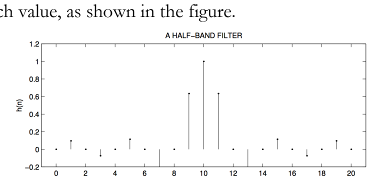
- When interpolating a signal x(n), the interpolation filter h(n) will in general change the samples of x(n) in addition to filling in the zeros.
- Can a filter be designed so as to preserve the original samples x(n)?
- To be precise, if $y(n) = h(n) * [\uparrow 2]\, x(n)$ then can we design h(n) so that $y(2n) = x(n)$?
  - Or more generally, so that $y(2n + n_o) = x(n)$?

Interpolation Filter Example 3

- When interpolating by a factor of 2, if h(n) is a half-band filter, then it will not change the samples x(n).
- A n_o-centered half-band filter h(n) is a filter that satisfies:

$$h(n) = \begin{cases} 1, & \text{for } n = n_o \\ 0, & \text{for } n = n_o \pm 2, 4, 6, \ldots \end{cases}$$

- That means, every second value of h(n) is zero, except for one such value, as shown in the figure.

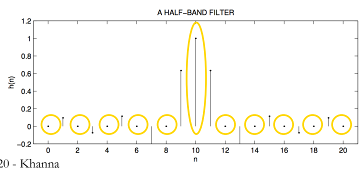


Interpolation Filter Example 3

- When interpolating by a factor of 2, if h(n) is a half-band filter, then it will not change the samples x(n).
- A n_o-centered half-band filter h(n) is a filter that satisfies:

$$h(n) = \begin{cases} 1, & \text{for } n = n_o \\ 0, & \text{for } n = n_o \pm 2, 4, 6, \ldots \end{cases}$$

- That means, every second value of h(n) is zero, except for one such value, as shown in the figure.

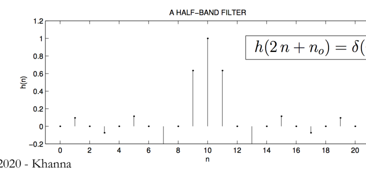


Interpolation Filter Example 3

- When interpolating by a factor of 2, if h(n) is a half-band filter, then it will not change the samples x(n).
- A n_o-centered half-band filter h(n) is a filter that satisfies:

$$h(n) = \begin{cases} 1, & \text{for } n = n_o \\ 0, & \text{for } n = n_o \pm 2, 4, 6, \ldots \end{cases}$$

- That means, every second value of h(n) is zero, except for one such value, as shown in the figure.



$$h(2n + n_o) = \delta(n),$$

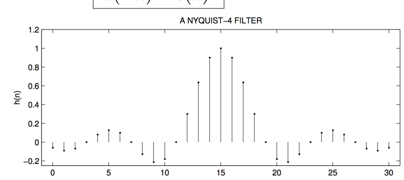
Interpolation Filter Example 4

- When interpolating a signal x(n) by a factor L, the original samples of x(n) are preserved if h(n) is a Nyquist-L filter.
- A Nyquist-L filter simply generalizes the notion of the halfband filter to L > 2.

Interpolation Filter Example 4

- When interpolating a signal x(n) by a factor L, the original samples of x(n) are preserved if h(n) is a Nyquist-L filter.
- A Nyquist-L filter simply generalizes the notion of the halfband filter to L > 2.
- A (0-centered) Nyquist-L filter h(n) is one for which

