Chapter 4

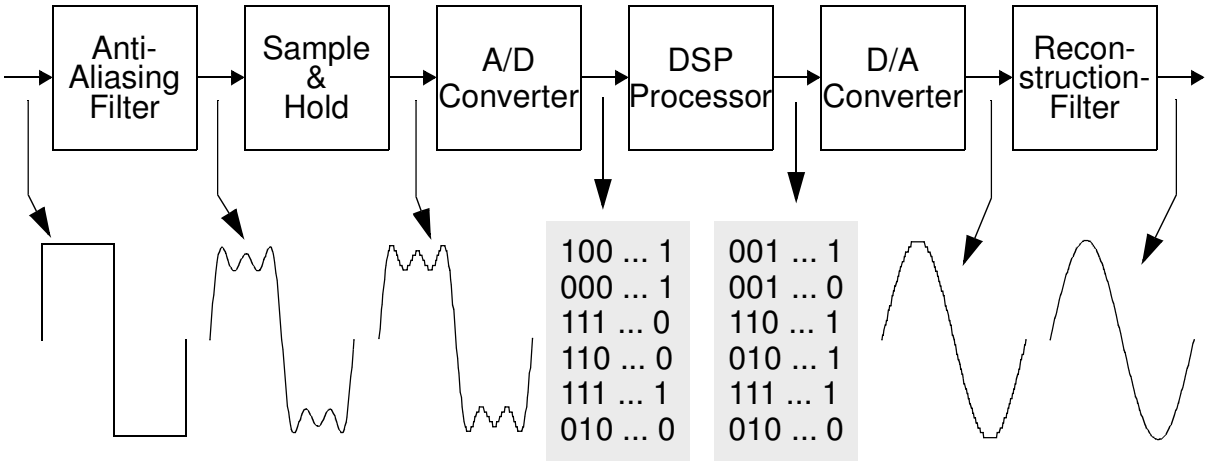
Sampling of Continuous-Time Signals

Contents

4.1 Introduction

Discrete-time signals very often occur as a result of periodic (uniform) sampling continuous-time signals. As we will see later it is not too difficult to design a sampling system that allows reasonably accurate reconstruction of a continuous-time signal from discrete-time samples.

An important result is that continuous-time signal processing can be implemented by first sampling, discrete-time processing, and then reconstruction of a continuous-time signal. The following block diagram illustrates such a system.



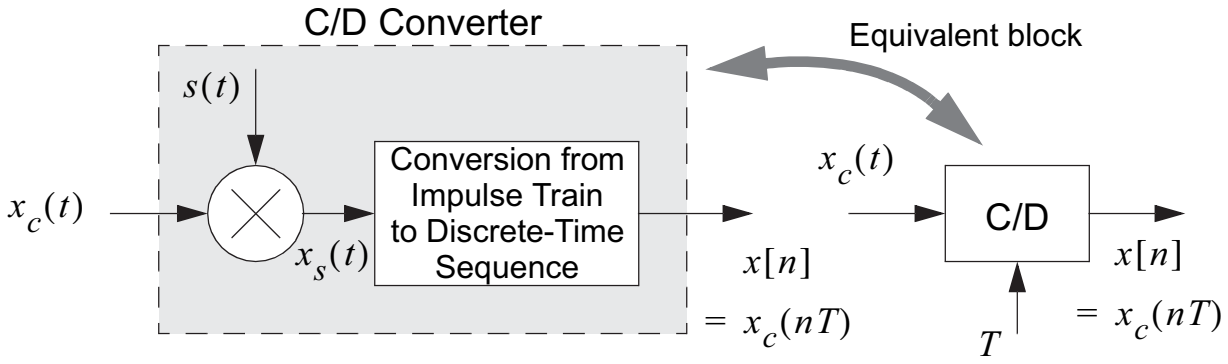
Continuous-time signal processing using discrete-time signal processing techniques

4.2 Periodic Sampling

The most common method of obtaining a discrete-time representation of a continuous-time signal is via periodic sampling. We obtain $x[n]$ from the continuous-time signal $x_c(t)$ by sampling every T seconds

$$x[n] = x_c(nT) \quad -\infty < n < \infty.$$

- The sample spacing T is the *sampling period*
- The reciprocal of T, $f_s = 1/T$ is the *sampling frequency (rate)* in samples per second
- A mathematically convenient way of representing ideal sampling is shown below



Continuous to discrete (C/D) converter

- Here $s(t)$ is an impulse train

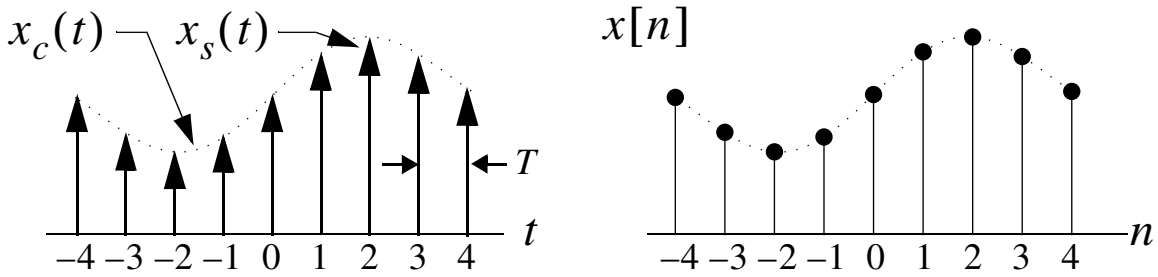
$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

and

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

is a weighted impulse train, related to $x[n]$

- Note that $x_s(t)$ and $x[n]$ differ in that $x[n]$ is simply the sample values of $x_c(t)$ while $x_s(t)$ is composed of impulses with area equal to the sample values of $x_c(t)$



Comparisons between $x_s(t)$ and $x[n]$

4.3 Frequency Domain Analysis

- The frequency domain relationship between the input and output of an ideal C/D converter can be obtained as follows:

$$S(j\Omega) = \mathcal{F}\{s(t)\} = \frac{2\pi}{T}\sum_{k=-\infty}^{\infty}\delta(\Omega - k\Omega_s),\ \Omega_s = 2\pi f_s$$

and since

$$x_s(t) = x_c(t)s(t)$$

$$\begin{aligned} X_s(j\Omega) = \mathcal{F}\{x_s(t)\} &= \frac{1}{2\pi}X_c(j\Omega) * S(j\Omega) \\ &= \frac{1}{2\pi}X_c(j\Omega) * \frac{2\pi}{T}\sum_{k=-\infty}^{\infty}\delta(\Omega - k\Omega_s) \\ &= \frac{1}{T}\sum_{k=-\infty}^{\infty}X_c(j\Omega - kj\Omega_s) \end{aligned}$$

4.3.1 Continuous-Time Fourier Transform–Review

- The continuous-time Fourier transform pair that we are using is

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

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

- For future reference define the continuous-time rectangular window function as

$$\Pi\left(\frac{t}{\tau}\right) \equiv \begin{cases} 1, & |t| < \tau/2 \\ 0, & \text{otherwise} \end{cases}$$

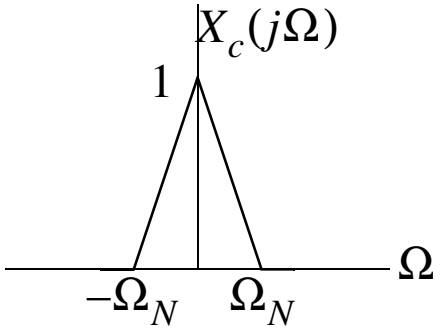
- Also define the sinc() function as

$$\text{sinc}(x) \equiv \frac{\sin \pi x}{\pi x}$$

Continuous-time Fourier transform pairs

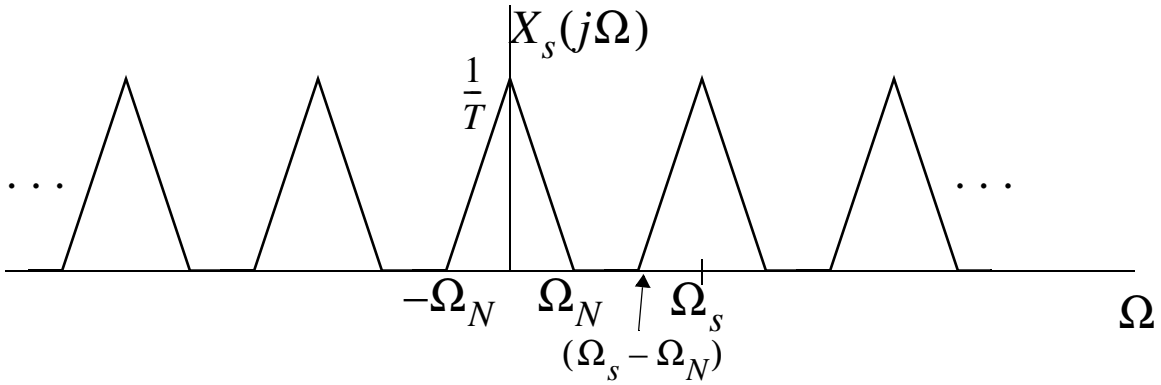
$x(t)$	$X(j\Omega)$
$x(t-t_o)$	$e^{-j\Omega t_o} X(j\Omega)$
$e^{j\Omega_o t} x(t)$	$X(j\Omega - j\Omega_o)$
$x(t)\cos(\Omega_o t + \varphi)$	$\frac{1}{2}[X(j\Omega - j\Omega_o)e^{j\varphi} + X(j\Omega + j\Omega_o)e^{-j\varphi}]$
$x_1(t) * x_2(t)$	$X_1(j\Omega)X_2(j\Omega)$
$x_1(t)x_2(t)$	$\frac{1}{2\pi}X_1(j\Omega) * X_2(j\Omega)$
$A\delta(t)$	A
A	$2\pi A\delta(\Omega)$
$e^{-at}\,u(t), \quad (\lvert a\rvert > 0)$	$1/(a + j\Omega)$
$x(t) = \Pi\left(\frac{t}{\tau}\right)$	$\tau\text{sinc}\left(\frac{\Omega\tau}{2\pi}\right)$
$2B\text{sinc}(2Bt)$	$\Pi\left(\frac{\Omega}{4\pi B}\right), \quad (B \text{ in Hz})$
$\sum_{k=-\infty}^{\infty}\delta(t-kT)$	$\Omega_o \sum_{n=-\infty}^{\infty}\delta(\Omega - n\Omega_o), \quad \Omega_o = \frac{2\pi}{T}$

- Returning to the frequency domain analysis of sampling, we see that the F.T. of $x_s(t)$ consists of replicas of $X_c(j\,\Omega)$ placed at all multiples of the sampling frequency Ω_s
- Suppose that $x_c(t)$ has bandlimited Fourier transform as shown below



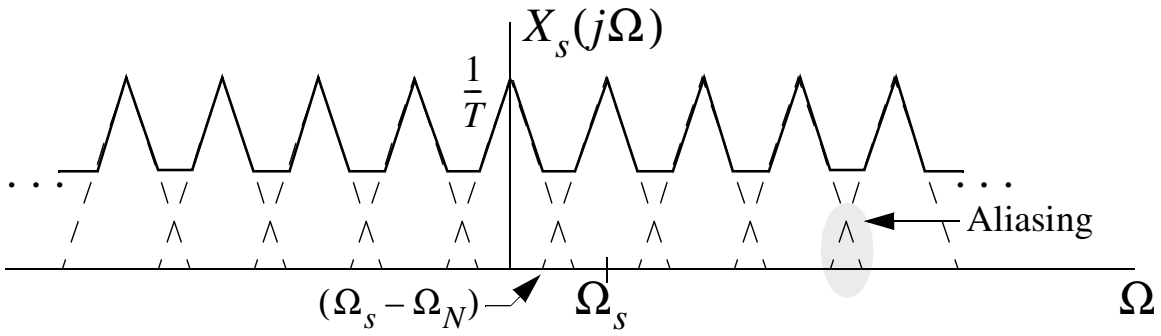
Fourier transform of $x_c(t)$

- If $x_c(t)$ is bandlimited to Ω_N we see from the following figure that no spectral overlap occurs in $X_s(j\,\Omega)$ provided $\Omega_s - \Omega_N > \Omega_N$ or $\Omega_s > 2\Omega_N$



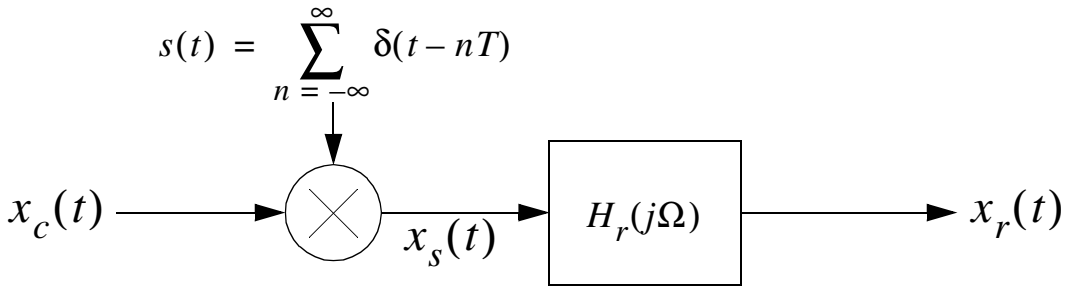
Spectrum of $x_s(t)$ for $\Omega_s > 2\Omega_N$

- Note that if $\Omega_s < 2\Omega_N$ then frequency domain overlap known as *aliasing* occurs

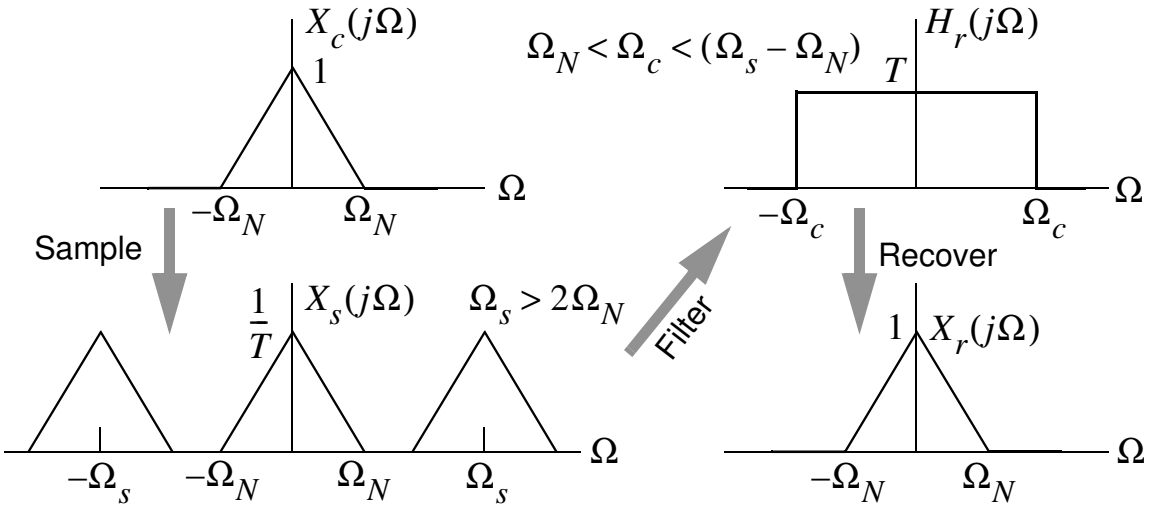


Spectrum of $x_s(t)$ for $\Omega_s < 2\Omega_N$

- Exact recovery of $x_c(t)$ from $x_s(t)$ is possible if the frequency translated replicas of $X_c(j\Omega)$ do not overlap when added together. Using an ideal lowpass filter we can recover $x_c(t)$ from $x_s(t)$ to within a $1/T$ scale factor.



Ideal lowpass filter recovery system



Exact recovery using an ideal lowpass filter

- Hence we conclude that if $H_r(j\Omega)$ is an ideal lowpass filter with gain T and cutoff frequency Ω_c chosen such that

$$\Omega_N < \Omega_c < \Omega_s - \Omega_N,$$

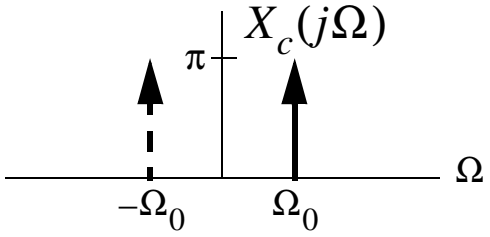
  we can write

$$X_r(j\Omega) = X_c(j\Omega) \text{ or } x_r(t) = x_c(t)$$

- Note that if $\Omega_s < 2\Omega_N$ then $x_c(t)$ **is not** recoverable from $x_s(t)$ due to the presence of aliasing

Example 4.1:

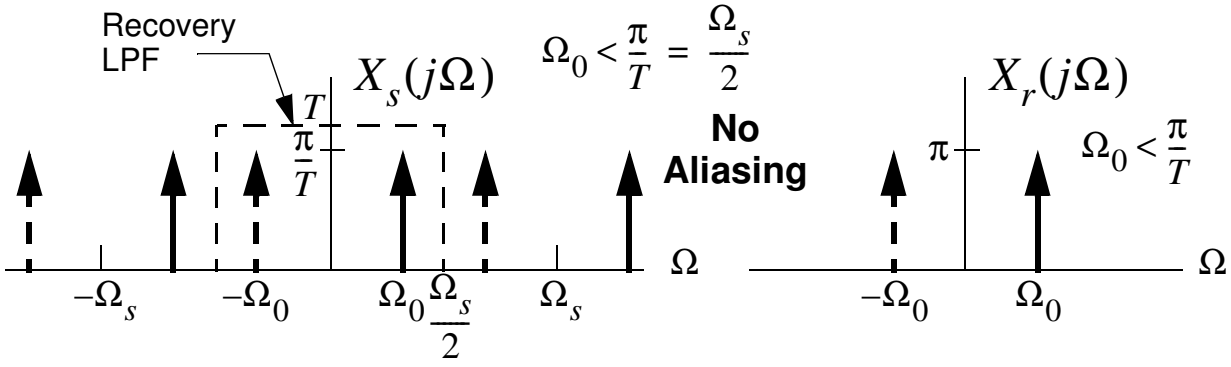
Suppose that $x_c(t) = \cos \Omega_o t$



The spectrum of $x_c(t)$

- With $\Omega_o < \Omega_s/2$ we see from the following figure that using an ideal lowpass filter the reconstructed output is

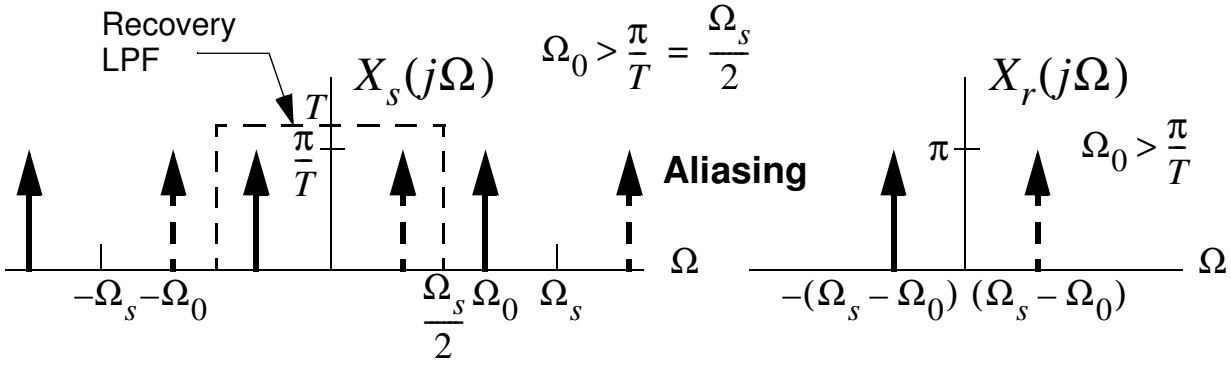
$$x_r(t) = \cos \Omega_o t$$



Pre and Post reconstruction spectra

- With $\Omega_s > \Omega_o > \Omega_s/2$ we see from the following figure that using an ideal lowpass filter we can not recover the true $x_c(t)$. The reconstructed output with aliasing is

$$x_r(t) = \cos(\Omega_s - \Omega_o)t$$



Pre and Post reconstruction spectra, with aliasing

Nyquist Sampling Theorem

Let $x_c(t)$ be a bandlimited signal such that

$$X_c(j\Omega) = 0 \quad \text{for } |\Omega| > \Omega_N$$

Then $x_c(t)$ can be recovered from its samples $x_c(nT)$
$\overset{\text{also}}{=} x[n]$, provided

$$\Omega_s = \frac{2\pi}{T} > 2\Omega_N$$

- Ω_N is referred to as the *Nyquist frequency*
- $2\Omega_N$ is referred to as the *Nyquist rate*

4.4 Relating $X(e^{j\omega})$ to $X_c(j\Omega)$ and $X_s(j\Omega)$

- To begin with consider the Fourier transform of $x_s(t)$ in terms of $x_c(nT)$ using the transform pair $A\delta(t - t_o) \overset{\mathcal{F}}{\longleftrightarrow} Ae^{-j\Omega t_o}$

$$X_s(j\Omega) = \sum_{n=-\infty}^{\infty} x_c(nT)e^{-j\Omega Tn}$$

- Note that $x[n] = x_c(nT)$ and

$$X(e^{j\Omega T}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\Omega Tn}$$

- It now follows by variable substitution that

$$X_s(j\Omega) = X(e^{j\omega})|_{w=\Omega T} = X(e^{j\Omega T})$$

  and hence that

$$X(e^{j\Omega T}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} X_c(j\Omega - jk\Omega_s)$$

  or equivalently that

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

- The transformation between $X(e^{j\omega})$ and $X_s(j\Omega)$ simply requires the frequency scaling $\omega = \Omega T$

- With the above frequency scaling we see that while $X_s(j\Omega)$ has period Ω_s, $X(e^{j\omega})$ always has period 2π. This is consistent with the fact that $x_s(t)$ has sample spacing $T = 1/f_s$ and $x[n]$ always has unit sample spacing.

4.5 Reconstruction of a Bandlimited Signal from Its Samples

At this point we know from the sampling theorem that it is possible to reconstruct a bandlimited signal from its samples. We are now interested in obtaining a time domain formula for reconstructing $x_c(t)$ directly from its samples $x[n]$.

- As the first step we can write

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

- The reconstruction filter with frequency response $H_r(j\,\Omega)$ has impulse response $h_r(t)$ so the filter output is

$$x_r(t) = x_s(t) * h_r(t) = \sum_{n=-\infty}^{\infty} x[n]h_r(t - nT)$$

- Recall that the ideal reconstruction filter is an ideal lowpass filter with gain T and cutoff frequency $\Omega_N < \Omega_c < \Omega_s/2 = \pi/T$, thus

$$\begin{aligned} h_r(t) &= \mathcal{F}^{-1}\left\{T \prod\left(\frac{\Omega}{2\Omega_c}\right)\right\} \\ &= \frac{T}{\pi}\Omega_c \,\mathrm{sinc}\left(\frac{\Omega_c t}{\pi}\right) \end{aligned}$$

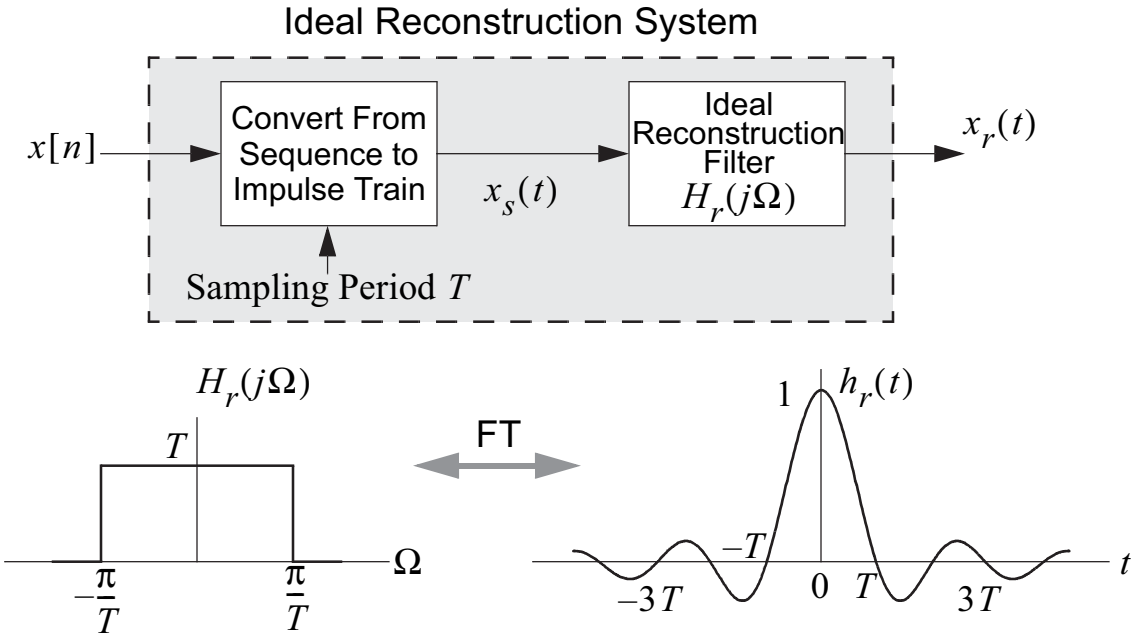
- The time domain formula for obtaining $x_r(t)$ from $x[n]$ then is

$$x_r(t) = \sum_{n=-\infty}^{\infty} x[n]\mathrm{sinc}[\Omega_c(t - nT)/\pi]$$

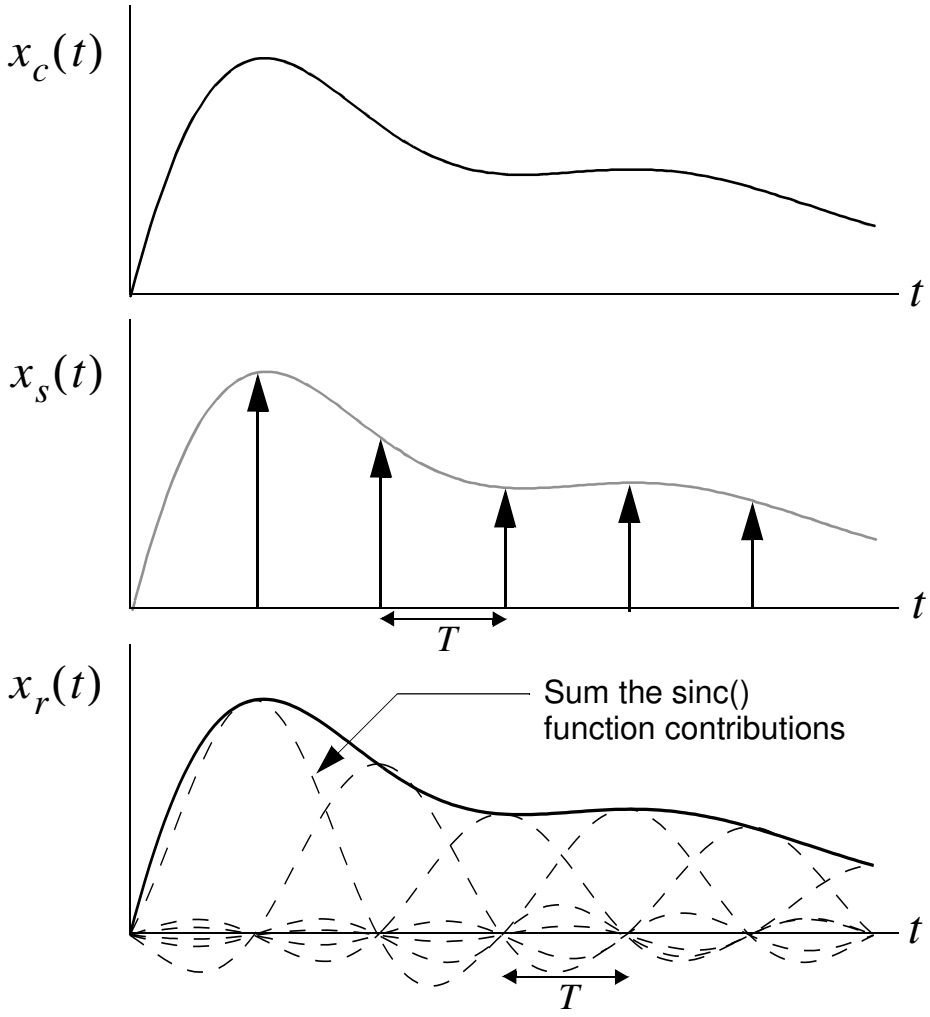
- As a special case we may choose as the cutoff frequency $\Omega_c = \Omega_s/2 = \pi/T$, then we can write

$$\begin{aligned} x_r(t) &= \sum_{n=-\infty}^{\infty} x[n]\text{sinc}[(t-nT)/T] \\ &= \sum_{n=-\infty}^{\infty} x[n]\frac{\sin[\pi(t-nT)/T]}{\pi(t-nT)/T} \end{aligned}$$

- Note that the reconstruction filter interpolates signal values between the impulses in $x_s(t)$ to form $x_r(t)$



Block diagram and waveforms associated with ideal reconstruction

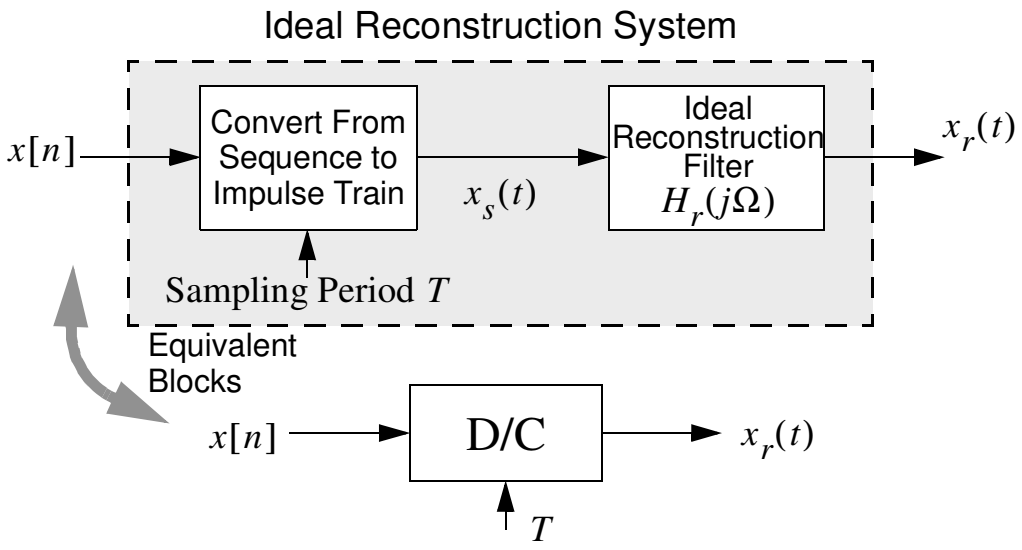


Ideal bandlimited interpolation with the lowpass cutoff frequency set to the Nyquist frequency

- In future developments we will refer to the ideal discrete-to-continuous-time (D/C) converter as a system which has frequency domain relationship

$$X_r(j\Omega) = H_r(j\Omega)X(e^{j\Omega T})$$

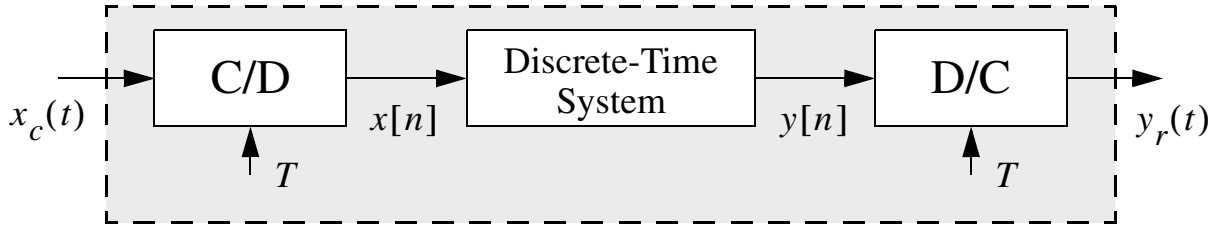
with the cutoff frequency of the lowpass reconstruction filter assumed to be π/T



Ideal D/C reconstruction system

4.6 Discrete-Time Processing of Continuous-Time Signals

A major application of discrete-time signal processing is in the processing of continuous-time signals. The processor blocks needed to accomplish this are shown in the following figure



Discrete-time processing of continuous-time signals using a C/D-$H(e^{j\omega})$-D/C system

- Without specifying anything about the discrete-time system we can at the very least say that

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

or in the frequency domain

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

and

$$y_r(t) = \sum_{n=-\infty}^{\infty} y[n]\text{sinc}[(t-nT)/T]$$

or in the frequency domain

$$\begin{aligned} Y_r(j\Omega) &= H_r(j\Omega)Y(e^{j\Omega T}) \\ &= \begin{cases} TY(e^{j\Omega T}), & |\Omega| < \pi/T \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

