Course Notes 5 – Sampling of Continuous Time Signals

5.0 Introduction

Thus far we have manipulated and analyzed sequences of numbers with no regard for their origin, or for any implied time duration between the individual samples.

Now suppose that the sequence $x(n)$ was obtained by periodic sampling of the analog signal $x(t)$ at intervals T, called the sampling period, and

$$F_s = \frac{1}{T} \qquad \text{the sampling frequency}$$

$$x(t)\Big|_{t=nT} = x(nT) \Rightarrow x(n) \qquad \text{for } -\infty < n < \infty$$

> Some texts uses the following notation: $x(n) = x_c(nT), \quad -\infty < n < \infty$

Before proceeding let's review some notational conventions:

- For analog signals the Fourier dual relationship is

$$x(t) \Leftrightarrow X(\Omega)$$

Continuous Time Fourier Transform

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\Omega) e^{j\Omega t} d\Omega$$

synthesis equation

$$X(\Omega) = \int_{-\infty}^{\infty} x(t) e^{-j\Omega t} dt$$

analysis equation

- For discrete-time signals the Fourier dual relationship is:

$$x(n) \Leftrightarrow X(\omega)$$

Discrete-Time Fourier Transform (DTFT)

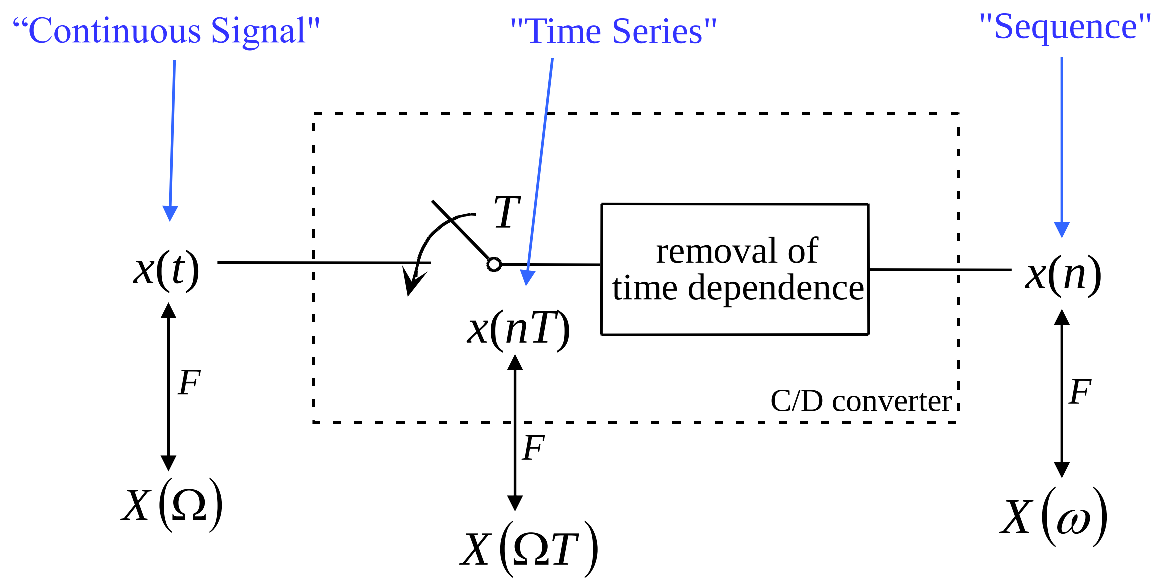
$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega) e^{j\omega n} d\omega$$

synthesis equation

$$X(\omega) = \sum_{n=\infty}^{\infty} x(n) e^{-j\omega n}$$

analysis equation

There is a precise relationship between these transforms, in the context of our discussion of sampling, as shown below:



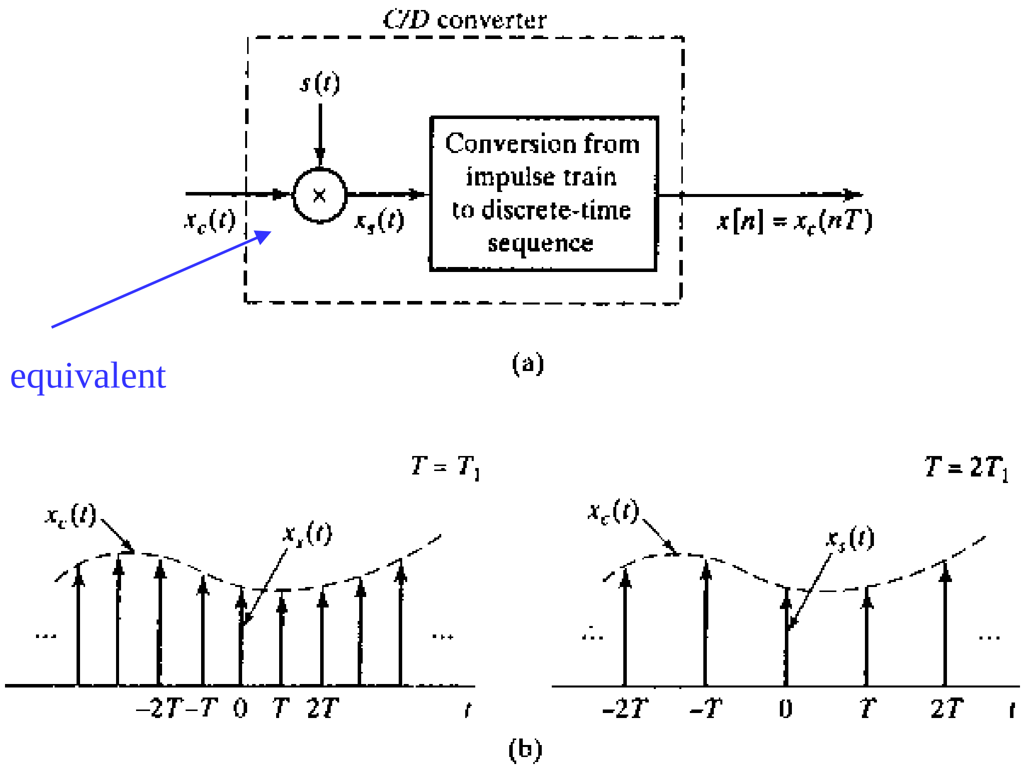
The task now is to establish the link (i.e., sampling) between the analog signal $x(t)$, and its discrete counterpart $x(n)$, by thoroughly analyzing the process above.

Before we begin, lets review two important identities that will be essential in understanding sampling:

$$x(t)\delta(t-t_0) = x(t_0)\delta(t-t_0) \qquad \text{Multiplication with impulses}$$

$$x(t)*\delta(t-t_0) = x(t-t_0) \qquad \text{Convolution with impulses}$$

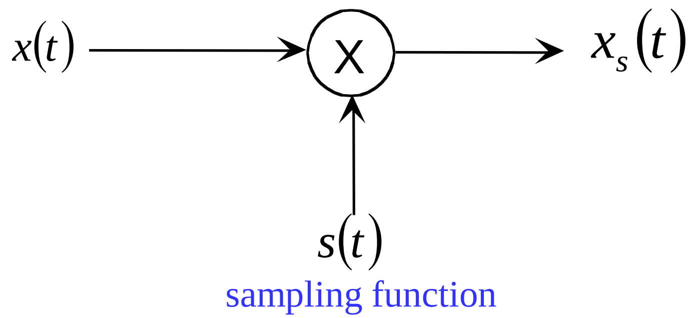
Continuous-to-Discrete converter



Switch is replaced by an equivalent mathematical operation

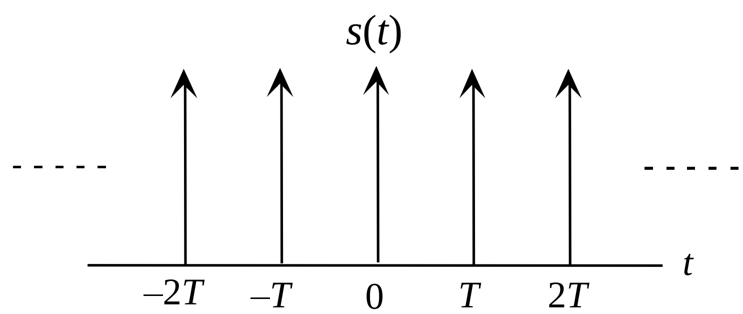
5.1 Periodic Sampling

We can model the sampling process mathematically as follows:



Let the sampling function be:

$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$$



$$x_s(t) = x(t) \cdot s(t) = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

$$x_s(t) = \sum_{n=-\infty}^{\infty} \underbrace{x(t{=}nT)}_{= x(n)} \delta(t - nT)$$ Sampling viewed from the time domain

$= x(n)$ in the discrete sequence

5.2 Frequency Domain Representation of Sampling

Now look at sampling from the frequency domain perspective. Begin with the sampling process and take the Fourier transform of both sides:

$$x_s(t) = x(t) \cdot s(t)$$

$$F\{x_s(t)\} = F\{x(t) \cdot s(t)\}$$

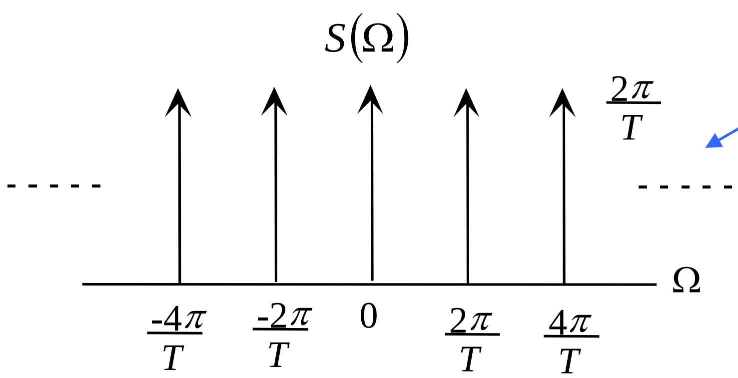
$$X_s(\Omega) = \frac{1}{2\pi} X(\Omega) * S(\Omega)$$

becomes convolution in frequency

from term in inverse Fourier transform

Using the sifting property of the impulse function, Fourier Series expansion of the periodic $s(t)$ yields

$$S(\Omega) = F\left\{\sum_{n=-\infty}^{\infty} \delta(t - nT)\right\} = \frac{2\pi}{T} \sum_{n=-\infty}^{\infty} \delta\left(\Omega - \frac{2\pi n}{T}\right)$$



An impulse train in the time domain is also an impulse train in the frequency domain.

Therefore, the sampling process in the frequency domain is:

$$X_s(\Omega) = \frac{1}{2\pi} X(\Omega) * \frac{2\pi}{T} \sum_{n=-\infty}^{\infty} \delta\left(\Omega - \frac{2\pi n}{T}\right)$$

Due to sampling, the spectrum is periodic. Therefore:

$$X_s(\Omega) = X(\Omega T)$$

$$X(\Omega T) = \frac{1}{T} \sum_{n=-\infty}^{\infty} X\left(\Omega - \frac{2\pi n}{T}\right)$$

and since $\omega = \Omega T$

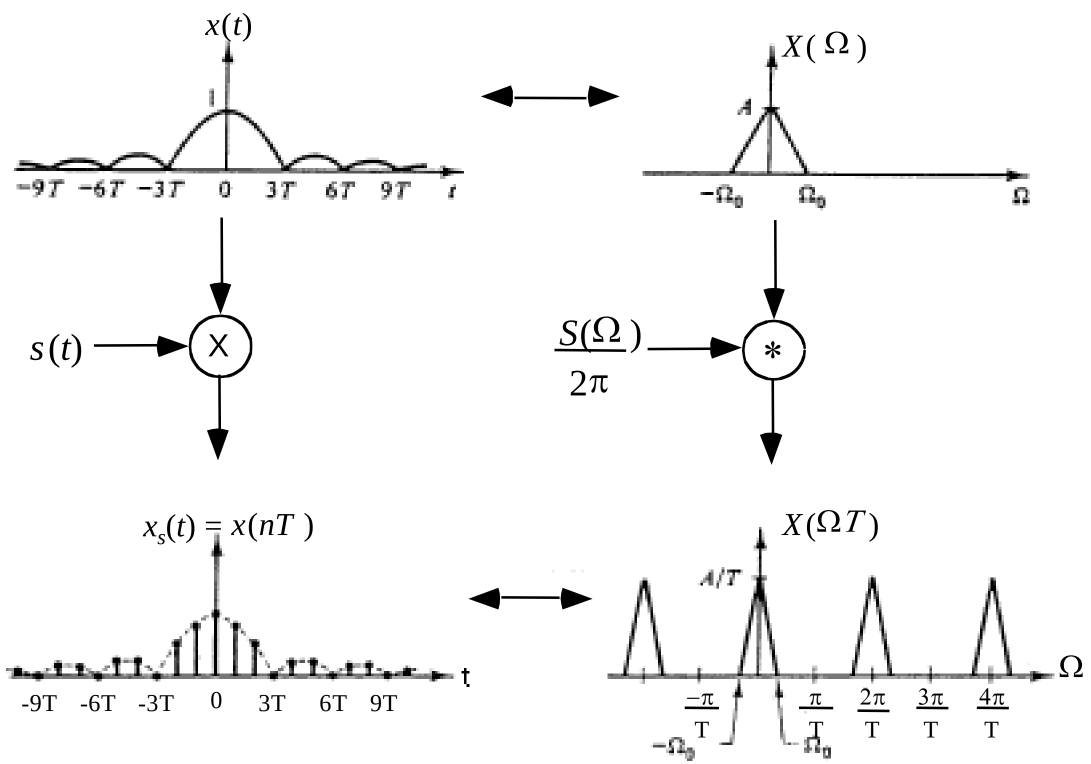
$$X(\omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} X\left(\frac{\omega}{T} - \frac{2\pi n}{T}\right)$$

So sampling period *T* affects the scaling of signals in the digital frequency domain, both in amplitude and how it spreads and replicates in ω

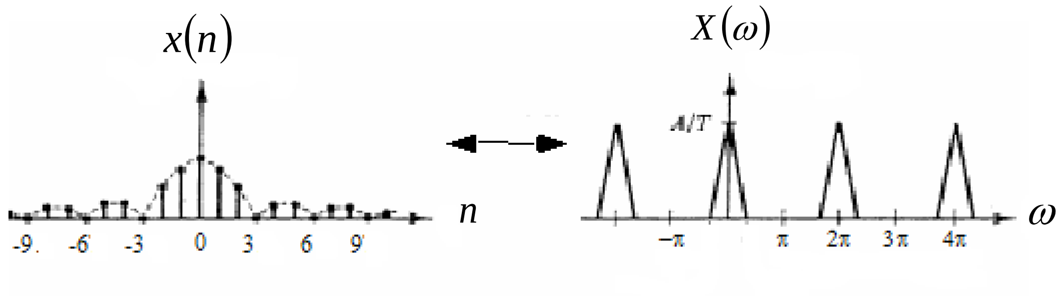
Remember: $X_s(\Omega)$ is the Fourier transform of a continuous time series

$X(\omega)$ is the Fourier transform of a discrete sequence

The important concepts associated with the sampling process can be interpreted graphically as shown below:



The time series can be expressed as a sequence by removing the time dependency to obtain the representation on the next page.



As long as the sampling frequency in $S(\Omega)$ equals or exceeds twice the highest frequency of $x(t)$, there will be no overlapping of the spectra in $x_s(t)$ -- in other words, the Nyquist criterion is satisfied

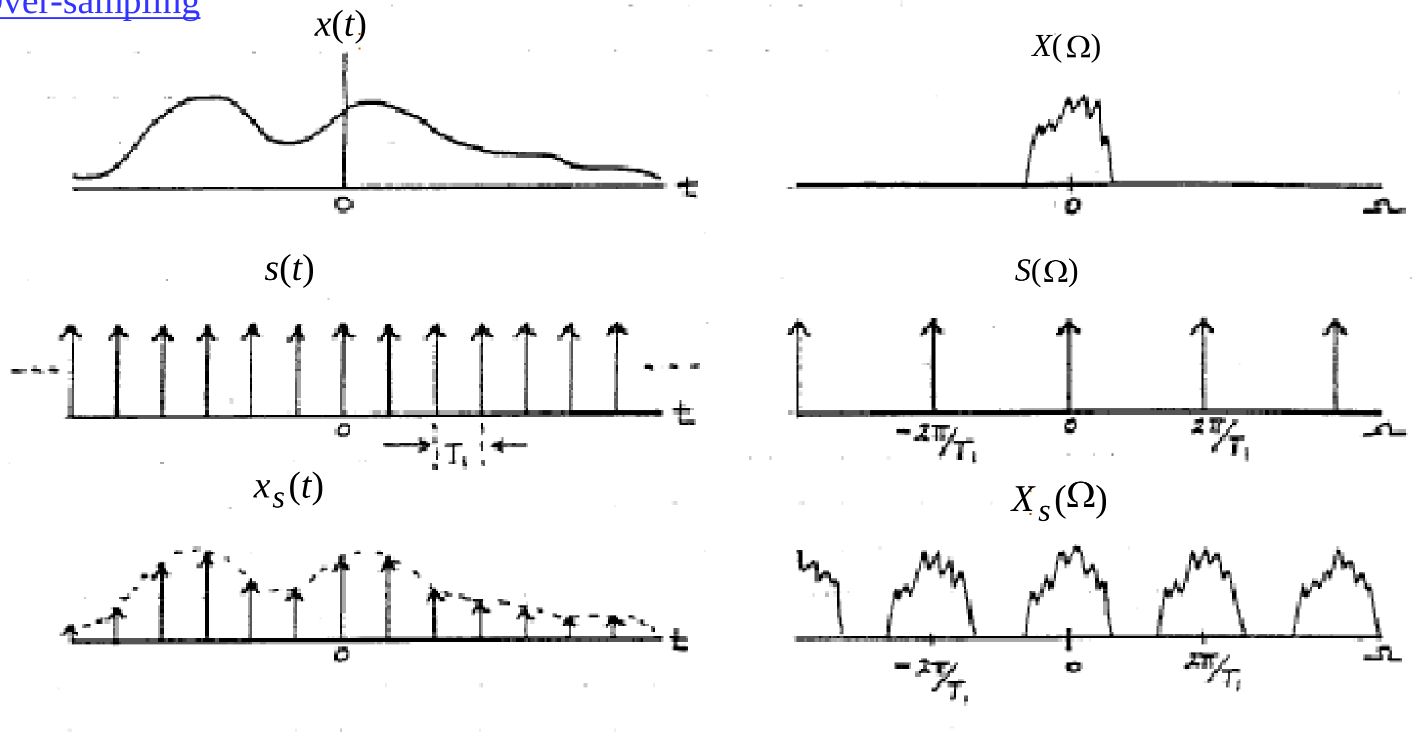
Nyquist Criterion - A bandlimited signal can be completely represented by a sequence of samples taken periodically at a rate equal to or exceeding twice the highest frequency contained in the bandlimited signal.