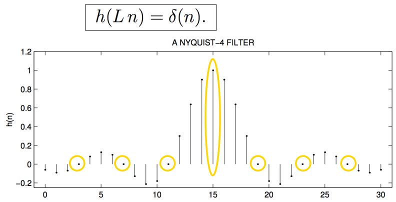
$$h(L\,n) = \delta(n).$$



Interpolation Filter Example 4

- When interpolating a signal x(n) by a factor L, the original samples of x(n) are preserved if h(n) is a Nyquist-L filter.
- A Nyquist-L filter simply generalizes the notion of the halfband filter to L > 2.
- A (0-centered) Nyquist-L filter h(n) is one for which

$$h(L\,n) = \delta(n).$$



Non-integer Resampling

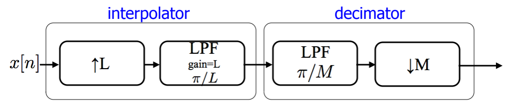
Penn

Non-integer Resampling

- T'=TM/L

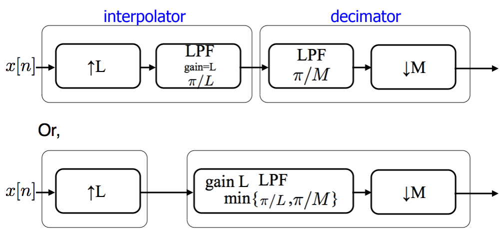
Non-integer Resampling

- T'=TM/L
  - Upsample by L, then downsample by M



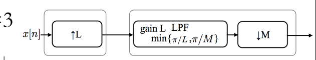
Non-integer Resampling

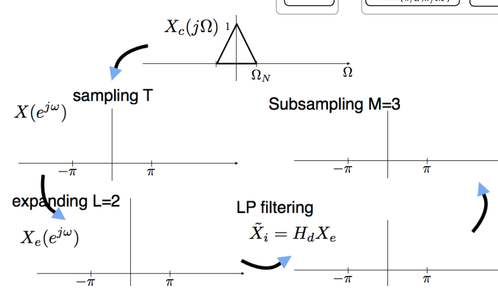
- T'=TM/L
  - Upsample by L, then downsample by M



Example

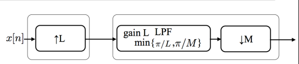
- T'=3/2T → L=2, M=3

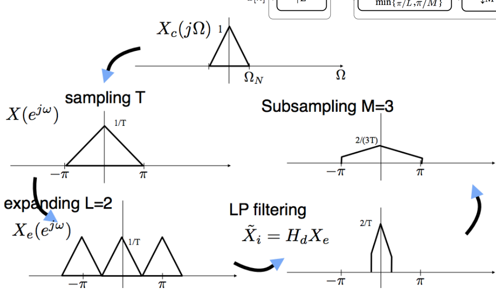




Example

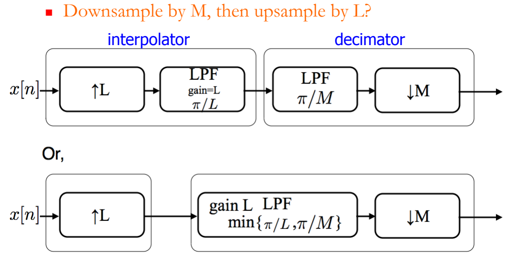
- T'=3/2T → L=2, M=3





Non-integer Sampling

- T'=TM/L
  - Downsample by M, then upsample by L?



Example

- What if we want to resample by 1.01T?

Example

- What if we want to resample by 1.01T?
  - Upsample by L=100
  - Filter π /101
  - Downsample by M=101

Example

- What if we want to resample by 1.01T?
  - Upsample by L=100
  - Filter π /101 ($$$$$)
  - Downsample by M=101
- Fortunately there are ways around it!
  - Called multi-rate signal processing
  - Uses compressors, expanders and filtering

Interchanging Operations

not LTI!