- To relate $x_c(t)$ to $y_r(t)$ we must know the relationship between $x[n]$ and $y[n]$

4.6.1 Linear Time-Invariant Discrete-Time Systems

The process of relating $x_c(t)$ to $y_r(t)$ is now straightforward since LTI systems have the property that

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

- Direct substitution of the above result into the previous expression for $Y_r(j\Omega)$ yields

$$\begin{aligned} Y_r(j\Omega) &= H_r(j\Omega)H(e^{j\Omega T})X(e^{j\Omega T}) \\ Y_r(j\Omega) &= H_r(j\Omega)H(e^{j\Omega T})\frac{1}{T}\sum_{k=-\infty}^{\infty} X_c\left(j\Omega - j\frac{2\pi k}{T}\right) \end{aligned}$$

- If we further assume that the input is bandlimited, that is $X_c(j\Omega) = 0$ for $|\Omega| \geq \pi/T$, then the ideal reconstruction filter will remove all but the fundamental term ($k = 0$) in $X(e^{j\omega})$, i.e.

$$Y_r(j\Omega) = \begin{cases} H(e^{j\Omega T})X_c(j\Omega), & |\Omega| < \pi/T \\ 0, & \text{otherwise} \end{cases}$$

- An alternate representation for $Y_r(j\Omega)$ is

$$Y_r(j\Omega) = H_{\text{eff}}(j\Omega)X_c(j\Omega)$$

  where the *effective* continuous-time frequency response is defined as

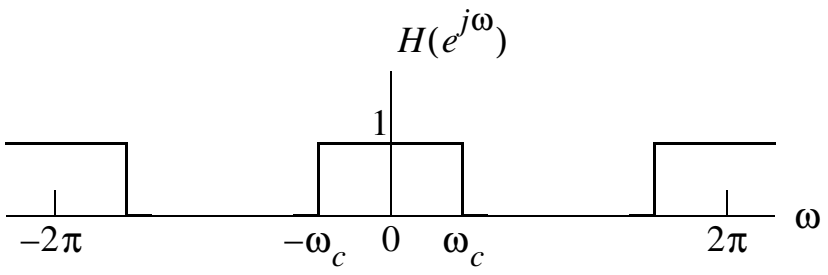
$$H_{\text{eff}}(j\Omega) = \begin{cases} H(e^{j\Omega T}), & |\Omega| < \pi/T \\ 0, & \text{otherwise} \end{cases}$$

- The above characteristic is valid only if (1) the discrete-time system is LTI, and (2) if $x_c(t)$ is bandlimited with Nyquist rate less than Ω_s

Example 4.2:

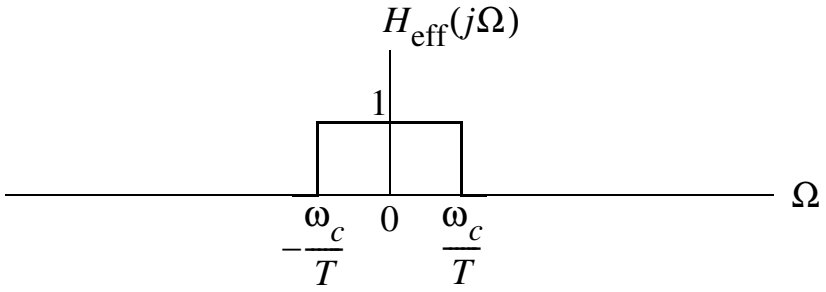
Consider an ideal lowpass filter with cutoff frequency ω_c. The discrete-time system response over the fundamental interval $[-\pi, \pi)$ is

$$H(e^{j\omega}) = \begin{cases} 1, & |\omega| < \omega_c \\ 0, & \text{otherwise} \end{cases}$$



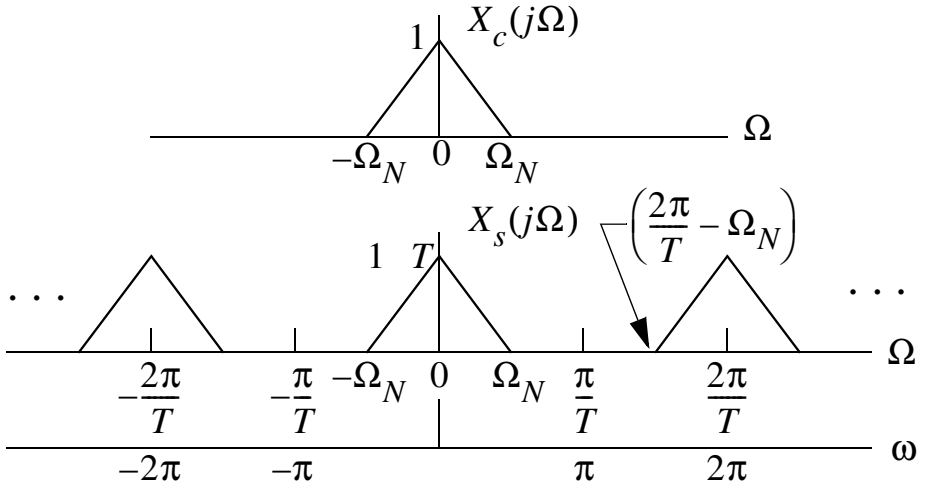
- Assuming the input is bandlimited and sampled above the Nyquist rate, the effective continuous-time frequency response is

$$H_{\text{eff}}(j\Omega) = \begin{cases} 1, & |\Omega| < \omega_c / T \\ 0, & \text{otherwise} \end{cases}$$

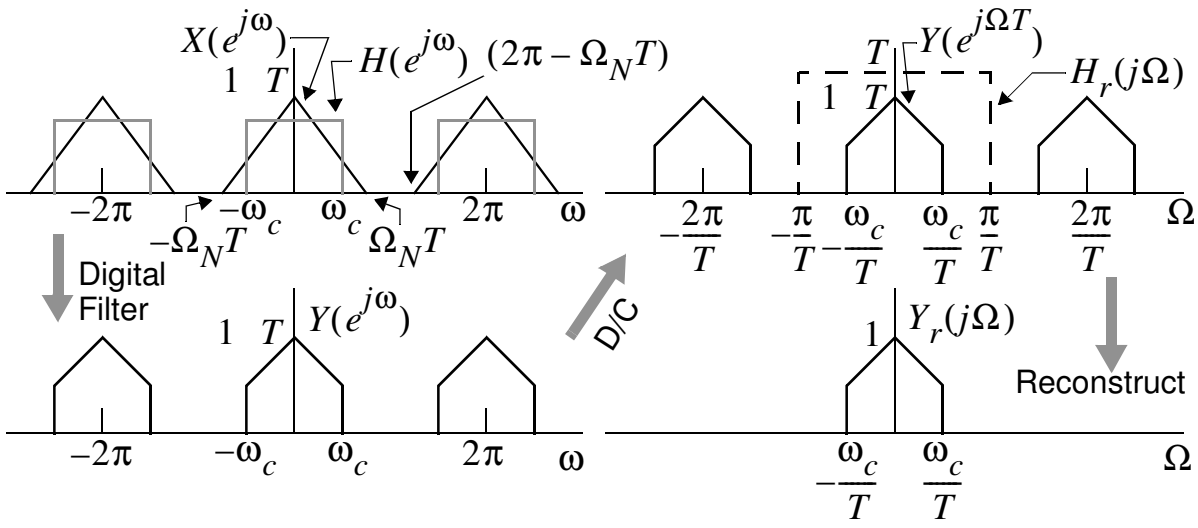


- Note that in the continuous-time domain the lowpass cutoff frequency is $\Omega_c = \omega_c / T$

- To be more specific suppose $x_c(t)$ has the bandlimited spectrum shown below



Plots of $X_c(j\Omega)$ and $X_s(j\Omega)$



Plots of $X(e^{j\omega})$, $Y(e^{j\omega})$, $Y(e^{j\Omega T})$, and $Y_r(j\Omega)$

Comments:

- The system is effectively an ideal lowpass filter with cutoff frequency $\Omega_c = \omega_c / T$
- A variable cutoff frequency lowpass filter can be implemented by simply varying the sample period T, in particular setting $\Omega_N T < \omega_c$ results in no filtering, i.e. $y_r(t) = x_c(t)$
- Since the discrete-time system provides additional bandlimiting, the condition that $\Omega_N < \Omega_s/2$ to avoid aliasing may be relaxed. Specifically the condition for no aliasing is

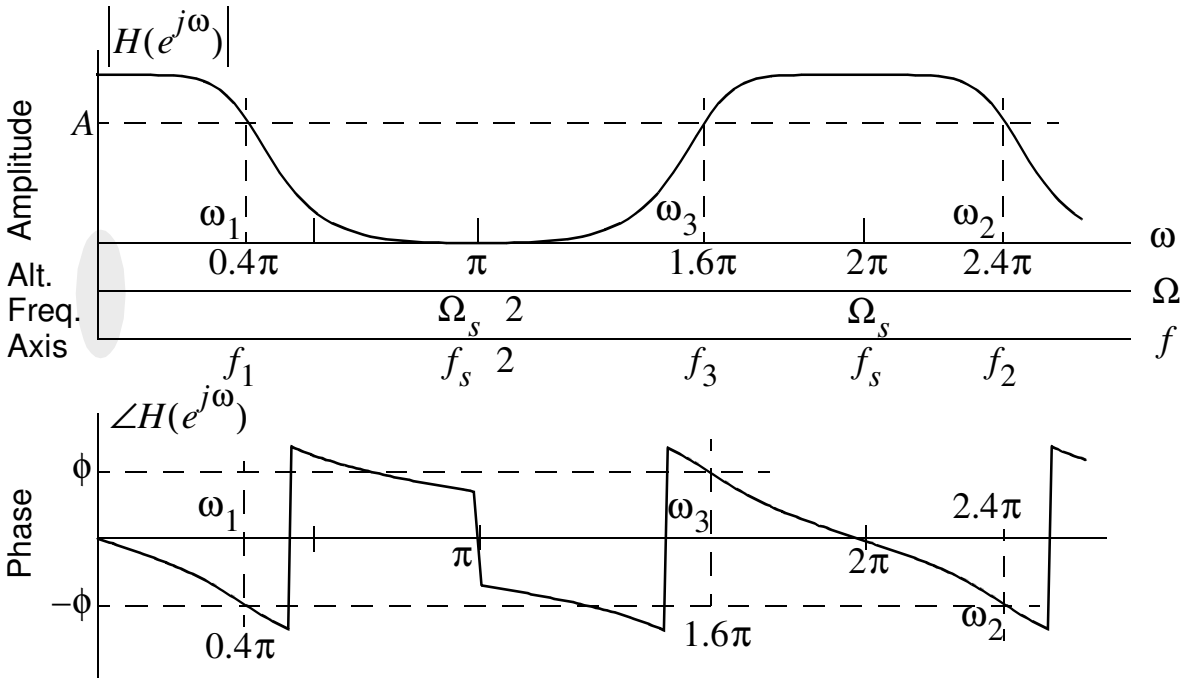
$$(2\pi - \Omega_N T) > \omega_c,$$

compared with the Nyquist theorem which requires

$$(2\pi - \Omega_N T) > \Omega_N T$$

Example 4.3:

Consider an LTI system with frequency response as shown below:



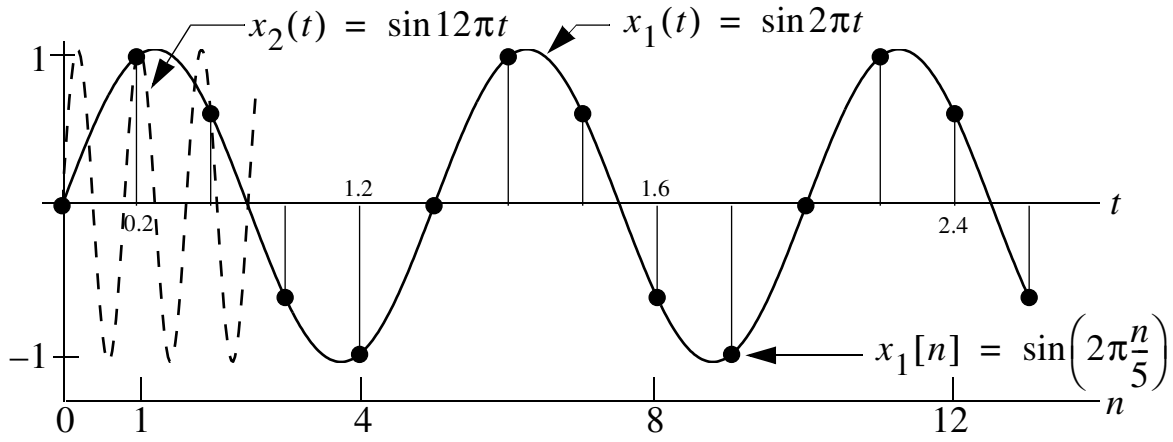
Magnitude and phase response of $H(e^{j\omega})$

Suppose this system is placed between ideal C/D and D/C converters to form a continuous-time processor with $T = 0.2$ seconds ($f_s = 5$ samples/sec.). To consider the effect of not sampling above the Nyquist rate consider the system response to a sinusoidal input of the form

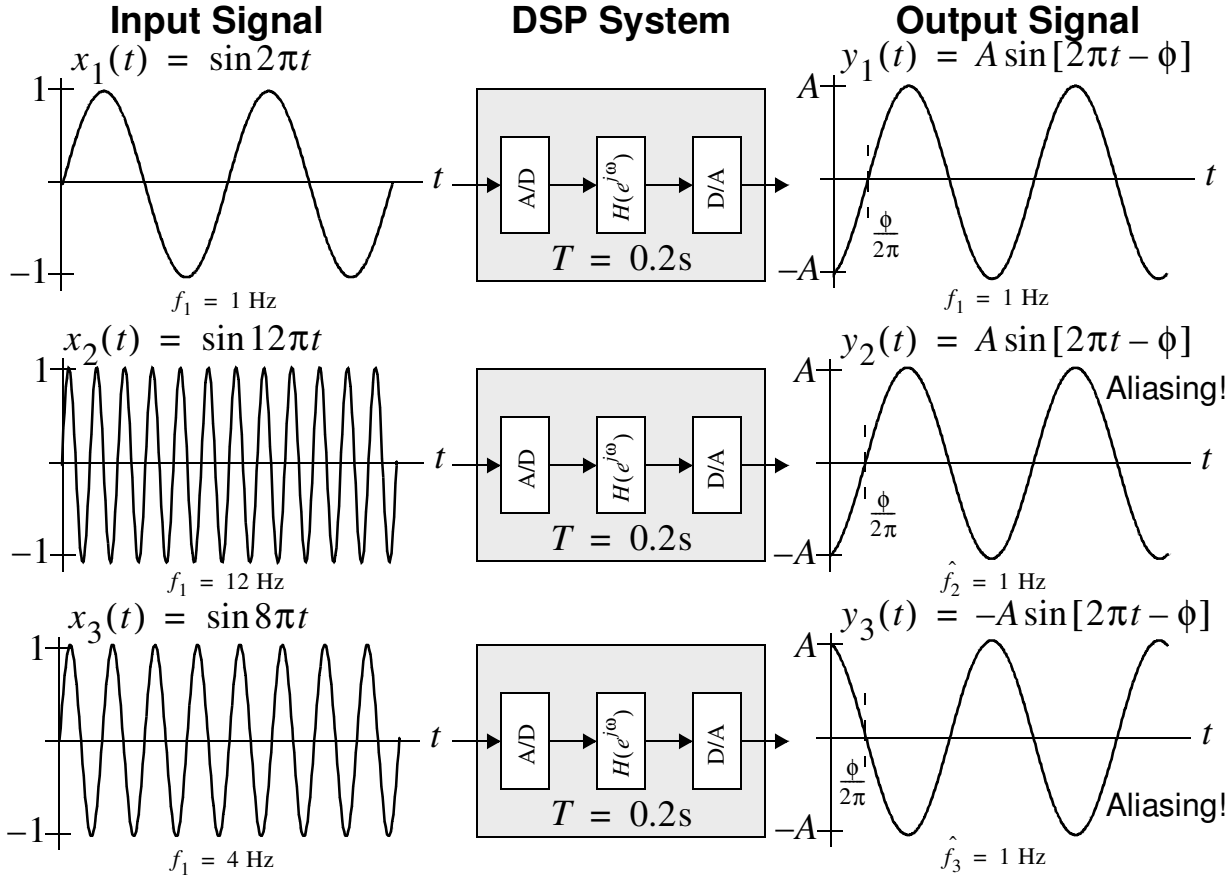
$$x_i(t) = \sin 2\pi f_i t, \quad i = 1, 2, 3$$

where $f_1 = 1$, $f_2 = 6$, and $f_3 = 4$ Hz.

- Note that after sampling with $T = 0.2$ we have $x_1[n] = x_2[n] = -x_3[n]$



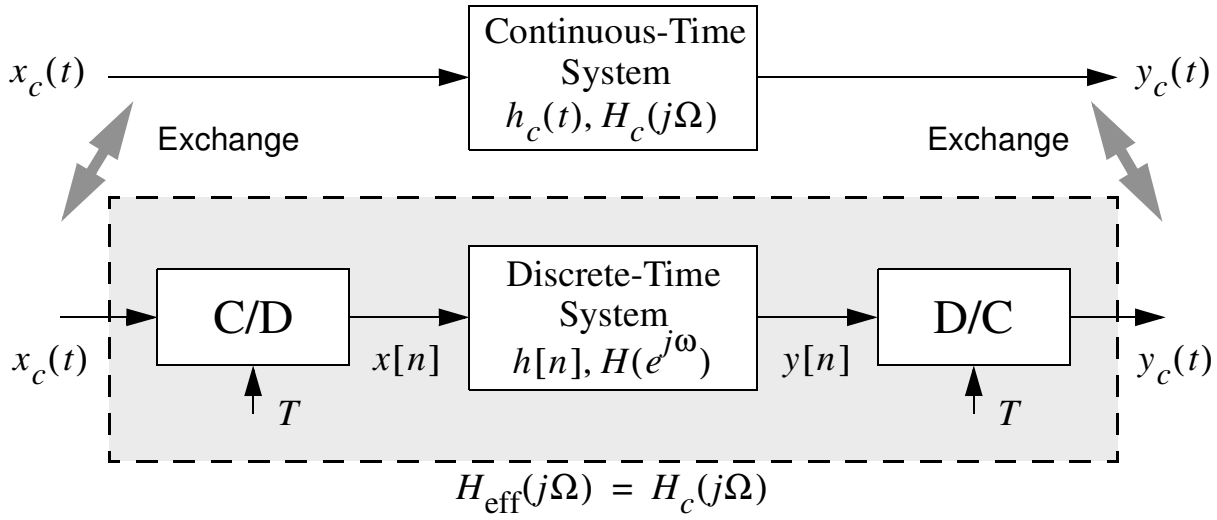
Two continuous-time signals giving the same discrete-time sequence after sampling every 0.2 sec.



System input/output response for all three inputs

4.6.2 Impulse-Invariant Design

We wish to implement a bandlimited continuous-time system, $H_c(j\Omega)$, using a C/D-$H(e^{j\omega})$-D/C system. We must choose $H(e^{j\omega})$ and T such that $H_{eff}(j\Omega) = H_c(j\Omega)$.



Desired LTI system; (top) continuous-time system (bottom) equivalent system for band limited inputs

- To obtain the desired response we must have

$$H(e^{j\omega}) = H_c(j\omega/T), \quad |\omega| < \pi$$

with T chosen such that

$$H_c(j\Omega) = 0 \quad \text{for } |\Omega| \geq \pi/T$$

- Note that if we let

$$h[n] = \mathcal{F}^{-1}\{H_c(j\Omega)\}|_{t \to nT} = h_c(nT)$$

then in the frequency domain we have

$$H(e^{j\omega}) = \frac{1}{T} \sum_{k=-\infty}^{\infty} H_c\left(j\frac{\omega}{T} - j\frac{2\pi k}{T}\right)$$

- On the interval $[-\pi, \pi)$ the frequency response is just

$$H(e^{j\omega}) = \frac{1}{T} H_c(j\omega/T),$$

  which is the desired result to within a constant

- To obtain an *impulse-invariant* version of the continuous-time system we thus specify that

$$h[n] = T\, h_c(nT)$$

Example 4.4: **Impulse Invariant Design**

- Both Python and MATLAB can be used to explore analog filters and impulse invariance based digital filter design

- For this introduction to impulse invariant design we will perform the transformation by hand since we only consider a simple first-order system

$$h(t) = Ae^{s_0 t}\, u(t)$$

- The corresponding Laplace transform is

$$H_c(s) = \frac{A}{s - s_0}$$

- In elementary circuit theory $H_c(s)$ would come from a simple RC lowpass filter with $A = 1/RC$ and $s_0 = -1/RC$, thus

$$h(t) = \frac{1}{RC} e^{-t/RC}\, u(t), \qquad H_c(s) = \frac{1}{1 + sRC}$$

  Note: The filter 3dB frequency is $f_3 = 1/(2\pi RC)$

- Applying the definition of impulse invariant simply requires that we set

$$h[n] = T h_c(nT) = \frac{T}{RC} e^{-nT/RC}\, u[n] = 2\pi f_3 T e^{-2\pi f_3 T n}\, u[n]$$

- The z-transform of the impulse invariant digital filter is

$$H(z) = \frac{2\pi f_3 T}{1 - e^{-2\pi f_3 T} z^{-1}}$$

- The frequency response of the filter is

$$H(e^{j\omega}) = \frac{2\pi f_3 T}{1 - e^{-2\pi f_3 T} e^{-j\omega}}$$

- We now use MATLAB to compare the frequency response of the two filters for $f_s = 1/T = 10$ kHz, and $f_3 = .1,\ 1$, and 5 kHz

  - As a practical matter we will first match the gains of the two filters at dc

$$H_c(j\Omega)|_{\Omega=0} = \left. \frac{1}{1 + j\Omega/2\pi f_3} \right|_{\Omega=0} = 1$$

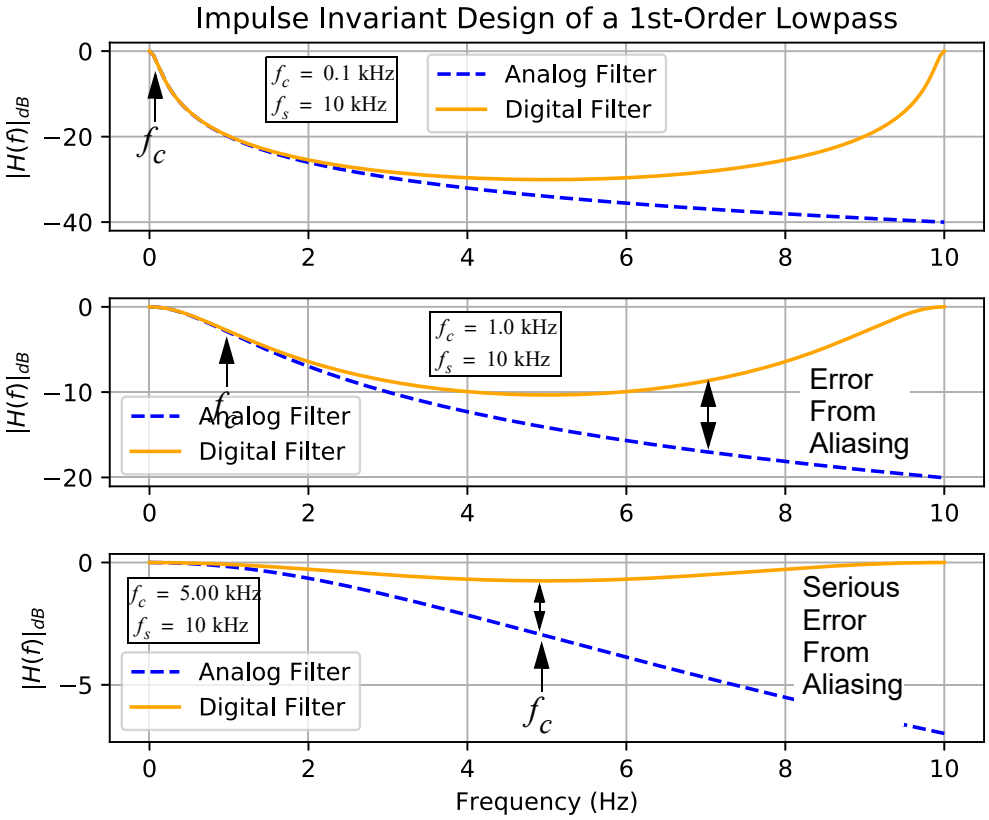
    Similarly with gain term G added to $H(e^{j\omega})$ we have,

$$H(e^{j\omega})\big|_{\omega=0} = G \frac{2\pi f_3 T}{1 - e^{-2\pi f_3 T} e^{-j0}}$$

    Thus to match the dc gain of the analog filter we must set

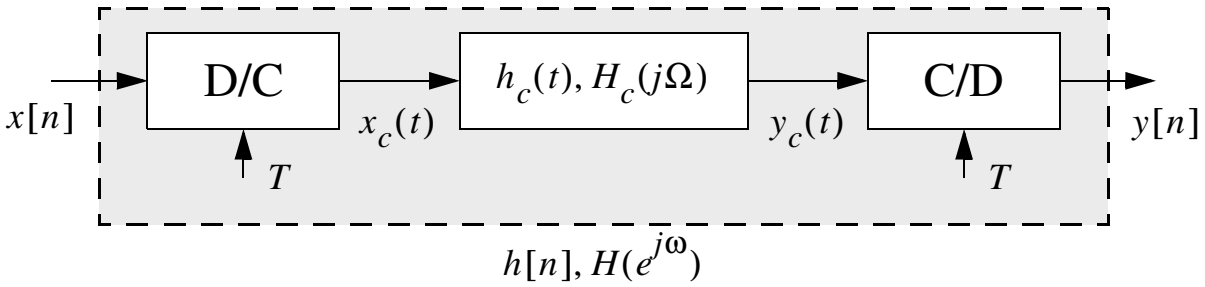
$$G = \frac{1 - e^{-2\pi f_3 T}}{2\pi f_3 T}$$

```
f = arange(0,10+0.05,0.05)
T = 1/10
def Hc_H(f,fc,T):
    Hc = 1/(1 + 1j*f/fc)
    H = (1 - exp(-2*pi*fc*T))/(1 - exp(-2*pi*fc*T)*exp(1j*2*pi*f*T))
    return Hc, H
subplot(311)
Hc, H = Hc_H(f,0.1,T)
plot(f, 20*log10(abs(Hc)),'b--')
plot(f, 20*log10(abs(H)),'b')
title(r'Impulse Invariant Design of a 1st-Order Lowpass')
ylabel(r'$|H(f)|_{dB}$')
grid()
subplot(312)
Hc, H = Hc_H(f,1.0,T)
plot(f, 20*log10(abs(Hc)),'b--')
plot(f, 20*log10(abs(H)),'b')
ylabel(r'$|H(f)|_{dB}$')
grid()
subplot(313)
Hc, H = Hc_H(f,5.0,T)
plot(f, 20*log10(abs(Hc)),'b--')
plot(f, 20*log10(abs(H)),'b')
ylabel(r'$|H(f)|_{dB}$')
xlabel(r'Frequency (Hz)')
grid()
tight_layout()
```



4.7 Continuous-Time Processing of Discrete-Time Signals

Using the definitions of the ideal D/C and C/D converters we can in theory construct a system for continuous-time processing of discrete-time signals.



Continuous-time processor for discrete-time signals using a D/C-$H_c(j\Omega)$-C/D system

- It is not difficult to show that the overall discrete-time system has frequency response

$$H(e^{j\omega}) = H_c(j\omega/T), \quad |\omega| < \pi$$

- Note that since $X_c(j\Omega) = 0$ for $|\Omega| \geq \pi/T$ (due to the design of the ideal D/C), the response of $H_c(j\Omega)$ above π/T is actually arbitrary

Example 4.5:

Use a D/C-$H_c(j\Omega)$-C/D system to implement a noninteger discrete-time delay.

- The discrete-time frequency response we desire is of the form

$$H(e^{j\omega}) = e^{-j\omega\Delta}, \quad |\omega| < \pi$$

  where Δ is not necessarily an integer

- Note that for the special case of Δ being an integer then $y[n] = x[n-\Delta]$

- A continuous-time system which gives arbitrary delay is

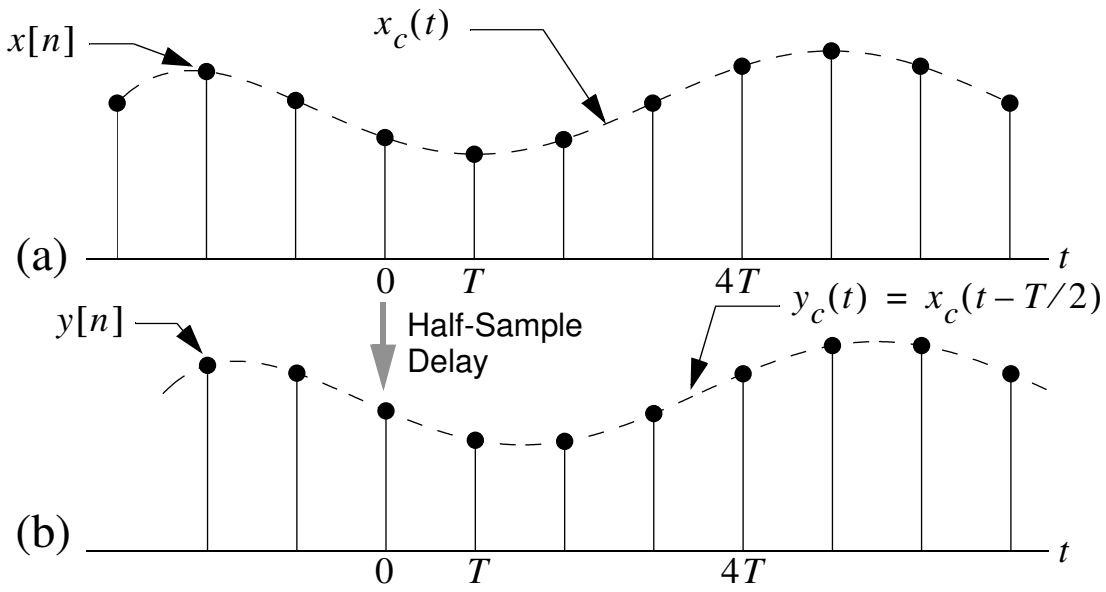
$$H_c(j\Omega) = H(e^{j\Omega T}) = e^{-j\Omega\Delta T}$$

  or in the time domain

$$y_c(t) = x_c(t - \Delta T)$$

- Since $x_c(t)$ is the bandlimited interpolation of $x[n]$, and $y[n] = y_c(t)|_{t=nT}$, we can write

$$\begin{aligned} y[n] &= y_c(nT) = x_c(nT - \Delta T) \\ &= \sum_{k=-\infty}^{\infty} x[k]\,\mathrm{sinc}[(nT - \Delta T - kT)/T] \end{aligned}$$



Plots of (a) $x[n]$ and (b) $y[n]$ for a half-sample delay system

Example 4.6: **Half-Sample Delay in Python**

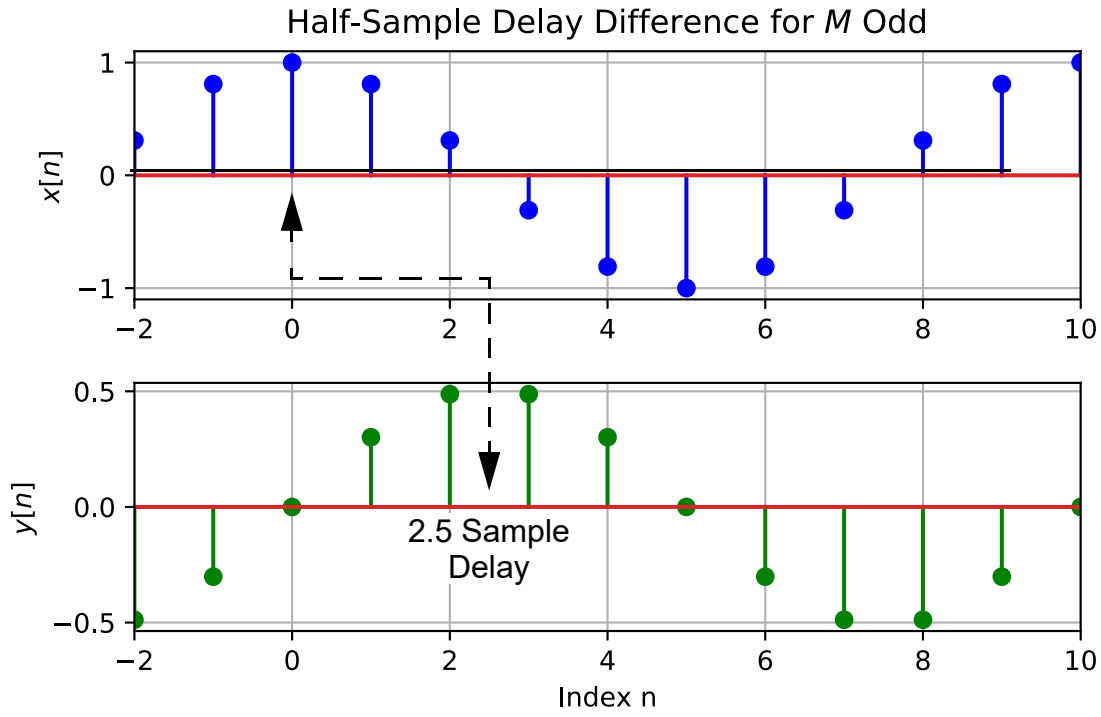
- A half-sample delay can be created entirely in the discrete-time domain as well
- One such system capable of doing this is the moving average system described in Chapter 2
- Recall that for a causal moving average system ($M_1 = 0$ and $M_2 = M$), the frequency response is

$$H(e^{j\omega}) = \frac{1}{M+1} \frac{\sin\left[\omega(M+1)/2\right]}{\sin(\omega/2)} e^{-j\omega M/2}$$

- The term $e^{-j\omega M/2}$ provides a pure delay in the time domain which we see is a half-sample if M is odd

- A simple Python simulation will demonstrate this

```
n = arange(-20,20)
x = cos(2*pi*0.1*n)
M = 5
y = signal.lfilter(ones(M+1)/(M+1),1,x)
subplot(211)
stem(n,x,linefmt='b',markerfmt='bo')
title(r'Half-Sample Delay Difference for $M$ Odd')
ylabel(r'$x[n]$')
xlim([-2,10])
grid();
subplot(212)
stem(n,y,linefmt='g',markerfmt='go')
xlabel('Index n')
ylabel(r'$y[n]$')
xlim([-2,10])
grid();
tight_layout()
```



$M/2$-Sample delay of a moving average filter; $M = 5$

4.8 Changing Sampling Rate using Discrete-Time Processing

In discrete-time signal processing systems we may wish to change the underlying sampling rate somewhere within the system. For example we may start with

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

and later find it more appropriate or efficient to work with

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

where $T \neq T'$ (typically T/T' is rational).