The key points in this statement are as follows:

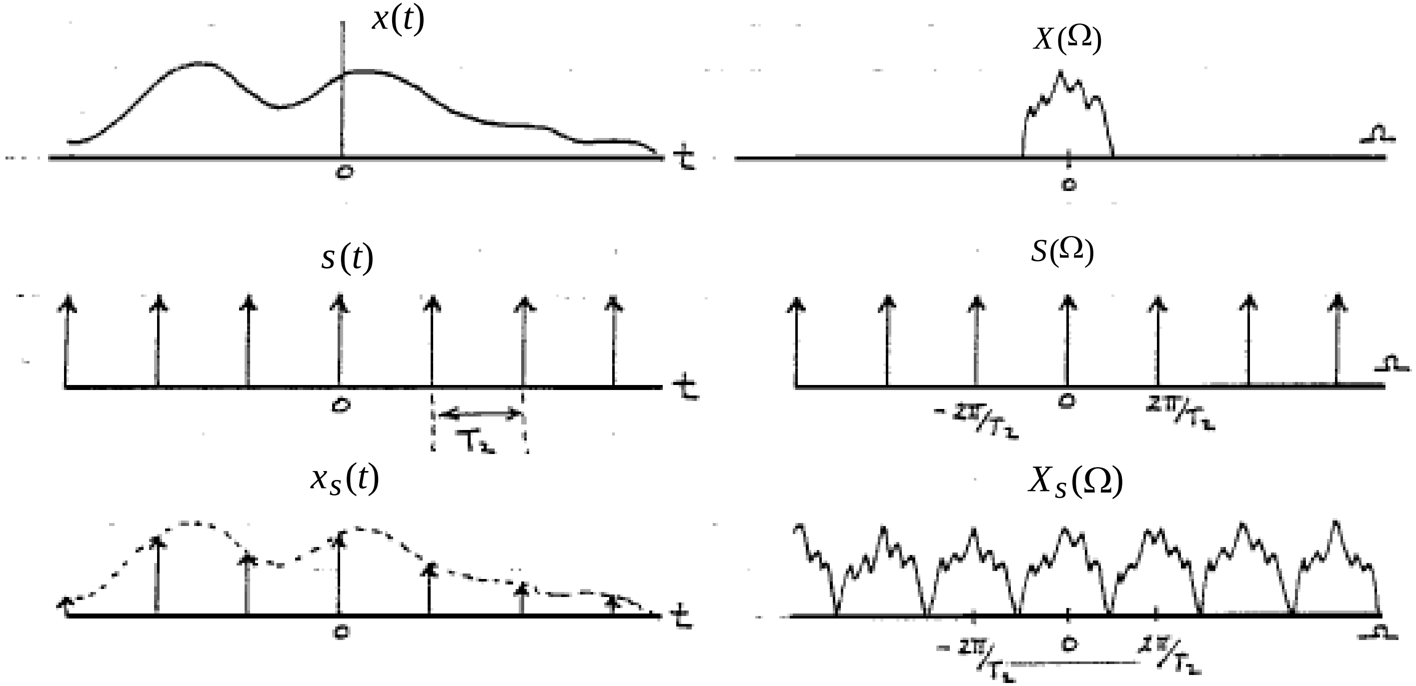
- The signal must be bandlimited. These signals **do not exist** in the real world.
- The signal is completely (not approximately) represented by its samples if Nyquist is met … but
- Periodic sampling is assumed
- The criterion is not exclusive. You can still represent a signal that has been sampled at less than the "Nyquist rate" … just not completely.

Let’s compare three interesting categories of sampling rates (under band-limited assumption):

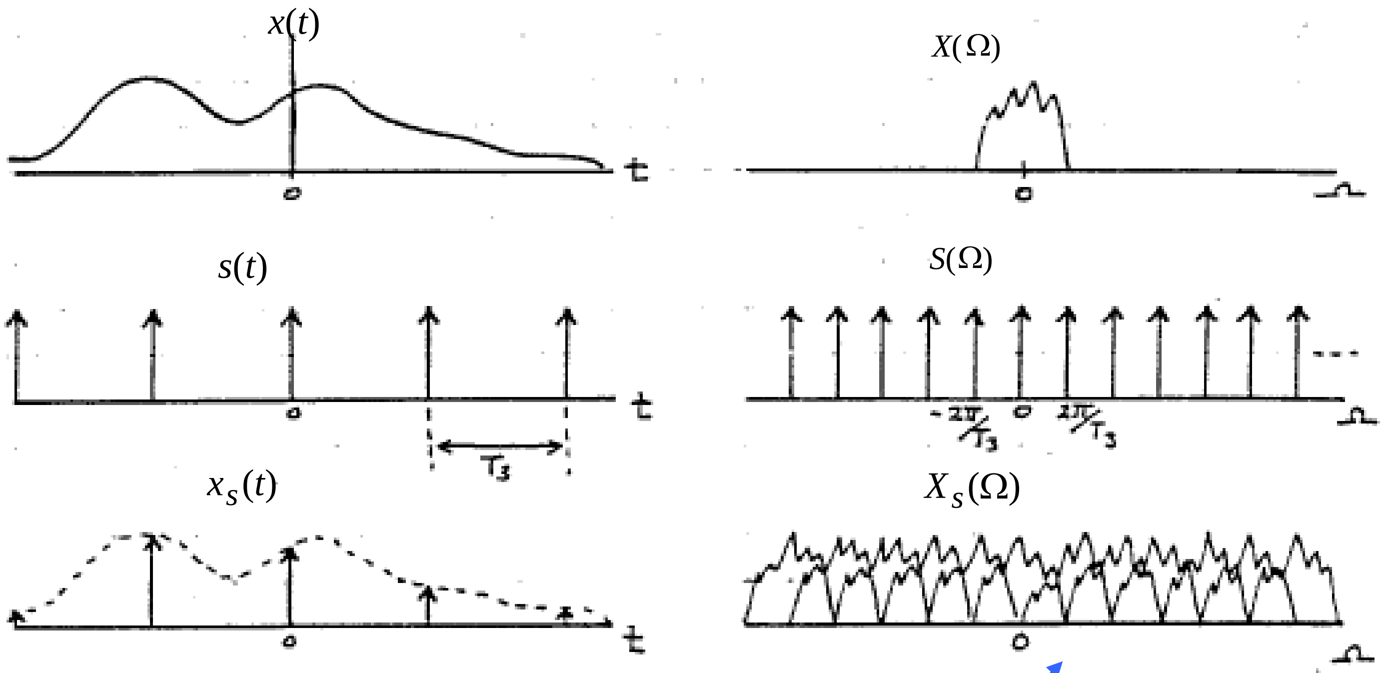
- Over-sampling



Nyquist sampling



Under-sampling

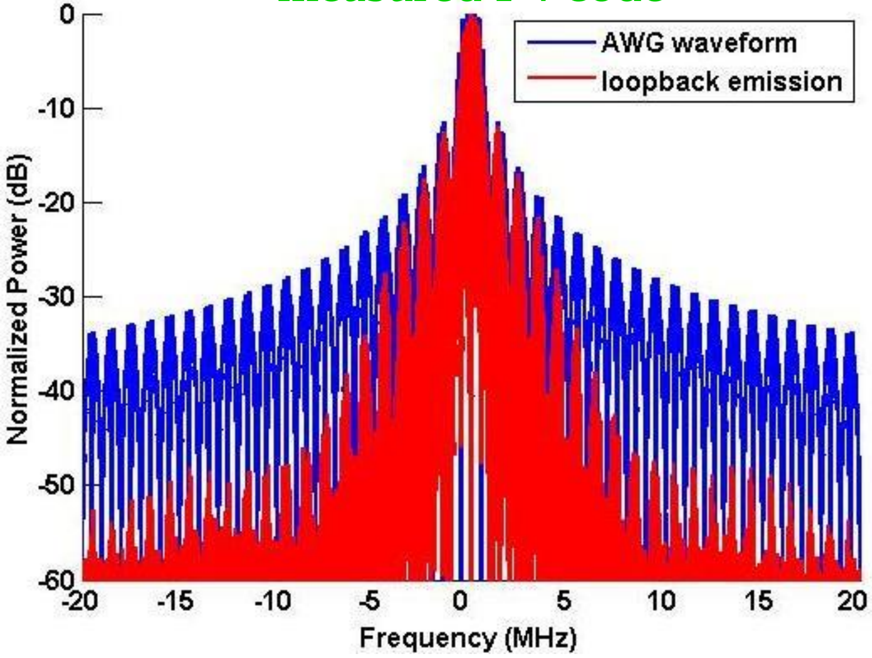


Aliasing

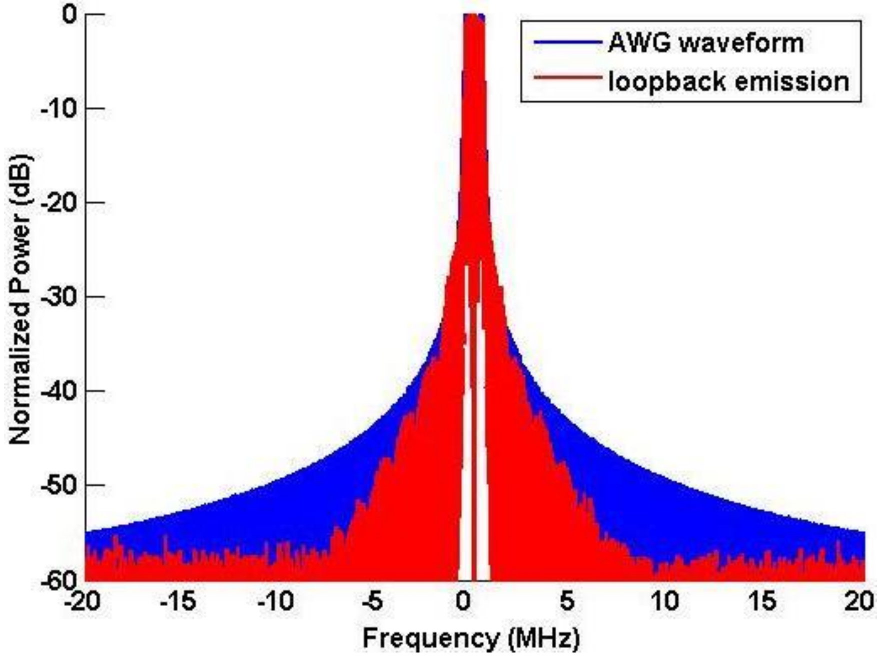
Technically, this is the only case that actually occurs in reality … though based on a given definition of bandwidth we still consider the other cases from a practical standpoint.

- For example, the figures below show the spectral content for two different pulsed waveforms, before (blue) and after (red) capture in hardware in a “loopback” configuration (used to assess transmitter distortion).
- What should be the Nyquist rate for each?
- What is the bandwidth?

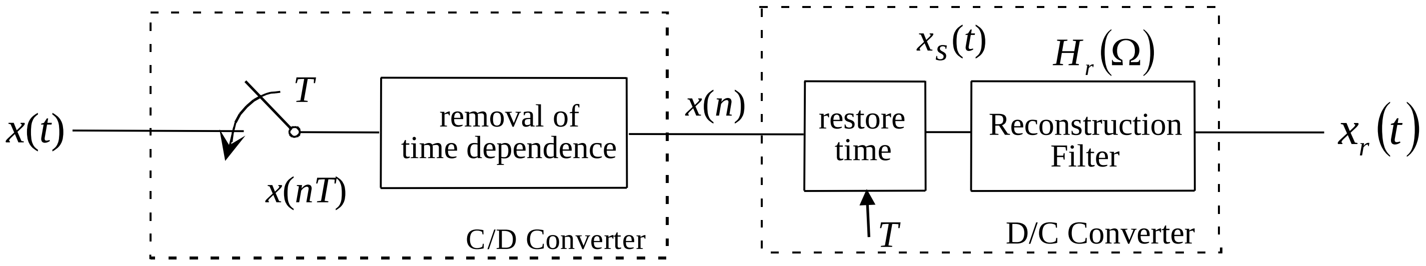
measured P4 code



measured LFM



5.3 Reconstruction of a Bandlimited Signal from its Samples



The time and frequency domain appearance of $x_s(t)$ has been established. The next task is to demonstrate that the original waveform can be recovered by simply lowpass filtering the sampled waveform (i.e., the time-series waveform).

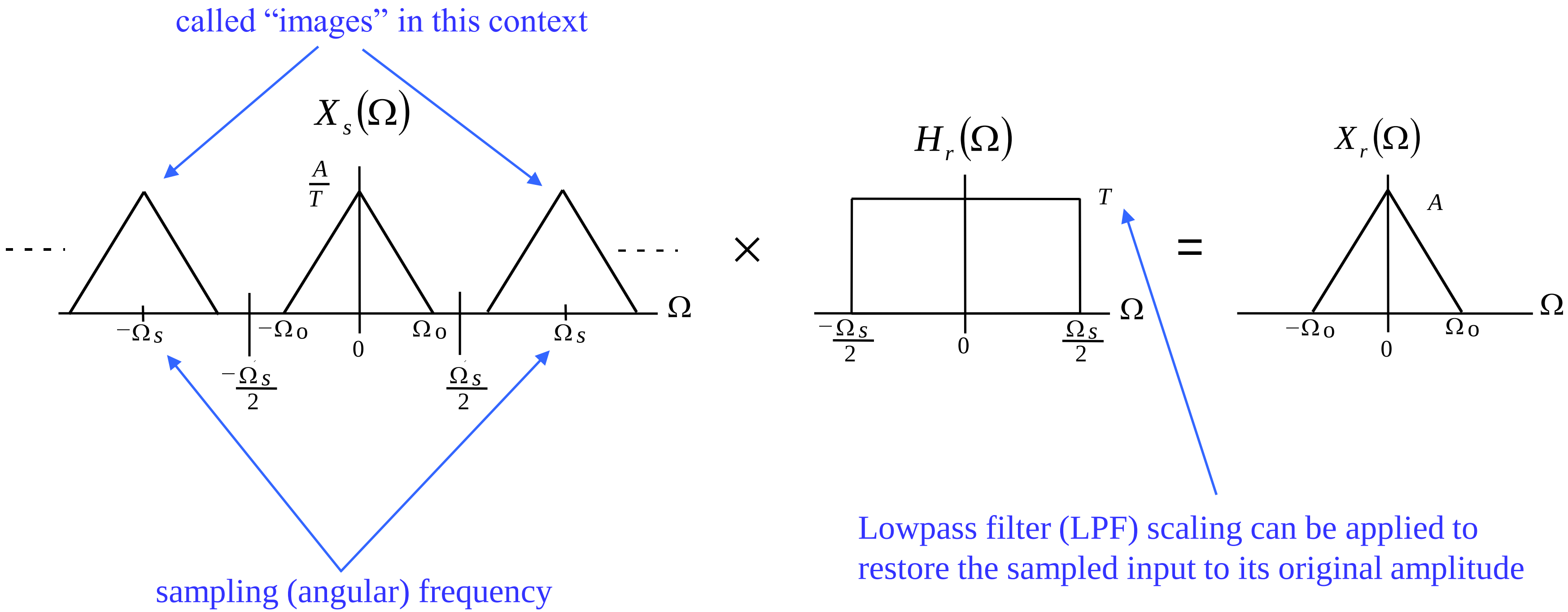
The frequency domain analysis of reconstruction (often called interpolation) is very simple to understand since

$$x_r(t) = h_r(t) * x_s(t)$$

$$\updownarrow$$

$$X_r(\Omega) = H_r(\Omega) \cdot X_s(\Omega)$$

This process can be illustrated as shown below:

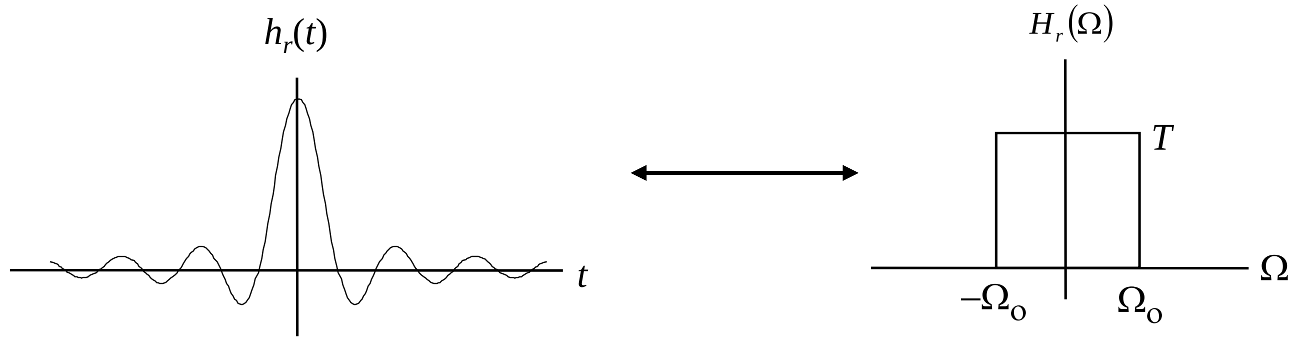


Finally, for a thorough understanding of the interpolation process, consider the time domain behavior. That is:

$$x_r(t) = h_r(t) * x_s(t)$$

Therefore, LPF reconstruction filters are also called "image rejection" filters.

recall the Fourier transform pair:



where:

$$h_r(t) = \frac{1}{2\pi} \int_{-\Omega_o}^{\Omega_o} T e^{j\Omega t} d\Omega = \frac{\Omega_O T}{\pi} \cdot \frac{\sin \Omega_O t}{\Omega_O t}$$

$\text{sinc}(\Omega_o t)$

It was previously shown that:

$$x_s(t) = \sum_{n=-\infty}^{\infty} x(nT)\delta(t - nT)$$

superposition of impulses

therefore, interpolation in the time domain can be represented as:

$$x_r(t) = x_s(t) * h_r(t) = \sum_{n=-\infty}^{\infty} x(nT)\delta(t - nT) * \frac{\Omega_O T}{\pi} \frac{\sin \Omega_o t}{\Omega_o t}$$

which is recognized as a convolution with impulses, and so can be evaluated by inspection as:

$$x_r(t) = \sum_{n=-\infty}^{\infty} x(nT) \cdot \frac{\Omega_o T}{\pi} \cdot \frac{\sin \Omega_o (t - nT)}{\Omega_o (t - nT)}$$

Note that for Nyquist sampling:

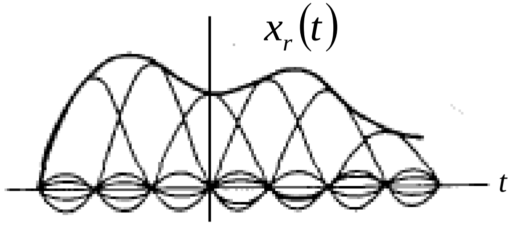
$$\Omega_s = 2\pi \frac{1}{T}$$

$$\Omega_o = \frac{\Omega_s}{2} = \frac{\pi}{T}$$

Therefore:

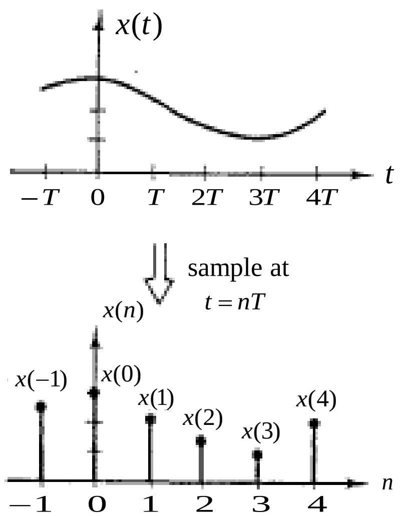
$$x_r(t) = \sum_{n=-\infty}^{\infty} x(nT) \frac{\sin \frac{\pi}{T}(t - nT)}{\frac{\pi}{T}(t - nT)}$$

$h_r(t)$ is called the "ideal interpolating function" since it is the shape of this filter that reconstructs the original signal perfectly. This concept is illustrated below and on the next page.