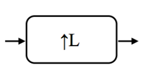
"expander"

Upsampling
-expanding in time
-compressing in frequency

"compressor"

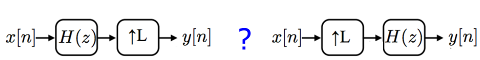
Downsampling
-compressing in time
-expanding in frequency

Interchanging Operations - Expander

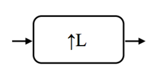


"expander"

Upsampling
-expanding in time
-compressing in frequency

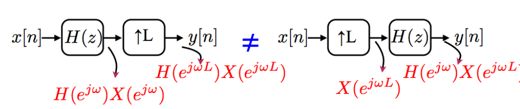


Interchanging Operations - Expander

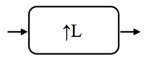


"expander"

Upsampling
-expanding in time
-compressing in frequency

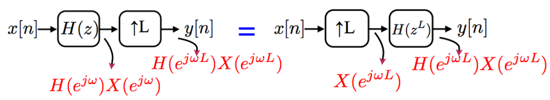


Interchanging Operations - Expander

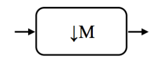


"expander"

Upsampling
-expanding in time
-compressing in frequency



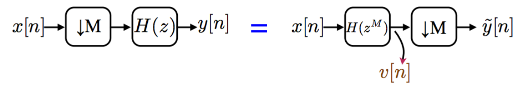
Interchanging Operations - Compressor



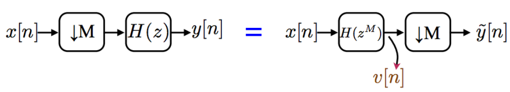
not LTI!

"compressor"

Downsampling
-compressing in time
-expanding in frequency



Interchanging Operations - Compressor

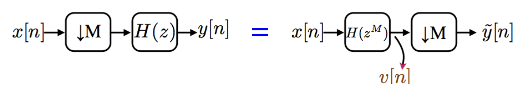


$$Y(e^{j\omega}) = H(e^{j\omega})\left(\frac{1}{M}\sum_{i=0}^{M-1} X\left(e^{j(\frac{\omega}{M}-\frac{2\pi i}{M})}\right)\right)$$

$$= \frac{1}{M}\sum_{i=0}^{M-1} \underbrace{H\left(e^{j(\omega-2\pi i)}\right)}_{H(e^{j\omega})} X\left(e^{j(\frac{\omega}{M}-\frac{2\pi i}{M})}\right)$$

$$= \frac{1}{M}\sum_{i=0}^{M-1} H\left(e^{jM(\frac{\omega}{M}-\frac{2\pi i}{M})}\right) X\left(e^{j(\frac{\omega}{M}-\frac{2\pi i}{M})}\right)$$

Interchanging Operations - Compressor



$$Y(e^{j\omega}) = \frac{1}{M}\sum_{i=0}^{M-1} H\left(e^{jM(\frac{\omega}{M}-\frac{2\pi i}{M})}\right) X\left(e^{j(\frac{\omega}{M}-\frac{2\pi i}{M})}\right)$$

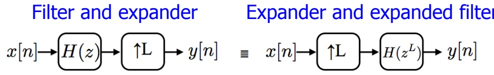
$$V(e^{j\omega}) = H(e^{j\omega M})X(e^{j\omega})$$

$$\tilde{Y}(e^{j\omega}) = \frac{1}{M}\sum_{i=0}^{M-1} V\left(e^{j(\frac{\omega}{M}-\frac{2\pi i}{M})}\right)$$

=

Interchanging Operations - Summary

Filter and expander — Expander and expanded filter*



Compressor and filter — Expanded filter* and compressor

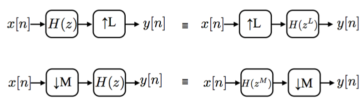
*Expanded filter = expanded impulse response, compressed freq response

Multi-Rate Signal Processing

- What if we want to resample by 1.01T?
  - Expand by L=100
  - Filter π /101 ($$$$$)
  - Compress by M=101
- Fortunately there are ways around it!
  - Called multi-rate
  - Uses compressors, expanders and filtering

Big Ideas

- Downsampling/Upsampling
- Practical Interpolation
- Non-integer Resampling
- Multi-Rate Processing
  - Interchanging Operations



Admin

- HW 4 due Sunday