4.8.1 Analog Techniques:

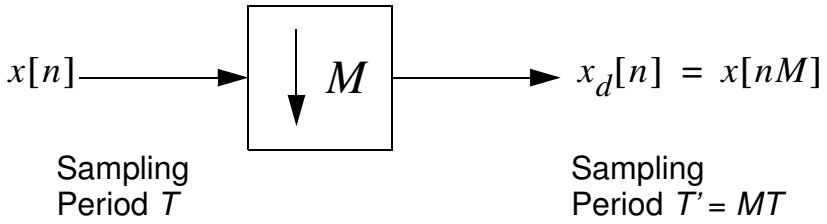
Conceptually the simplest way to change the sampling rate is to first reconstruct $x_c(t)$ from $x[n]$ using an appropriate analog interpolation filter, then resample $x_c(t)$ with period T' to obtain $x'[n]$.

4.8.2 Sampling Rate Reduction by an Integer Factor

Define a new sequence $x_d[n]$ with rate reduced by a factor M with respect to $x[n]$ as

$$x_d[n] = x[nM] = x_c(nMT)$$

- The system which performs the sampling rate reduction (prefiltering may also be included) is referred to as a *downsampler*, or a *decimator*



Downsampler or decimator (no prefilter shown)

- To study the operation of the downsampler we will consider $x_d[n]$ and $x[n]$ in the frequency domain
- Recall that since $x[n] = x_c(nT)$ we can write

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

- Similarly since $x_d[n] = x_c(nMT)$ we can write

$$\begin{aligned} X_d(e^{j\omega}) &= \frac{1}{T'}\sum_{r=-\infty}^{\infty} X_c\left(j\frac{\omega}{T'} - j\frac{2\pi r}{T'}\right) \\ &= \frac{1}{MT}\sum_{r=-\infty}^{\infty} X_c\left(j\frac{\omega}{MT} - j\frac{2\pi r}{MT}\right) \end{aligned}$$

- The above equation shows that $X_d(e^{j\omega})$ consists of replicas of $X_c(j\omega/MT)$ placed at all multiples of 2π, while in contrast $X(e^{j\omega})$ consists of replicas of $X_c(j\omega/T)$ at 2π spacings
- To relate $X(e^{j\omega})$ to $X_d(e^{j\omega})$ we can reindex the sum in $X_d(e^{j\omega})$ using $r = i + kM$ with $-\infty < k < \infty$ and $0 \le i \le M-1$ (note we still have $-\infty < r < \infty$)

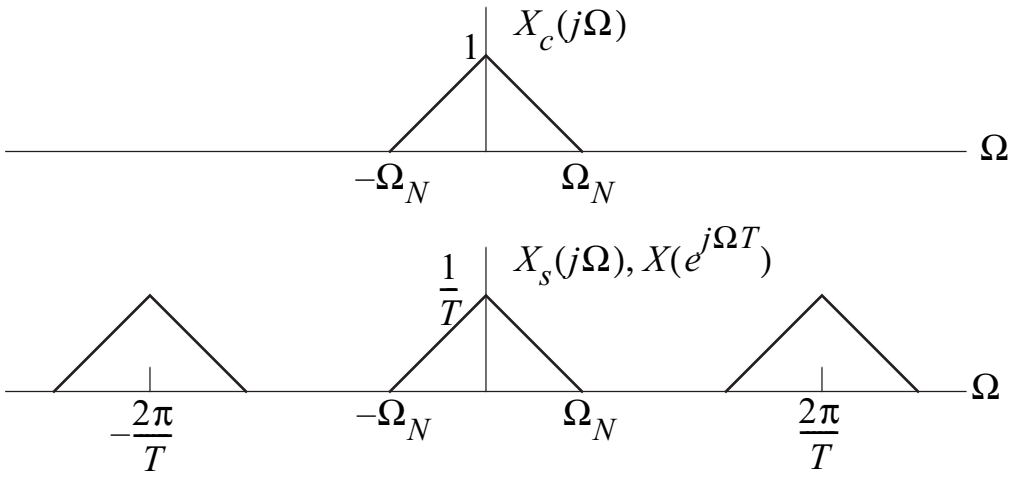
- It can then be shown that

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

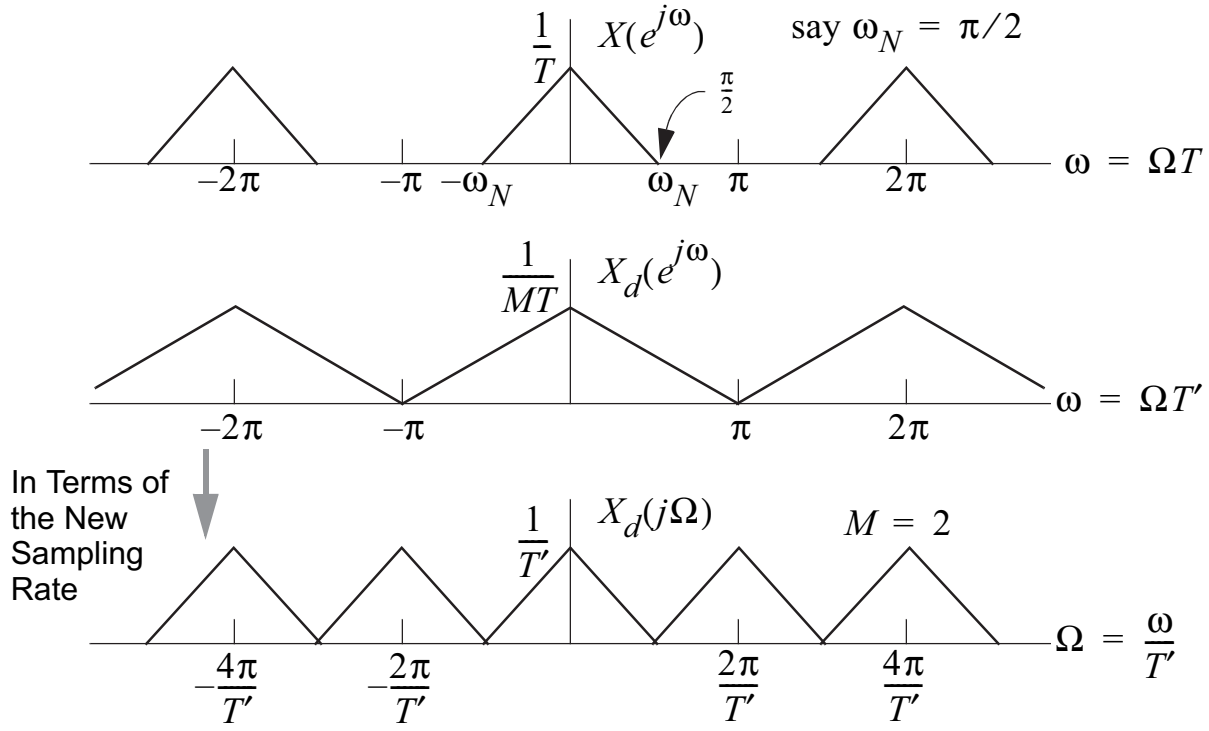
- The above equation shows that an alternative interpretation of $X_d(e^{j\omega})$ is that it consists of M copies of $X(e^{j\omega})$ frequency scaled by M (i.e. $\omega \rightarrow \omega/M$), each shifted by integer multiples of $2\pi/M$
- No matter how $X_d(e^{j\omega})$ is viewed its period is always 2π
- With downsampling the potential for aliasing exits since the new sampling rate may actually be below the Nyquist rate of $x_c(t)$.

Example 4.7:

Consider downsampling using $M = 2$ a signal with bandlimited spectrum as shown below



Spectra of $X_c(j\Omega)$ and $X(e^{j\Omega T})$



Spectra of $X(e^{j\omega})$, $X_d(e^{j\omega})$, and $X_d(j\Omega)$ for $M = 2$

- Note that in the above example Ω_N has been chosen so that after downsampling with $M = 2$, no aliasing results (i.e. $4\Omega_N = 2\pi/T$ and $\Omega_N \overset{\text{just}}{=} (\pi/T')$ or $\pi/(2T)$)

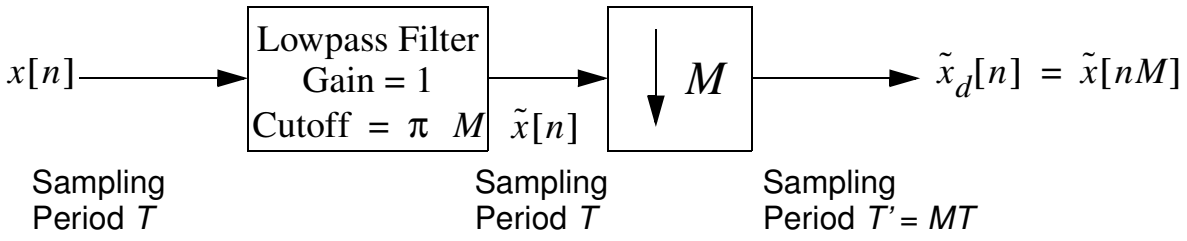
- In general to avoid aliasing when downsampling by M we must ensure that $x[n]$ is bandlimited such that

$$\omega_N M < \pi \quad \text{or} \quad \omega_N < \pi/M$$

  where ω_N is the lowpass bandwidth of $x[n]$ (i.e. $\omega_N = \Omega_N T$)

- In some cases we may need to filter $x[n]$ with an ideal lowpass filter with cutoff frequency $\omega_c = \pi/M$ before downsampling

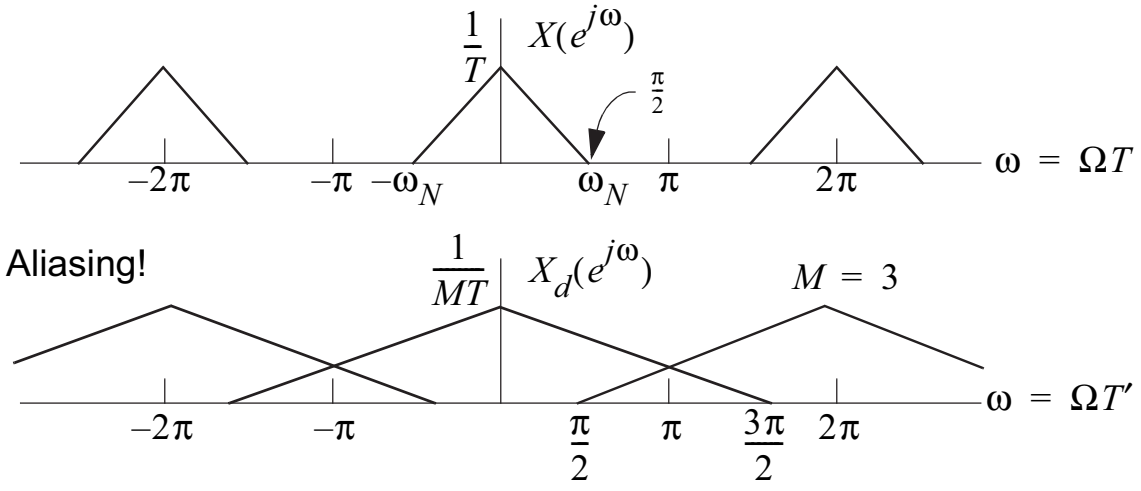
- The filter output, denoted $\tilde{x}[n]$, may then be downsampled without aliasing to obtain $\tilde{x}_d[n]$
- Note that with the introduction of the prefilter $\tilde{x}_d[n] \neq x_c(nMT)$, but is equivalent to sampling $x_c(t)$ after filtering with an ideal lowpass filter having cutoff frequency $\Omega_c = \pi/(MT)$



General system for decimation by M

Example 4.8:

Consider a second example where $4\Omega_N \overset{\text{still}}{=} 2\pi/T$, but now $M = 3$.

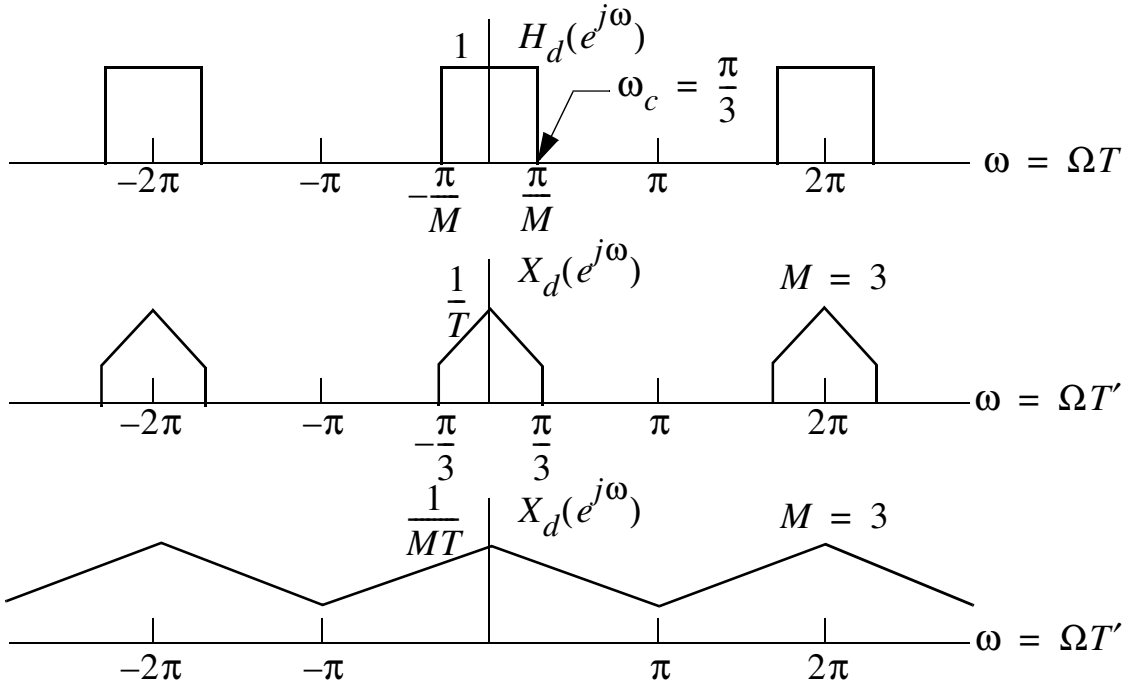


Downsampling with aliasing

- The downsampled sequence $x_d[n]$ will contain aliasing since the downsampled signal bandwidth in the ω domain is $\Omega_N MT = 3\pi/2 > \pi$

- Note that an ideal lowpass prefilter with cutoff frequency $\omega_c = \pi/3$ can be used to eliminate aliasing

- Spectral plots of the modified downsampling system are shown on the following page



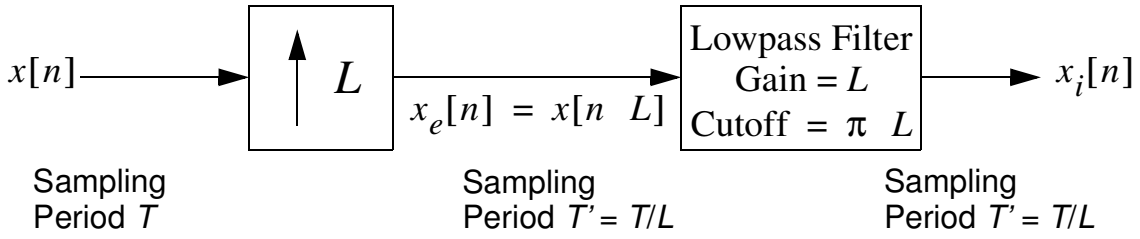
Downsampling with prefiltering to avoid aliasing

4.9 Increasing the Sampling Rate by an Integer Factor

Define a new sequence $x_i[n]$ with rate increased by a factor L as a lowpass filtered version of

$$x_e[n] = x[n/L] = x_c(nT/L), \quad n = 0, \pm L, \pm 2L, \ldots$$

- The system which performs the sampling rate increase (including postfiltering) is referred to as an *upsampler*, or an *interpolator*



- The first stage of the upsampler is also known as a *sampling rate expander* or just *expander*.

- We can write the expander output as

$$x_e[n] = \begin{cases} x[n/L], & n = 0, \pm L, \pm 2L, \ldots \\ 0, & \text{otherwise} \end{cases}$$
$$= \sum_{k=-\infty}^{\infty} x[k]\delta[n - kL]$$

- To obtain $x_i[n]$ from $x_e[n]$ requires discrete-time reconstruction (interpolation), since $x_e[n]$ is an impulse train consisting of nonzero samples every L samples

- The operation of the upsampler is best viewed in the frequency domain

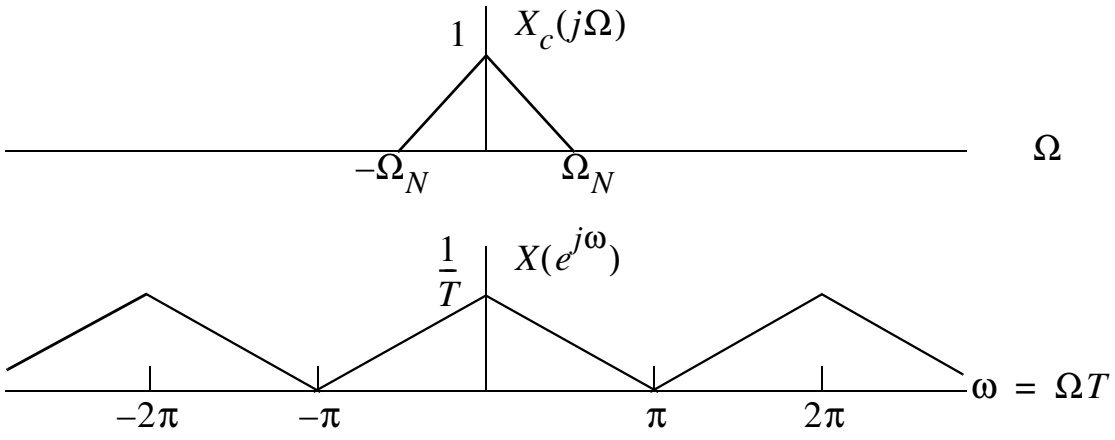
- To begin with we can write

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

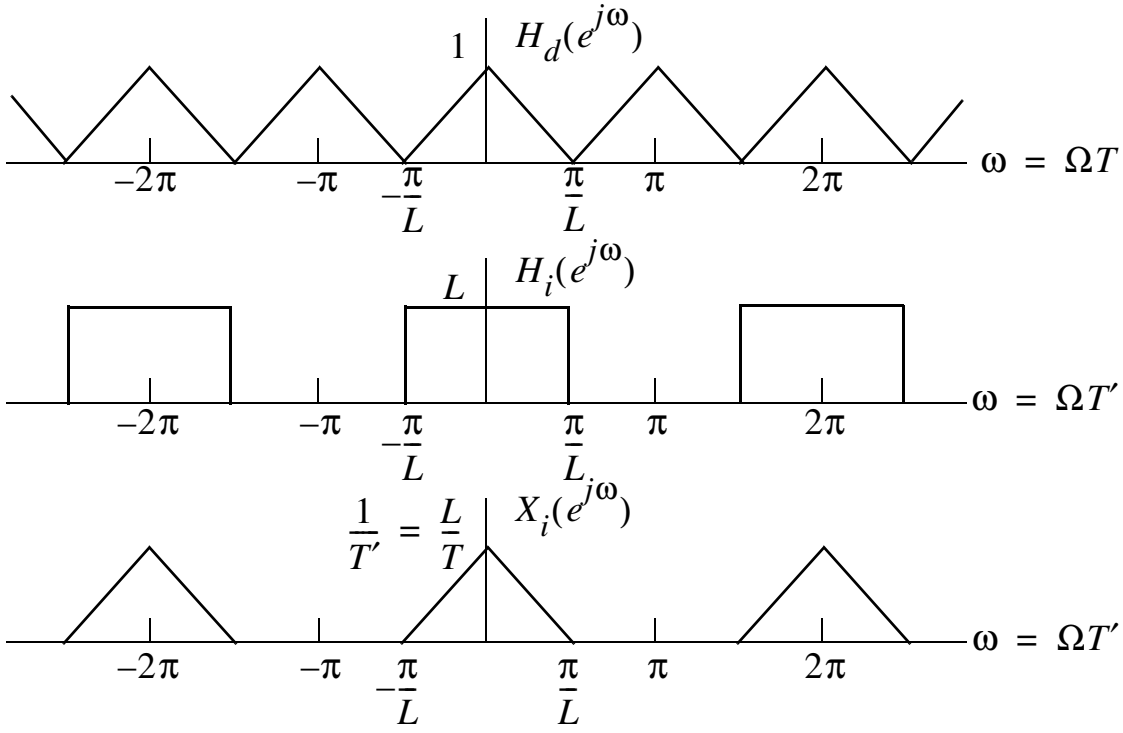
- From the above equation we see that the expander output is simply a frequency scaled version of $X(e^{j\omega})$ (i.e. $\omega \rightarrow \omega L = \Omega LT$)

Example 4.9:

Consider upsampling a signal with bandlimited spectrum as shown below using $L = 2$



Spectra of $X_c(j\Omega)$ and $X(e^{j\omega})$



Spectra of $X_e(e^{j\omega})$ and $X_i(e^{j\omega})$ for $L2$ when $H_i(e^{j\omega})$ is ideal

- In the previous example we saw that the reconstruction filter was used to remove all the frequency-scaled images of $X_c(j\Omega)$ except those located at integer multiples of 2π
- In general if we use an ideal lowpass filter for interpolation the filter gain must be L (this gives an overall gain of $L/T = 1/T'$), and the cutoff frequency must be π/L
- Assuming an ideal lowpass interpolation filter we can then write the upsampler output as

$$x_i[n] = \sum_{k=-\infty}^{\infty} x[k]\,\mathrm{sinc}[(n-kL)/L]$$

- In practice the ideal lowpass filter cannot be implemented exactly, however very good approximations are possible

- As an alternative to the ideal lowpass filter one may wish to consider using linear interpolation which corresponds to a filter with impulse response

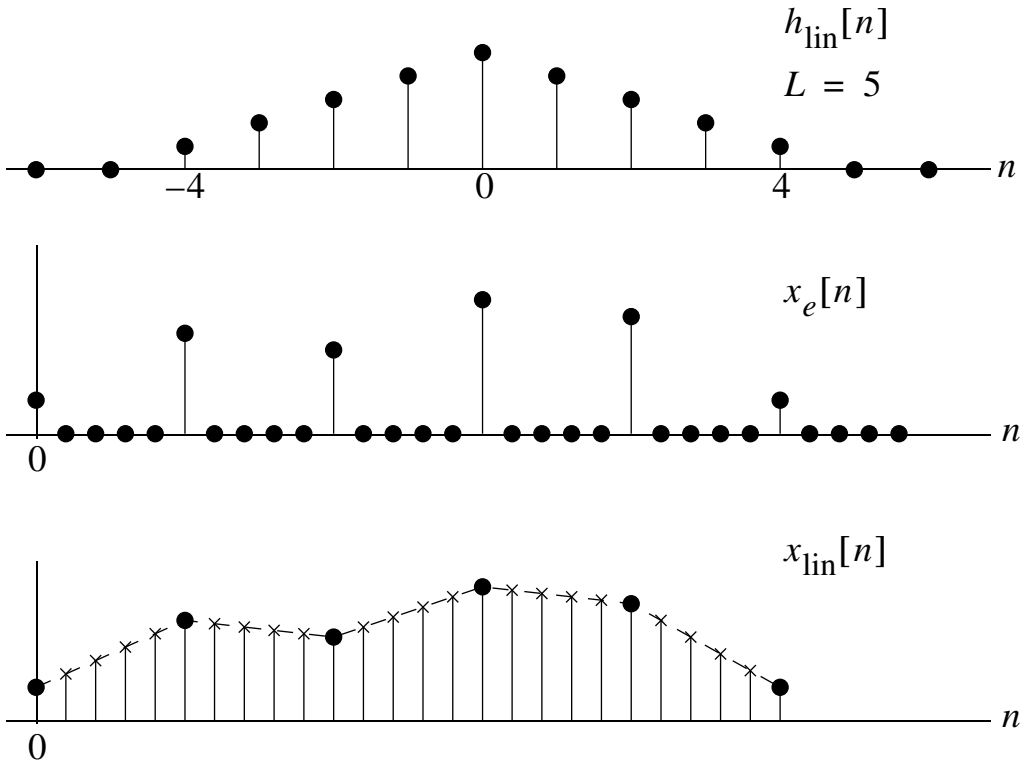
$$h_{\text{lin}}[n] = \begin{cases} 1 - |n|/L, & |n| \leq L \\ 0, & \text{otherwise} \end{cases}$$

- For the case of linear interpolation the upsampler output is

$$\begin{aligned} x_i[n] &= \sum_{k=-\infty}^{\infty} x_e[k] h_{\text{lin}}[n-k] \\ &= \sum_{k=-\infty}^{\infty} x[k] h_{\text{lin}}[n-kL] \end{aligned}$$

Example 4.10:

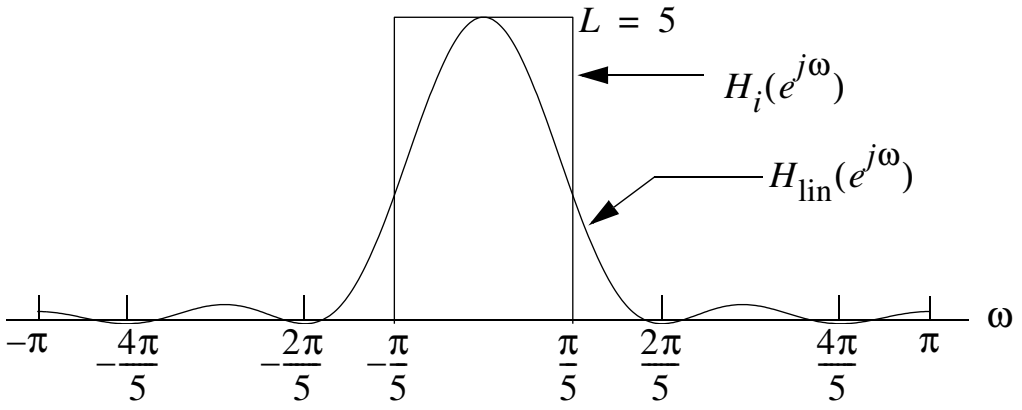
Shown below is an example of linear interpolation for the case $L = 5$. Note that in data plotting linear interpolation is inherent when straight line segments are used to connect the data points.



Linear interpolation; impulse response, & I/O

- The limitations of using a simple linear interpolation filter can be seen in the frequency response of the filter
- It can be shown that

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

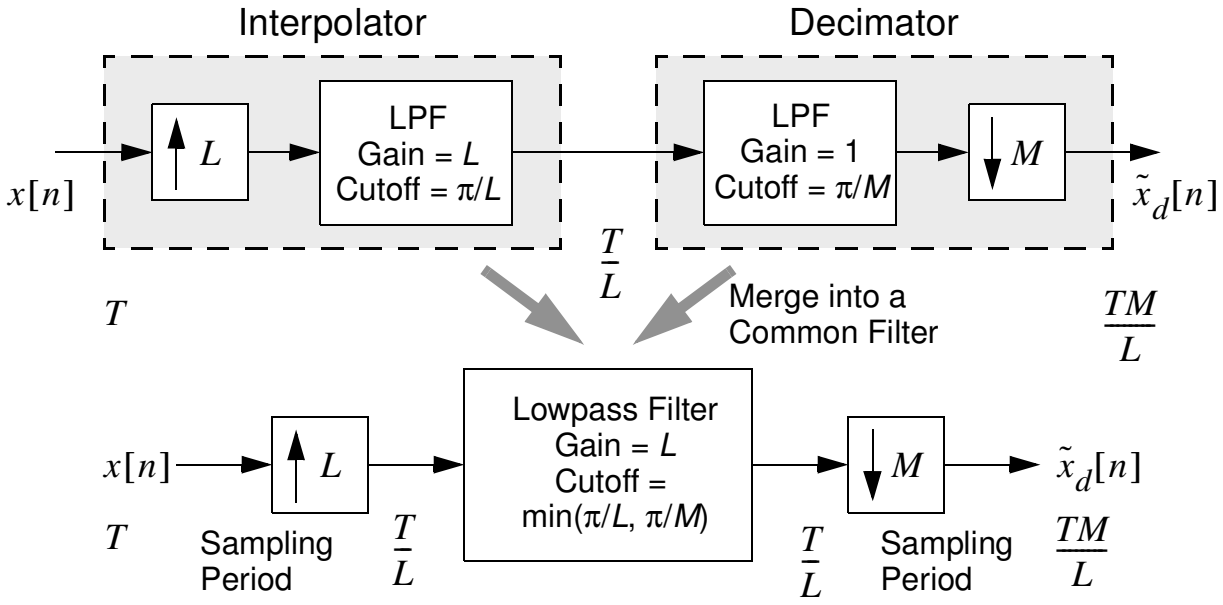


- Note that unless the original sampling rate is well above the Nyquist rate, the interpolation filter will allow adjacent band energy ($\pi/L < |\omega| \leq \pi$) to leak into $x_i[n]$
- Intuitively we can reason that if the sampling rate is well above the Nyquist rate, then the signal does not vary much between samples, thus linear interpolation gives accurate results

4.9.1 Changing the Sampling Rate by a Rational Number Factor

To change the sampling rate by a rational number such as $T' = TM/L$ all we need to do is cascade an interpolator (upsampler) with a decimator (downsampler).