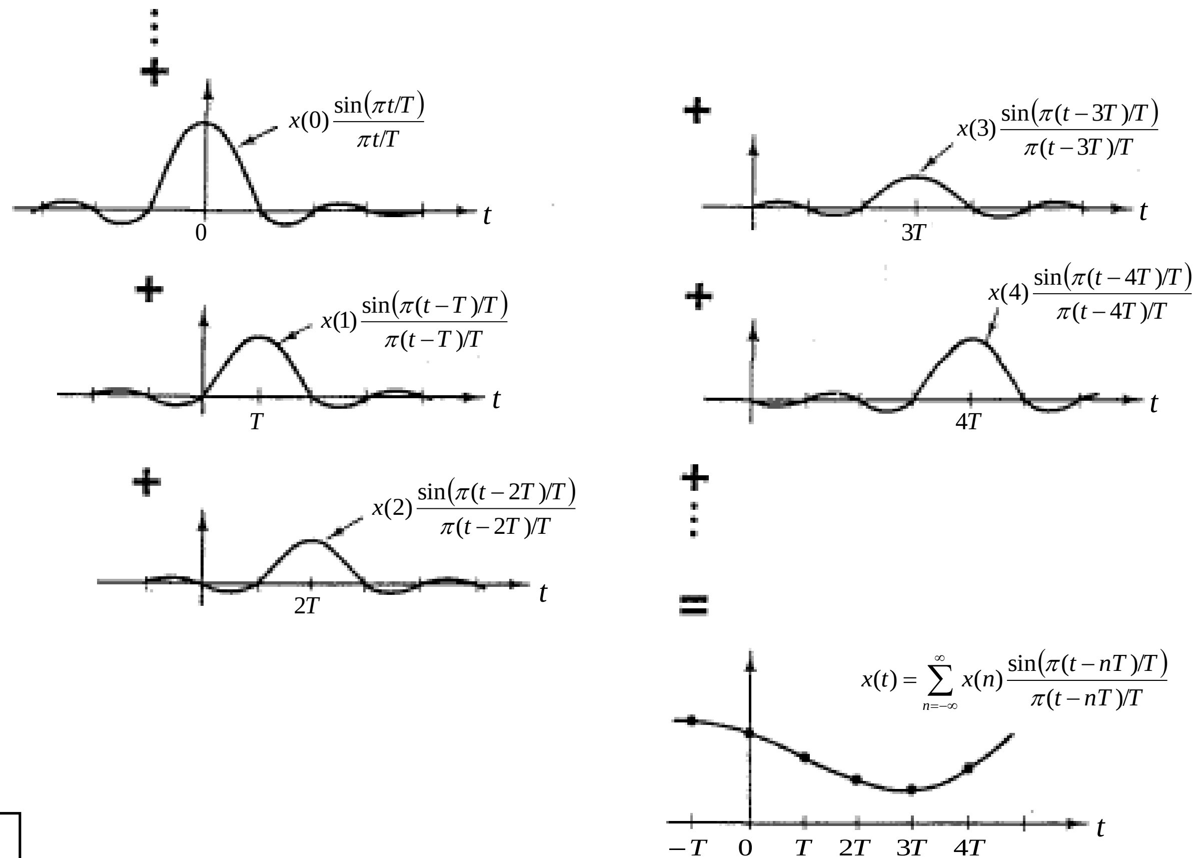


We can illustrate the reconstruction process in the time domain as shown below:

<u>Sampling</u>:



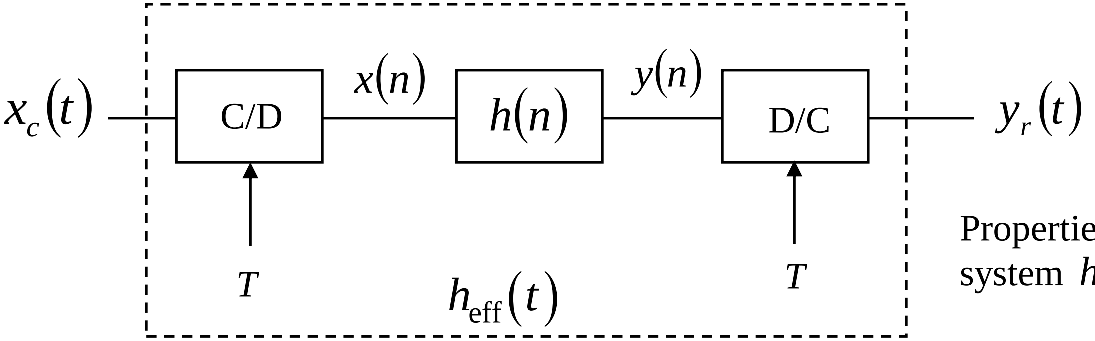
<u>Reconstruction</u>:



While theoretically correct, there's a little practical problem with the ideal reconstruction filter … it has infinite time extent.

5.4 Discrete-Time Processing of Continuous-Time Signals

The following "looks" like a continuous-time system:



Properties of the overall "effective" system $h_{\text{eff}}(t)$ depend on $h(n)$ and T.

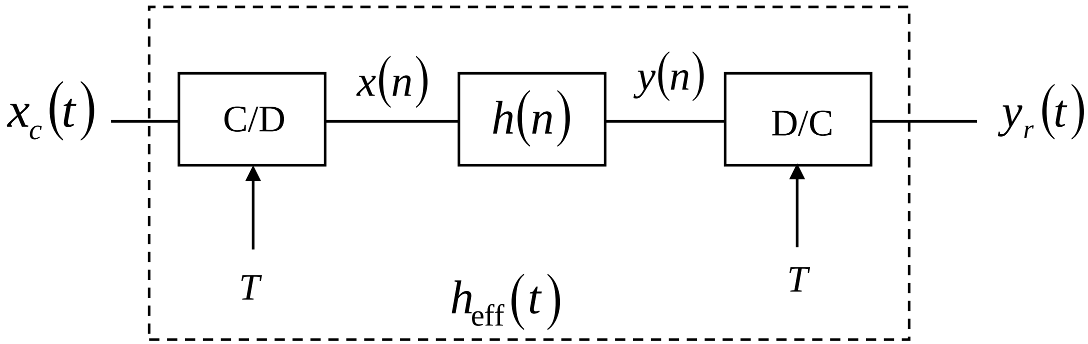
What is $h_{\text{eff}}(t)$ or $H_{\text{eff}}(\Omega)$ with respect to $h(n)$ or $H(\omega)$? Begin in the frequency domain:

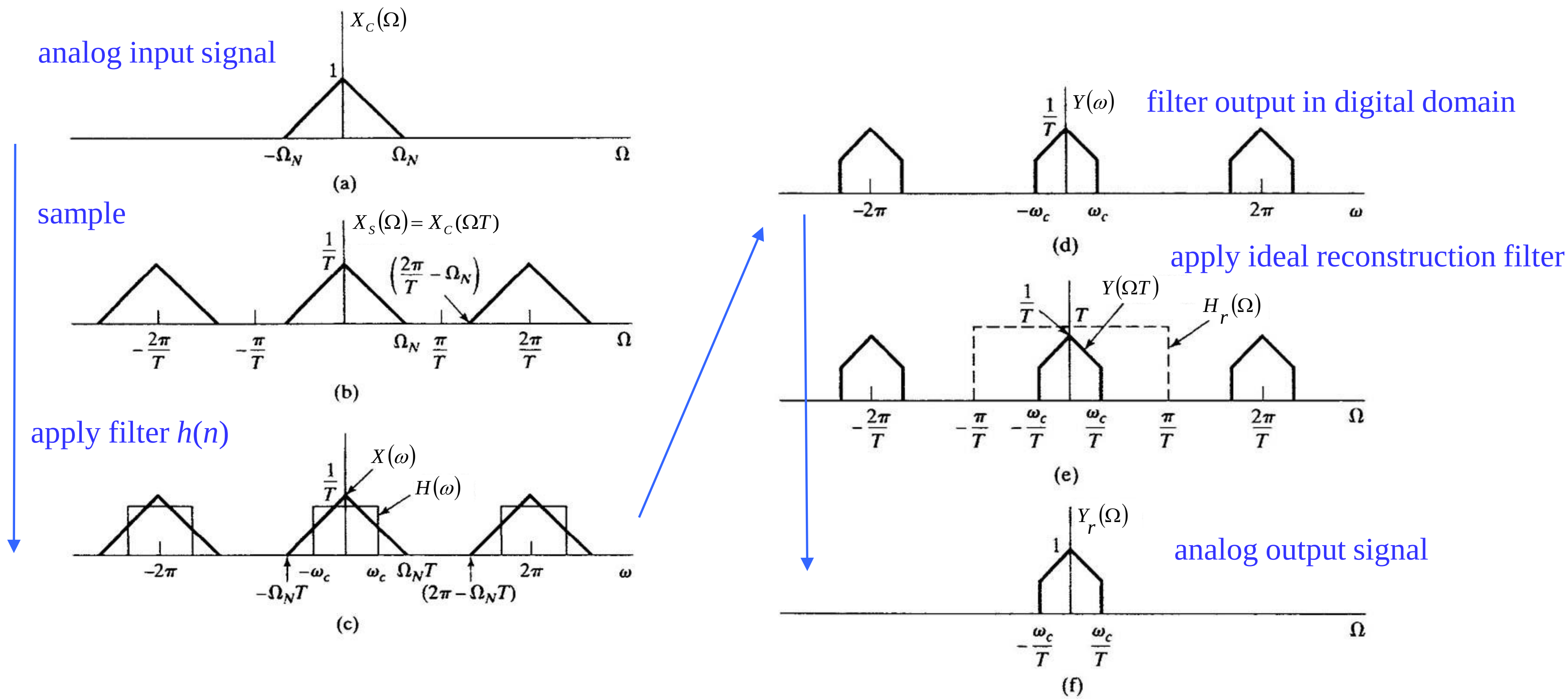
$$
\begin{aligned}
Y_r(\Omega) &= H_{\text{eff}}(\Omega) X_c(\Omega) \\
&= H_r(\Omega) H(\Omega T) X_C(\Omega T)
\end{aligned}
$$

As long as $x_c(t)$ is bandlimited and sampling is above the Nyquist rate (i.e., no aliasing):

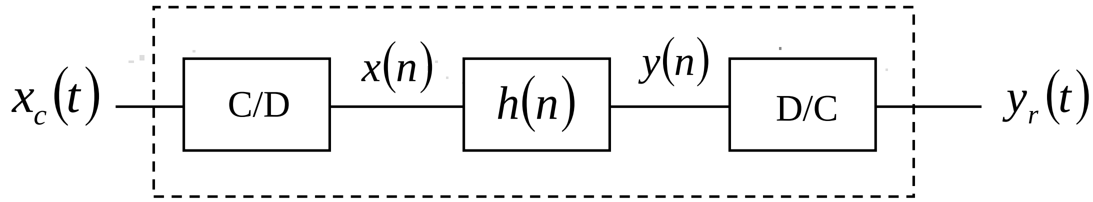
$$
H_{\text{eff}}(\Omega) = \begin{cases} H(\Omega T) & |\Omega| < \pi/T \\ 0 & |\Omega| \geq \pi/T \end{cases}
$$

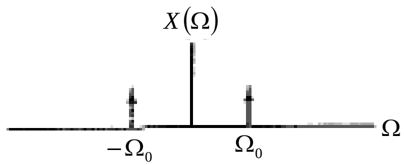
Example: Digital lowpass filtering

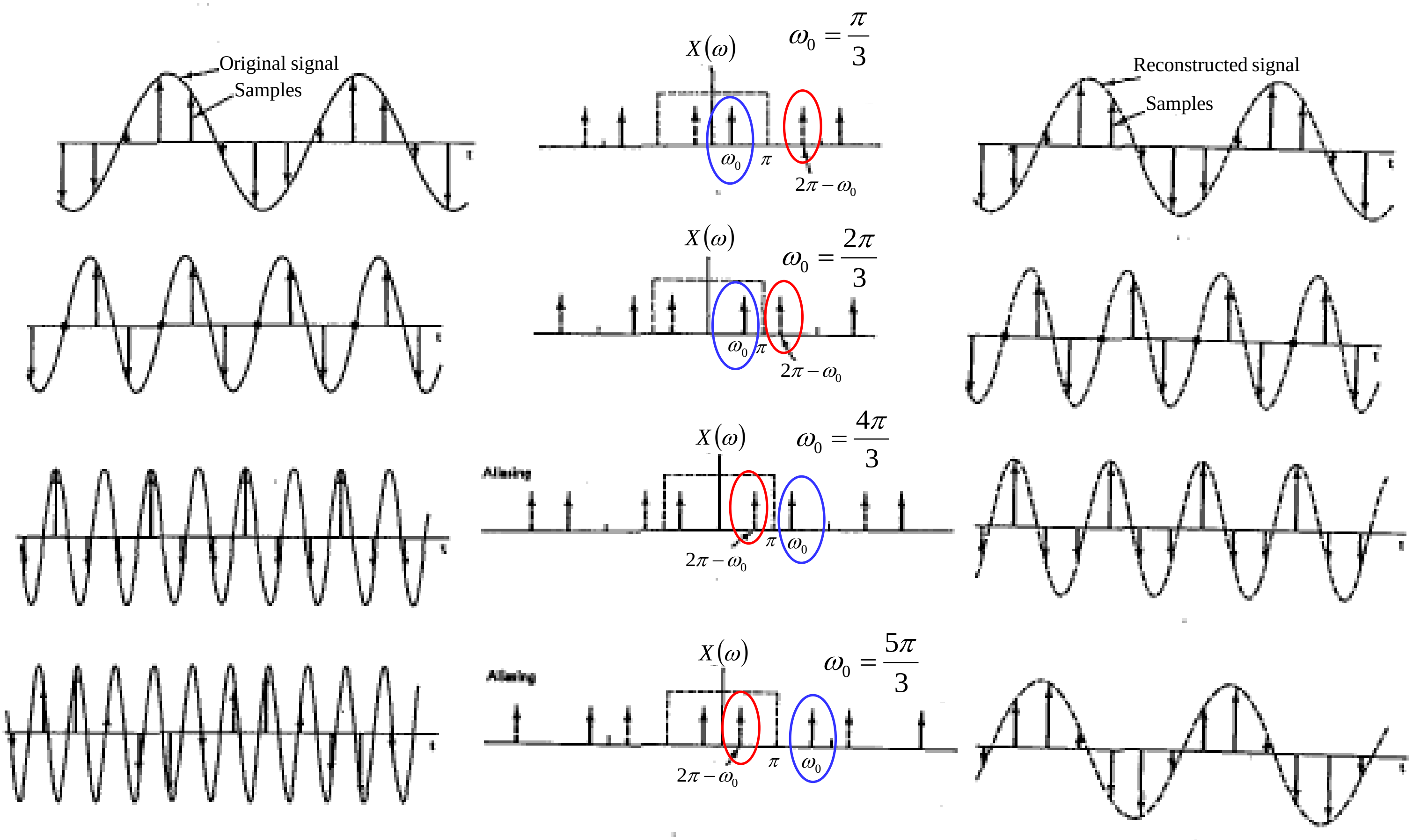




Example - Aliasing







5.5 Changing the Sampling Rate using Discrete-Time Processing

It is often more efficient or convenient to perform different parts of a processing algorithm at different sampling rates (often due to relative computational cost).

Suppose we have a sequence, $x(n)$ obtained by sampling an analog signal $x(t)$ at $t = nT$:

$$x(t)\Big|_{t=nT} = x(nT) \Rightarrow x(n) \qquad \text{for } -\infty < n < \infty$$

and we want a new sequence $y(n)$ that could be obtained by sampling $x(t)$ at $t = nT'$ (*i.e.*, at some other sampling rate).

<u>Question</u>: Can we obtain the desired sequence $y(n)$ directly from $x(n)$?

<u>Answer</u>: Yes, under certain circumstances.

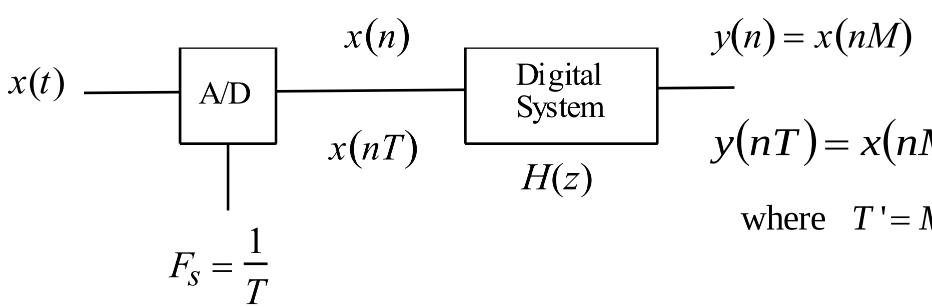
To see how this is accomplished we examine two processes:

a. **<u>Decimation</u> (down-sampling)** - compress or reduce the sampling rate. That is, produce a new sequence that would have been obtained from the original analog signal by *sampling slower*.

b. **<u>Interpolation</u> (up-sampling)** - expand or increase the sampling rate. That is, produce a new sequence that could have been obtained from the original analog signal if it had been *sampled faster*.

Decreasing the Sample Rate by an ***Integer Factor***

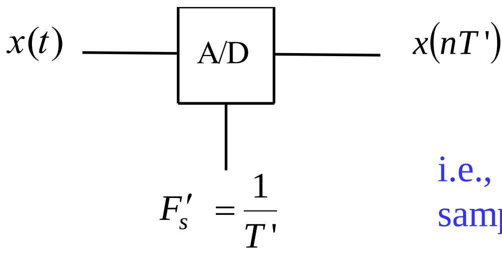
The down-sampling (decimation) process can be illustrated as



$$y(nT) = x(nMT) = x(nT')$$

where $T' = MT$

Let’s consider how we can implement $H(z)$ from the previous slide so that decimation looks like the process shown below:



i.e., it looks like $x(n)$ was obtained by sampling $x(t)$ at F_s'

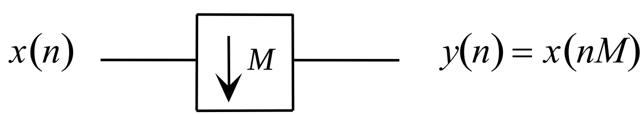
In decimation, the <u>sampling **interval** is increased</u> by an integer factor M, so that

$$T' = MT$$

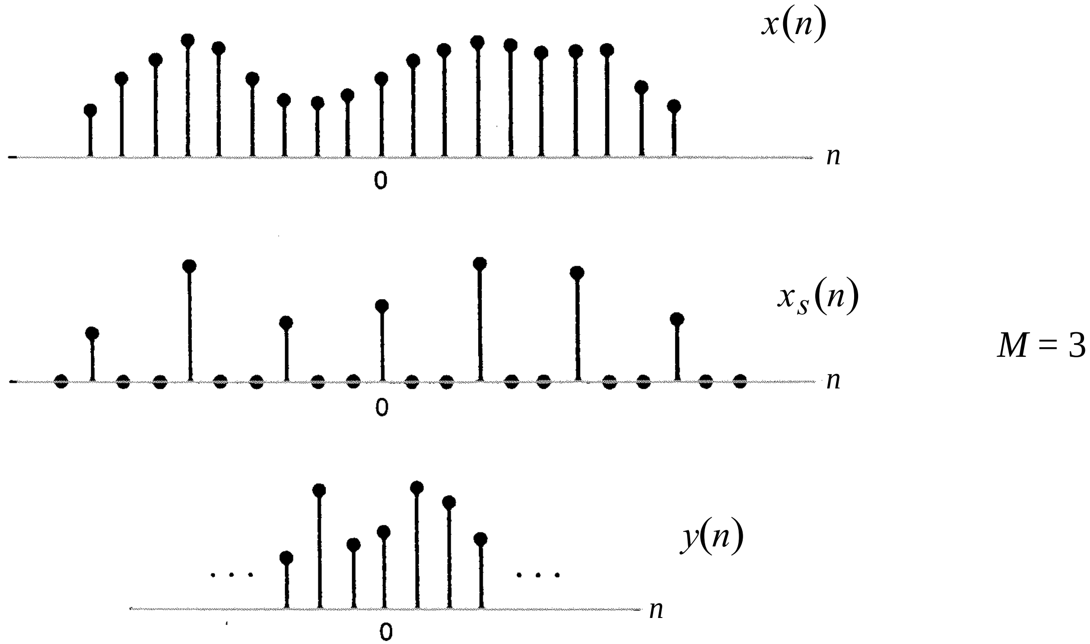
Thus the <u>sampling **rate** is reduced</u> by the same factor:

$$F_s' = \frac{1}{T'} = \frac{1}{MT} = \frac{F_s}{M}$$

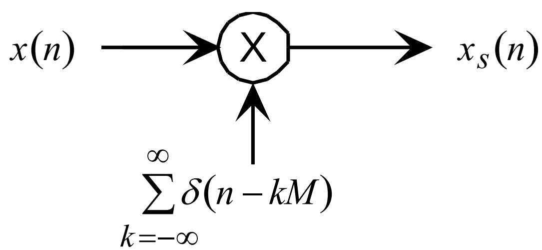
The decimation process is often represented as



In the time domain, this process is illustrated as:



Note that $x_S(n)$ could have been obtained from $x(n)$ by "sampling the samples."



Now examine decimation in the frequency domain.

$$Y(\omega) = \sum_{n=-\infty}^{\infty} y(n) e^{-j\omega n} \qquad \text{where} \quad y(n) = x(nM)$$

$$= \sum_{n=-\infty}^{\infty} x(nM) e^{-j\omega n} \qquad \text{then let } r = nM$$

$$= \sum_{r=-\infty}^{\infty} x(r) e^{-j\omega \frac{r}{M}}$$

But note that as the index r counts from $-\infty$ to ∞, <u>only certain values</u> of r within this range <u>are allowed</u>. In other words,

$$x(nM) = x(r) \qquad \text{for } r = 0,\ \pm M,\ \pm 2M, \cdots$$

That is, $x(nM) = x(r)$ at the sampling points determined by M.

A convenient representation for $x(r)$:

$$x(r)\Big|_{r=0,\pm M,\pm 2M,\cdots} = x(r)\left[\frac{1}{M}\sum_{k=0}^{M-1} e^{j\frac{2\pi kr}{M}}\right] \qquad -\infty < r < \infty$$

discrete Fourier series representation of a periodic impulse train with a period of M samples.

Therefore,

$$Y(\omega) = \sum_{r=-\infty}^{\infty} x(r)\left[\frac{1}{M}\sum_{k=0}^{M-1} e^{j\frac{2\pi kr}{M}}\right] e^{-j\frac{\omega r}{M}}$$

$$= \frac{1}{M}\sum_{k=0}^{M-1}\sum_{r=-\infty}^{\infty} x(r) e^{j\frac{2\pi kr}{M}} e^{-j\frac{\omega r}{M}}$$

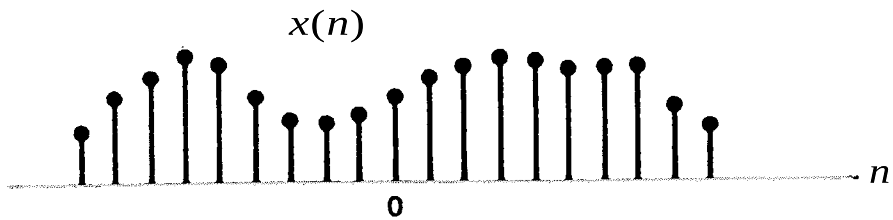
$$= \frac{1}{M}\sum_{k=0}^{M-1}\sum_{r=-\infty}^{\infty} x(r) e^{-j\left(\frac{\omega}{M}-\frac{2\pi k}{M}\right)r}$$

So that the spectrum of the output sequence $y(n)$ in terms of the spectrum of the input sequence $x(n)$ is:

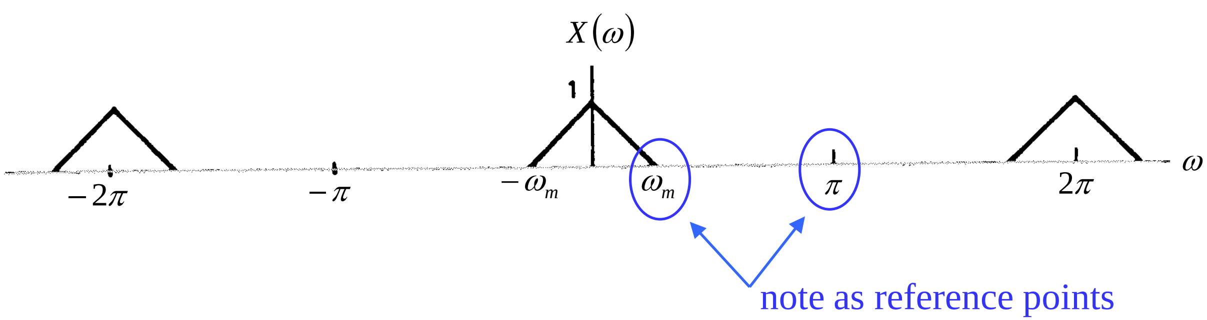
$$Y(\omega) = \frac{1}{M} \sum_{k=0}^{M-1} X\left(\frac{\omega}{M} - \frac{2\pi k}{M}\right)$$

But what does this mean?

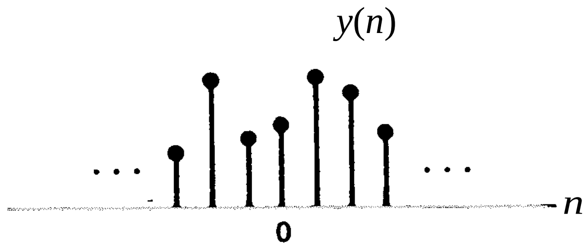
Let's look at an example. Given $x(n)$



which has the following spectrum (not really, but for the sake of illustration):



Next, decimate by $M = 3$, to obtain

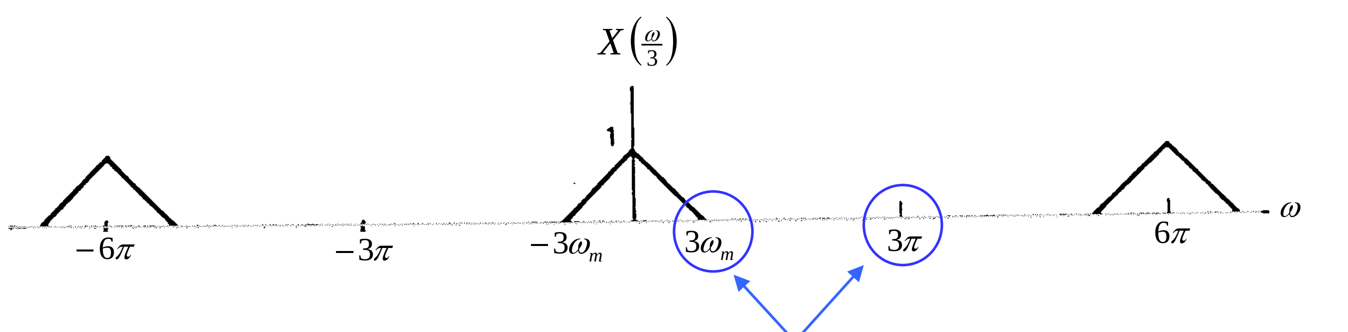


where $Y(\omega) = \frac{1}{3}\sum_{k=0}^{2} X\left(\frac{\omega}{3} - \frac{2\pi k}{3}\right)$

So that $Y(\omega) = \frac{1}{3}X\left(\frac{\omega}{3}\right) + \frac{1}{3}X\left(\frac{\omega}{3} - \frac{2\pi}{3}\right) + \frac{1}{3}X\left(\frac{\omega}{3} - \frac{4\pi}{3}\right)$

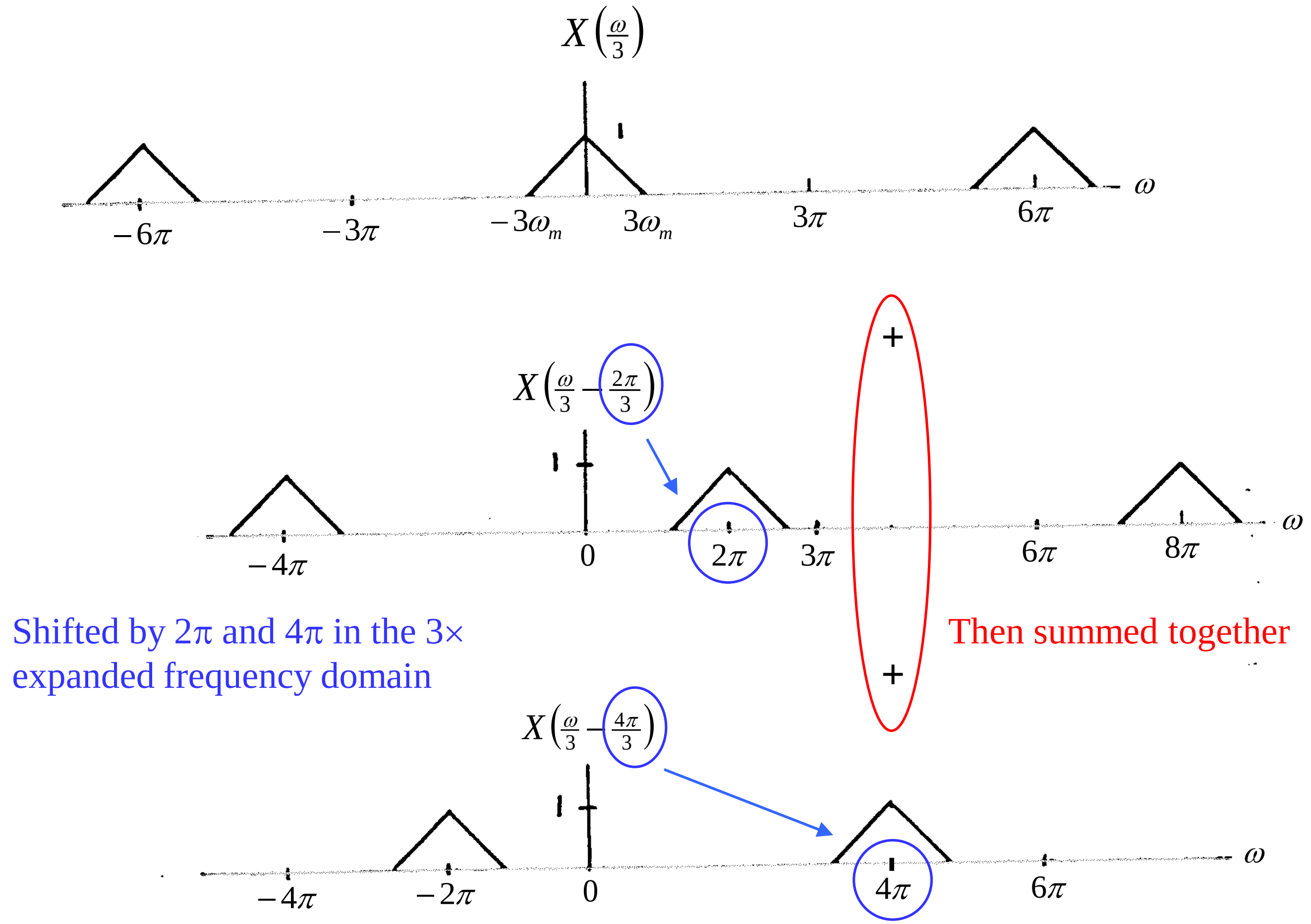
$Y(\omega)$ is composed of a <u>superposition of frequency-shifted replicas</u> of $X(\omega)$ that have been scaled (along the frequency axis) by 3.

It might be helpful to first look at

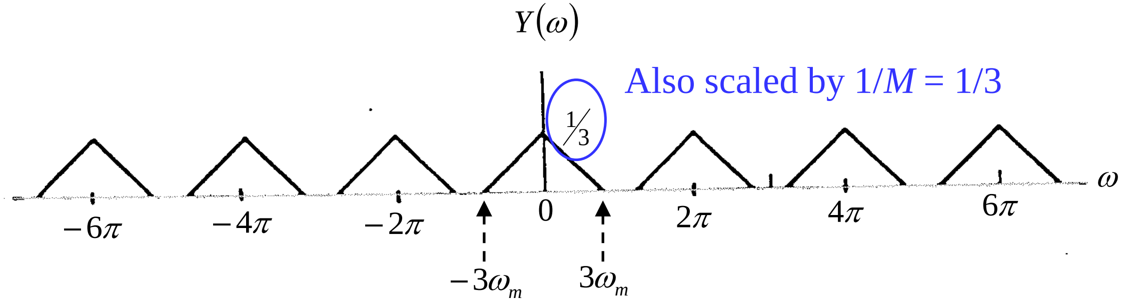


Expanded in frequency by factor of 3

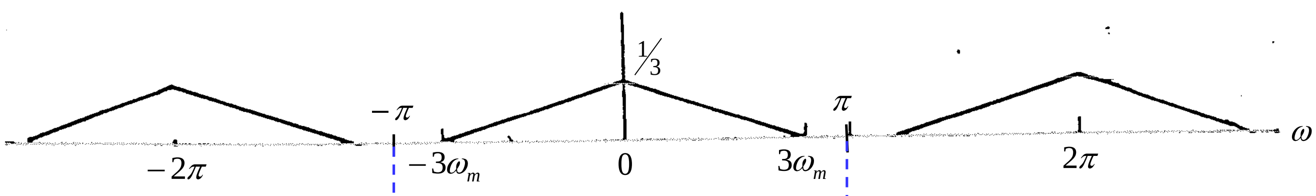
So $Y(\omega)$ is made up of the superposition of the following three spectra



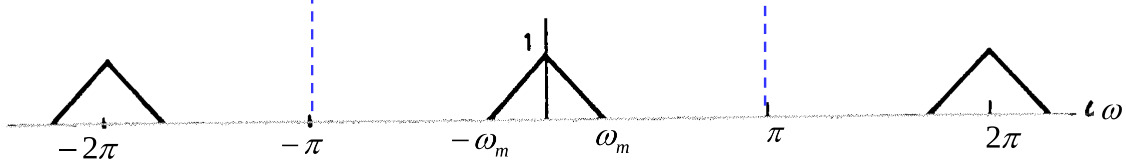
Resulting in



Or equivalently



Compare with the spectrum of $X(\omega)$



Since decimation requires throwing away samples, it is important that the decimated signal satisfy the Nyquist criterion. That is:

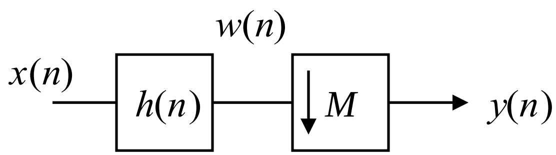
$$F_s^{'} \geq 2F_N$$

where F_N is the highest frequency in the analog signal.