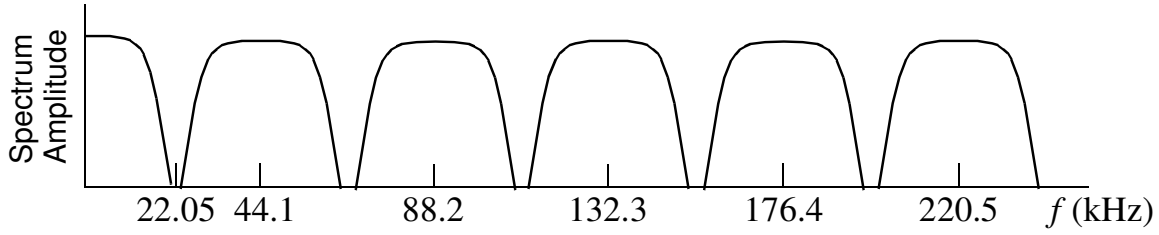
- Note that the interpolation filter of the upsampler will be in cascade with the downsampler prefilter, thus the two filters may be combined into a single lowpass filter with cutoff frequency equal to the minimum of π/L and π/M



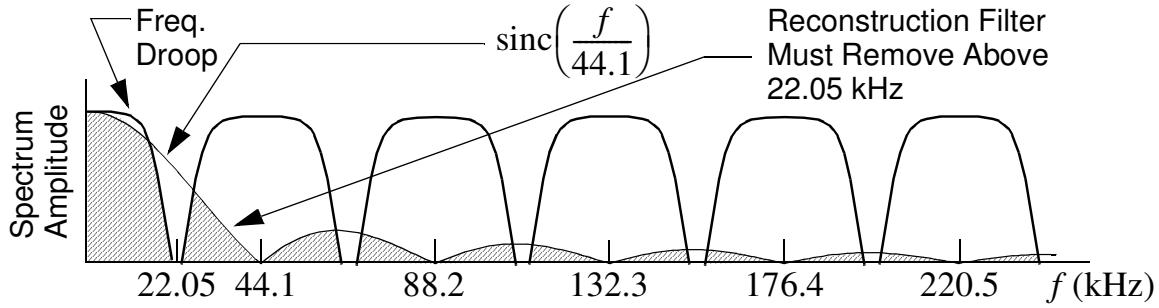
Example 4.11:

In commercial compact disc (CD) players upsampling using $L = 2$ or 4 is often employed to simplify the design of the analog reconstruction filter, and also to avoid the use of a sharp cutoff filter which introduces unwanted phase shifts near the audio frequency bandedge.

- In the CD system the initial sampling rate is fixed at 44.1 kHz
- The spectrum of the output signal following conversion to an impulse train is shown below

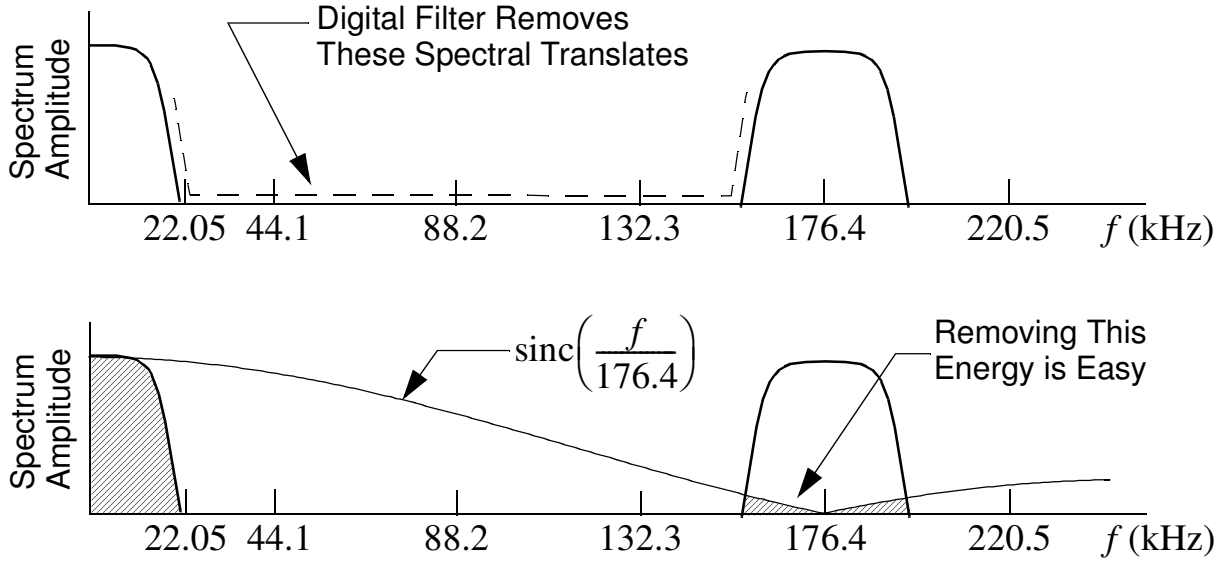


- The digital-to-analog converter (DAC) performs a zero-order hold operation on the impulse train so that the output becomes a stair-step approximation (more on this in Section 4.12.4). The frequency response of the zero-order hold is of the form of a sinc() function



Audio spectrum including zero-order hold (DAC) response

- By using say $L = 4$ upsampling (4 times oversampling) and a digital interpolation filter, the nearest spectral image that the analog reconstruction filter has to remove is shifted from 44.1 kHz up to 176.4 kHz



Spectrum following upsampler & ZOH (DAC)

Example 4.12: Digital Radio Receiver Techniques

The design of modern wireless receivers relies heavily upon digital signal processing techniques. Capturing a radio frequency (rf) signal from the cellular band (∼850 MHz), PCS band (∼1900 MHz), or from any microwave carrier frequency for that matter, and converting it into a *complex baseband* discrete-time signal, is the subject of this example.

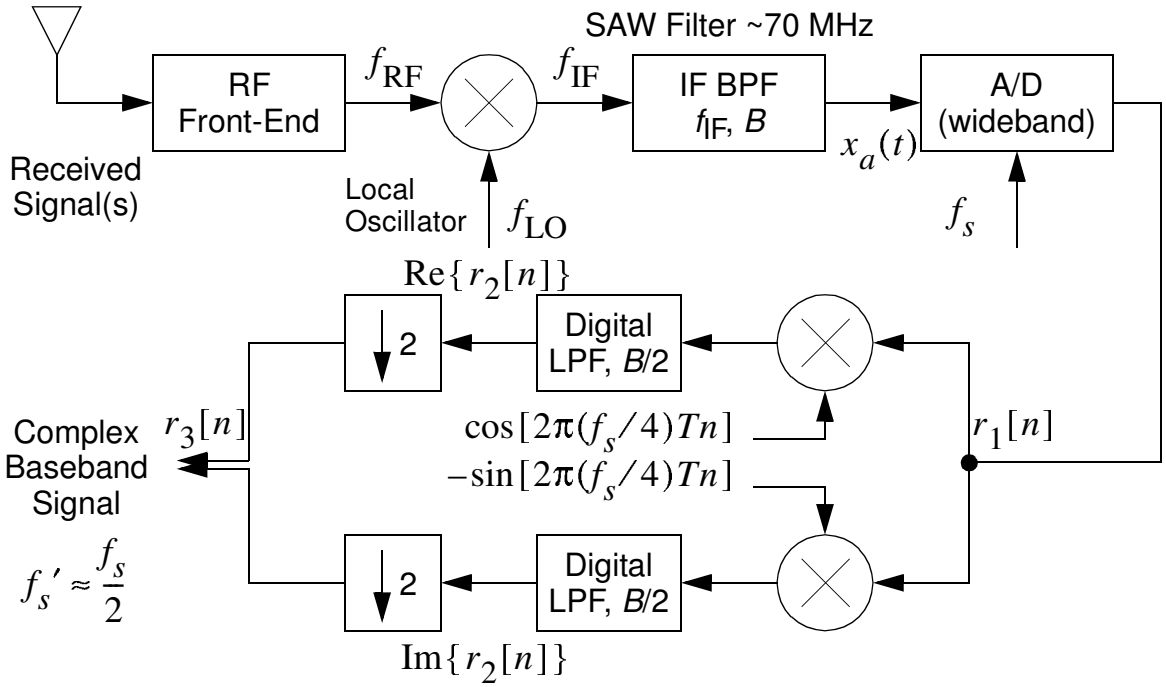
- Three primary options exist:
  1. Convert to an in-phase and quadrature (I&Q) baseband signals using analog circuitry, then sample using two A/D converters to form a discrete-time complex baseband signal
  2. Convert only to an intermediate frequency (IF) using analog circuitry, then bandpass sample (undersample) the IF signal using a wideband input A/D, and digitally mix (frequency translate) to a complex baseband signal
  3. Directly bandpass sample the received rf signal using a minimum of analog circuitry, and then form a discrete-time complex baseband signal using combinations of digital mixing, filtering, and decimation
- In this example we explore the second option which is often referred to as IF sampling[1][2][3]

[1]Matt Easley, "Digital Processing in the IF Domain," *Communication System Design*, August 1999, pp. 46–51.

[2]Joseph Liberti and Theodore Rappaport, *Smart Antennas for Wireless Communications*, Prentice Hall, Upper Saddle River, NJ, 1999.

[3]Gary Saulnier, et al, "A VLSI Demodulator for Digital RF Network Applications: Theory and

- A block diagram of an IF sampling digital receiver is shown below



IF sampling digital receiver block diagram

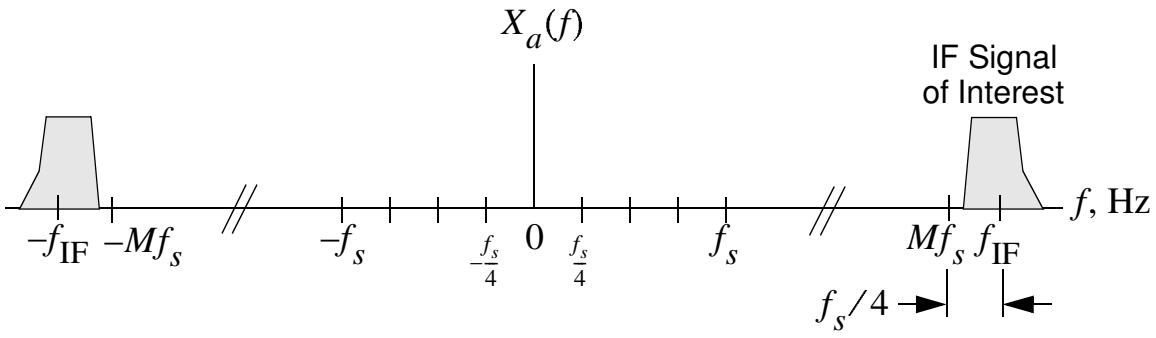
- The rf front-end bandpass filters and low noise amplifies the received signal

- The rf bandpass filter is wide with respect to the signal of interest since the receiver is typically capable of tuning to many different rf carrier frequencies

- The received rf signal is converted to a common IF frequency using an analog mixer followed by a narrow bandpass filter, often a surface acoustic wave (SAW) device if the IF is in the vicinity of 70 MHz

Results," *IEEE Journal and Selected Areas in Communications*, October 1990, pp. 1500–1511.

- The IF filter is very important here in that it must pass the desired signal yet reject adjacent channel signals and noise
- The choice of the exact IF frequency, f_{IF}, and the corresponding bandwidth B are design parameters
- The signal spectrum of $x_a(t)$, prior to sampling, is shown below



The spectrum of IF signal $x_a(t)$

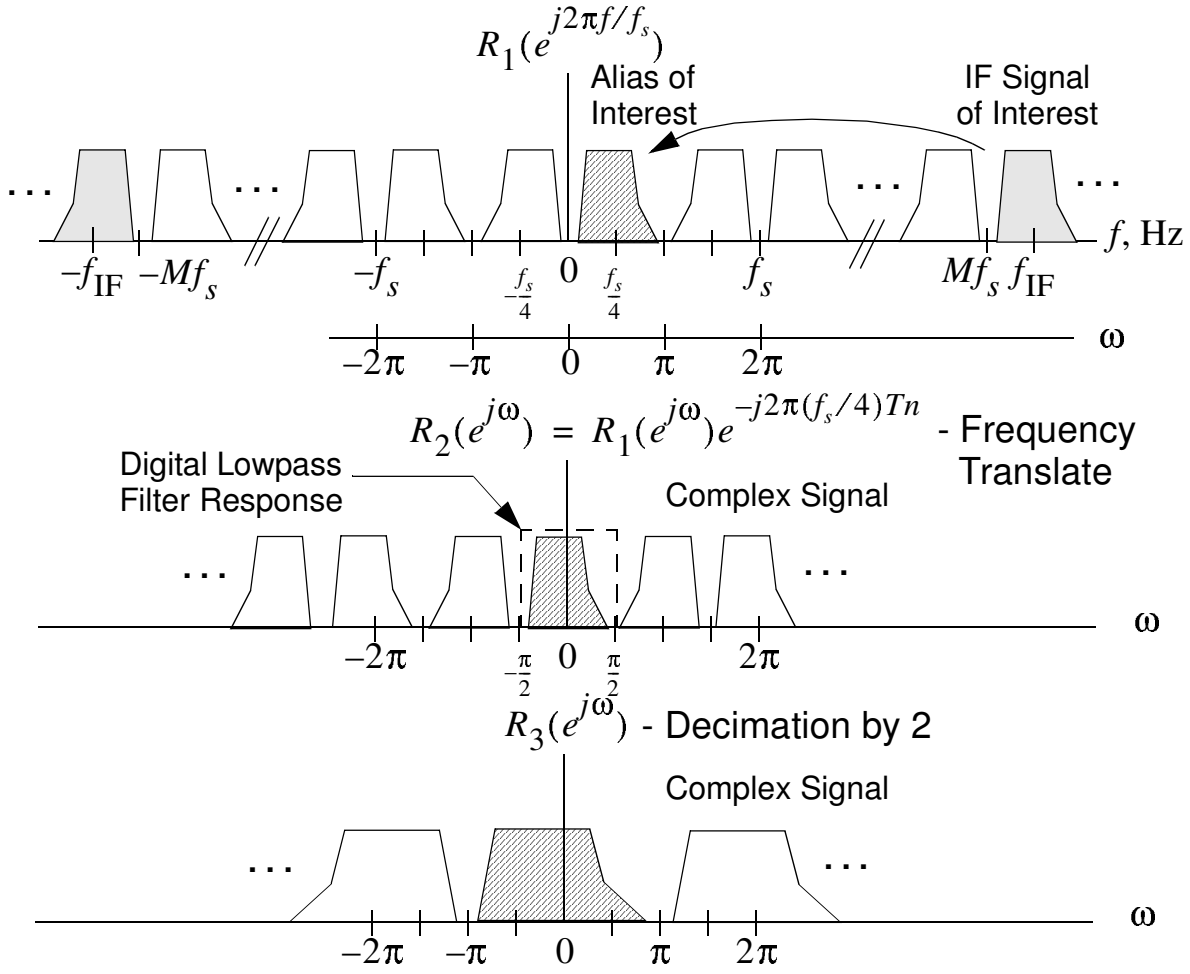
- By undersampling the IF signal we can accomplish both digitizing and frequency translation in one step
- Undersampling introduces *controlled aliasing* such that the desired signal can be futher processed in the fundamental interval $[-f_s/2,\ f_s/2]$ Hz or $[-\pi,\ \pi]$ rad/s on the ω axis
- The design constraints (formulas) are that we choose

$$|f_{\text{IF}} - Mf_s| = f_s/4$$
$$B \leq f_s/2$$

Note: As a matter of practice B is fixed by the signal that is being received and f_s is chosen to allow for filter transition bands, but also to be an integer multiple of the bit rate; f_{IF} and

M are chosen accordingly within physical design constraints of devices

- The above two expressions insure that following the undersampling the IF spectrum $X_a(f)$ ($X_a(j\Omega)$ with Ω replaced by $2\pi f$) will be centered on $\pm f_s/4$ and the various spectral translates will not overlap (alias) with each other
  - Note: If Mf_s is above f_{IF} the signal spectrum will flip



- The entire spectrum will next be shifted down or up in frequency by $f_s/4$ Hz ($\pi/2$ on the ω axis); in general down unless the spectrum is flipped

- This frequency translation is performed in the discrete-time domain using a complex local oscillator of the form

$$\begin{aligned} e^{\pm j2\pi(f_s/4)Tn} &= \cos[2\pi(f_s/4)Tn] \pm j\,\sin[2\pi(f_s/4)Tn] \\ &= \cos[\pi n/2] \pm j\,\sin[\pi n/2] \end{aligned}$$

  Note: This multiplication is very simple as cos and sin take only the values 0, 1, and -1 (very nice!)

- Frequency translation results in a complex signal, which in this case is now centered on dc, so we call it a complex baseband signal, which is composed of real (I) and imaginary (Q) parts

- Each signal path, real and imaginary, is now typically filtered by a linear phase FIR filter to provide signal *matched filtering*

- Since the signal of interest is constrained to lie on the interval of $[\pi/2,\ \pi/2]$ we can also decimate the signal by two to further save processing resources

- Further simplifications are possible and are discussed in the references

Specific Frequency Plan

- Consider a PCS band application for the design of an IS-95 Code Division Multiple Access (CDMA) receiver

- The Block E receive band is 1885–1890 MHz, which in Colorado Springs is used by Qwest

- Three CDMA carrier frequencies reside in this block, the center carrier frequency is 1887.5 MHz with a bandwidth of 1.25 MHz

- The IS-95 CDMA signal has a *chip rate* of 1.2288 Mchips/s and is pulse shaped to occupy an rf bandwidth of 1.25 MHz
- Here we choose $f_s = 7.3728$ Msamples/s (6 samples/chip) so $f_s/4 = 1.8432$ MHz is our *digital IF* frequency
- The analog IF frequency must be either 1.8432 MHz above or below $M \times 7.3728$ MHz
- Choosing $M = 9$ and then $f_{\text{IF}} = 68.1984$ MHz works, which then requires $f_{\text{LO}} = 1819.3016$ MHz

4.10 Multirate Signal Processing

In *multirate signal processing* more than one sampling rate is employed in the DSP system. This in general means one or more decimator, interpolator, and the associated digital filters.

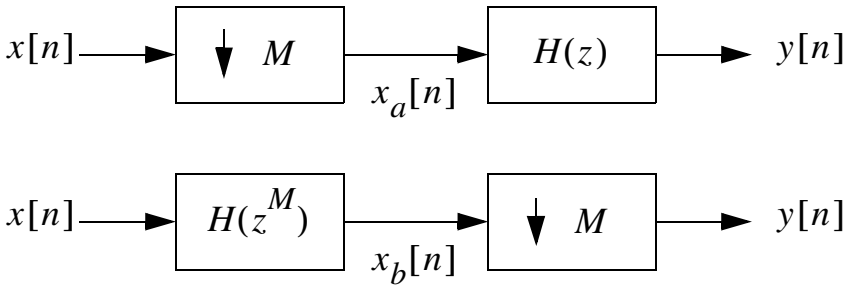
- The CD player example (Example 4.11) is a multirate system
- A rate changing system where the ratio of input over output sampling rates is say 101/100, would nominally require interpolation using $L = 100$ followed by decimation using $M = 101$; intermediate rate changes by smaller amounts can reduce the computational burden

The theme of this section is how to efficiently combine filtering downsampling/upsampling operations into what is known as a *polyphase decomposition*.

4.10.1 Interchanging of Filtering and Downsampling/Upsampling

The polyphase decomposition approach relies on a downsampling and an upsampling identity.

Downsampling Identity



- To show the equivalence we work in the frequency domain on the lower system, and write that

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

- From the previous analysis of decimation by M, we know that

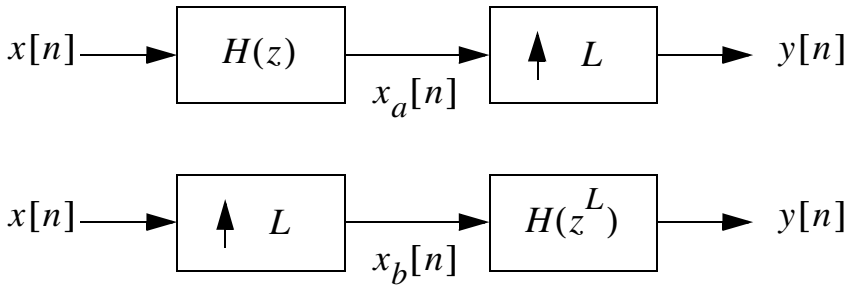
$$Y(e^{j\omega}) = \frac{1}{M}\sum_{i=0}^{M-1} X(e^{j(\omega/M - 2\pi i/M)})H(e^{j(\omega-2\pi i)})$$

- Due to the periodicity of $H(e^{j\omega})$, $H(e^{j(\omega-2\pi i)}) = H(e^{j\omega})$, so the above becomes

$$\begin{aligned} Y(e^{j\omega}) &= H(e^{j\omega})\frac{1}{M}\sum_{i=0}^{M-1} X(e^{j(\omega/M-2\pi i/M)}) \\ &= H(e^{j\omega})X_a(e^{j\omega}) \end{aligned}$$

  by application of the decimator results to $x[n]$

Upsampling Identity



Two equivalent upsampling systems

- Beginning with the top system, we can write that

$$\begin{aligned} Y(e^{j\omega}) &= X_a(e^{j\omega L}) \\ &= X(e^{j\omega L})H(e^{j\omega L}) \end{aligned}$$

- Using the previously developed results for an interpolator, we also know that

$$X_b(e^{j\omega}) = X(e^{j\omega L})$$

so it follows upon substitution that in the lower system,

$$Y(e^{j\omega}) = H(e^{j\omega L})X_b(e^{j\omega})$$

4.10.2 Polyphase Decompositions

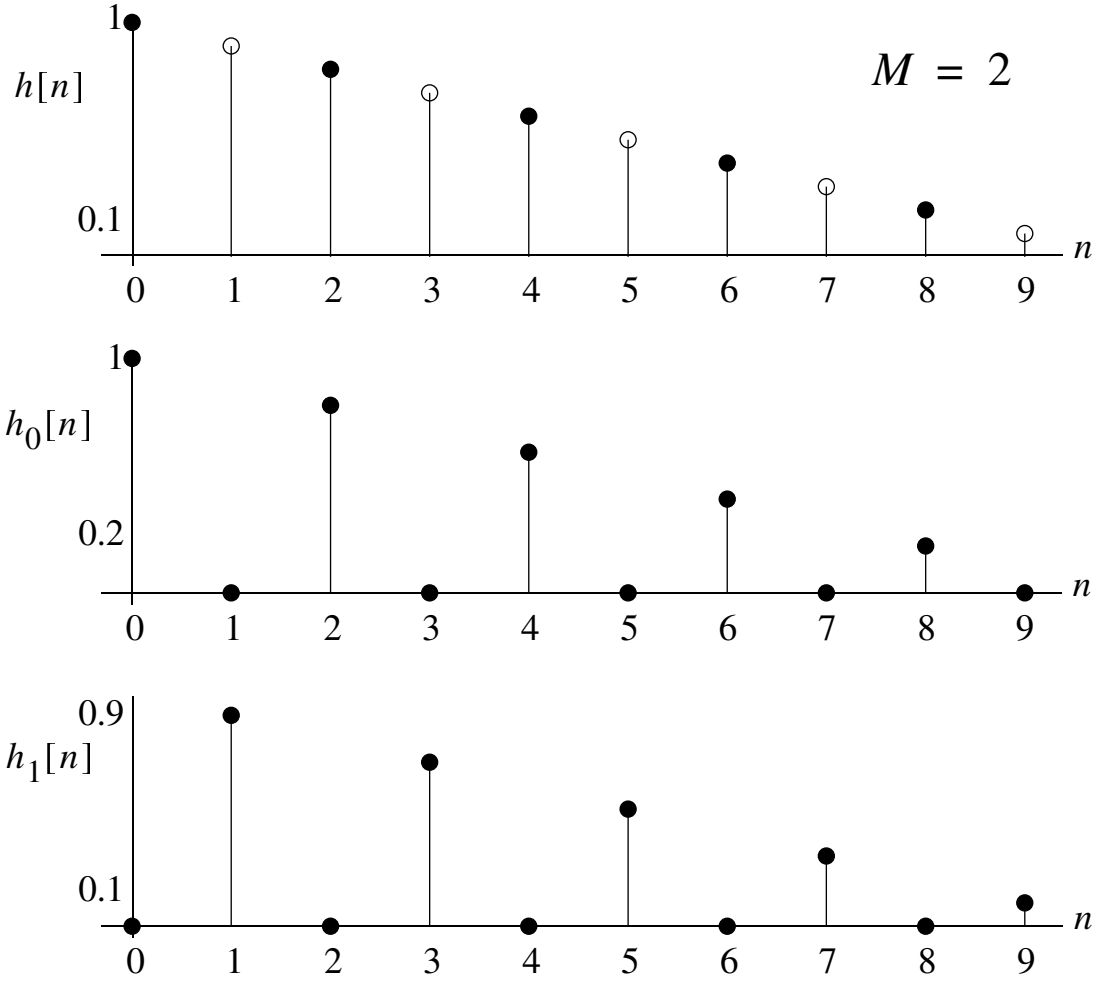
- The polyphase decomposition begins by breaking impulse response $h[n]$ into M subsequences, $h_k[n]$, as follows:

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

- To return to the original impulse response, $h[n]$, from the subsequences just form the sum

$$h[n] = \sum_{k=0}^{M-1} h_k[n-k]$$

- As a *simple example* consider the following 10-point $h[n]$ with $M = 2$:

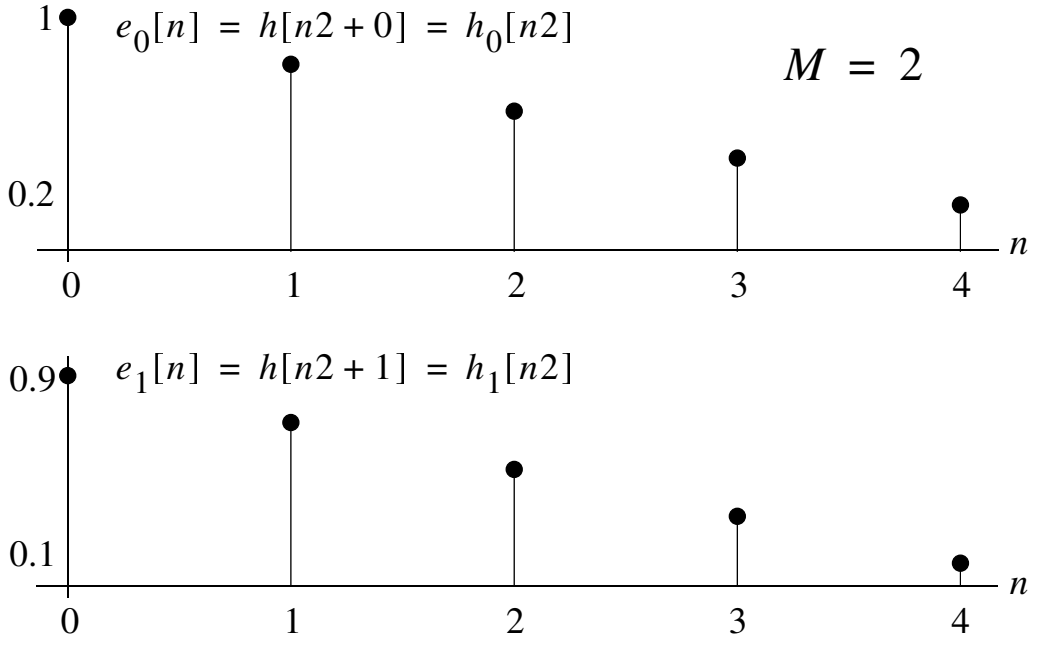


Decomposition of $h[n]$ for $M = 2$ into $h_k[n]$ sequences

- By delaying and decimating $h[n]$ or the $h_k[n]$ sequences by M, we obtain sequences denoted $e_k[n]$

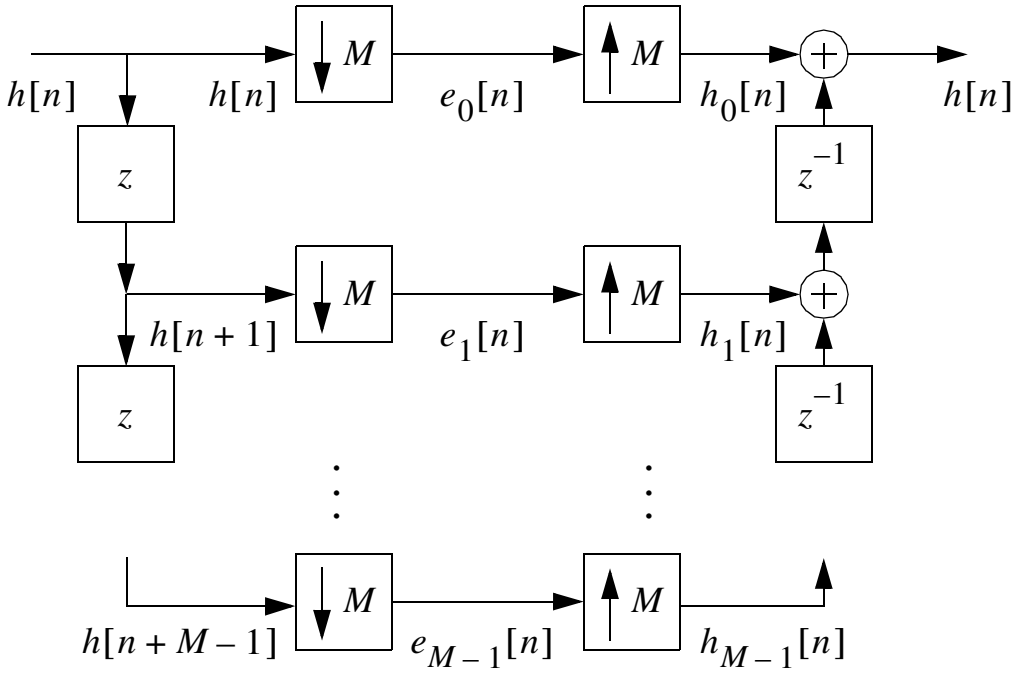
$$e_k[n] = h[nM + k] = h_k[nM]$$

- Continuing the *simple example* above, $e_0[n]$ and $e_1[n]$ are as follows:



Decomposition of $h[n]$ for $M = 2$ into $e_k[n]$ sequences

- A block diagram representation that breaks $h[n]$ down into the components $e_k[n]$ and then reassembles $h[n]$ is useful in understanding how to build an efficient decimator and interpolator



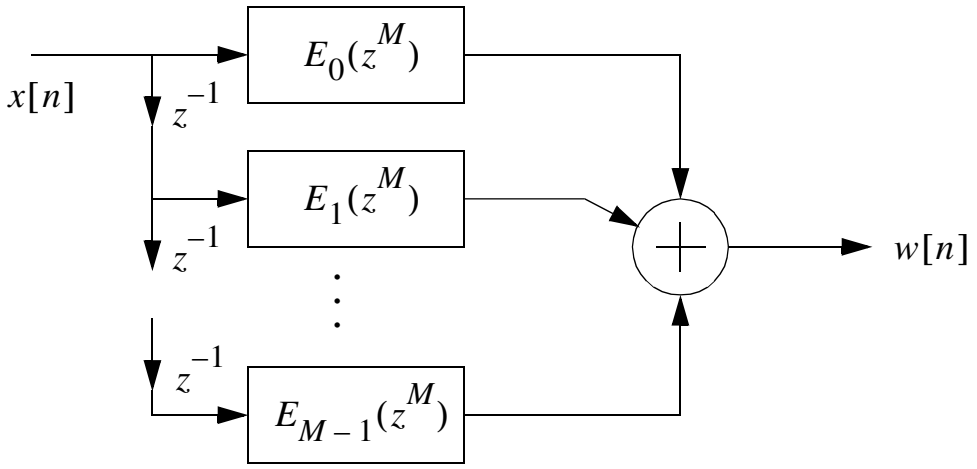
Polyphase decomposition of $h[n]$ using $e_k[n]$ components

- The block diagram shown above is not yet practical since the z blocks are unrealizable

- Working from $E_k(z) = \mathcal{Z}\{e_k[n]\}$ we can write $H(z)$ as

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

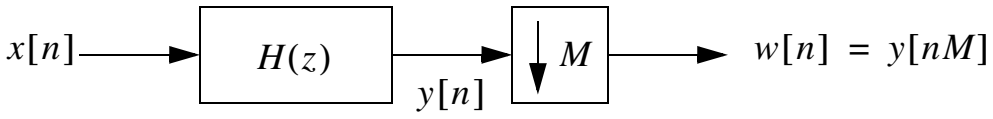
- Suppose $w[n] = x[n] * h[n]$, then polyphase filtering of $x[n]$ to produce $w[n]$ can be represented as shown below