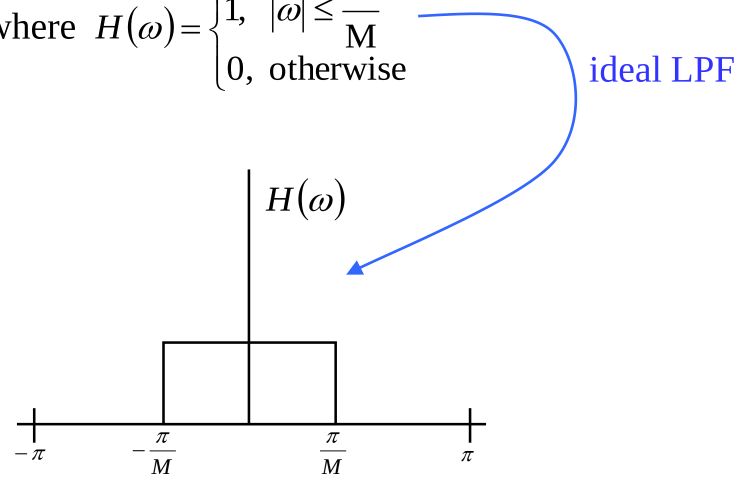
In this case, no aliasing occurs and the decimation equation can be expressed as:

$$Y(\omega) = \frac{1}{M} X\left(\omega/M\right) \qquad \text{for } |\omega| \le \pi$$

Since aliasing is undesirable, the decimator is often preceded by a lowpass filter that insures aliasing will not occur (in theory)



$$\text{where } H(\omega) = \begin{cases} 1, & |\omega| \le \dfrac{\pi}{\text{M}} \\ 0, & \text{otherwise} \end{cases}$$



Consider two (ideal) examples in the frequency domain:

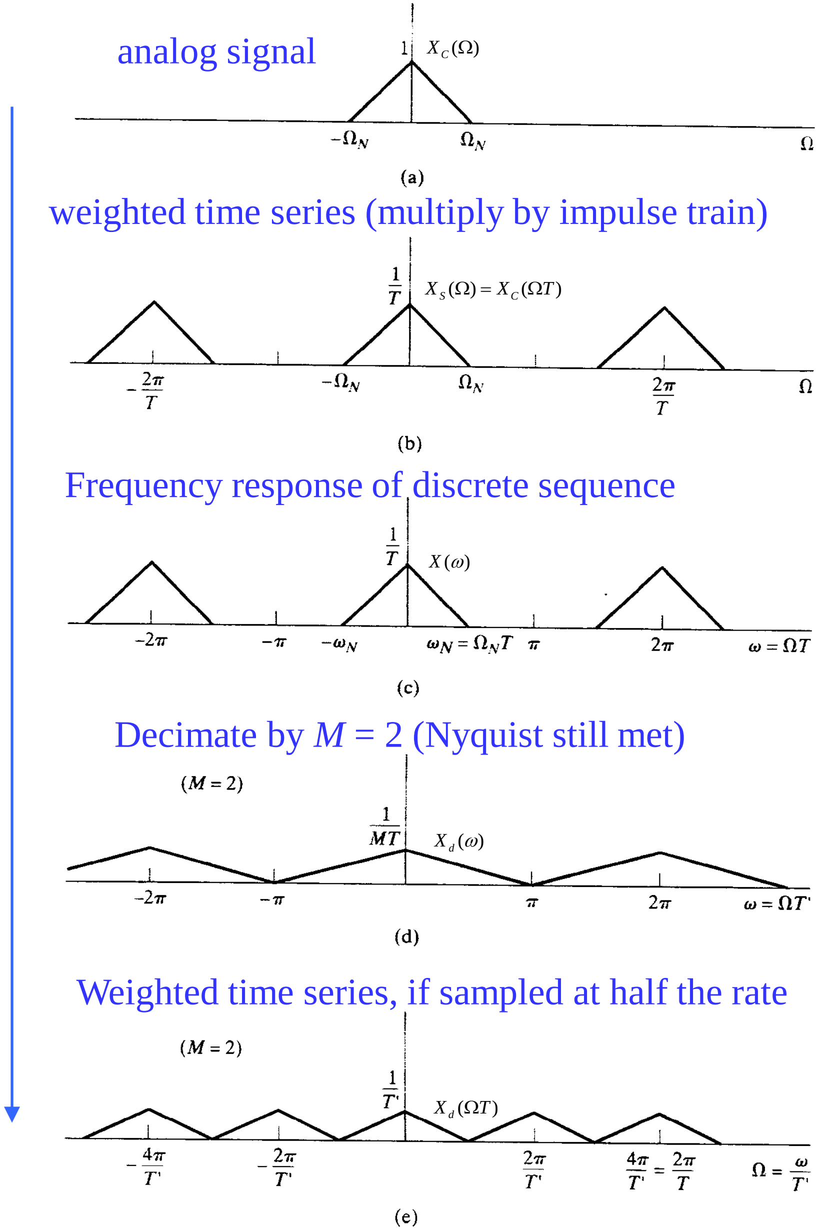


Figure 4.21 Frequency-domain illustration of downsampling.

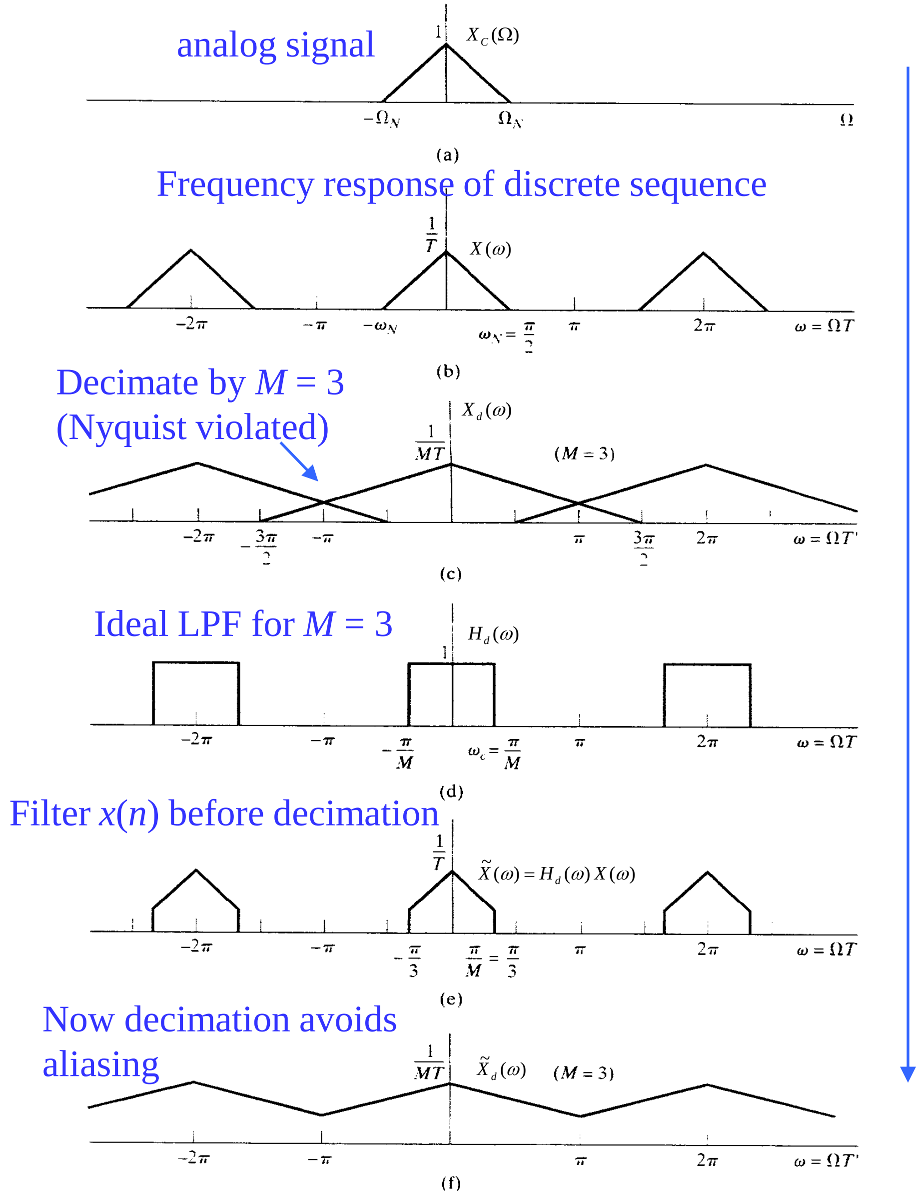


Figure 4.22 (a)–(c) Downsampling with aliasing. (d)–(f) Downsampling with prefiltering to avoid aliasing.

Increasing the Sampling Rate by an Integer Factor

When the sampling interval is decreased by an integer factor L

$$T' = \frac{T}{L}$$

then the sampling rate is increased by the same factor

$$F_s' = \frac{1}{T'} = \frac{L}{T} = LF_s$$

Now suppose we have a sequence $x(n)$ obtained by sampling $x(t)$ at $t = nT$:

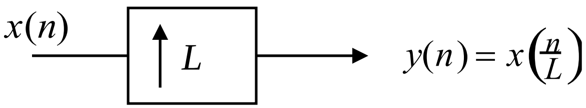
$$x(t)\Big|_{t=nT} = x(nT) \Rightarrow x(n) \qquad \text{for } -\infty < n < \infty$$

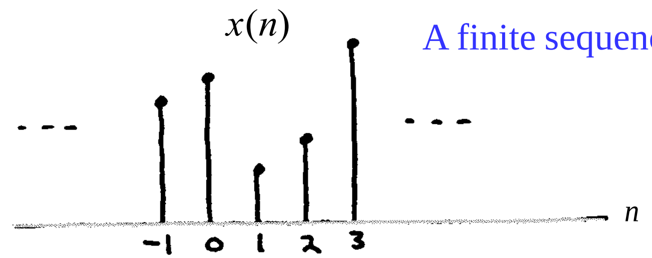
and we want a new sequence $y(n)$ that could have been obtained by sampling $x(t)$ at $t = nT'$. This implies we must interpolate L–1 new sample values between each pair of sample values of $x(n)$.

Therefore, $$y(nT) = x(nT') = x\left(\tfrac{n}{L}T\right)$$

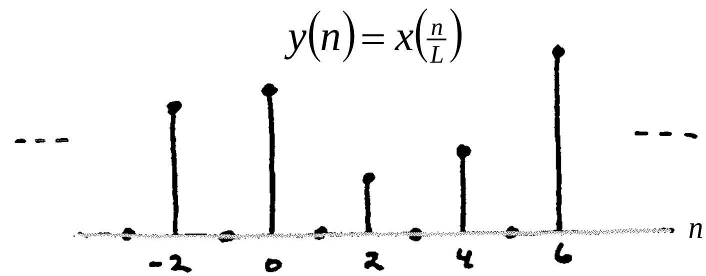
$$y(n) = x\left(\tfrac{n}{L}\right) \quad \text{for } n = 0,\ \pm L,\ \pm 2L, \cdots \ (\text{i.e., } \tfrac{n}{L} \text{ is an integer})$$

and $y(n) = 0$ otherwise

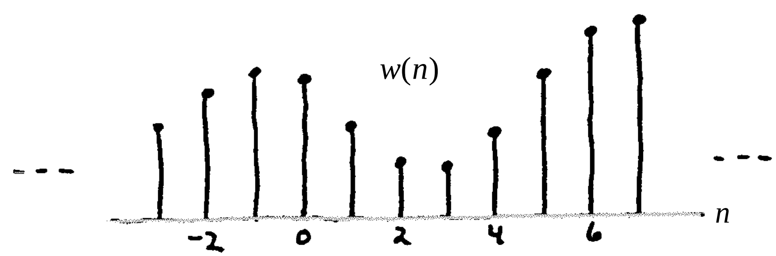




A finite sequence



L = 2 (i.e., zero-fill between each sample of $x(n)$)



A new sequence, but up-sampling does not give us this. So where does it come from?

To answer this question let's examine interpolation in the frequency domain.

$$Y(\omega) = \sum_{n=-\infty}^{\infty} y(n) e^{-j\omega n} \qquad \text{where} \qquad y(n) = x\left(\frac{n}{L}\right)$$

so

$$Y(\omega) = \sum_{n=-\infty}^{\infty} x\left(\tfrac{n}{L}\right) e^{-j\omega n}$$

then

$$Y(\omega) = \sum_{r=-\infty}^{\infty} x(r) e^{-j\omega L r} \qquad \text{where } r = \frac{n}{L} \text{ is an integer}$$

therefore

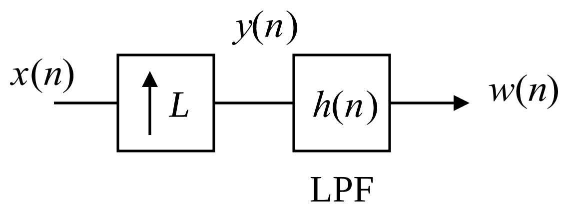
$$\boxed{Y(\omega) = X(\omega L)}$$

Note that while decimation implied a shifting & scaling of the frequency axis, up-sampling implies only a **scaling** of the frequency axis.

To complete the interpolation process, some mechanism must be included to filter out the unwanted images so that the output signal sequence truly looks as if it were sampled at a higher rate.

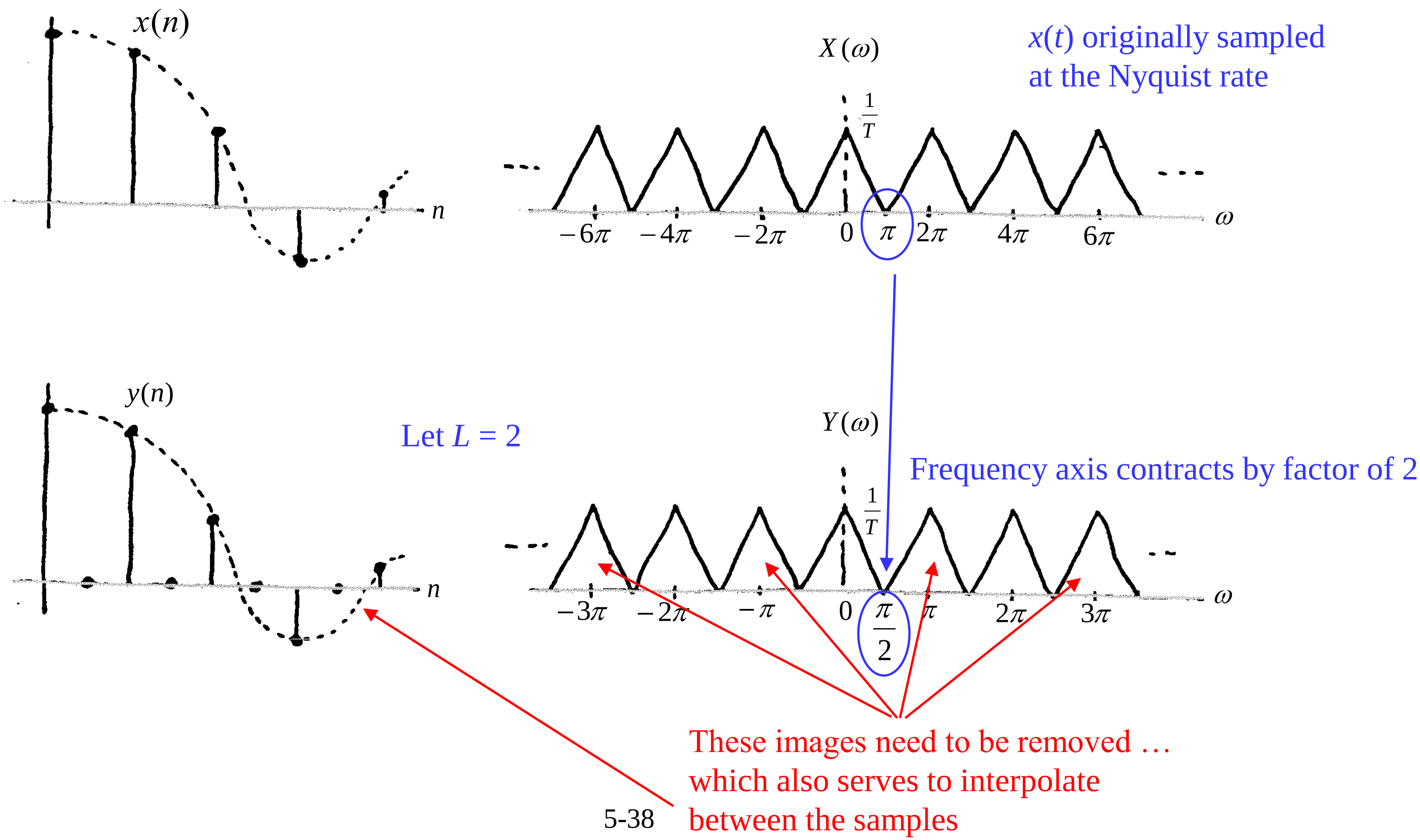
Sounds a bit like an "image rejection" filter.

The complete system is

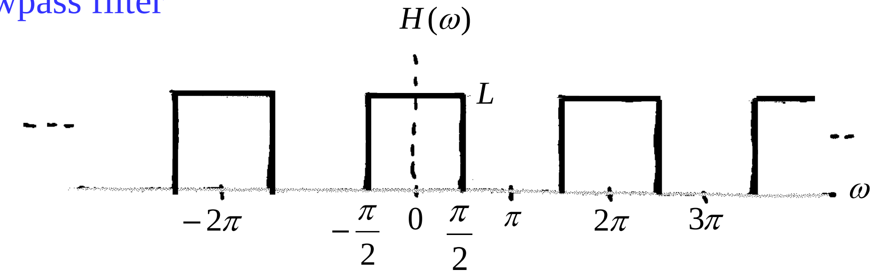


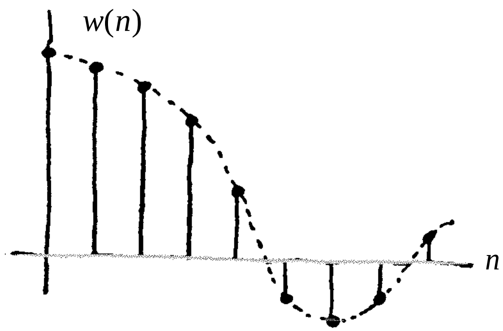
$$H(\omega)=\begin{cases} L, & |\omega|\le \frac{\pi}{L} \\ 0, & \text{otherwise} \end{cases}$$

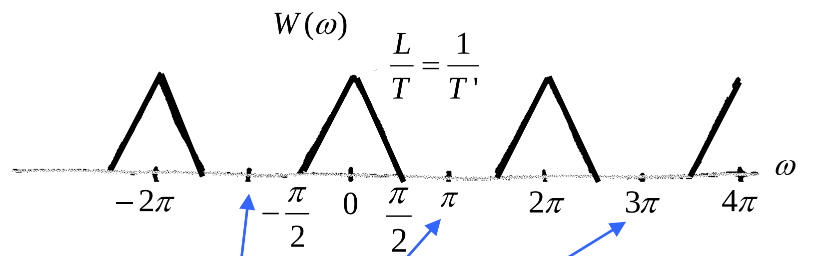
An illustration of the process follows:



Lowpass filter







Lowpass filtering removes the unwanted images,

which provides the interpolated signal.

Compact version of same example we just went through

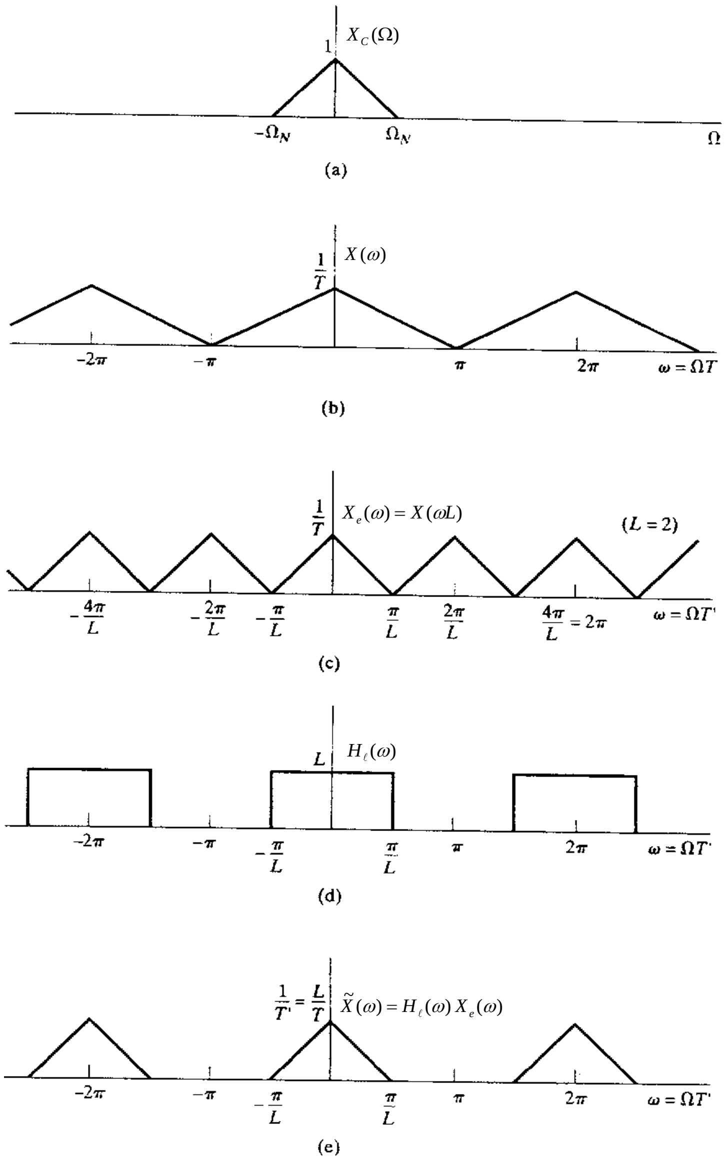


Figure 4.25 Frequency-domain illustration of interpolation.