Polyphase filtering structure for $w[n] = x[n] * h[n]$

4.10.3 Polyphase Implementation of Decimation Filters

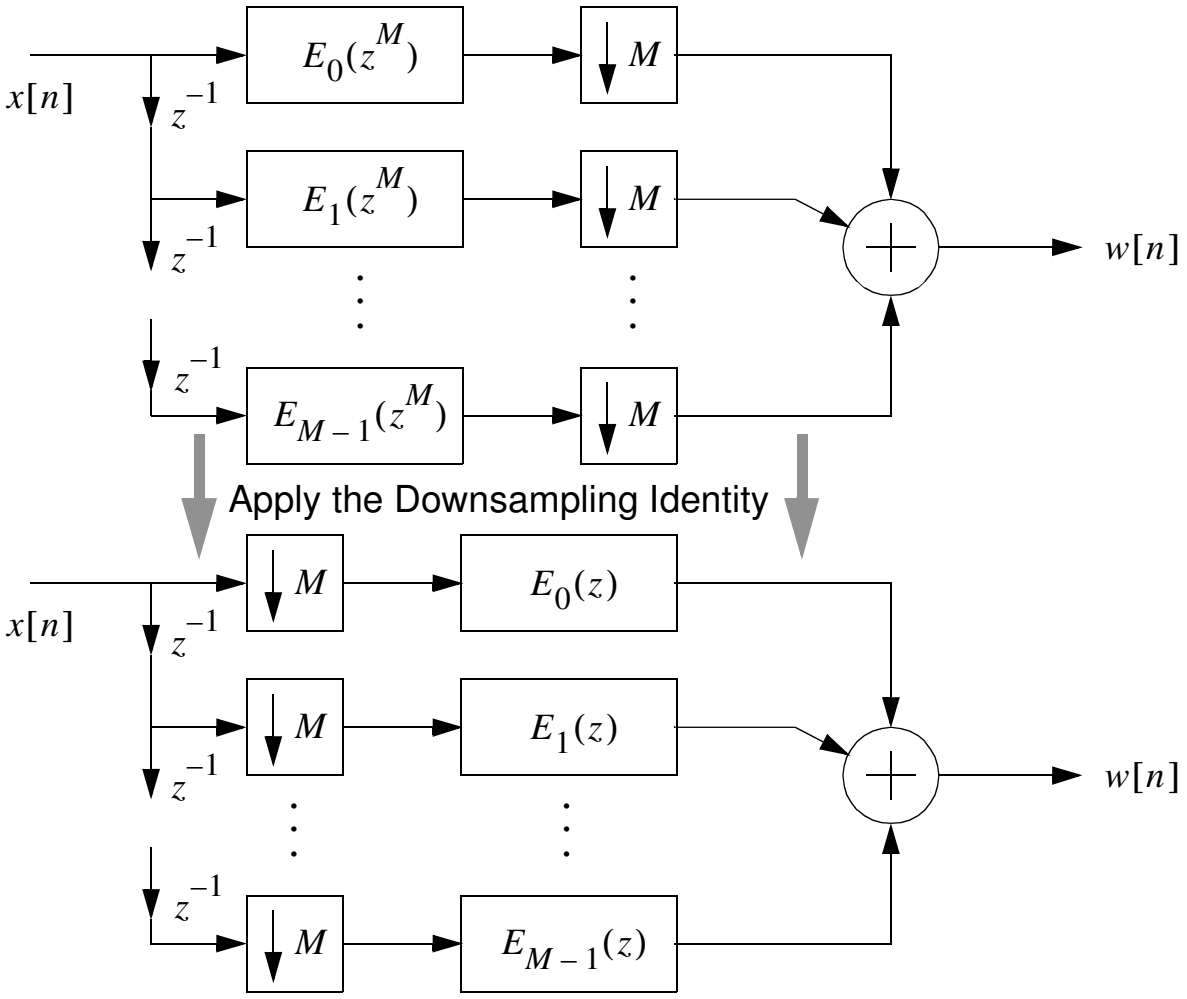
- A complete decimation system requires a lowpass prefilter followed by a decimator



A decimation by M system

- Since the decimator only keeps every Mth sample, it would be nice to make this operation more efficient

- Using the polyphase decomposition of $h[n]$ followed by a decimator we can form the following block diagram:



Polyphase decomposition decimation

- The second version of the above figure results from the down sampling identity for an N-point FIR filter
- In the final polyphase implementation each filter, $E_k(z)$, requires $\frac{1}{M}\left(\frac{N}{M}\right)$ multiplications and $\frac{1}{M}\left(\frac{N}{M}-1\right)$ additions per input sample
- Comparing the total computations count per input sample for the direct N-tap FIR decimator and the polyphase decimator we have the following:

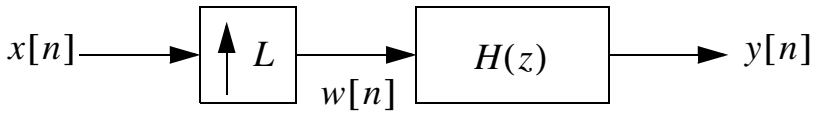
Decimator Operations per Input Sample		
Scheme	**Additions**	**Multiplications**
Direct Implementation	$N-1$	N
Efficient Polyphase	$\left(\frac{N}{M}-1\right)+(M-1)$	$\frac{N}{M}$

- For $N = 50$ and $M = 4$ the saving in multiplications is 50 versus $50/4 = 12.5$ or a factor of 4

- The additions count is 49 versus 14.5 or a factor of about 3

4.10.4 Polyphase Implementation of Interpolation Filters

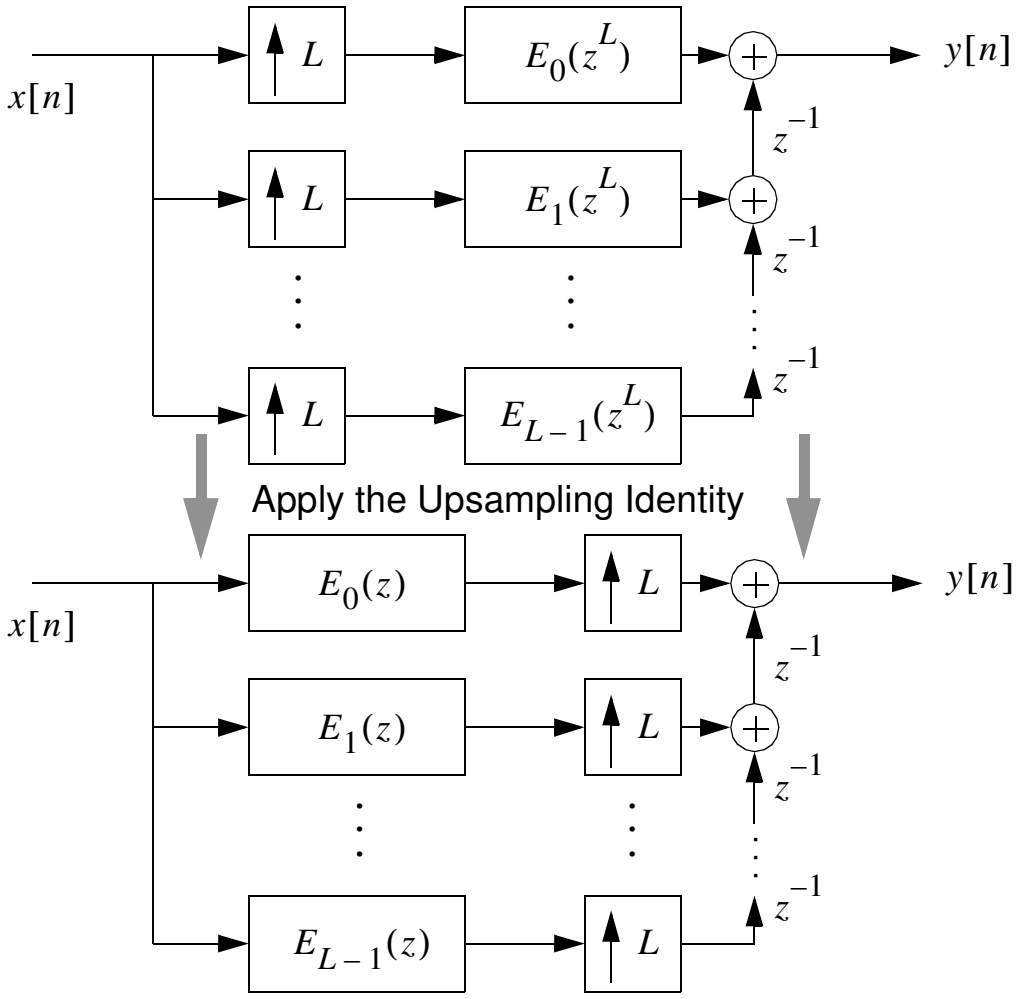
- A complete interpolation system requires an interpolator followed by a lowpass postfilter to fill in the zero samples



An interpolation by L system

- Since the interpolation inserts $L-1$ zero samples between each input sample, it would be nice to make this operation more efficient as with the decimator described earlier

- Apply a polyphase decomposition of $h[n]$ following interpolation by L



Polyphase decomposition interpolation

- The second version of the above figure results from the up sampling identity
- In the final polyphase implementation each filter, $E_k(z)$, requires $\left(\frac{N}{L}\right)$ multiplications and $\left(\frac{N}{L}-1\right)$ additions per input sample
- Comparing the total computations count per input sample for the direct N-tap FIR interpolator and the polyphase interpolator we have the following:

Interpolator Operations per Input Sample		
Scheme	**Additions**	**Multiplications**
Direct Implementation	$NL-1$	NL
Efficient Polyphase	$L\left(\frac{N}{L}-1\right)+(L-1)$	$L\left(\frac{N}{L}\right)$

- For $N = 50$ and $L = 4$ the saving in multiplications is 200 versus 50 or a factor of 4
- The additions count is 199 versus 49 or a factor of about 4

4.11 Discrete-Time Random Signals

To prepare for the study of quantization noise in digital processing of analog signals and other related modeling issues, we now need to take time out to introduce the basics of random process theory.

In this section the emphasis will be on the characterization of discrete-time random processes as opposed to both continuous and discrete-time random processes. The major concepts are statistical averages, correlation and covariance functions, and the power spectrum. Of special importance is the analysis of random signals in linear time invariant systems. Before getting into the random process topics we will briefly review probability and random variables.

4.11.1 Probability

Consider an experiment which has possible outcomes ξ_i.

- The ξ_i's are elements of the space $\mathcal{S}$ known as the sample space
- A subset of $\mathcal{S}$ is a collection of elements ξ_i i.e. for

$$A \subset \mathcal{S} \qquad B \subset \mathcal{S}$$

we might have

$$A = \{\xi_1\} \qquad B = \{\xi_1,\ \xi_3,\ \xi_5\}$$

Note that here $A \subset B$

- **Events:**
  - Subsets of $\mathcal{S}$ which make up a σ-field are *events* on $\mathcal{S}$
  - The sample space $\mathcal{S}$ is called the *certain event*
  - The empty set, $\emptyset$, is called the *impossible event*
- Associated with each event is a measure called *probability*
- **Axioms of Probability:**
  1. $P(A) \geq 0$ for all A in $\mathcal{S}$
  2. $P(\mathcal{S}) = 1$
  3. If $AB = A \cap B = \emptyset$ then

$$P(A + B) = P(A \cup B) = P(A) + P(B)$$

Note that (3) extends to

$$P\left(\bigcup_{i=1}^{n} A_i\right) = P\left(\sum_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} P(A_i)$$

if $A_i A_j = \emptyset\ \forall\ i \neq j$

- **Joint Probability:**

$$P(AB) = P(A \cap B) = P(A, B)$$

- **Conditional Probability:**

  Given two events A and B, the probability of A given B is

$$P(A|B) = \frac{P(AB)}{P(B)}$$

  also

$$P(B|A) = \frac{P(AB)}{P(A)}$$

  $\Longrightarrow$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad \text{(Bayes Rule)}$$

- **Theorem of Total Probability:**

  Given n mutually exclusive (disjoint) events

$$A_i A_j = \emptyset \;\; i \neq j = 1, 2, \ldots, n$$

  such that $\sum_{i=1}^{n} A_i = \mathcal{S}$, then for an arbitrary $B \subset \mathcal{S}$

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

  proof

$$B = B\mathcal{S} = B\left(\sum_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} BA_i$$

  so

$$P(B) = P\left(\sum_{i=1}^{n} BA_i\right) = \sum_{i=1}^{n} P(BA_i)$$

  but $P(BA_i) = P(B|A_i)P(A_i)$

- **Independence:**

  Events A and B are statistically independent if and only if

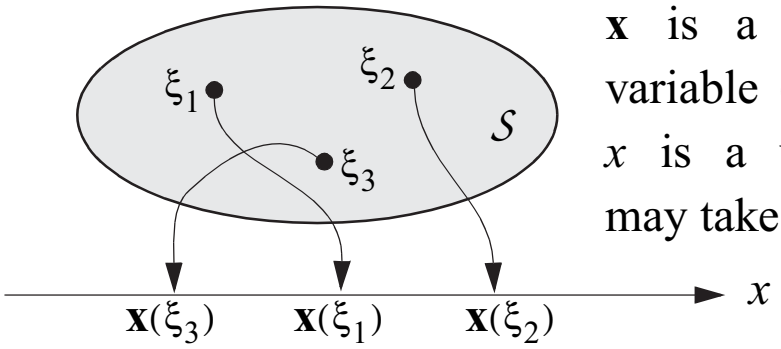
$$P(AB) = P(A)P(B)$$

  It then follows that

$$P(A|B) = P(A) \quad \text{and} \quad P(B|A) = P(B)$$

4.11.2 Random Variables

A *random variable* is a rule that assigns a numerical value to outcomes of a chance experiment



$\mathbf{x}$ is a random variable (rv) and x is a value it may take on.

Elementary outcomes mapped to the real line

It may be continuous, discrete, or mixed. A random variable (rv) will be denoted as $\mathbf{x}(\xi_i) = \mathbf{x}$ or $X(\xi_i) = X$. The values it takes on will be denoted by x.

- **Cumulative Distribution Function (CDF):** or probability distribution function

$$P_{\mathbf{x}}(x) = P[\{\xi : \mathbf{x}(\xi) \leq x\}] = P[\mathbf{x} \leq x]$$

Properties

1. $P_{\mathbf{x}}(-\infty) = 0, \quad P_{\mathbf{x}}(\infty) = 1$
2. Right continuous i.e.

$$\lim_{x \to x_o^+} P_{\mathbf{x}}(x) = P_{\mathbf{x}}(x_o)$$

3. A nondecreasing function i.e.

$$P_{\mathbf{x}}(x_1) \leq P_{\mathbf{x}}(x_2), \text{ if } x_1 < x_2$$

- **Probability Density Function (PDF):**

$$p_{\mathbf{x}}(x) = \frac{dP_{\mathbf{x}}(x)}{dx}$$

Note that

$$P_{\mathbf{x}}(x) = \int_{-\infty}^{x} p_{\mathbf{x}}(u)\, du$$

Properties

1. $p_{\mathbf{x}}(x) \geq 0$
2. $\int_{-\infty}^{\infty} p_{\mathbf{x}}(x)\, dx = 1$
3. $P[x_1 < \mathbf{x} \leq x_2] = P_{\mathbf{x}}(x_2) - P_{\mathbf{x}}(x_1) = \int_{x_1}^{x_2} p_{\mathbf{x}}(x)\, dx$

- **Joint RVs:**

CDF

$$P_{\mathbf{x},\mathbf{y}}(x, y) = P[\mathbf{x} \leq x, \mathbf{y} \leq y]$$

PDF

$$p_{\mathbf{x},\mathbf{y}}(x, y) = \frac{\partial^2 P_{\mathbf{x},\mathbf{y}}(x, y)}{\partial x\, \partial y}$$

Note:

$$P_{\mathbf{x},\mathbf{y}}(x,y) = \int_{-\infty}^{x}\int_{-\infty}^{y} p_{\mathbf{x},\mathbf{y}}(u,v)\,du\,dv$$

- **Conditional pdfs:**

$$p_{\mathbf{y}|\mathbf{x}}(y|x) = \frac{p_{\mathbf{x},\mathbf{y}}(x,y)}{p_{\mathbf{x}}(x)}$$
$$p_{\mathbf{x}|\mathbf{y}}(x|y) = \frac{p_{\mathbf{x},\mathbf{y}}(x,y)}{p_{\mathbf{y}}(y)}$$

and Bayes rule

$$p_{\mathbf{y}|\mathbf{x}}(y|x) = \frac{p_{\mathbf{x}|\mathbf{y}}(x|y)\,p_{\mathbf{y}}(y)}{p_{\mathbf{x}}(x)}$$

- **Independent RVs:**

$$\begin{aligned} P_{\mathbf{x},\mathbf{y}}(x,y) &= P_{\mathbf{x}}(x)\,P_{\mathbf{y}}(y) \\ p_{\mathbf{x},\mathbf{y}}(x,y) &= p_{\mathbf{x}}(x)\,p_{\mathbf{y}}(y) \end{aligned}$$

- **Statistical Averages:**

Mean or first moment

$$m_{\mathbf{x}} = \mathcal{E}\{\mathbf{x}\} = \int_{-\infty}^{\infty} x\,p_{\mathbf{x}}(x)\,dx$$

Fundamental Theorem of Expectation

$$\mathcal{E}\{g(\mathbf{x})\} = \int_{-\infty}^{\infty} g(x)\,p_{\mathbf{x}}(x)\,dx$$

mean square $\mathcal{E}\{\mathbf{x}^2\}$ (second moment)

variance

$$\sigma_{\mathbf{x}}^2 = \mathcal{E}\{(\mathbf{x} - m_{\mathbf{x}})^2\} = \mathcal{E}\{\mathbf{x}^2\} - m_{\mathbf{x}}^2$$

Note: $\sigma_{\mathbf{x}} =$ *standard deviation*

Definition: Random variables $\mathbf{x}$ and $\mathbf{y}$ are uncorrelated if

$$\mathcal{E}\{\mathbf{xy}\} = \mathcal{E}\{\mathbf{x}\}\mathcal{E}\{\mathbf{y}\}$$

Note if either $\mathbf{x}$ or $\mathbf{y}$ has zero mean, then uncorrelated rvs are also *orthogonal* i.e. $\mathcal{E}\{\mathbf{xy}\} = 0$

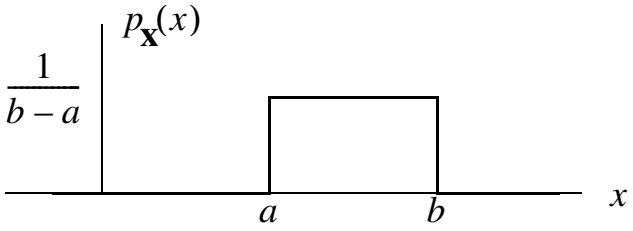
- **Important PDFs:**

  Uniform

  If rv $\mathbf{x}$ is uniform on the interval $[a, b]$, then

$$p_{\mathbf{x}}(x) = \begin{cases} \frac{1}{b-a}, & a \le x \le b \\ 0, & \text{otherwise} \end{cases}$$

  A shorthand notation is simply to say $\mathbf{x} \sim U[a, b]$.



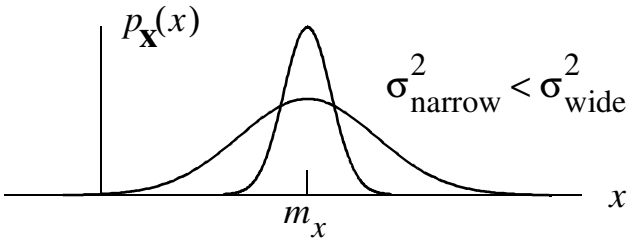
The rv $\mathbf{x}$ uniform on $[a, b]$

  Gaussian or Normal:

  If rv $\mathbf{x}$ is normal with mean $m_{\mathbf{x}}$ and variance $\sigma_{\mathbf{x}}^2$, then

$$p_{\mathbf{x}}(x) = \frac{1}{\sqrt{2\pi\sigma_{\mathbf{x}}^2}} e^{\frac{-(x-m_x)^2}{2\sigma_x^2}}$$

  A shorthand notation is simply to say $\mathbf{x} \sim N[m_{\mathbf{x}}, \sigma_{\mathbf{x}}^2]$.



The rv $\mathbf{x}$ normal with parameters m_x and σ_x^2

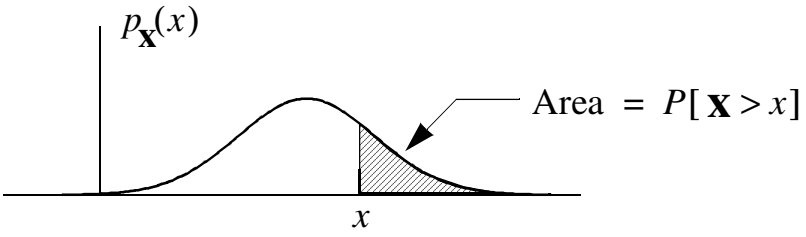
- The area under the tail of a Gaussian PDF can be written in terms of the $Q()$-function, that is

$$P[\mathbf{x} > x] = Q\left(\frac{x - m_{\mathbf{x}}}{\sigma_{\mathbf{x}}}\right)$$

where

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2}\, dt$$

Note that $Q(0) = 1/2$ and $Q(-x) = 1 - Q(x)$.



Tail of normal pdf

A simple, reasonably accurate approximation for $Q(x)$ over $0 < x < \infty$ is

$$Q(x) \simeq \left[\frac{1}{(1-a)x + a\sqrt{x^2 + b}}\right] \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

where $a = 1/\pi$ and $b = 2\pi$

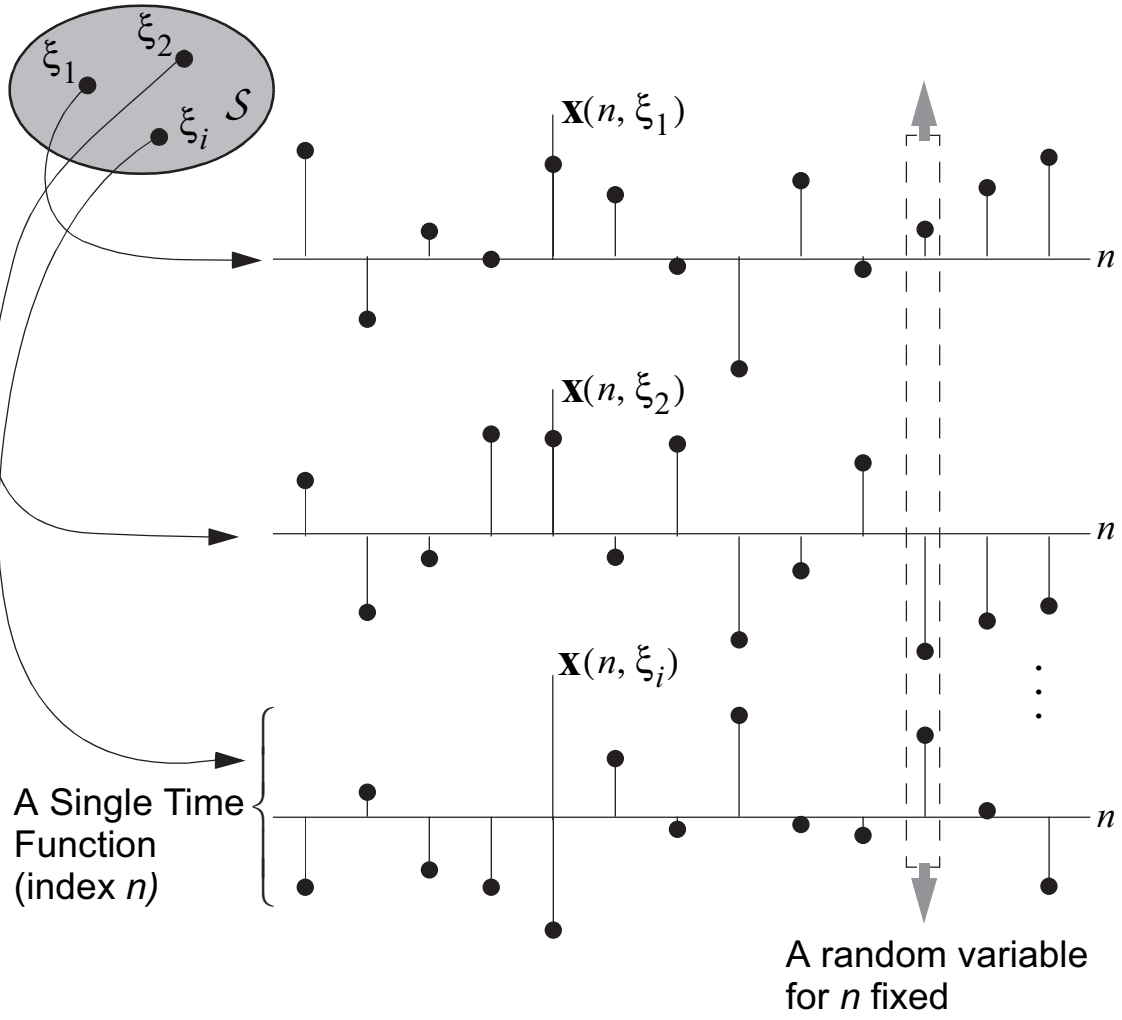
Example 4.13: **A short table of values for $Q(x)$**

x	$Q(x)$	Approx.	x	$Q(x)$	Approx.
0.5	3.09E-01	3.05E-01	3.0	1.35E-03	1.35E-03
1.0	1.59E-01	1.57E-01	3.5	2.33E-04	2.32E-04
1.5	6.68E-02	6.63E-02	4.0	3.17E-05	3.16E-05
2.0	2.28E-02	2.26E-02	4.5	3.40E-06	3.40E-06
2.5	6.21E-03	6.19E-03	5.0	2.87E-07	2.87E-07

4.11.3 Random Processes

Formally a random process (rp) is a family (ensemble) of functions $\mathbf{x}(t,\xi)$ for continous-time or $\mathbf{x}[n,\xi]$ for discrete-time, such that each outcome ξ_i forming the space $\mathcal{S}$ is assigned a time function $\mathbf{x}(t,\xi_i)$ or a sequence $\mathbf{x}[n,\xi_i]$.

- A discrete-time random process can also be viewed as being a sequence of random variables indexed by the discrete-time variable n



Ensemble of sample functions when all ξ_i's are considered

- Interpretations of $\mathbf{x}[n]$ are:
  - A family of time functions, n and $\mathbf{x}$ variables
  - A single time function, n variable and $\mathbf{x}$ fixed
  - A random variable, n fixed and $\mathbf{x}$ variable
  - A single number, n and $\mathbf{x}$ both fixed
- Notationally it is easier to write rp $\mathbf{x}[n] = x[n]$ and let it be understood from the discussion which of the four interpretations is intended

Random Process Characterization

- Since rp $x[n]$ is just an rv for any particular index value, n, we can define the mean and variance as:

$$m_x[n] = \mathcal{E}\{x[n]\}$$

and

$$\begin{aligned}\sigma_x^2[n] &= \mathcal{E}\{|x[n] - m_x[n]|^2\} \\ &= \mathcal{E}\{|x[n]|^2\} - |m_x[n]|^2\end{aligned}$$

- In general the mean and variance are functions of time index n, but if the rp $x[n]$ is *stationary*, then they are both constants

- The *autocorrelation sequence* of $x[n]$ is defined as

$$\phi_{xx}[n,m] = \mathcal{E}\{x[n]x^*[m]\}$$

for the general case of a complex rp

- The *autocovariance sequence* is defined as

$$\begin{aligned}\gamma_{xx}[n,m] &= \mathcal{E}\{(x[n] - m_x[n])(x[m] - m_x[m])^*\} \\ &= \phi_{xx}[n,m] - m_x[n]m_x^*[m]\end{aligned}$$

Note that $\gamma_{xx}[n,n] = \sigma_x^2[n]$

- If $x[n]$ is *wide-sense stationary (WSS)*, then

$$\begin{aligned}m_x[n] &= m_x \\ \sigma_x^2[n] &= \sigma_x^2 \\ \phi_{xx}[n+m,n] &= \phi_{xx}[m] \\ \gamma_{xx}[n+m,n] &= \gamma_{xx}[m] = \phi_{xx}[m] - |m_x|^2\end{aligned}$$

- The *cross-correlation* between jointly WSS sequences $x[n]$ and $y[n]$ is

$$\phi_{xy}[m] = \mathcal{E}\{x[n+m]y^*[n]\}$$

- Similarly the *cross-covariance* is defined as

$$\gamma_{xy}[m] = \mathcal{E}\{(x[n+m] - m_x)(y[n] - m_y)^*\}$$

- If jointly WSS sequence $x[n]$ and $y[n]$ are uncorrelated, then

$$\mathcal{E}\{x[n+m]y^*[n]\} = \mathcal{E}\{x[n+m]\}\mathcal{E}\{y^*[n]\} = m_x m_y^*$$

  If either process has zero mean, then as a special case of being uncorrelated $\mathcal{E}\{x[n+m]y^*[n]\} = 0$

- The power spectral density of a WSS rp is

$$P_{xx}(\omega) = \Phi_{xx}(e^{j\omega}) = \mathcal{F}\{\phi_{xx}[m]\}$$

  Conversely

$$\phi_{xx}[m] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_{xx}(e^{j\omega})e^{j\omega m}\, d\omega$$

- The mean square of $x[n]$ is also interpreted as the average power

$$\mathcal{E}\{|x[n]|^2\} = \phi_{xx}[0] = \frac{1}{2\pi}\int_{-\pi}^{\pi} P_{xx}(\omega)\, d\omega$$

  Note that the ac power or variance is just

$$\sigma_x^2 = \gamma_{xx}(0) = \frac{1}{2\pi}\int_{-\pi}^{\pi} P_{xx}(\omega)\, d\omega - |m_x|^2$$

Time Averages

- Define the time average mean as

$$\langle x[n] \rangle = \lim_{L \to \infty} \frac{1}{2L+1} \sum_{n=-L}^{L} x[n]$$

- Similarly define the time average autocorrelation function as

$$\langle x[n+m]x^*[n] \rangle = \lim_{L \to \infty} \frac{1}{2L+1} \sum_{n=-L}^{L} x[n+m]x^*[n]$$

- From a practical implementation point-of-view we would like to replace the ensemble averages with corresponding time averages over a single observed sample function

- If a process is *mean-ergodic*, then the statistical mean and the time average mean are equivalent

- If a process is *correlation-ergodic*, then the statistical autocorrelation and the time average autocorrelation are equivalent

- Unless stated otherwise in this course we will assume that we are dealing with ergodic processes (also WSS), thus the operations $\langle \cdot \rangle$ and $\mathcal{E}\{\cdot\}$ may be interchanged

- In practice it is not possible for $L \to \infty$, thus m_x, σ_x^2, and $\phi_{xx}[m]$ must be estimated using say, a record of L samples from $x[n]$

- Given a finite-length record of $x[n]$, $n = 0, 1, \ldots, L-1$, an estimator for the autocorrelation sequence assuming $x[n]$ is real, is

$$\hat{\phi}_{xx}[m] = \begin{cases} \frac{1}{L}\sum_{n=0}^{L-|m|-1} x[n]x[n+|m|], & |m| \leq L-1 \\ 0, & \text{otherwise} \end{cases}$$

- Important properties of this estimator are its mean and variance (Note: The estimate $\hat{\phi}_{xx}[m]$ is an rv). The variance is difficult to calculate.

- The mean is easily shown to be

$$\mathcal{E}\{\hat{\phi}_{xx}[m]\} = \begin{cases} \left(\frac{L-|m|}{L}\right)\phi_{xx}[m], & |m| \leq L-1 \\ 0, & \text{otherwise} \end{cases}$$

  Thus the estimate is biased, but the bias is small for $|m| \ll L$.

- An unbiased estimator is

$$\check{\phi}_{xx}[m] = \left(\frac{L}{L-|m|}\right)\hat{\phi}_{xx}[m]$$

- A general observation with regard to the variance of both estimators is that as $|m|$ increases fewer samples are used in the estimate, thus it is expected that the variance should also increase

LTI Systems with RP Inputs

Let $x[n]$ be a real rp input sequence and $h[n]$ a real impulse response, the output rp $y[n]$ is

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

- If $\mathcal{E}\{x[n]\} = m_x$, then

$$\begin{aligned} m_y &= \mathcal{E}\left\{\sum_{k=-\infty}^{\infty} h[k]x[n-k]\right\} \\ &= \sum_{k=-\infty}^{\infty} h[k]\underbrace{\mathcal{E}\{x[n-k]\}}_{m_x} \\ &= m_x \underbrace{\sum_{k=-\infty}^{\infty} h[k]}_{H(e^{j0})} \end{aligned}$$

- Since $x[n]$ is WSS $\phi_{xx}[n+m,n] = \phi_{xx}[m]$ then it can be shown that

$$\phi_{yy}[m] = \phi_{xx}[m] * h[-m] * h[m]$$

An alternate expression is

$$\phi_{yy}[m] = \sum_{l=-\infty}^{\infty} \phi_{xx}[m-l]c[l]$$

where $c[l]$, called the *autocorrelation sequence* of $h[n]$, is defined as

$$\begin{aligned} c[l] &= \sum_{k=-\infty}^{\infty} h[k]h[l+k] = h[-l] * h[l] \\ &= \mathcal{Z}^{-1}[H(z)H(1/z)] \end{aligned}$$

- Taking the Fourier transform of both sides of the above expres-

sion for $\phi_{yy}[m]$ gives

$$\begin{aligned}\Phi_{yy}(e^{j\omega}) &= C(e^{j\omega})\Phi_{xx}(e^{j\omega}) \\ &= H(e^{j\omega})H^*(e^{j\omega})\Phi_{xx}(e^{j\omega}) \\ &= |H(e^{j\omega})|^2\Phi_{xx}(e^{j\omega})\end{aligned}$$

- **White Noise**

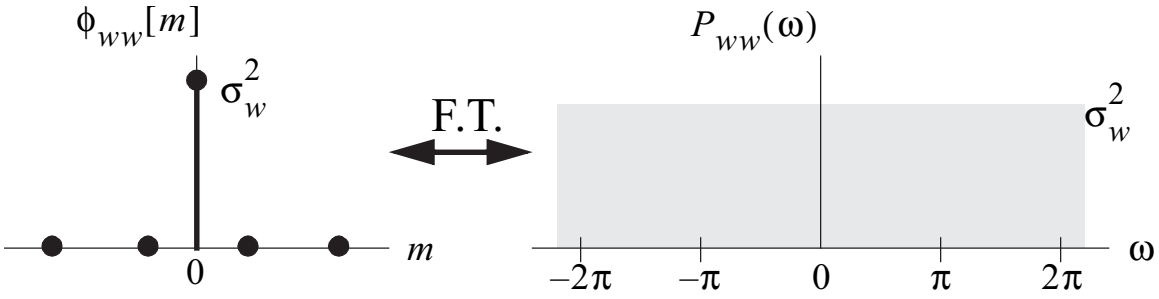
  A WSS rp $w[n]$ is a white noise sequence if

  $$\mathcal{E}\{w[n]\} = 0 \text{ and } \phi_{ww}[m] = \sigma_w^2\delta[m]$$

  Thus the power density spectrum is constant

  $$P_{ww}(\omega) = \Phi_{ww}(e^{j\omega}) = \sigma_w^2, \quad |\omega| < \pi$$

  – Note: White and Gaussian are unrelated random process properties, although they often occur together in practice, e.g., *white Gaussian noise*...



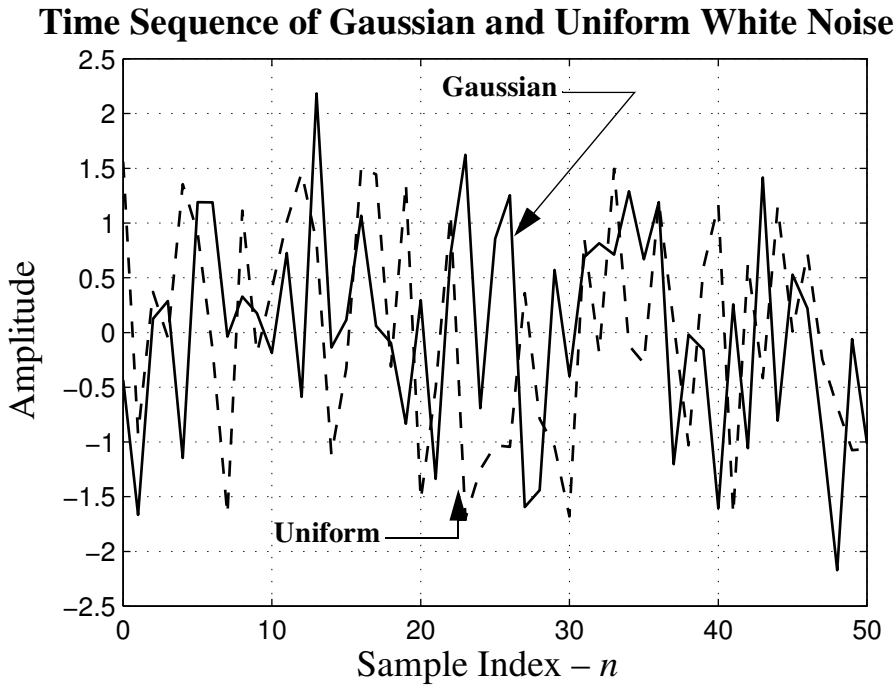
Autocorrelation and power spectrum for white noise

Example 4.14: Exploring White Noise Processes in MATLAB

In this example we will use MATLAB to investigate the correlation and spectral properties of white noise processes. To begin with we generate both uniform and Gaussian white noise sequences, $x_u[n]$ and $x_g[n]$ respectively, each of length $L = 4096$ points.

- The sequence $x_g[n]$ is Gaussian with zero mean and unit variance
- The sequence $x_u[n]$ has uniform amplitude pdf with zero mean and unity variance, i.e., $x_u[n] \sim U(-\sqrt{3}, \sqrt{3})$, why?

```
>> xg = randn(4096,1);
>> xu = sqrt(3)*(2*rand(4096,1)-1);
```



Unit variance uniform and Gaussian white noise

- To estimate the autocorrelation function of these sequences we use the signal processing toolbox function `xcorr()`, which can for example form either the biased or unbiased estimates of $\phi_{xx}[m]$

```
XCORR Cross-correlation function estimates.
   C = XCORR(A,B), where A and B are length M vectors (M>1), returns
   the length 2*M-1 cross-correlation sequence C. If A and B are of
   different length, the shortest one is zero-padded.  C will be a
```

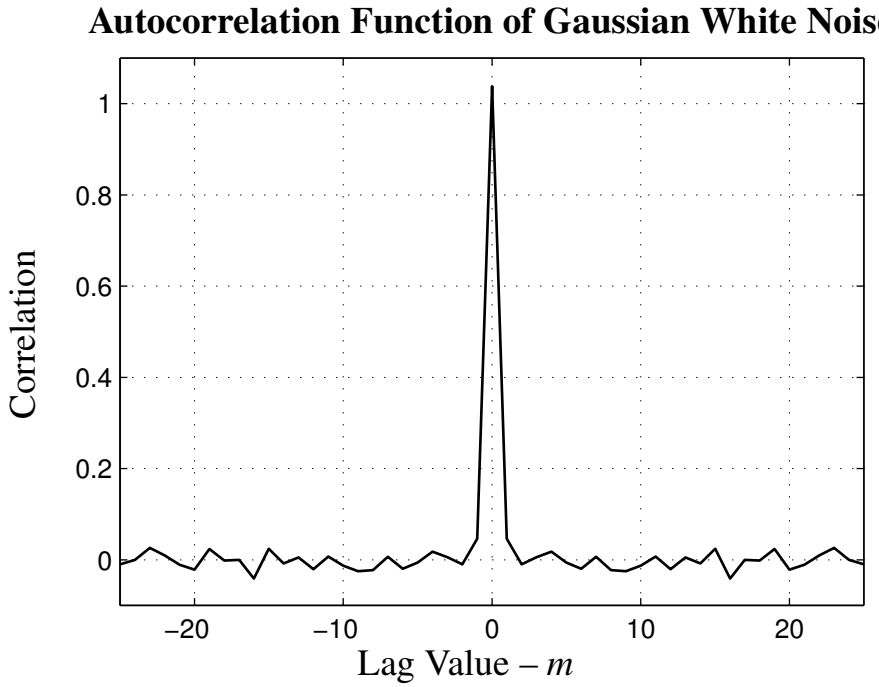
```
row vector if A is a row vector, and a column vector if A is a
column vector.

XCORR(A), when A is a vector, is the auto-correlation sequence.
XCORR(A), when A is an M-by-N matrix, is a large matrix with
2*M-1 rows whose N^2 columns contain the cross-correlation
sequences for all combinations of the columns of A.
The zeroth lag of the output correlation is in the middle of the
sequence, at element or row M.

XCORR(A,B,MAXLAG) computes the cross-correlation over the
range of lags:  -MAXLAG to MAXLAG, i.e., 2*MAXLAG+1 lags.
XCORR(A,MAXLAG) computes the auto-correlation over the
range of lags.   If missing, default is MAXLAG = M-1.

[C,LAGS] = XCORR  returns a vector of lag indices (LAGS).

XCORR(A,'flag'), XCORR(A,B,'flag') or XCORR(A,B,MAXLAG,'flag')
normalizes the correlation according to 'flag':
    biased   - scales the raw cross-correlation by 1/M.
    unbiased - scales the raw correlation by 1/(M-abs(lags))
    coeff    - normalizes the sequence so that the auto-correlations
               at zero lag are identically 1.0.
    none     - no scaling (this is the default).

See also XCOV, CORRCOEF, CONV, COV and XCORR2.
```

- We will use the unbiased estimator for $L = 4096$ point records

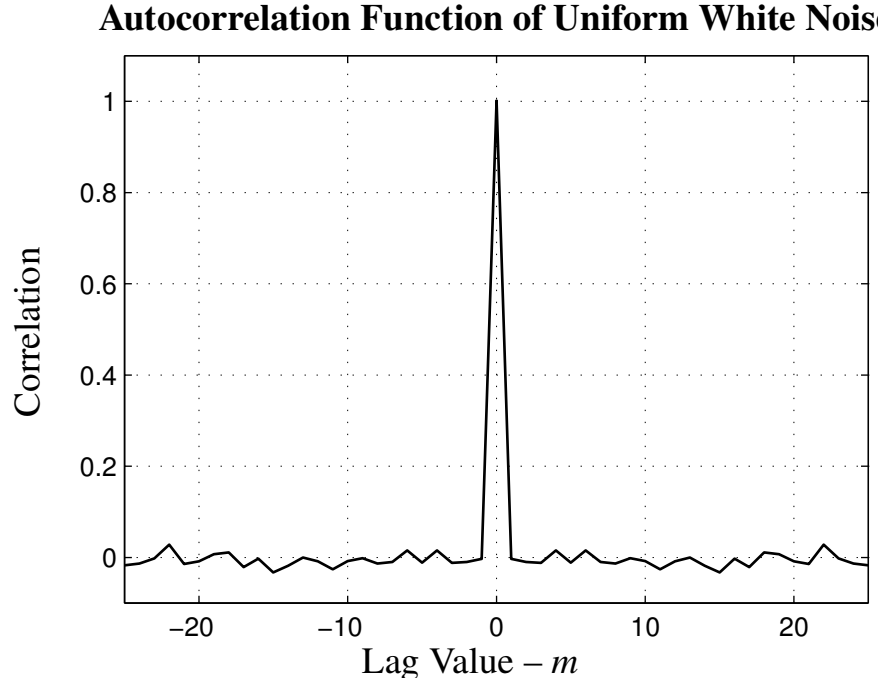
- The theoretical autocorrelation function for both sequences is of course

$$\phi_{xx}^{g}[m] = \phi_{xx}^{u}[m] = \sigma_x^2 \delta[m] = \delta[m]$$

```
>> [phig_xx, lag] = xcorr(xg,25,'unbiased');
>> [phiu_xx, lag] = xcorr(xu,25,'unbiased');
```



Sample autocorrelation of Gaussian white noise



Sample autocorrelation of uniform white noise

- An estimate of the power spectral density (psd) can be obtained by Fourier transforming the autocorrelation function estimate of directly using the signal processing toolbox function `psd()`
- The function `psd()` uses Welch's method of averaging modified periodograms
- A periodogram is basically the discrete Fourier transform (DFT) of a block of signal samples in magnitude-squared form (we will study this more later for now psd is just a tool)

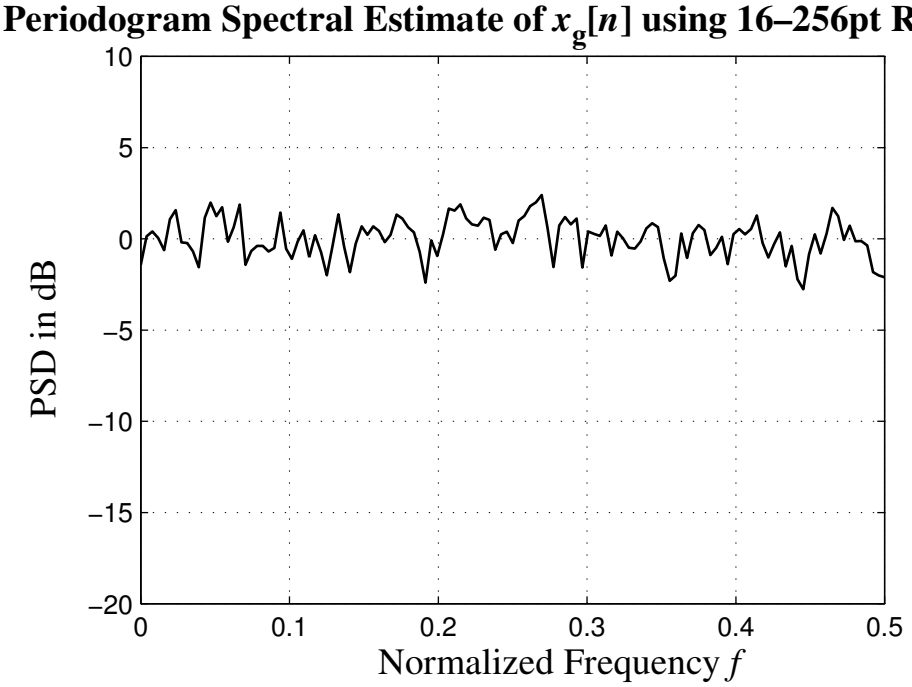
```
PSD Power Spectral Density estimate.
   Pxx = PSD(X,NFFT,Fs,WINDOW) estimates the Power Spectral Density of
   a discrete-time signal vector X using Welch's averaged, modified
   periodogram method.

   X is divided into overlapping sections, each of which is detrended
   (according to the detrending flag, if specified), then windowed by
   the WINDOW parameter, then zero-padded to length NFFT.  The magnitude
   squared of the length NFFT DFTs of the sections are averaged to form
   Pxx.  Pxx is length NFFT/2+1 for NFFT even, (NFFT+1)/2 for NFFT odd,
   or NFFT if the signal X is complex.  If you specify a scalar for
   WINDOW, a Hanning window of that length is used.  Fs is the sampling
   frequency which doesn't affect the spectrum estimate but is used
   for scaling the X-axis of the plots.

   [Pxx,F] = PSD(X,NFFT,Fs,WINDOW,NOVERLAP) returns a vector of frequen-
   cies the same size as Pxx at which the PSD is estimated, and overlaps
   the sections of X by NOVERLAP samples.

   [Pxx, Pxxc, F] = PSD(X,NFFT,Fs,WINDOW,NOVERLAP,P) where P is a scalar
   between 0 and 1, returns the P*100% confidence interval for Pxx.

   PSD(X,...,DFLAG), where DFLAG can be 'linear', 'mean' or 'none',
   specifies a detrending mode for the prewindowed sections of X.
   DFLAG can take the place of any parameter in the parameter list
   (besides X) as long as it is last, e.g. PSD(X,'mean');

   PSD with no output arguments plots the PSD in the current figure window,
   with confidence intervals if you provide the P parameter.

   The default values for the parameters are NFFT = 256 (or LENGTH(X),
```

```
whichever is smaller), NOVERLAP = 0, WINDOW = HANNING(NFFT), Fs = 2,
P = .95, and DFLAG = 'none'.  You can obtain a default parameter by
leaving it off or inserting an empty matrix [], e.g. PSD(X,[],10000).

NOTE: For Welch's method implementation which scales by the sampling
      frequency, 1/Fs, see PWELCH.
```

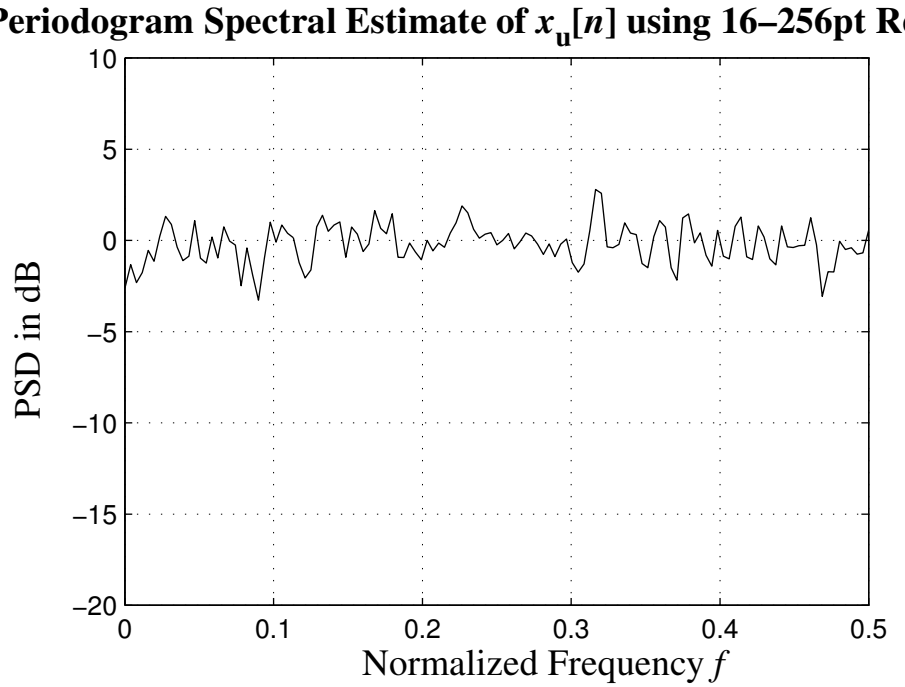
- Here we force `psd` to average sixteen periodograms by setting `NFFT` to 256, e.g., `Px = psd(x,256,1);`

- The theoretical psd is of course

$$P_{xx}(\omega) = \sigma_x^2 = 1, \ \forall \ \omega$$



Power spectrum estimate of Gaussian white noise



Power spectrum estimate of uniform white noise

Linear Systems and White Noise

In the second part of this example we will investigate the correlation, crosscorrelation, and spectral properties of white noise that has been passed through a first-order lowpass filter. We will consider the following system with real a

$$y[n] = ay[n-1] + (1-a)x[n] \;\Rightarrow\; H(z) = \frac{1-a}{1-az^{-1}},\ |z| > |a|$$

- The theoretical output autocorrelation function is

$$\begin{aligned}\phi_{yy}[m] &= \sum_{l=-\infty}^{\infty} \phi_{xx}[m-l]c[l] = \sum_{l=-\infty}^{\infty} \sigma_x^2 \delta[m-l]c[l] \\ &= \sigma_x^2 c[m]\end{aligned}$$

where

$$c[l] = \sum_{k=-\infty}^{\infty} h[k]h[l+k] = h[l] * h[-l]$$

- We can find $c[l]$ using z-transform methods:

$$c[l] = \mathcal{Z}^{-1}\left[H(z)H(z^{-1})\right] \text{ (from conv. thm.)}$$

Now,

$$\begin{aligned} H(z)H(z^{-1}) &= (1-a)^2 \frac{1}{1-az^{-1}} \cdot \frac{1}{1-az} \\ &= \frac{-(1-a)^2}{a} \frac{z^{-1}}{(1-az^{-1})(1-a^{-1}z^{-1})} \\ &= \frac{-(1-a)^2}{a}\left[\frac{A_1}{1-az^{-1}} + \frac{A_2}{1-a^{-1}z^{-1}}\right], \\ & \qquad |a| < |z| < \left|\frac{1}{a}\right| \end{aligned}$$

Solve for A_1 and A_2 and inverse transform

$$\begin{aligned} A_1 &= \left.\frac{z^{-1}}{1-a^{-1}z^{-1}}\right|_{z^{-1}=a^{-1}} = \frac{-a}{1-a^2} \\ A_2 &= \left.\frac{z^{-1}}{1-az^{-1}}\right|_{z^{-1}=a} = \frac{a}{1-a^2} \Rightarrow \end{aligned}$$

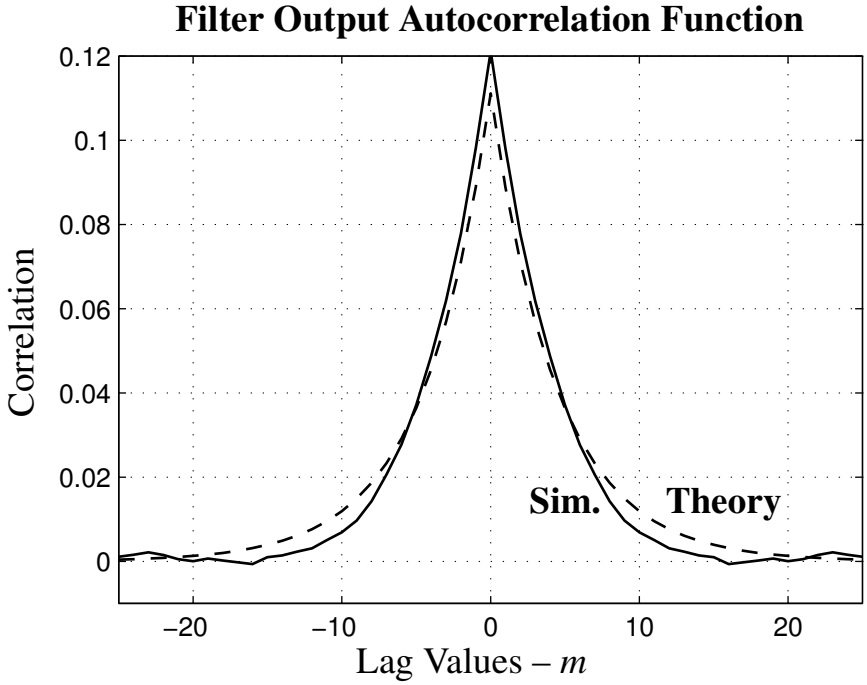
$$\begin{aligned} c[m] &= \frac{(1-a)^2}{1-a^2}\left[a^m u[m] + \left(\frac{1}{a}\right)^m u[-m-1]\right] \\ &= \frac{(1-a)^2}{1-a^2} a^{|m|}, \; -\infty < m < \infty \end{aligned}$$

- Finally,

$$\phi_{yy}[m] = \sigma_x^2 \frac{(1-a)^2}{1-a^2} a^{|m|}$$

- Let $a = 0.8$ and compare the theoretical autocorrelation function with that measured (for just $x_g[n]$), again using `xcorr()`

```
>> yg = filter(1-0.8,[1 -0.8],xg);
>> [phig_yy, lag] = xcorr(yg,25,'unbiased');
>> phig_yythy = .2^2/(1 - .8^2)*0.8.^(abs(lag));
```



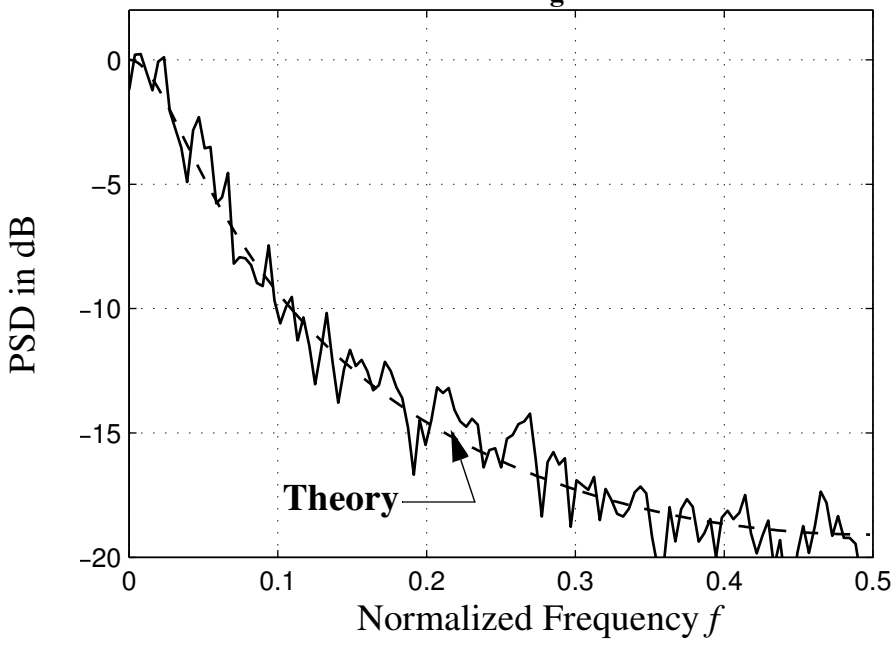
- We can also compare the theoretical power spectrum with the averaged periodogram

- The theoretical psd at the filter output is

$$P_{yy}(\omega) = \Phi_{yy}(e^{j\omega}) = \sigma_x^2 |H(e^{j\omega})|^2$$

```
>> psd(yg,256,1);
>> hold
>> [Py, F] = freqz(0.2,[1 -.8],256,1);
>> plot(F,20*log10(abs(Py)),'r')
```

Periodogram Spectral Estimate of $y_g[n]$ using 16–256pt Records



Output power spectrum estimate of filtered white noise, $P_y(\omega)$

- The cross-correlation between the input and output sequences is defined as

$$\begin{aligned}\phi_{xy}[m] &= E\{x[n+m]y[n]\} \\ &= E\left\{x[n+m]\sum_{k=-\infty}^{\infty} h[k]x[n-k]\right\} \\ &= \sum_{k=-\infty}^{\infty} h[k]\phi_{xx}[m+k]\end{aligned}$$

- The cross power spectrum is obtained by Fourier transforming both sides in the above

$$\Phi_{xy}(e^{j\omega}) = H(e^{j\omega})\Phi_{xx}^*(e^{j\omega}) \overset{x \text{ real}}{=} H(e^{j\omega})\Phi_{xx}(e^{j\omega})$$

- If the input is a zero mean white noise random process, it then follows that

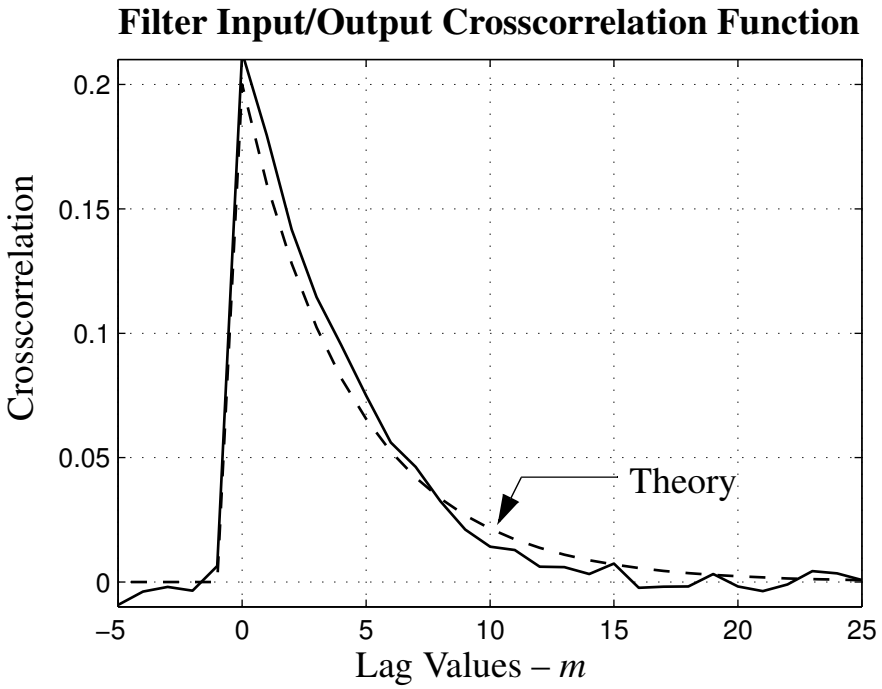
$$\phi_{xy}[m] = \sigma_x^2 h[m]$$

and the cross spectrum is

$$\Phi_{xy}(e^{j\omega}) = \sigma_x^2 H(e^{j\omega})$$

- Again with $a = 0.8$ the theoretical impulse response and estimated cross-correlation estimate are shown below

```
>> [phig_xy, lag] = xcorr(xg,yg,25,'unbiased');
>> phig_xythy = zeros(size(lag));
>> s1 = find(lag >= 0);
>> phig_xythy(s1) = 0.2*0.8.^(lag(s1));
>> plot(lag,phig_xythy,'r')
>> hold
>> plot(lag,phig_xythy,'r')
```

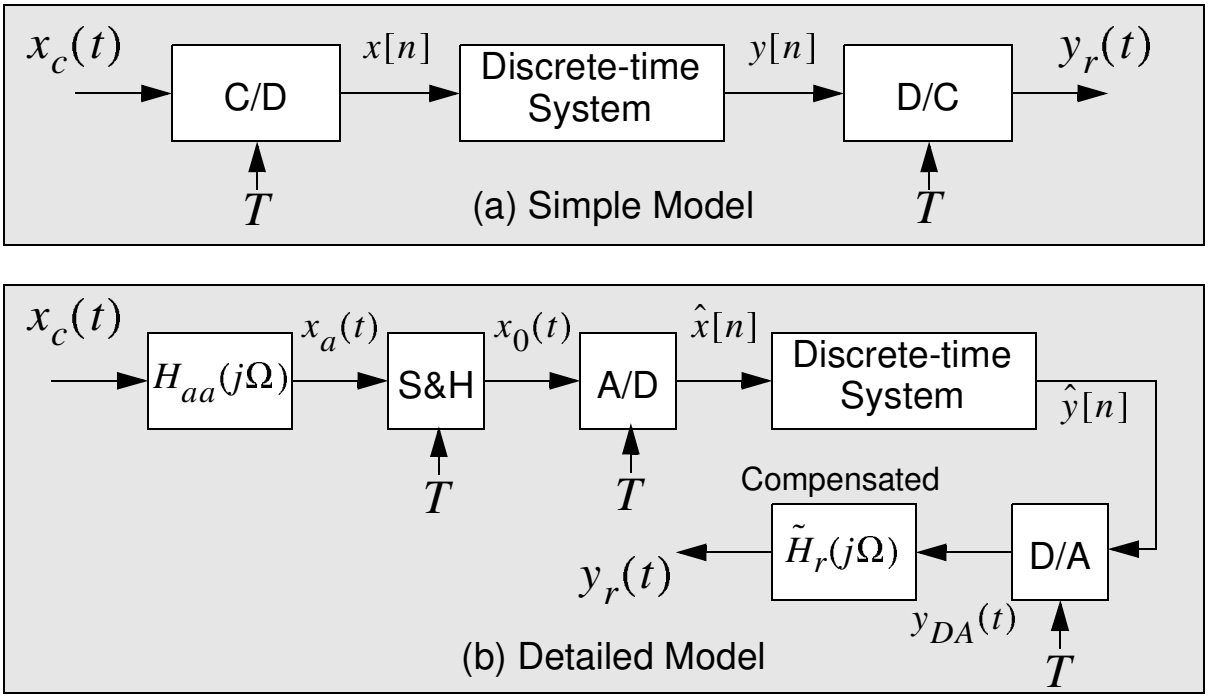


Input/output cross correlation $\phi_{xy}[m]$

4.12 Digital Signal Processing of Analog Signals

In this section practical issues associated with the idealized C/D-$H(z)$-D/C system are investigated. The section begins with the *antialiasing filter* which bandlimits the input signal to satisfy the sampling theorem. We will see that multirate sampling can be used to ease analog design requirements. Next we model the analog-to-digital (A/D) conversion process in more detail than the simple C/D presented earlier. The quantization of the analog input will be modeled using stochastic (random) process theory. Finally, the digital-to-analog (D/A) conversion process is modeled in more detail.

The modeling enhancements provided by this section of the notes is summarized by comparing the differences between subfigures (a) and (b) of the following figure.



DSP of analog signals: (a) Simple Model, (b) Detailed Model

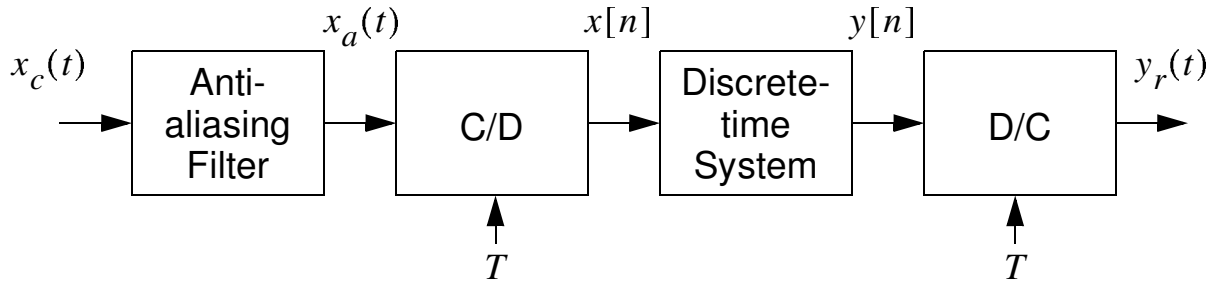
4.12.1 Prefiltering to Avoid Aliasing

A general rule in digital signal processing is to minimize the sampling rate for core processing algorithms whenever possible. The analog signal of interest, $x_c(t)$, may need analog preprocessing before conversion to a discrete-time signal.

1. The signal $x_c(t)$ my need to be bandlimited to contain just a *useful* band of frequencies

2. The signal $x_c(t)$ may be adequately bandlimited already, but high frequency noise is present above the Nyquist frequency and will hence alias into the band of interest

Preprocessing is performed by passing $x_c(t)$ through an antialiasing filter with frequency response $H_{aa}(j\Omega)$.