Another interesting example is linear interpolation. It is not highly accurate but it is simple and popular.

Linear interpolation filter

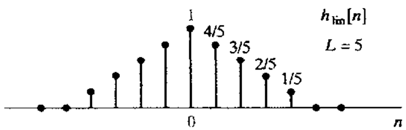
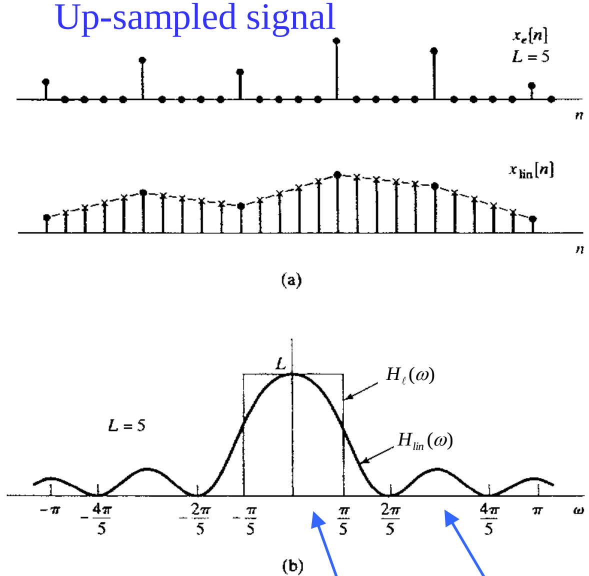


Figure 4.26 Impulse response for linear interpolation.

Up-sampled signal

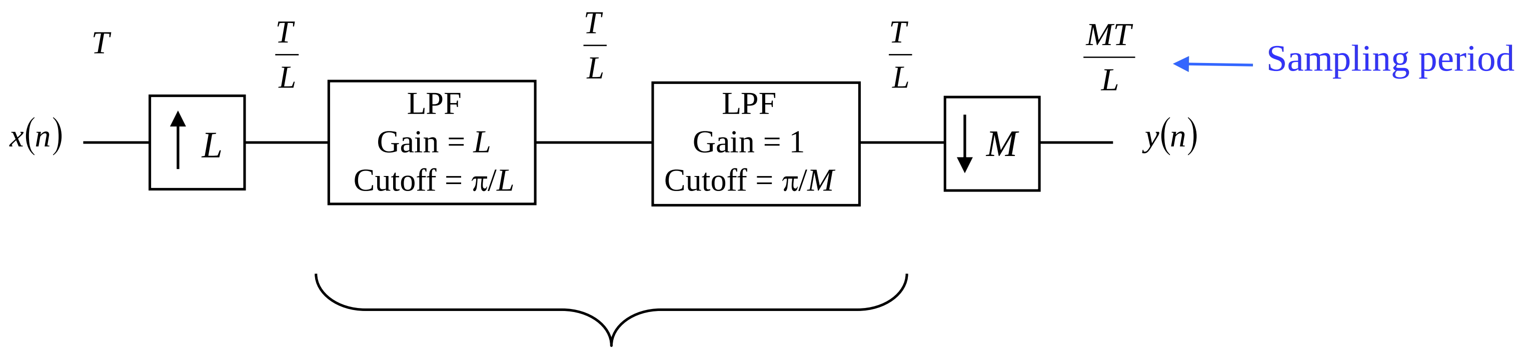


linearly interpolated signal (basically it is "connect the dots")

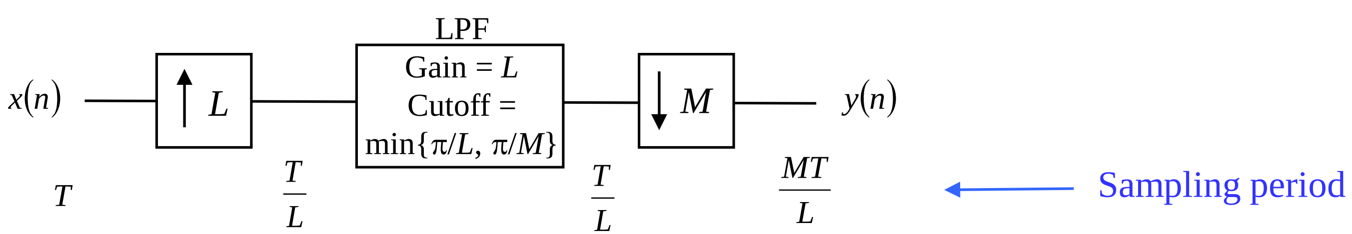
Figure 4.27 (a) Illustration of linear interpolation by filtering. (b) Frequency response of linear interpolator compared with ideal lowpass interpolation filter.

Compared to ideal filter spectral leakage now occurs

Changing the Sampling Rate by a ***Non-Integer Factor*** *(but rational)*



These two linear filters can be implemented as one

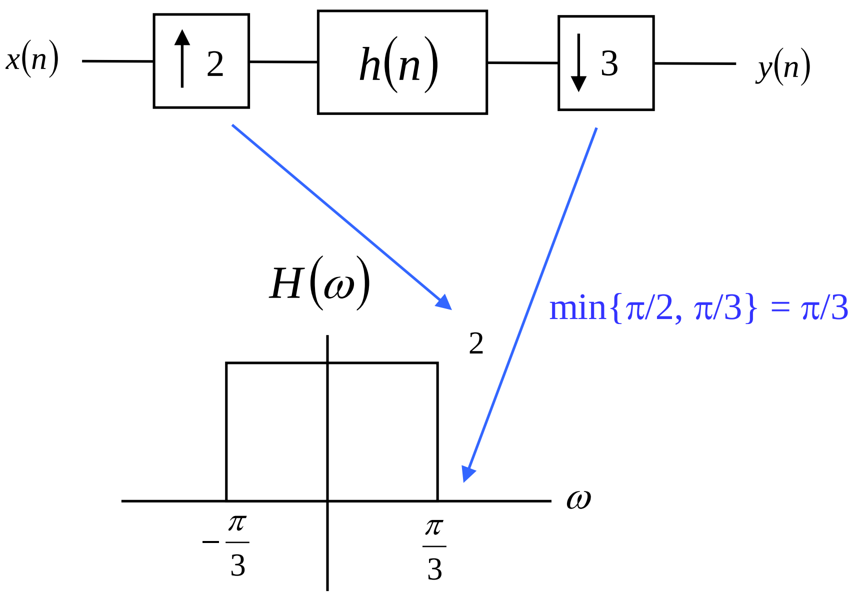


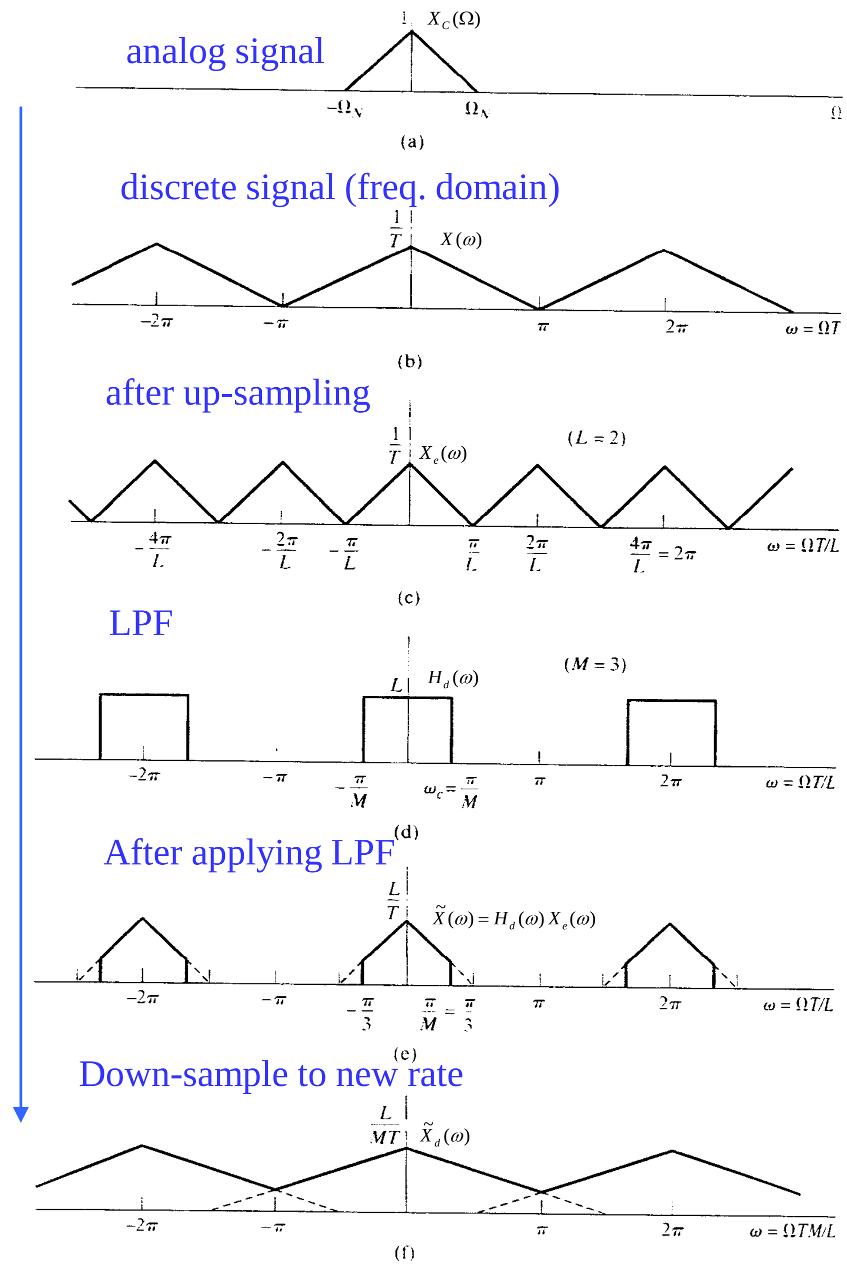
For example, if we want $T' = 1.01T$, then let $L = 100$ and $M = 101$

This 1% change does require filtering at a much higher sample rate in the middle, however.

Here’s an example:

Change the original sampling rate by 2/3:





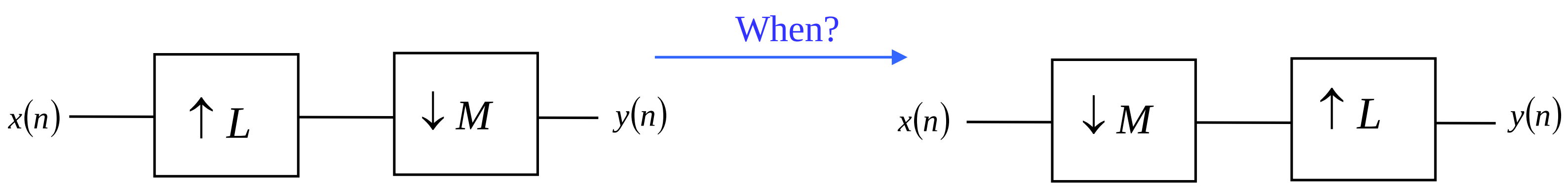
5.6 Multirate Signal Processing

From a practical point of view, implementing large decimation and interpolation factors is not efficient since high order filters and processing at a very high intermediate rate are often needed.

Multirate techniques focus on the upsampler/downsampler to increase the efficiency of DSP systems.

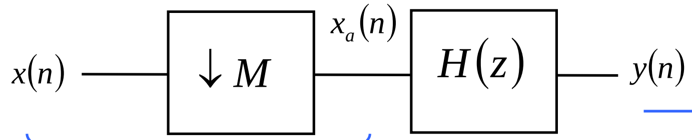
<u>*Interchanging of Filtering and Downsampling/Upsampling*</u>

To implement a fractional change in the sampling rate it follows that a cascade of an upsampler and a downsampler should be used. We would like to determine under what conditions a cascade of a factor of *M* downsampler and a factor of *L* upsampler can be interchanged. That is:



It can be shown that this interchange is possible only if *M* and *L* are relatively prime. That is, *M* and *L* do not have a common factor that is an integer $k > 1$.

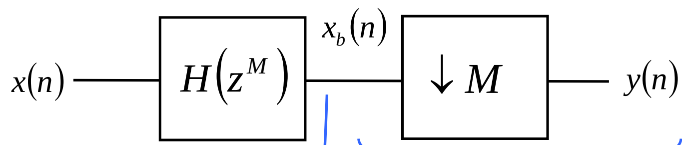
A related case is presented with decimation shown below. In one case a filter follows the down-sampler (operating at a lower sampling rate) and in the other case a filter precedes the down-sampler (operating at a higher sampling rate). Both systems are equivalent, as shown in the equations below:



$$Y(\omega) = H(\omega) X_a(\omega)$$

(i)

$$X_a(\omega) = \frac{1}{M}\sum_{k=0}^{M} X\left(\frac{\omega}{M} - \frac{2\pi k}{M}\right)$$



(ii)

$$X_b(\omega) = H(\omega M)\, X(\omega)$$

(iii)

$$Y(\omega) = \frac{1}{M}\sum_{k=0}^{M} X_b\left(\frac{\omega}{M} - \frac{2\pi k}{M}\right)$$

$$Y(\omega) = \frac{1}{M}\sum_{k=0}^{M} X\left(\frac{\omega}{M} - \frac{2\pi k}{M}\right) H(\omega - 2\pi k)$$ inserting (ii) into (iii)

$$H(\omega - 2\pi k) = H(\omega)$$

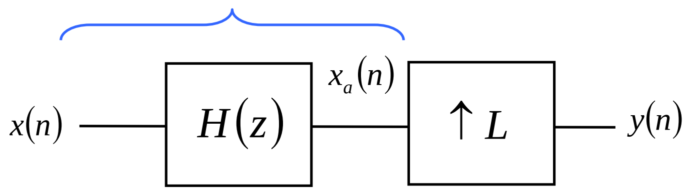
$$Y(\omega) = H(\omega)\frac{1}{M}\sum_{k=0}^{M} X\left(\frac{\omega}{M} - \frac{2\pi k}{M}\right) = H(\omega) X_a(\omega)$$

this is just $X_a(\omega)$ from (i)

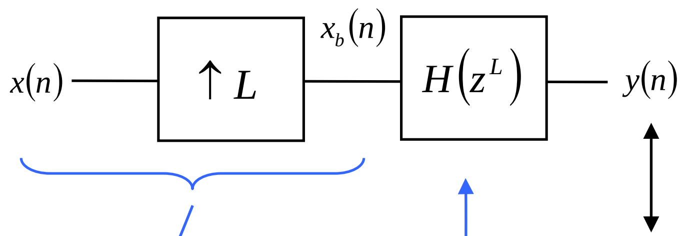
<u>Take away</u>: we can swap the down-sampler / filter order if the filter has a special up-sampled structure

A similar principle applies to interpolation:

$$X_a(\omega) = X(\omega)H(\omega)$$



$$Y(\omega) = X_a(\omega L) = X(\omega L)H(\omega L)$$



$$Y(\omega) = H(\omega L)X_b(\omega) = H(\omega L)X(\omega L)$$

$$X_b(\omega) = X(\omega L)$$

<u>Take away</u>: if we can express the LPF filter in an "up-sampled" form, then we can move it before the actual up-sampling stage.

But why would we ever have a filter like that … and how do these decimation/interpolation filter order changes help us?

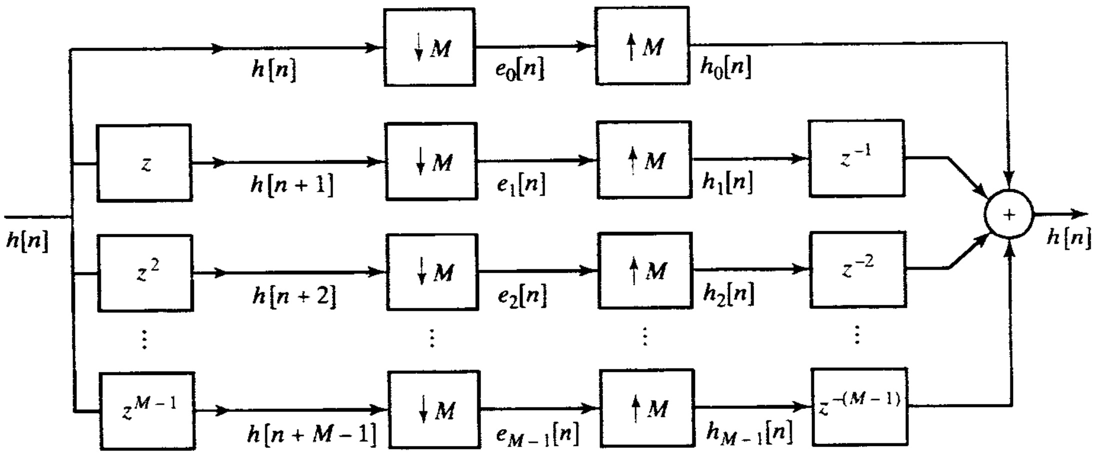
Polyphase Decomposition

Representing a sequence (or impulse response) as M subsequences, each consisting of every M^{th} value of successively delayed versions of the sequence.

This approach can lead to efficient implementation of filters

$$h_k(n) = \begin{cases} h(n+k) & n = \text{integer multiple of } M \\ 0 & \text{otherwise} \end{cases}$$

Note: Decomposition, not filtering operation!