Use of an antialias prefilter $H_{aa}(j\Omega)$

- An ideal antialiasing filter is of the form

$$H_{aa}(j\Omega) = \begin{cases} 1, & |\Omega| < \Omega_c < \pi/T \\ 0, & \text{otherwise} \end{cases}$$

- Bandlimiting $x_c(t)$ to have bandwidth at or below π/T radians/s ($f_s/2$ Hz) insures for the case of an ideal $H_{aa}(j\Omega)$, that no aliasing can possibly occur

- If this is true then the effective analog frequency response is

$$H_{\text{eff}}(j\Omega) = \begin{cases} H(e^{j\Omega T}), & |\Omega| < \Omega_c < \pi/T \\ 0, & \text{otherwise} \end{cases}$$

  where $H(e^{j\omega})$ is the discrete-time system frequency response

- As a matter of practice no time-limited signal is strictly bandlimited and $H_{aa}(j\Omega)$ cannot be made bandlimited, thus we often settle for a minimum attenuation of aliased signal and/or noise components with respect to the desired signal

- As a simple approximation, if we can make $H_{aa}(j\Omega)$ small for $|\Omega| > \pi/T$ then

$$H_{\text{eff}}(j\Omega) \approx H_{aa}(j\Omega)H(e^{j\Omega T})$$

  and the spectrum of $y_r(t)$ can be modeled as

$$Y_r(j\Omega) \simeq X_c(j\Omega)H_{\text{eff}}(j\Omega) + \underbrace{Y_{\text{alias-sig}}(j\Omega) + Y_{\text{alias-noise}}(j\Omega)}_{\text{result of imperfect } H_{aa}}$$

Using Multi-Rate Sampling to Simplify the Analog Design

When the desired signal bandwidth, Ω_N, is close to $\pi/T = \Omega_s/2$ (f_N is close to $f_s/2$) the design requirements for the antialiasing filter become more challenging

- A sharp-cutoff (high-order) design is needed

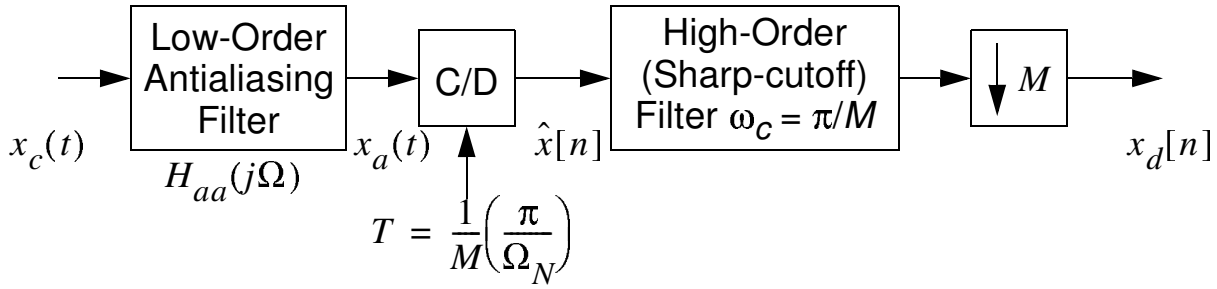
- The filter must begin to roll-off before Ω_N, which introduces amplitude distortion

  - Once in the discrete-time domain this distortion can be partially compensated for with a digital equalization filter

- Phase distortion near the band edge also occurs in sharp-cutoff analog filters

- Realizing a sharp-cutoff analog filter can be both difficult and expensive

- A way to simplify the analog design requirements is via multi-rate sampling



Using oversampling to simplify the design of $H_{aa}(j\,\Omega)$

- Assuming the usable lowpass bandwidth is again Ω_N we choose an oversampling factor of M, so the sampling rate is now $\Omega_s = 2M\Omega_N$

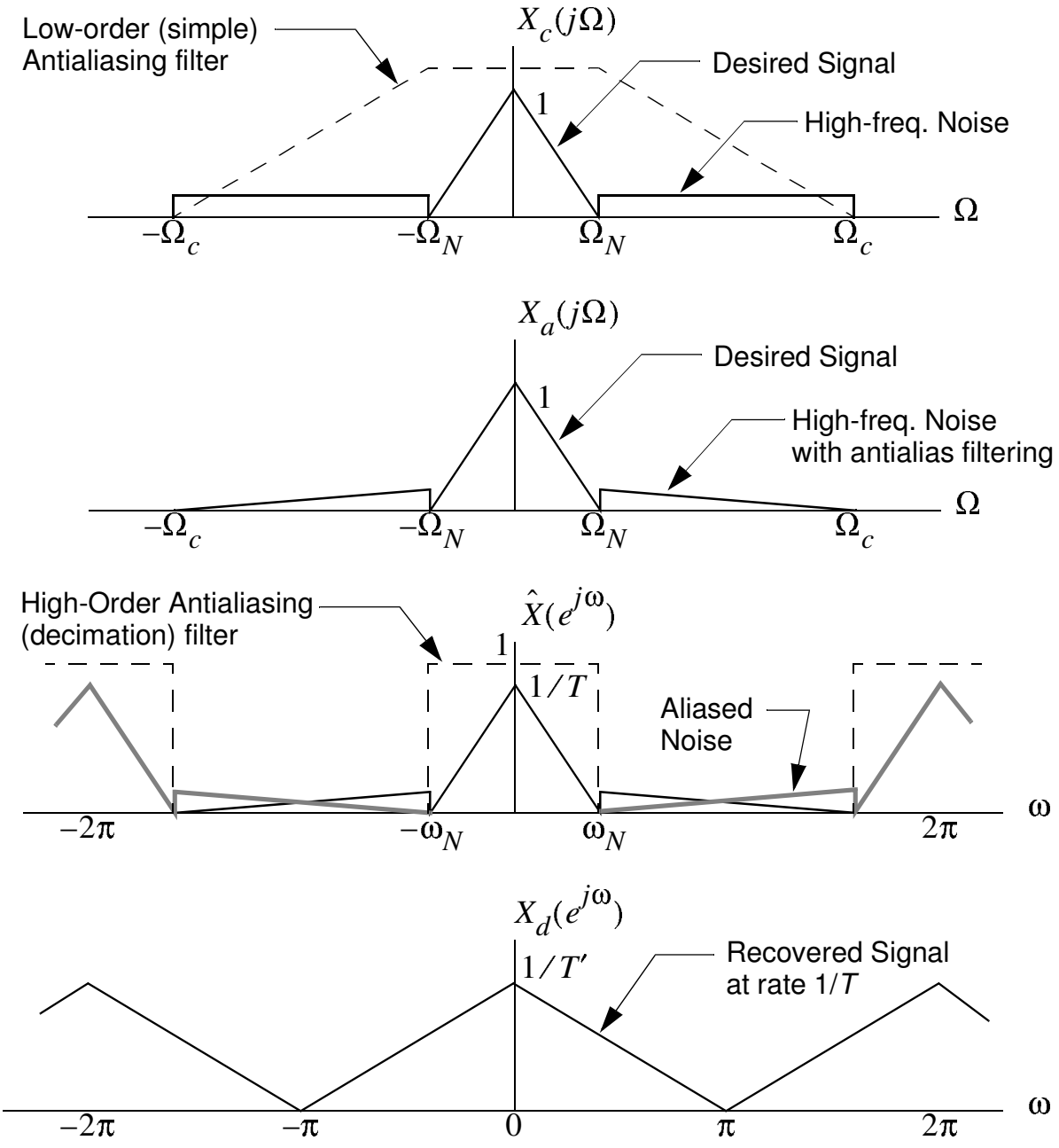
- The simple antialiasing filter can have a very gradual cutoff, so long as it has significant attenuation at $M\Omega_N$

- In the discrete-time domain we now implement a sharp-cutoff antialiasing filter with digital cutoff frequency of $\omega_c = \pi/M$ and then decimate by M

  - The sharp-cutoff digital filter can have linear phase and thus provide near perfect bandlimiting behavior

  - An obvious drawback is the need for additional processing as a trade for the more difficult/expensive analog filter design

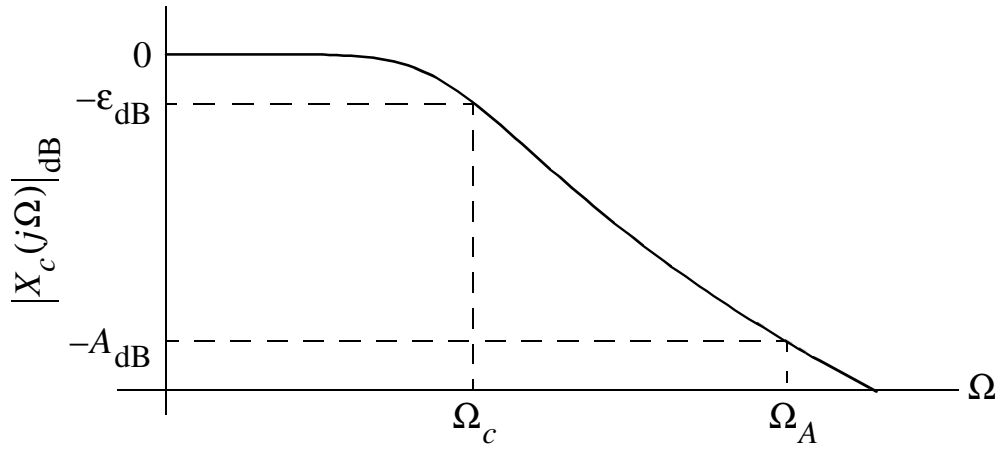
- A pictorial example of how this works is shown in the following figure



A spectral view of how oversampling works

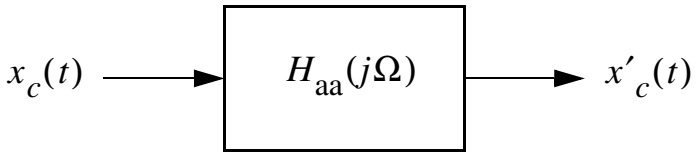
Example 4.15: **Anti-Aliasing Filter Design**

Consider a continuous-time signal $x_c(t)$ with Fourier transform $X_c(j\Omega)$ (note the spectrum is scaled to 0 dB passband gain).



The Analog input signal amplitude spectrum

We wish to pass $x_c(t)$ through a lowpass filter with frequency response $H_{aa}(j\Omega)$ to provide additional bandlimiting prior to sampling.

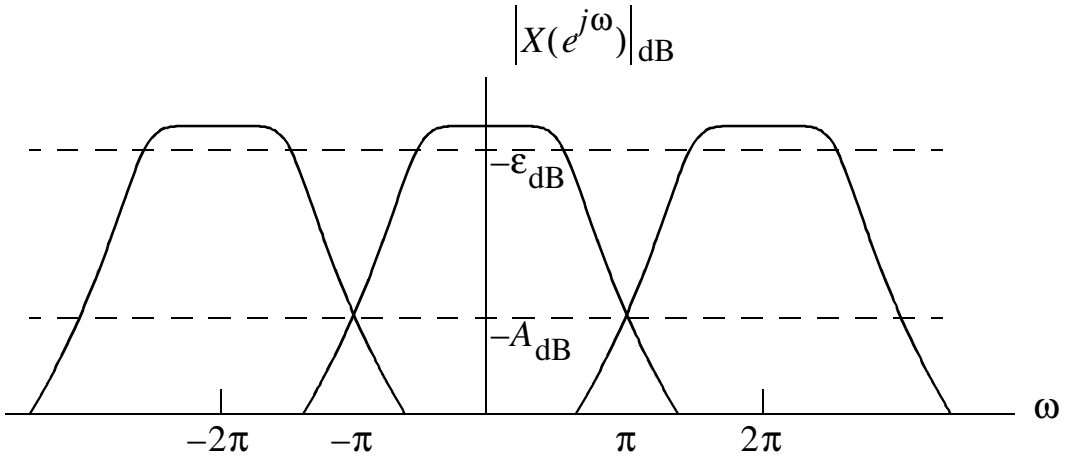


Filtering of the input signal $x_c(t)$ with the anti-aliasing filter $H_{aa}(j\Omega)$

- Assume that the ϵ_{dB} bandwidth of $x_c(t)$ is Ω_c (f_c Hz) rad/sec and at Ω_A the spectrum is down A_{dB}

- Further assume that the signal spectrum is monotonically decreasing with frequency, i.e. $|X_c(j\Omega_2)| < |X_c(j\Omega_1)|$ for $\Omega_2 \geq \Omega_1$

- If $x_c(t)$ is sampled at frequency Ω_s (f_s Hz) to produce the sequence $x[n]$, then the Fourier transform of $x[n]$ is a superposition of scaled replicas of $X_c(j\Omega)$ at multiples of the normalized sampling rate (i.e., 2π)



Discrete-time domain spectra following sampling without the antialiasing filter

Antialiasing Filter Requirements:

- A reasonable design specification is to require that the aliased spectral content from 0 to π be $A_{a-\text{dB}}$ down from the 0 dB spectral reference level of the input

- In particular note that if $f_A = f_s/2$ then the minimum aliased spectral content is down only A_{dB}

- An additional constraint is that the spectral content of the input signal between 0 and f_c Hz not be modified by the antialiasing filter

- To increase $A_{a-\text{dB}}$ one may increase f_s or choose to prefilter $x_c(t)$ with an antialising filter such that the spectrum of $x_c'(t)$ meets the desired aliased spectral content requirement

- Note: A combination of the above may also be considered

The Filter Design Problem

The filter design problem is simply to design a lowpass filter with cutoff frequency at f_c and filter attentuation at $f_s/2$ of

$$\underbrace{|X_c(j\,\Omega_s/2)|_{\text{dB}}}_{-A_{\text{dB}}} + |H_{aa}(j\,\Omega_s/2)|_{\text{dB}} \leq -A_{a-\text{dB}}$$

or

$$-|H_{aa}(j\,\Omega_s/2)|_{\text{dB}} \geq A_{a-\text{dB}} - A_{\text{dB}}$$

The lowpass prototype may a Butterworth, Chebyshev (type I or II), or perhaps an elliptic. For more information on these prototype transfer functions see Chapter 7 of the text.

- A a specific example suppose the analog filter is a Butterworth with f_c the 3 dB cutoff frequency

- Note: The input signal spectrum near this cutoff frequency will receive additional attenuation which may be compensated for in the discrete-time domain, see text problem 4.56

- An Nth-order Butterworth lowpass filter has frequency response of the form

$$|H_b(j\,\Omega)|^2 = \frac{1}{1 + \left(\frac{\Omega}{\Omega_c}\right)^{2N}}$$

- We must find the minimum N such that

$$10\log_{10}\left[1+\left(\frac{\Omega_s}{2\Omega_c}\right)^{2N}\right] \geq A_{a-\text{dB}} - A_{\text{dB}}$$

- Rearranging gives

$$\left(\frac{\Omega_s}{2\Omega_c}\right)^{2N} = 10^{\left(\frac{A_{a-\text{dB}}-A_{\text{dB}}}{10}\right)} - 1$$

or

$$N \geq \frac{\log_{10}\left[10^{\left(\frac{A_{a-\text{dB}}-A_{\text{dB}}}{10}\right)} - 1\right]}{2\log_{10}[\Omega_s/2\Omega_c]}$$

- Note that N must be an integer!

An Example Case:

Suppose that the analog input signal has a Butterworth spectrum of the form

$$|X_c(j\,\Omega)| = \frac{1}{\sqrt{1+(\Omega/(2\pi\times 350,000))^6}}$$

and the sampling rate is chosen to be $f_s = 1$ MHz. The aliased spectral components are to be attenuated by at least 60 dB. Find the filter cutoff frequency and minimum filter order, N, to meet the design goals.

- To begin with we observe that the signal spectrum has a cutoff frequency of 350 kHz, thus $f_c = 350\times 10^3$

- At $f_s/2 = 500$ kHz the input spectrum is attenuated by only 9.77 dB

- Assuming a Butterworth antialiasing filter

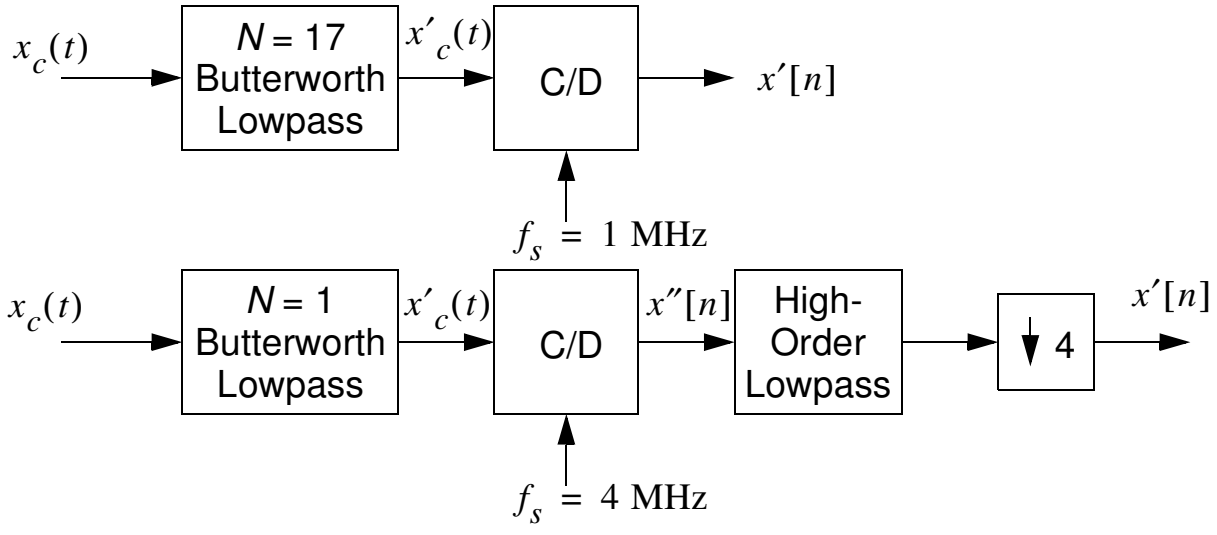
$$N \geq \frac{\log_{10}\left[10^{[(60-9.77)/10]}-1\right]}{2\log_{10}[500/350]} = 16.2$$

- $N = 17$ is unreasonable, so to lessen the requirements on the antialiasing filter we either use a different prototype, such as an elliptic since it has a very narrow transition band, or up the sampling rate

- Try increasing f_s to 4 MHz, once the signal is in the discrete-time domain we can then build a high order digital filter to band limit the signal and then decimate back down to the desired 1 MHz rate

$$N \geq \frac{\log_{10}\left[10^{[(60-45.41)/10]}-1\right]}{2\log_{10}[2000/350]} = 0.953$$

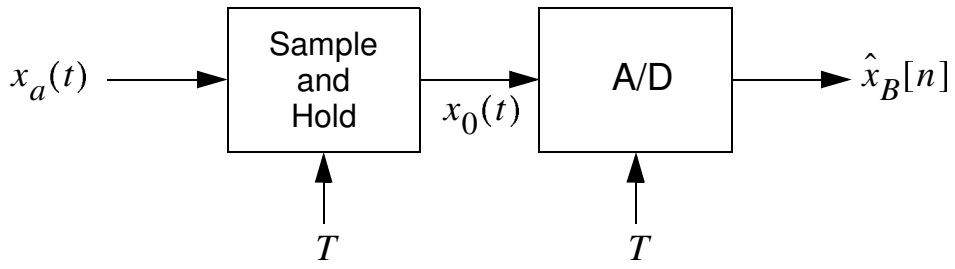
- $N = 1$ is very reasonable, perhaps the sampling rate can be reduced to something in between 1 MHz and 4 MHz



C-to-D conversions systems, with anti-aliasing filters

4.12.2 Analog-to-Digital (A/D) Conversion

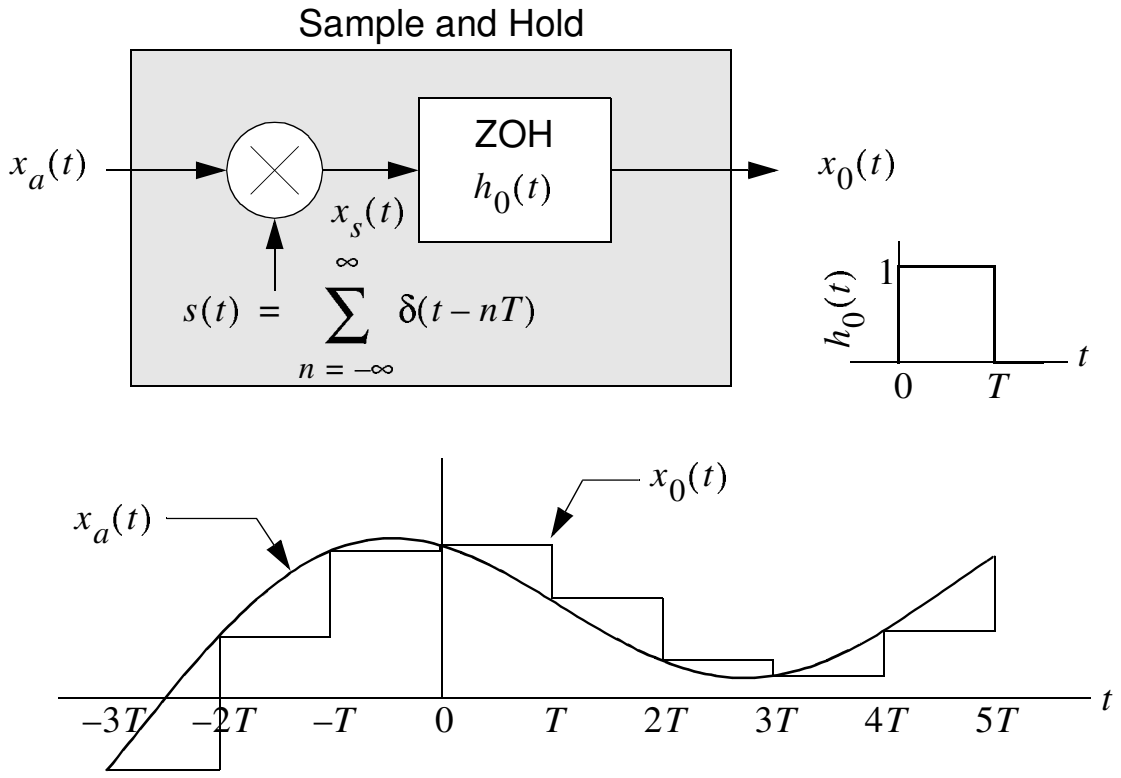
In the model of the C/D converter, developed previously, we have a means for converting a continuous-time signal into a discrete-time signal with each sample having infinite precision. In reality the C/D can be better approximated as a sample-and-hold (S&H) followed by an analog-to-digital (A/D or ADC) converter. The S&H holds each analog voltage or current sample long enough for the A/D conversion cycle to convert the input into a binary code that most closely represents the input amplitude.



Analog-to-digital converter modeling

- The S&H can be modeled as an ideal sampler followed by a zero-order hold (ZOH) filter $h_0(t)$ of the form

$$h_0(t) = \begin{cases} 1, & 0 < t < T \\ 0, & \text{otherwise} \end{cases}$$



- Mathematically we can write the ZOH output as

$$x_0(t) = h_0(t) * \sum_{n=-\infty}^{\infty} x_a(nT)\delta(t - nT)$$

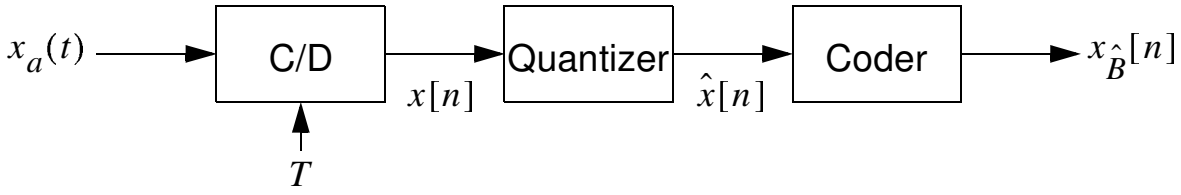
- The filtering action of $h_0(t)$, although important for operation of the A/D, does not alter the signal samples $\hat{x}_B[n]$

- We have assumed that the S&H output is constant until the next sample is taken

  - More advanced modeling of S&H issues can be found elsewhere

- As described here the S&H allows ideal sampling to occur

- A mathematical representation of an analog-to-digital (A/D) converter is thus a cascade of three stages: (1) ideal C/D converter; (2) *quantizer*; (3) *coder*



Refined Model of an A/D

- The combination of the quantizer and coder represent the A/D converter

- The quantizer is a nonlinear system which transforms the ideal real number input sample $x[n]$ into a number from a finite set

- Mathematically the operator notation is

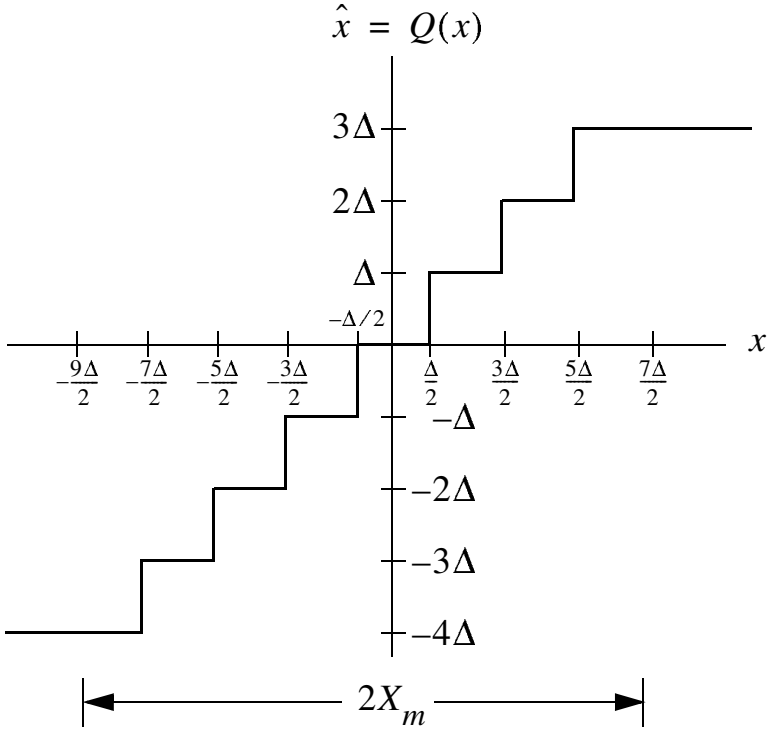
$$\hat{x}[n] = Q(x[n])$$

  where $\hat{x}[n]$ is used to represent a quantized signal sample

  - The quantization levels may be uniformly spaced or nonuniformly spaced

- In the following figure a uniform quantizer is shown which accepts a bipolar input and has eight quantization levels

- Note that in this figure the number of quantization levels is even, thus it is not possible to have a level at zero and have an equal number of positive and negative levels



Two's Complement Code	Offset Binary Code
011	111
010	110
001	101
000	100
111	011
110	010
101	001
100	000

A uniform quantizer that uses *rounding*

- The coder block in the mathematical model maps each quantization level to a code word
- If binary symbols are used, then 2^{B+1} levels can be coded using $(B+1)$-bit binary code words
- The quantizer step size, Δ, is given by

$$\Delta = \frac{2X_m}{2^{B+1}} = \frac{X_m}{2^B}$$

  where X_m is the full-scale level of the A/D
- Two popular binary coding schemes are *offset binary* and *two's complement*
- The two's complement format is the most convenient representation of signed numbers

- The leftmost bit or most-significant-bit (MSB) is also termed the sign bit

- The right most bit is the least significant bit or LSB

- For a binary $(B+1)$-bit two's complement number of the form

$$b_{0\diamond}b_1b_2\ldots b_B$$

we can write the quantizer output as

$$\begin{aligned}\hat{x}[n] &= X_m\hat{x}_B[n] \\ &= X_m\left[-b_0+\sum_{i=1}^{B}b_i2^{-i}\right]\end{aligned}$$

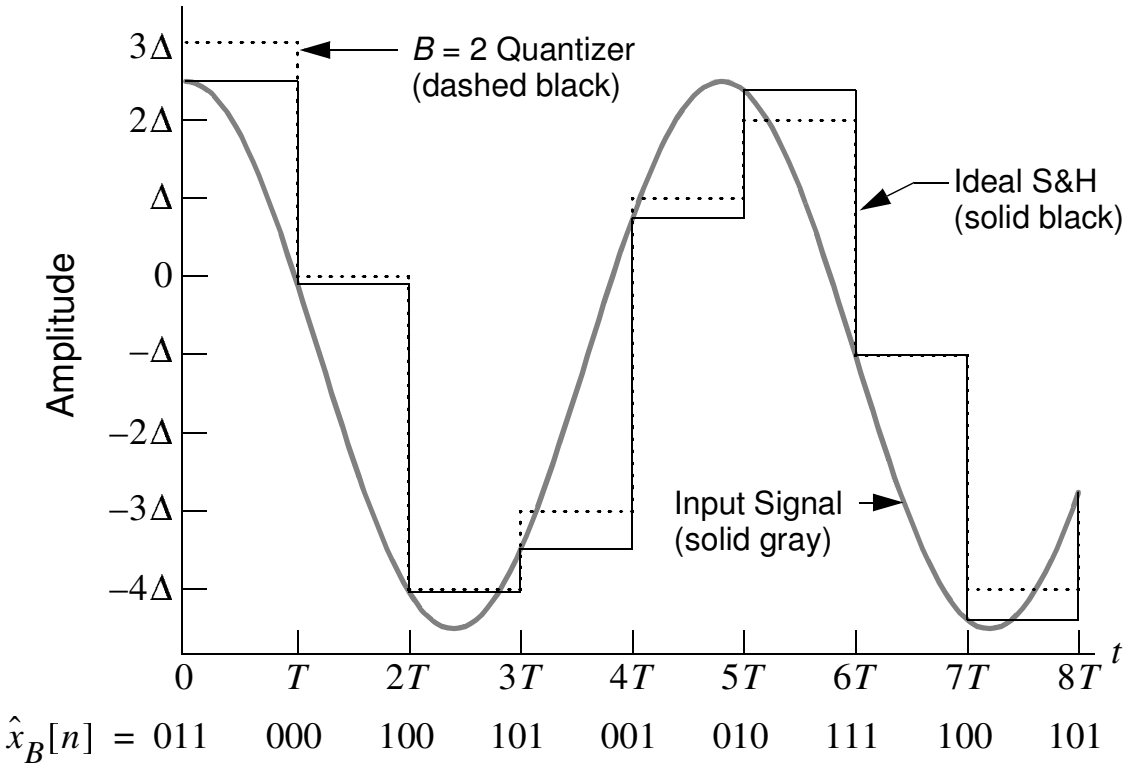
Note that $\hat{x}_B[n]$ is the fractional part of $\hat{x}[n]$ i.e., $-1 \leq \hat{x}_B[n] < 1$

- In a two's complement representation a positive fraction is in straight binary form with a zero in the sign bit (MSB). To obtain a negative fraction subtract the positive number from 2.0 or just complement the positive fraction and add one LSB e.g.,

$$\begin{aligned}\frac{3}{8} &= 0011 \\ -\frac{3}{8} &= 1100+0001=1101\end{aligned}$$

Two's complement binary symbols and $\hat{x}_B$ for $B=2$

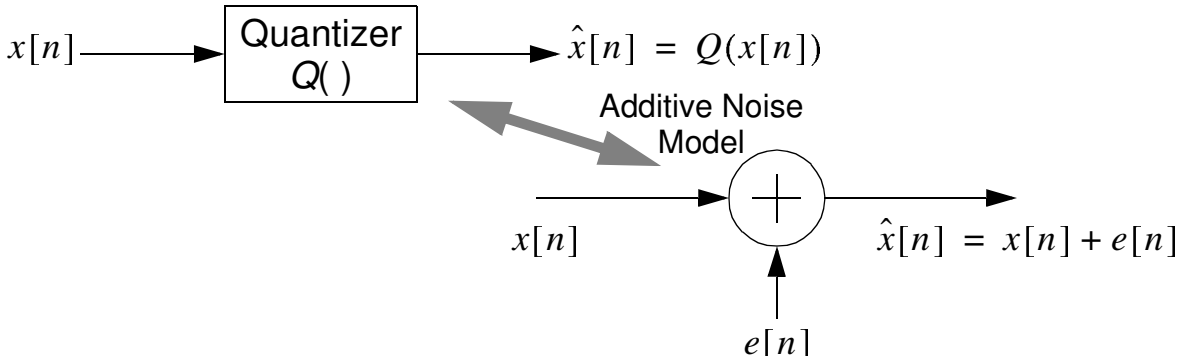
Binary Symbol	Numeric Value, $\hat{x}_B$
$0_\diamond 11$	$3/4$
$0_\diamond 10$	$1/2$
$0_\diamond 01$	$1/4$
$0_\diamond 00$	0
$1_\diamond 11$	$-1/4$
$1_\diamond 10$	$-1/2$
$1_\diamond 01$	$-3/4$
$1_\diamond 00$	-1



Sampling and quantizing with 3-bits

4.12.3 Analysis of Quantization Errors

- In the analysis of quantization errors $x[n]$ is the true sequence value and $\hat{x}[n]$ is the quantized value