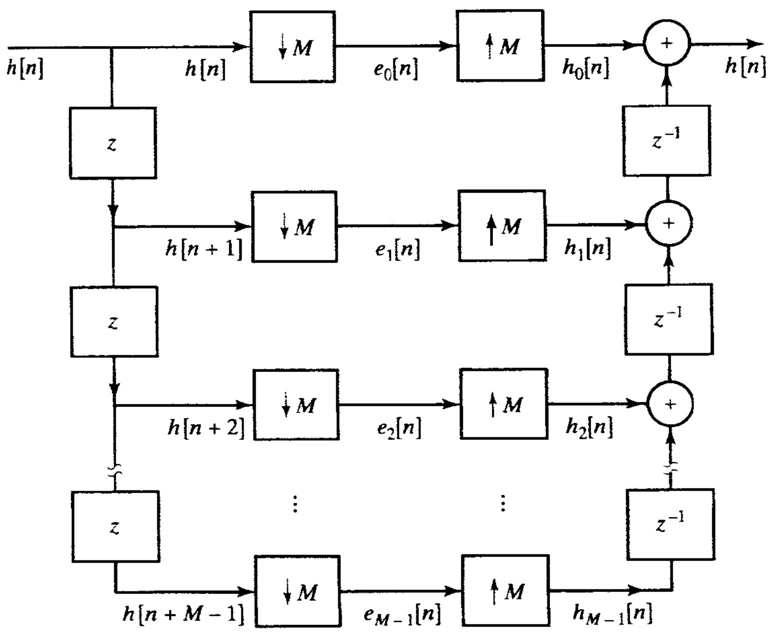


$$h(n) = \sum_{k=0}^{M-1} h_k(n-k)$$

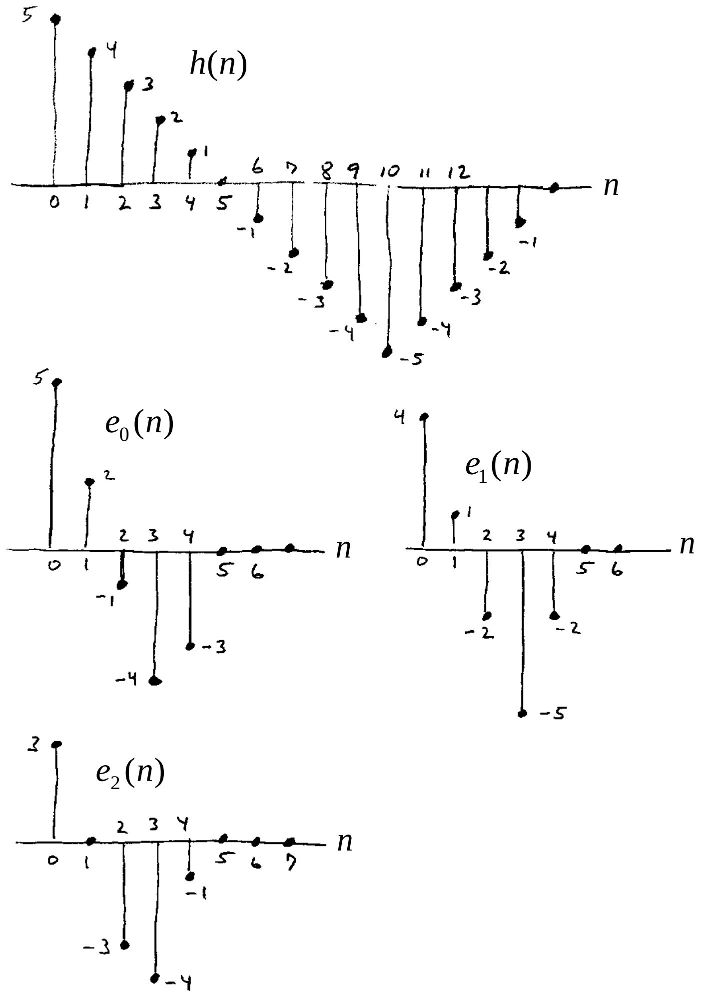
$$e_k(n) = h(nM+k) = h_k(nM)$$

These are the “polyphase components” of $h(n)$

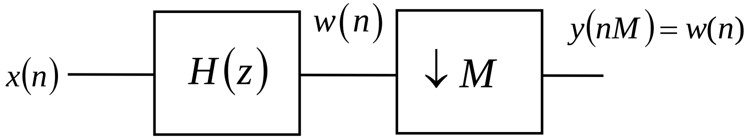
An equivalent representation is shown below. Note these are not filter realizations, but rather, they show how a filter can be decomposed into parallel filters.



Example of a polyphase decomposition with $M = 3$

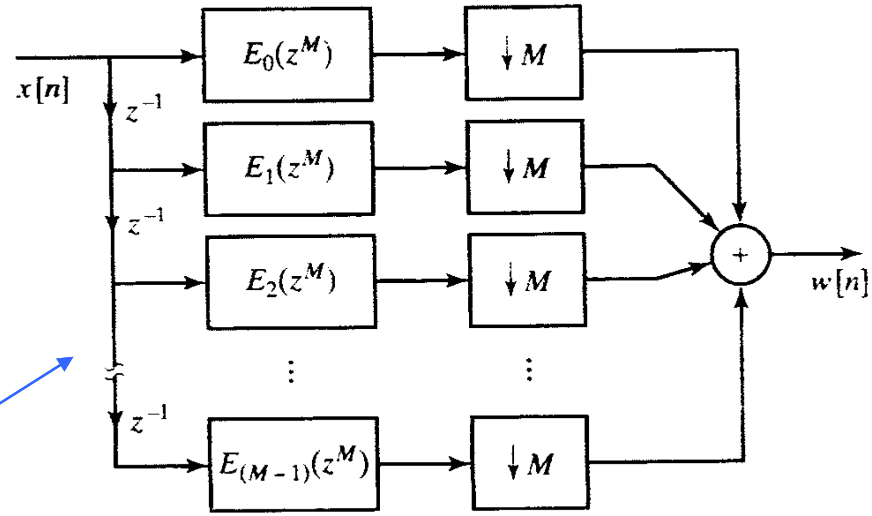


Polyphase Implementation of Decimation Filters



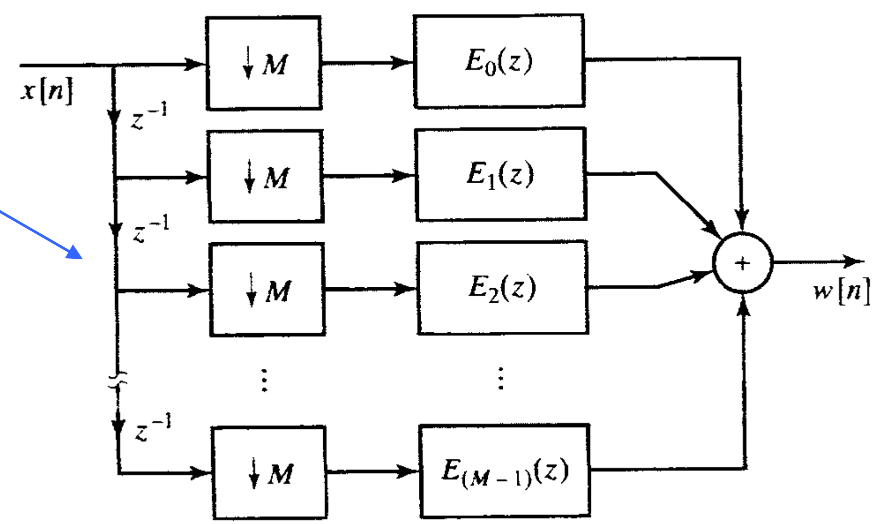
$$e_k(n) = h(nM + k) = h_k(nM)$$

$$H(z) = \sum_{k=0}^{M-1} E_k\left(z^M\right) z^{-k}$$



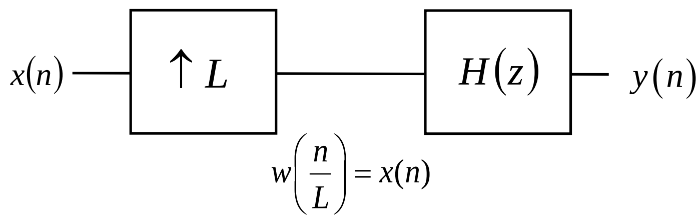
filter-then-decimate

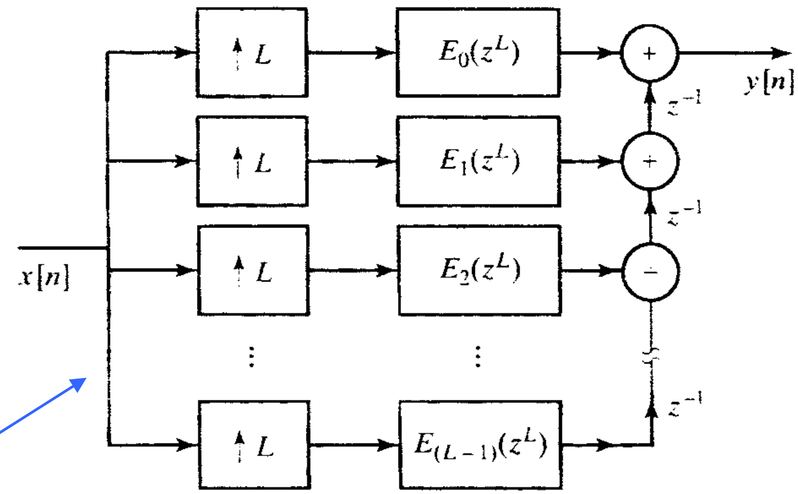
decimate-then-filter



So we can perform filtering at the lower rate (after the down-sampling)

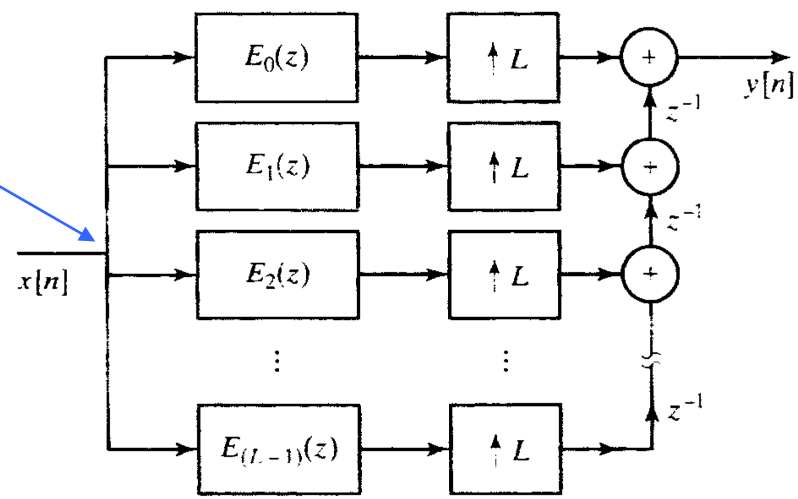
Polyphase Implementation of Interpolation Filters





interpolate-then-filter

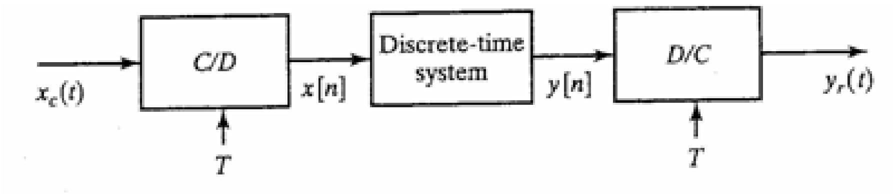
filter-then-interpolate



So we can perform filtering at the lower rate (before the up-sampling)

5.7 Digital Processing of Analog Signals

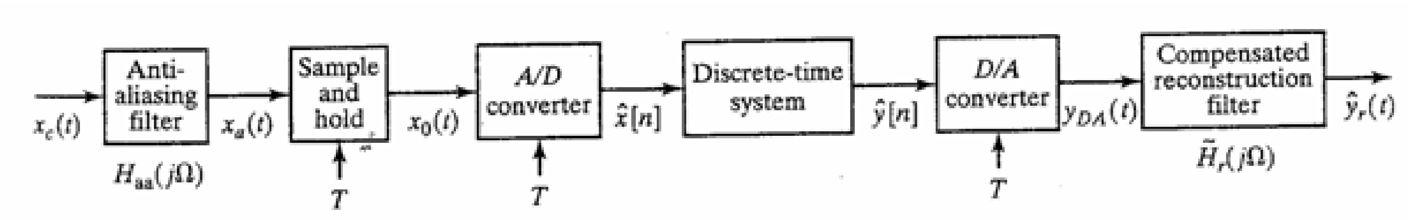
All previous signal conversions were ideal as described in the diagram below:



However, practical considerations reveal the following sources of error:

- continuous-time signals are not bandlimited
- ideal filters cannot be realized
- ideal C/D and D/C converters cannot be realized

The way in which signals are converted in actuality is shown below



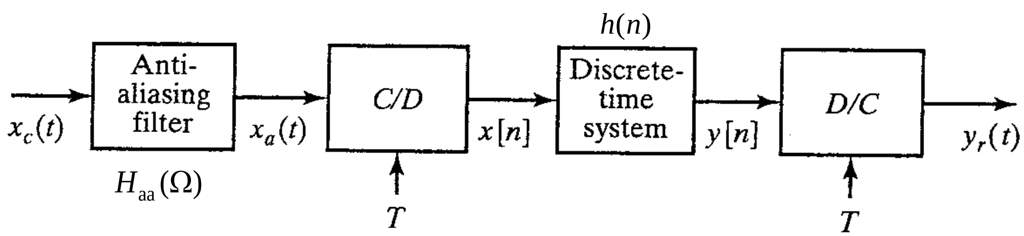
A/D and D/A converters have non-ideal behavior that impacts the entire system.

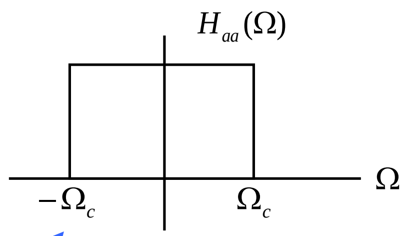
Of course, the general approach is to minimize the sampling rate. Why?

Note that even for ideal C/D converters an anti-aliasing (aa) filter is needed in order to "bandlimit" the input signal.

highest frequency component after $H_{aa}(\Omega)$

$$|\Omega| < \Omega_c$$





$$\Omega_c \le \frac{\pi}{T}$$

Note that:

$$H_{\text{eff}} = \begin{cases} H(\Omega T), & |\Omega| \le \dfrac{\pi}{T} \\ 0, & |\Omega| > \dfrac{\pi}{T} \end{cases}$$

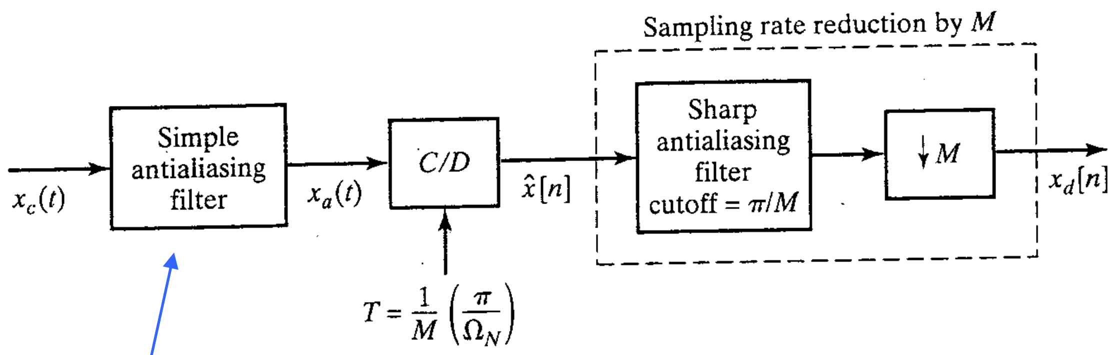
But with the anti-aliasing filter, this becomes

$$H_{\text{eff}}(\Omega) = H_{\text{aa}}(\Omega) H(\Omega), \quad |\Omega| < \Omega_c$$

Note: still shown as ideal, but

With this approach, a sharp cutoff anti-aliasing LPF is required. We would like to avoid the cost of such filters by using DSP techniques.

Consider the following system:



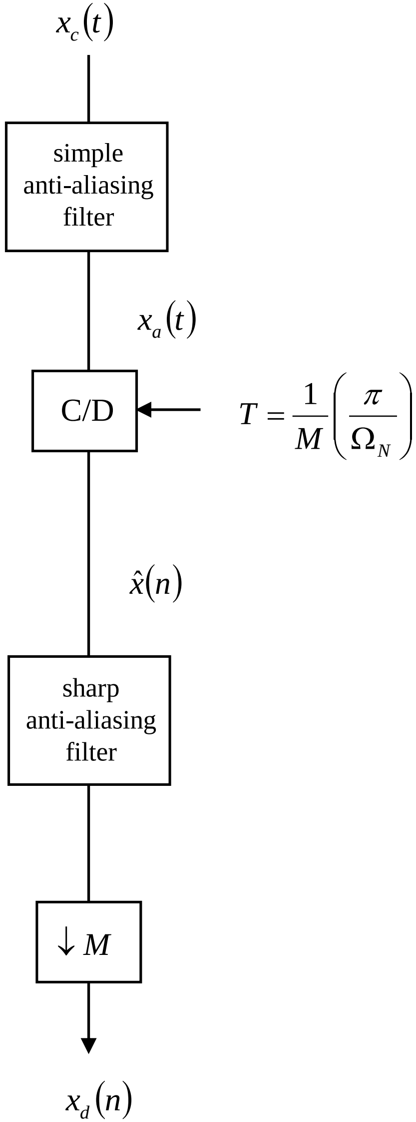
Ω_N is the highest frequency component after overall anti-aliasing filtering

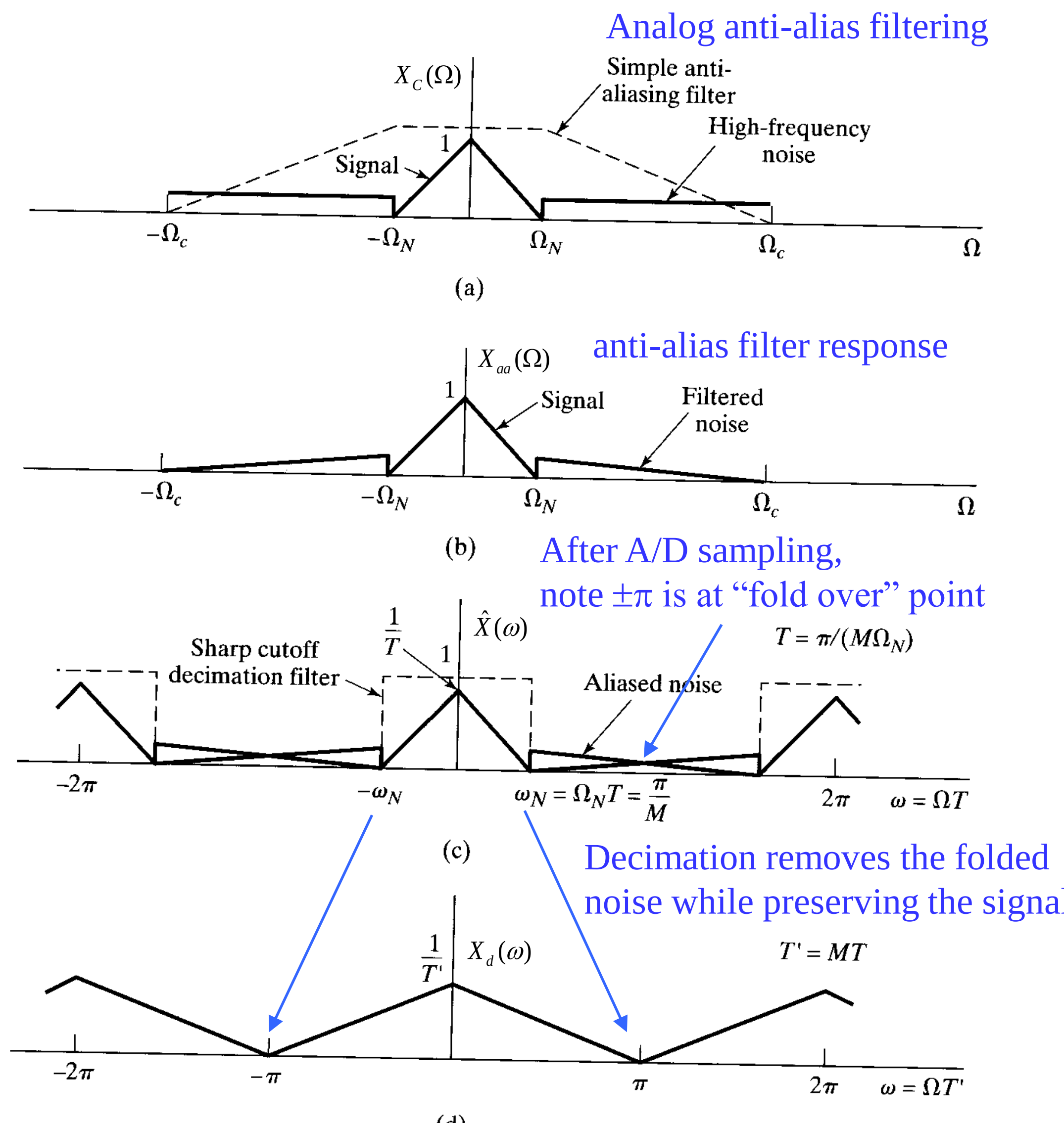
The simple anti-aliasing filter has gradual cutoff but high attenuation at $M\Omega_N$

Here is the approach:

1. Sample much higher than $2\Omega_N$, say $2M\Omega_N$
2. Follow by sharp anti-aliasing filter in digital domain
3. Then down-sample by M

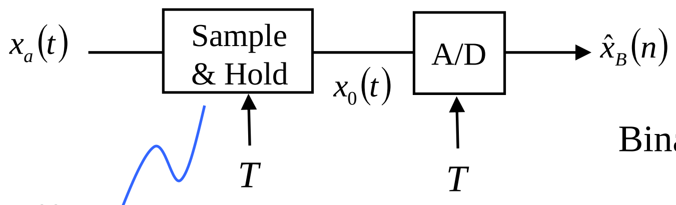
Combined, these two steps are decimation


$x_c(t)$
simple
anti-aliasing
filter
$x_a(t)$
C/D
$T = \frac{1}{M}\left(\frac{\pi}{\Omega_N}\right)$
$\hat{x}(n)$
sharp
anti-aliasing
filter
$\downarrow M$
$x_d(n)$

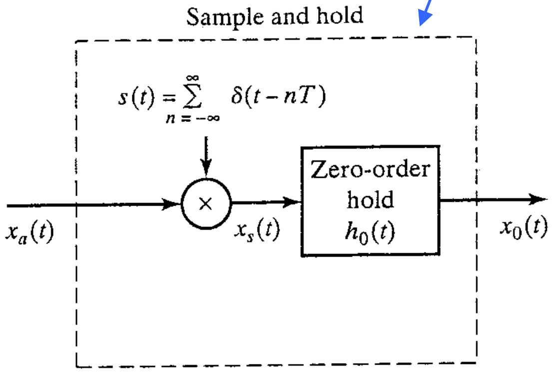

Analog anti-alias filtering
$X_c(\Omega)$
Simple anti-
aliasing filter
High-frequency
noise
Signal
1
$-\Omega_c$
$-\Omega_N$
Ω_N
Ω_c
Ω
(a)
anti-alias filter response
$X_{aa}(\Omega)$
1
Signal
Filtered
noise
$-\Omega_c$
$-\Omega_N$
Ω_N
Ω_c
Ω
(b)
After A/D sampling,
note ±π is at "fold over" point
$\frac{1}{T}$
$\hat{X}(\omega)$
1
Sharp cutoff
decimation filter
Aliased noise
$T = \pi/(M\Omega_N)$
-2π
$-\omega_N$
$\omega_N = \Omega_N T = \frac{\pi}{M}$
2π
$\omega = \Omega T$
(c)
Decimation removes the folded
noise while preserving the signal
$\frac{1}{T'}$
$X_d(\omega)$
$T' = MT$
-2π
$-\pi$
π
2π
$\omega = \Omega T'$

A/D Conversion

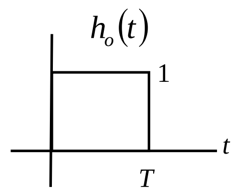
The A/D converter is an approximation to the C/D converter as shown below:



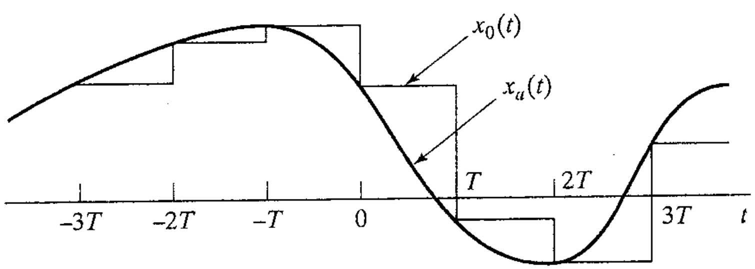
Binary numbers of B bit accuracy



(a)



$$x_o(t) = h_o(t) * \sum_{n=-\infty}^{\infty} x_a(nT)\delta(t-nT)$$



(b)

We would like the sampling to be instantaneous and the hold to keep the sample value constant for T-seconds.

The next step in the conversion process is the quantizer and encoder. Although represented separately, quantizing and encoding are usually one step

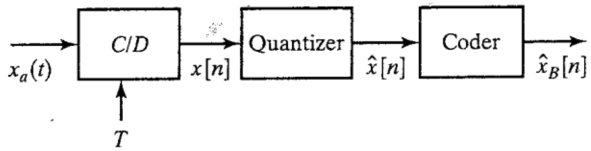
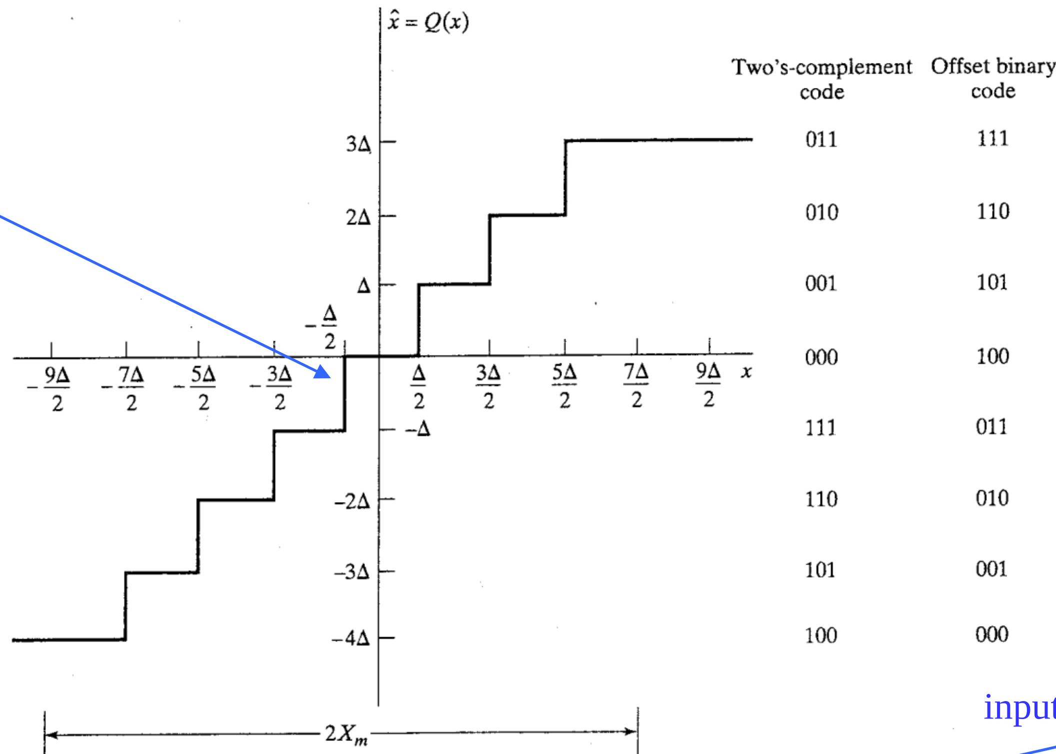


Figure 4.47 Conceptual representation

The quantizer is a nonlinear system having this as its input/output characteristic

This example is a "mid-tread" quantizer. In contrast, a "mid-rise" quantizer is symmetric about zero.



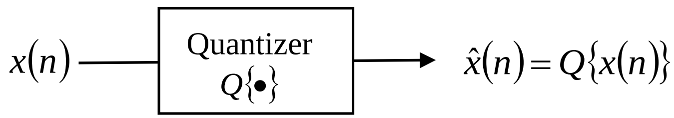
input dynamic range

Quantizing – rounding off sample values to the nearest permissible output value

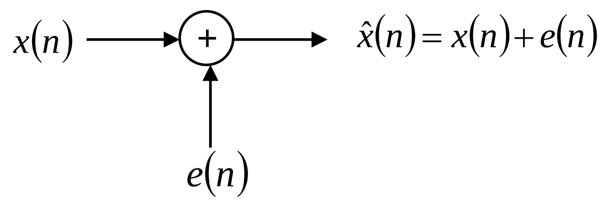
Encoding – representing each permissible sample value with a binary digital number

Analysis of Quantization Errors

Quantization is an approximation, and therefore introduces error in the quantized signal sample.



We can define the quantizing error as $e(n) = \hat{x}(n) - x(n)$ so that



The quantizer error is bounded by

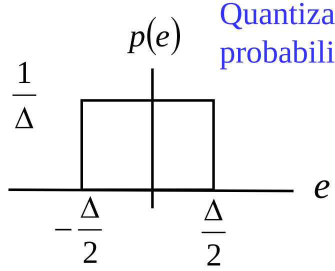
$$-\frac{\Delta}{2} < e(n) \leq \frac{\Delta}{2} \qquad \text{where} \qquad \Delta = \frac{2X_m}{2^{B+1}} = \frac{X_m}{2^B}$$

X_m is the full scale level

$2X_m$ is the dynamic range

Quantization error is represented statistically, which requires the following assumptions:

- The error sequence is a stationary random process.
- The error sequence is uncorrelated with input sequence $x(n)$.
- The error sequence is a white noise random process (i.e. error sequence samples are uncorrelated).
- The PDF of the error function is uniform over the range of quantization error.



Quantization error probability distribution

$$\Delta = 2^{-B} X_m$$

The variance of the error sequence is effectively the quantizer "noise power"

$$\sigma_e^2 = \int_{-\Delta/2}^{\Delta/2} e^2 \frac{1}{\Delta} de = \frac{\Delta^2}{12}$$

in terms of the quantizer parameters

$$\sigma_e^2 = \frac{2^{-2B} X_m^2}{12}$$

Noise power alone is not the most useful way of characterizing the impact of quantizing error. The signal-to-quantization-noise ratio (SQNR) is more valuable as a metric:

"signal power"

$$\text{SQNR} = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_e^2}\right) = 10\log_{10}\left(\frac{12 \cdot 2^{2B} \sigma_x^2}{X_m^2}\right) = 6.02B + 10.8 - 20\log_{10}\left(\frac{X_m}{\sigma_x}\right)$$

$$\text{SQNR} = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_e^2}\right) = 10\log_{10}\left(\frac{12\cdot 2^{2B}\sigma_x^2}{X_m^2}\right) = 6.02B + 10.8 - 20\log_{10}\left(\frac{X_m}{\sigma_x}\right)$$

SQNR increases 6 dB for each additional bit of quantizer accuracy (i.e. doubling # of quantizing levels)

- if $\sigma_x > X_m$ clipping will occur
- when σ_x gets smaller, the SQNR decreases
- the "optimum" case encompasses the entire signal,
- assuming Gaussian input, "optimal" is approx. by

$X_m = 3\,\sigma_x$ which covers >99% of the signal

Example: A sinusoid $x(t) = A\cos(\Omega_o T)$

With signal power: $\sigma_x^2 = \frac{1}{T}\int_0^T \left[A\cos(\Omega_o T)\right]^2 dt = \frac{A^2}{2}$ → $\sigma_x = \frac{A}{\sqrt{2}}$

Adjust the amplitude of $x(t)$ to extend across the dynamic range of the quantizer: $X_m = A$

$$\text{SQNR} = 6.02B + 10.8 - 20\log_{10}\left(\sqrt{2}\right)$$

$$\text{SQNR} = 6.02B + 7.79$$

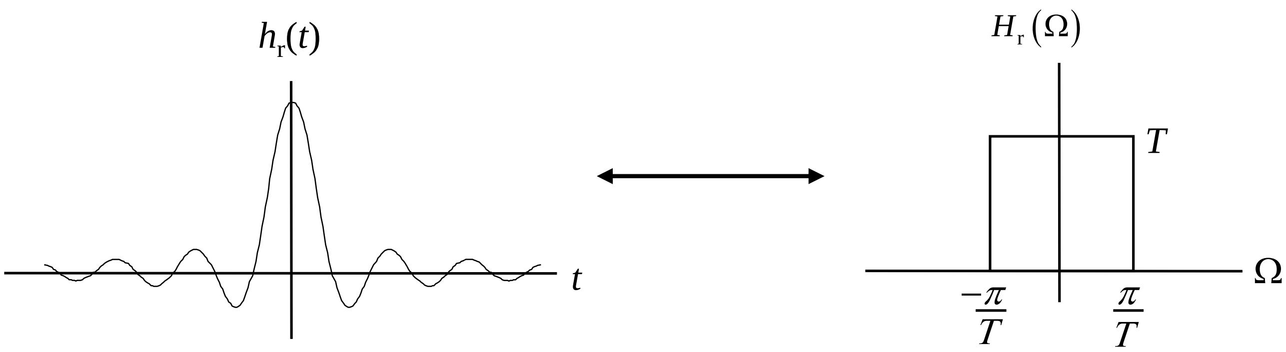
Can use to determine # bits needed to achieve a desired SQNR.

=> Remember to add a sign bit afterwards (this only accounts for interval [0 X_m], and need to quantize over [$-X_m$ X_m]

D/A Conversion

Previously, for a bandlimited signal sampled at (or above) the Nyquist rate we showed that the ideal analog reconstruction filter (or interpolation function) has the form

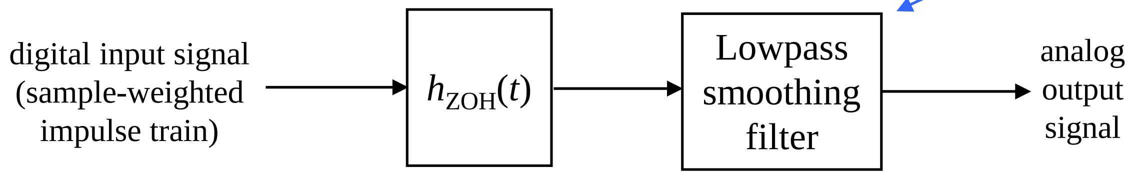
$$h_r(t) = \frac{\sin \frac{\pi}{T} t}{\frac{\pi}{T} t} \longleftrightarrow H_r(\omega) = \begin{cases} T, & |\Omega| \le \dfrac{\pi}{T} = 2\pi \cdot \dfrac{F_S}{2} \\ 0, & |\Omega| > \dfrac{\pi}{T} \end{cases}$$



Note that the ideal reconstruction filter extends over all time. Hence, it is noncausal and physically unrealizable. We now consider some practical reconstruction filters.

Zero-Order Hold

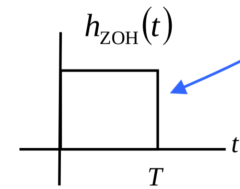
The simplest reconstruction filter is implemented by holding the sample value as a constant over the length of the sampling period T. Note that this is the same approach used for A/D conversion.



sharp transition creates undesirable high frequency components, need LPF to remove

where

$$h_{ZOH}(t) = \begin{cases} 1, & 0 \le t \le T \\ 0, & \text{otherwise} \end{cases}$$

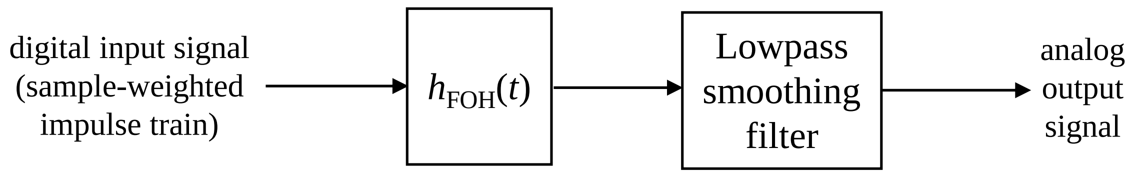


or

$$\hat{x}(t) = x(nT) \qquad nT \le t < (n+1)T$$

First-Order Hold

First-order hold approximates $x(t)$ by straight-line segments that have a slope determined by the current sample $x(nT)$ and the previous sample $x(nT-T)$.



FOH still requires LPF
because it generates undesirable
high frequency components

where

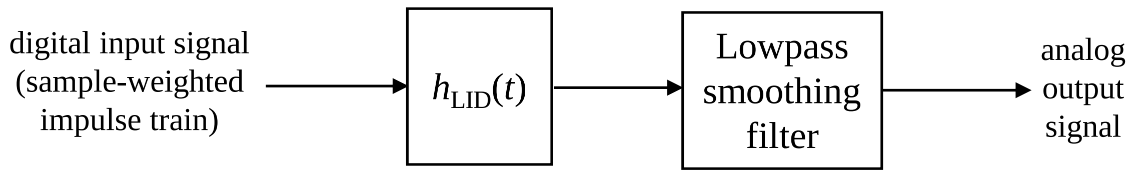
$$h_{FOH}(t) = \begin{cases} 1 + \dfrac{t}{T}, & 0 \le t \le T \\ 1 - \dfrac{t}{T}, & T \le t \le 2T \\ 0, & \text{otherwise} \end{cases}$$

or

$$\hat{x}(t) = x(nT) + \frac{x(nT) - x(nT-T)}{T}(t - nT) \qquad nT \le t < (n+1)T$$

Linear Interpolation with Delay

This technique linearly extrapolates the next sample based on the samples $x(nT)$ and $x(nT-T)$. A one sample delay is required to avoid jumps at the sample points.



where

$$h_{LID}(t) = \begin{cases} \dfrac{t}{T}, & 0 \le t \le T \\ 2 - \dfrac{t}{T}, & T \le t \le 2T \\ 0, & \text{otherwise} \end{cases}$$

or

$$\hat{x}(t) = x(nT-T) + \frac{x(nT) - x(nT-T)}{T}(t - nT) \qquad nT \le t < (n+1)T$$

Note that all non-ideal reconstruction filters must be followed by LPF to remove the distortions they induce.