Quantizer and additive noise model

- Define the quantization error sequence as

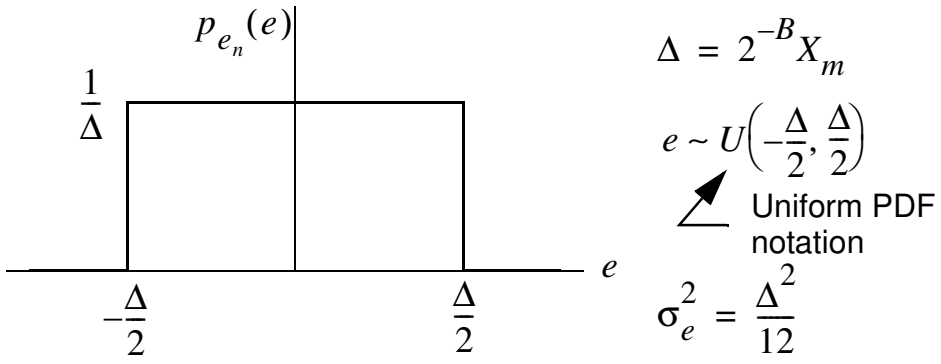
$$e[n] = \hat{x}[n] - x[n]$$

  where if we assume rounding in the quantizer $-\Delta/2 < e[n] \leq \Delta/2$ provided $(-X_m - \Delta/2) < x[n] \leq (X_m - \Delta/2)$

- A useful quantizer interpretation is to view $\hat{x}[n]$ as $x[n]$ plus an additive noise sequence $e[n]$

- Simplistic, yet useful modeling assumptions are the following:
  - Let $e[n]$ be a stationary random process
  - Assume $e[n]$ and $x[n]$ are uncorrelated (this is difficult to justify)
  - Assume $e[n]$ is a white noise sequence
  - Assume the PDF of $e[n]$ is uniform over the error signal range

- Assuming rounding and the modeling assumptions discussed above $e[n]$ has the following statistical description:



Quantizer and additive noise model

Example 4.16: **Quantization of a Sinusoid**

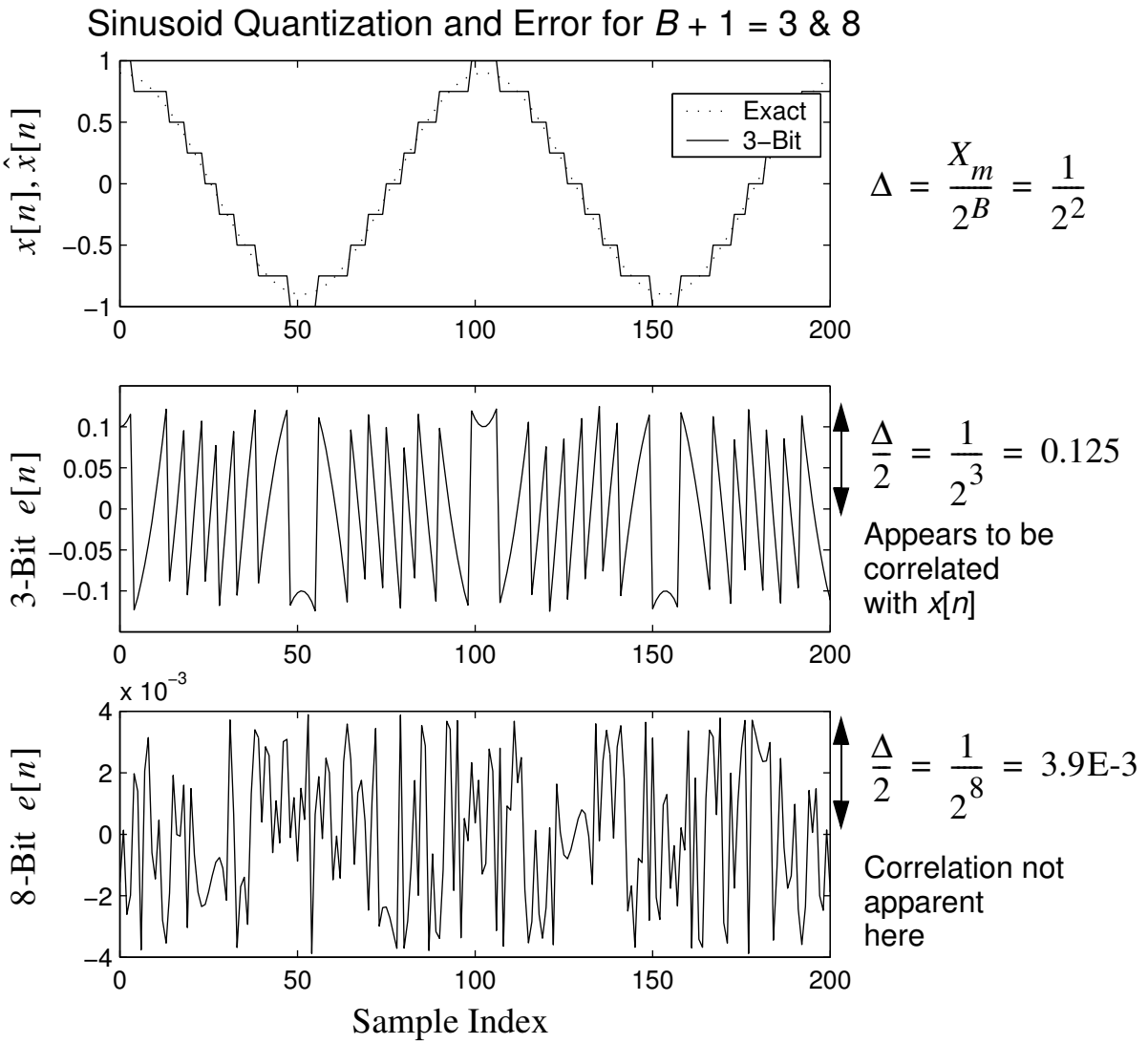
- Consider the uniform quantization of a sinusoid of the form

$$x[n] = 0.9\cos[2\pi(5/512)n], \;\; f_o = 5 \text{ Hz and } f_s = 512 \text{ Hz}$$

- We quantize this sinusoid in MATLAB using approximately $B = 3$ and 8 bits as follows

```
» n = 0:511;
» x = 0.9*cos(2*pi*5/512*n);
» x3 = round(2^3/2*x)/(2^3/2);
» e3 = x3 - x;
» x8 = round(2^8/2*x)/(2^8/2);
» e8 = x8 - x;
```

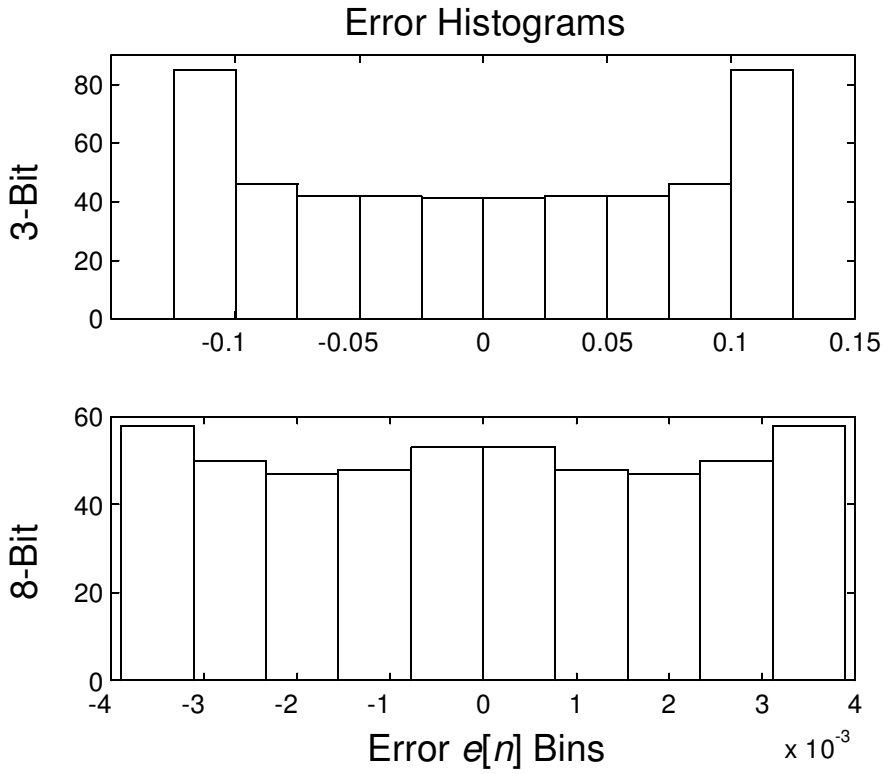
- Note: the rounding operation in MATLAB actually creates $2^3 + 1$ and $2^8 + 1$ levels respectively



A quantized sinusoid (top), error $e[n]$ for 3-bits (middle), error $e[n]$ for 8-bits

- From the above figure we see that the quantization error pattern is not apparent in the 8-bit error as it is in the 3-bit error
- The error values are bounded as expected by $\pm\Delta/2 = 1/2^B$
- The experimental distribution of the quantization errors can be computed using a histogram, .e.g,

```
» hist(x8,10); % Auto plot histogram with 10 bins
```



Histogram of error statistics

- The histogram appears more uniform for 8-bits than for 3-bits
- Looking at the autocorrelation sequence or power spectrum of $e[n]$ are other statistical measures worth considering

---

- A useful performance measure is the signal-to-quantization noise ratio

$$\begin{aligned} \text{SNR}_{dB} &= 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_e^2}\right) \\ &= 6.02B + 10.8 - 20\log_{10}\left(\frac{X_m}{\sigma_x}\right) \end{aligned}$$

- Since X_m is typically fixed we see that reducing σ_x by two decreases the SNR by about 6 dB
- If $x[n]$ should exceed X_m then severe distortion results i.e. $|e[n]| > \Delta/2$. The degree of distortion depends on just how large $|e[n]|$ is and how often an exceedance occurs.
- In general the signal amplitude must "match" as best as possible to the full-scale amplitude

---

Example 4.17: Normally Distributed Signal

Suppose $x[n]$ has a normal distribution i.e. $x[n] \sim N[0, \sigma_x]$.

- To insure that $x[n]$ "rarely" exceeds $\pm X_m$ we might set $X_m = 3\sigma_x$, then
$$P[|x[n]| > 3\sigma_x] = 2.70 \times 10^{-3}$$
or $|x[n]|$ exceeds X_m only 0.27% of the time
- Under this condition
$$\begin{aligned} \text{SNR}_{dB} &\approx 6B + 10.8 - 20\log_{10}\left(\frac{3\sigma_x}{\sigma_x}\right) \\ &\approx 6B + 1.26 \text{ dB} \end{aligned}$$
- For 16 bit digital audio $B = 16 - 1 = 15$ and with $X_m = 3\sigma_x$ it follows that
$$\text{SNR}_{dB} \approx 91.3 \text{ dB}$$

4.12.4 D/A Conversion

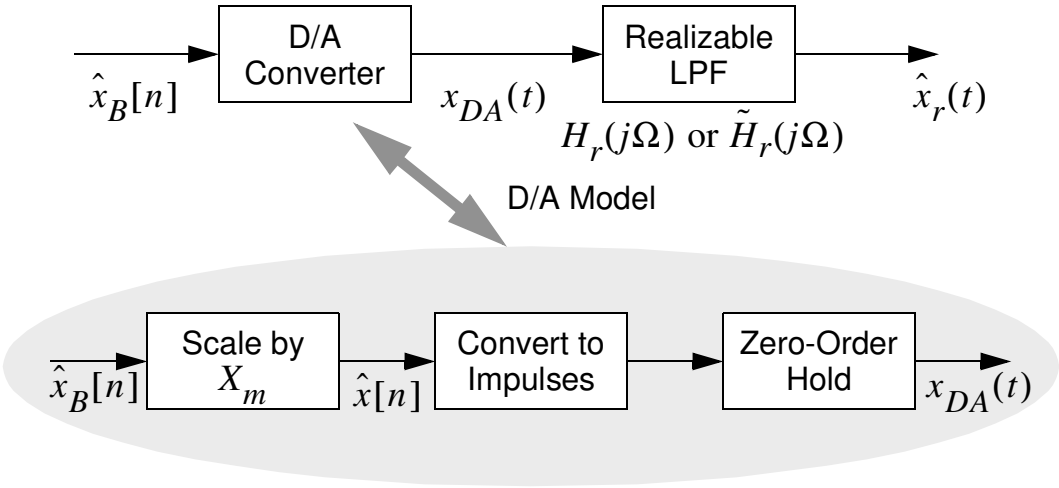
For a bandlimited input signal the ideal D/C produces an output spectrum of

$$X_r(j\Omega) = X(e^{j\Omega T})H_r(j\Omega)$$

where $H_r(j\Omega)$ is the frequency response of the ideal lowpass reconstruction filter. From our earlier discussion we know that in the time-domain this becomes the sinc function interpolation formula

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

A more complete model of the D/C is an analog-to-digital (D/A or DAC) converter followed by a *realizable* reconstruction filter



A/D converter model with reconstruction filter

- Initially we will consider just the D/A model

- The ZOH converts the impulse train sampled signal into a *stair-step* approximation as follows

$$x_{DA}(t) = \sum_{n=-\infty}^{\infty} X_m \hat{x}_B[n] h_0(t - nT)$$
$$= \sum_{n=-\infty}^{\infty} \hat{x}[n] h_0(t - nT)$$

  where the ZOH impulse response, $h_0(t)$, is a rectangular pulse of duration T seconds

- A quantized signal sample is held fixed at the D/A output for T seconds

- If we consider $\hat{x}[n]$ as resulting from an additive noise model for quantization noise, we can write

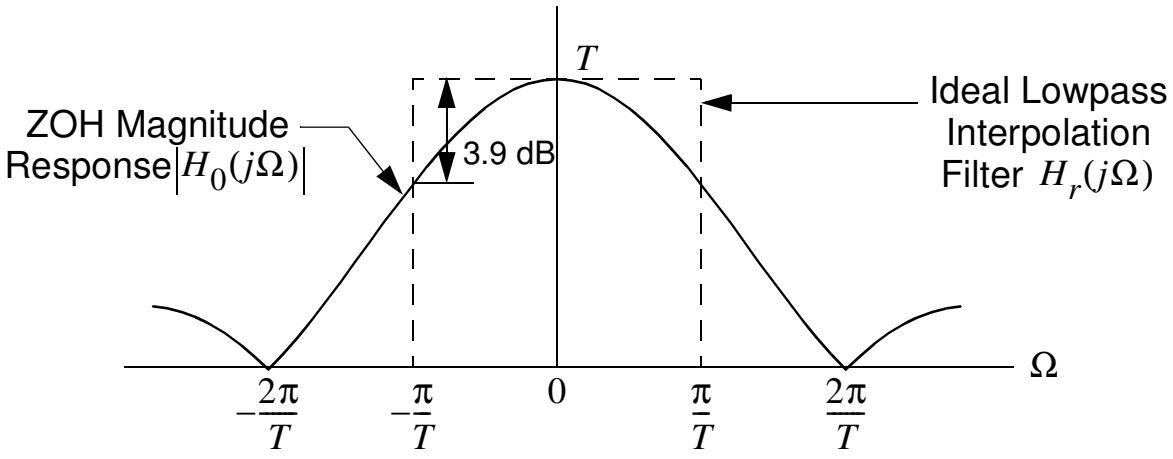
$$x_{DA}(t) = \underbrace{\sum_{n=-\infty}^{\infty} x[n] h_0(t - nT)}_{x_0(t)} + \underbrace{\sum_{n=-\infty}^{\infty} e[n] h_0(t - nT)}_{e_0(t)}$$

- Note that the signal term $x_0(t)$ is related to the input $x[n] = x_a(nT)$, and $e_0(t)$ is in part related to the quantization noise generated by the input A/D

- The term $e_0(t)$ may also take contributions from other quantization elements in the system, for example if the D/A has a shorter word-length than the internal processing achitecture, additional quantization noise is created at the D/A interface (more on this in text Chapter 6)

- In the frequency domain we can write for the signal component that

$$X_0(j\Omega) = X(e^{j\Omega T})H_0(j\Omega)$$



ZOH frequency response compared with ideal $H_r(j\Omega)$

- The frequency domain shaping imposed by $H_0(j\Omega)$ will in general interfere with $X(e^{j\Omega T})$ inside the $|\Omega| < \pi/T$ frequency band

- The final filtering stage, $H_r(j\Omega)$, my compensate for $H_0(j\Omega)$ by choosing

$$\tilde{H}_r(j\Omega) = \frac{H_r(j\Omega)}{H_0(j\Omega)}$$

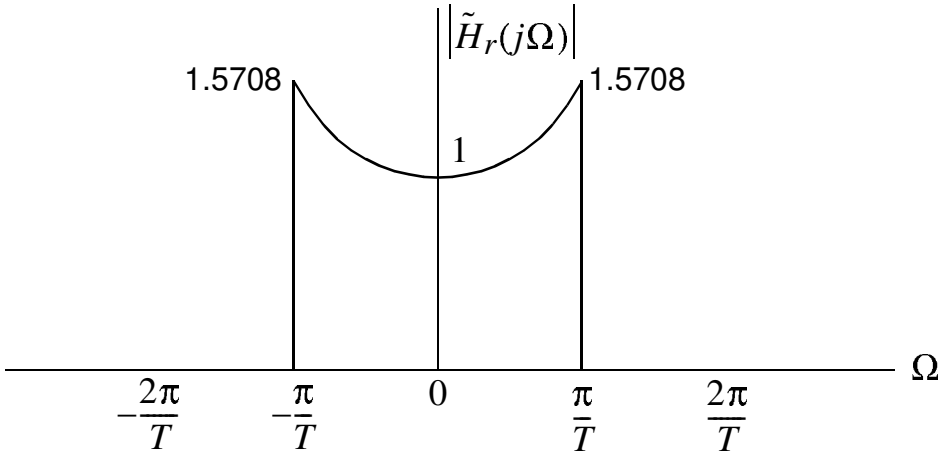
- The ZOH has frequency response given by

$$H_0(j\Omega) = \mathcal{F}\left\{\prod\left(\frac{t}{T}\right)\right\} = \frac{2\sin(\Omega T/2)}{\Omega}e^{-j\Omega T/2}$$

- An ideal compensated reconstruction filter is

$$\tilde{H}_r(j\Omega) = \begin{cases} \frac{\Omega T/2}{\sin(\Omega T/2)}e^{j\Omega T/2}, & |\Omega| < \pi/T \\ 0, & |\Omega| \geq \pi/T \end{cases}$$

- A plot of the magnitude of $\tilde{H}_r(j\Omega)$ is shown below



Ideal compensated reconstruction filter

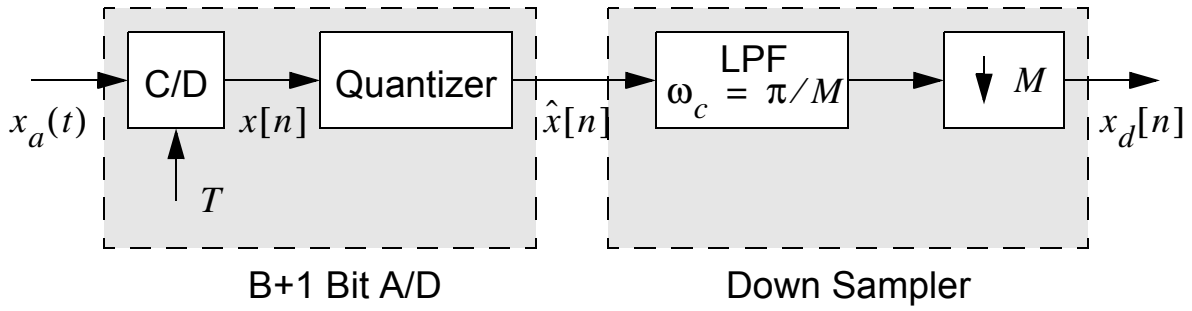
- As a matter of practice the time delay represented in the term $e^{j\Omega T/2}$ would not be realized

4.13 Oversampling and Noise Shaping in A/D and D/A Conversion

We have already seen that by using oversampling the sharp cutoff antialising filter can be moved from the analog front-end to the interior of the DSP design, where it becomes a digital filter. When oversampling is employed it turns out that the number of quantization levels can also be reduced. In particular if quantization *noise shaping* is employed a further reduction in the number of A/D bits can be achieved. The same techniques for reducing quantization bits in the overall C/D operation can be employed in the D/C.

4.13.1 Oversampled A/D Conversion with Direct Quantization

Before studying a complete oversampling and noise shaping based A/D, we will first consider a basic C/D system that employs straight oversampling followed by decimation. The mathematical framework requires the use of both continuous and discrete-time random processes.



Oversampling $B+1$-bit A/D conversion with a downsampler

- We assume that the input $x_a(t)$ is a zero-mean WSS random process with continuous-time autocorrelation function

$$\phi_{x_a x_a}(\tau) = \mathcal{E}\{x_a(t+\tau)x_a(t)\}$$

and continuous-time power spectral density

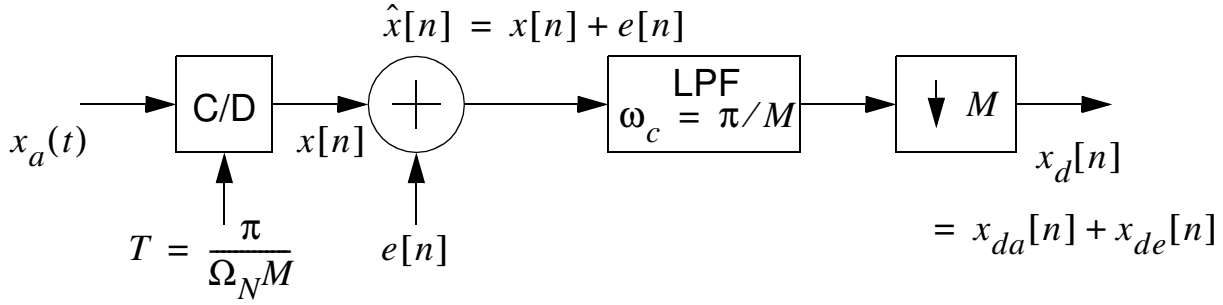
$$\Phi_{x_a x_a}(j\Omega) = \mathcal{F}\{\phi_{x_a x_a}(\tau)\}$$

- We will also assume that $x_a(t)$ is bandlimited, i.e.,

$$\Phi_{x_a x_a}(j\Omega) = 0, \quad |\Omega| \geq \Omega_N$$

and that $\pi/T = M\Omega_N$, where M is the *oversampling ratio*

- To analyze the quantization noise performance resulting from oversampling we replace the quantizer with a linear noise model as shown below



- As a result of superposition, the decimator output is composed of a signal related term $x_{da}[n]$, and noise related term $x_{de}[n]$

- We would like to find the ratio of output signal power to output quantization noise power, i.e,

$$\left(\text{SNR}_Q\right)_d = \frac{\mathcal{E}\{x_{da}^2[n]\}}{\mathcal{E}\{x_{de}^2[n]\}} = \frac{P_{da}}{P_{de}}$$

  in terms of the number of quantizer bits B and the oversampling ratio M

- As previously developed, the signal and noise are modeled as independent random processes

Signal Analysis

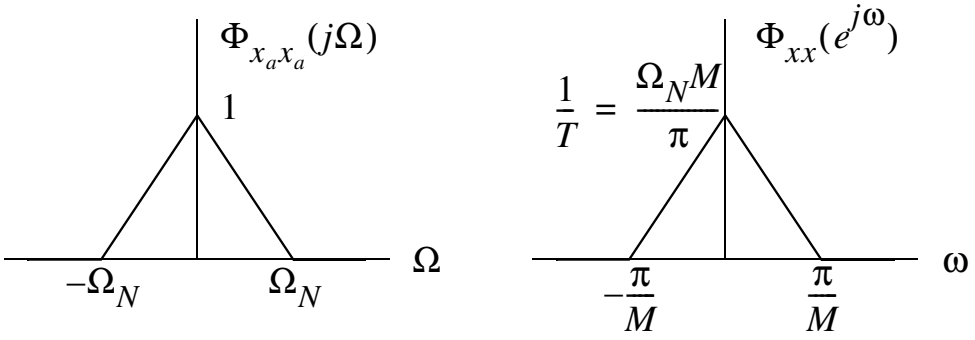
- To find the signal power we will use the power spectral density of $x_{da}[n]$ which is related to the power spectral density of $x_a(t)$

- Following the development in Appendix A of this chapter, we know that $\phi_{xx}[m] = \phi_{x_a x_a}(mT)$ and hence

$$\Phi_{xx}(e^{j\Omega T}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} \Phi_{x_a x_a}\left(j\left(\Omega - \frac{2\pi k}{T}\right)\right)$$

- The bandlimited assumption about $x_a(t)$ allows us to write

$$\Phi_{xx}(e^{j\omega}) = \begin{cases} \frac{1}{T}\Phi_{x_a x_a}\left(j\frac{\omega}{T}\right), & |\omega| < \pi/M, \\ 0, & \pi/M < |\omega| \leq \pi \end{cases}$$



Signal $x_a(t)$ spectra before and after sampling

- The power in the input analog signal is given by

$$\mathcal{E}\{x_a^2(t)\} = \frac{1}{2\pi}\int_{-\Omega_N}^{\Omega_N} \Phi_{x_a x_a}(j\,\Omega)\,d\,\Omega$$

- The power in the sampled analog signal is

$$\begin{aligned} \mathcal{E}\{x^2[n]\} &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_{xx}(e^{j\omega})\,d\omega \\ &= \frac{1}{2\pi}\int_{-\pi/M}^{\pi/M} \frac{1}{T}\Phi_{x_a x_a}\left(j\frac{\omega}{T}\right)\,d\omega \end{aligned}$$

- If we change variables of integration in the above by letting $\omega \rightarrow \Omega T$ and noting that $\pi/M \rightarrow \Omega_N$, we can also write that

$$\mathcal{E}\{x^2[n]\} = \frac{1}{2\pi}\int_{-\Omega_N}^{\Omega_N} \Phi_{x_a x_a}(j\,\Omega)\,d\,\Omega = \mathcal{E}\{x_a^2(t)\}$$

  and hence that the power in the input analog signal and sampled signal are identical

- The decimation lowpass filter is ideal and $x[n]$ is bandlimited so we can write

$$x_{da}[n] = x[nM] = x_a(nMT)$$

  which further implies that the signal power is constant through the entire system

- Viewed in terms of the power spectrum

$$\Phi_{x_{da}x_{da}}(e^{j\omega}) = \frac{1}{M}\Phi_{xx}(e^{j\omega/M}), \quad |\omega| < \pi$$

  and

$$\begin{aligned}\mathcal{E}\{x_{da}^2[n]\} &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \Phi_{x_{da}x_{da}}(e^{j\omega})\,d\omega \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{1}{M}\Phi_{xx}(e^{j\omega/M})\,d\omega \\ &= \frac{1}{2\pi}\int_{-\pi/M}^{\pi/M} \Phi_{xx}(e^{j\omega})\,d\omega = \mathcal{E}\{x^2[n]\} = P_{da}\end{aligned}$$

Noise Analysis

- The quantization noise analysis is different as it is not bandlimited to $|\omega| < \pi/M$

- We model $e[n]$ as a WSS zero mean white-noise process with

$$\phi_{xx}[m] = \sigma_e^2 \delta[m] = \frac{\Delta^2}{12}\delta[m]$$

- The noise power spectrum is simply

$$\Phi_{ee}(e^{j\omega}) = \sigma_e^2 \quad |\omega| < \pi$$

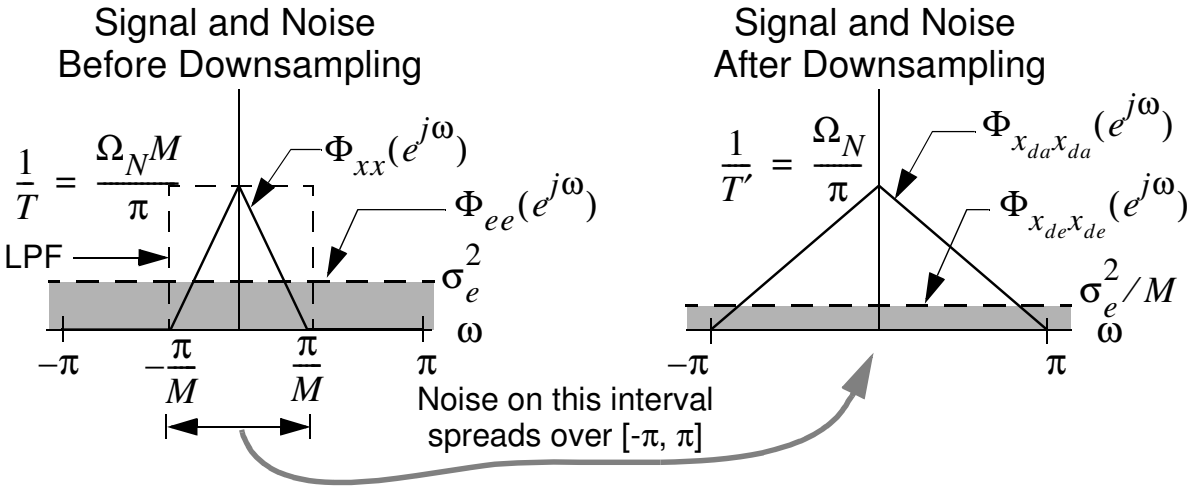
- The noise power in $e[n]$ is of course σ_e^2

- The noise power following the decimation lowpass filter and decimator is

$$\begin{aligned}\mathcal{E}\{x_{de}^2[n]\} &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \sigma_e^2 |H_{\text{LPF}}(e^{j\omega})|^2 \, d\omega \\ &= \frac{1}{2\pi}\int_{-\pi/M}^{\pi/M} \sigma_e^2 \, d\omega = \frac{\sigma_e^2}{M} = P_{de}\end{aligned}$$

- From this we can also infer that

$$\Phi_{x_{de}x_{de}}(e^{j\omega}) = \frac{\sigma_e^2}{M}$$



Signal and noise spectra before and after downsampling

- The bandlimiting action of the decimation lowpass filter has reduced the quantization noise power by the factor M
- Note that since $\Delta = X_m/2^B$ we can also write that

$$\mathcal{E}\{x_{de}^2[n]\} = \frac{1}{12M}\left(\frac{X_m}{2^B}\right)^2$$

SNR_Q Observations

- We can now write the signal-to-quantization noise ratio (SNR_Q) as

$$\left(\text{SNR}_Q\right)_d = \frac{P_{da}}{P_{de}} = \mathcal{E}\{x_{da}^2[n]\} \cdot 12M\left(\frac{2^B}{X_m}\right)^2$$

- Clearly as M increases SNR_Q increases linearly
- Also observe that for a fixed SNR_Q increasing M by four is the same as increasing B by one bit, i.e., every doubling of M increases the effective number of bits by 1/2

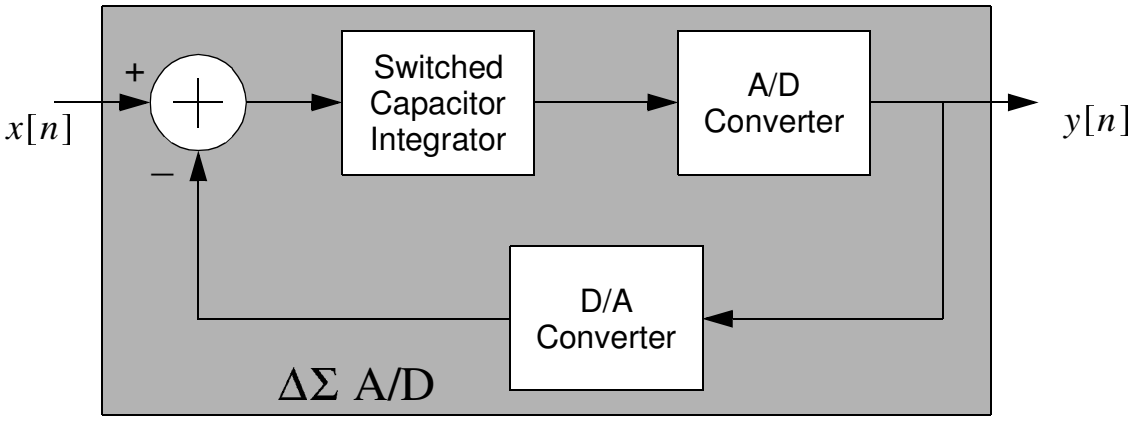
- To increase the effective number of bits from say 10 to 16 would require an oversampling factor of $M = 2^{2(16-10)} = 2^{12}$!

4.13.2 Oversampled A/D Conversion with Noise Shaping

In the previous section we saw that by the using oversampling the effective number of quantization bits can be increased by $\log_2(M)/2$ bits. In this section we will see that with noise shaping the increase can be significantly greater than this.

- We would like to make the quantization noise spectrum non-uniform so that most of the noise power is outside the band $|\omega| < \pi/M$, and thus removed by the downsampling lowpass filter

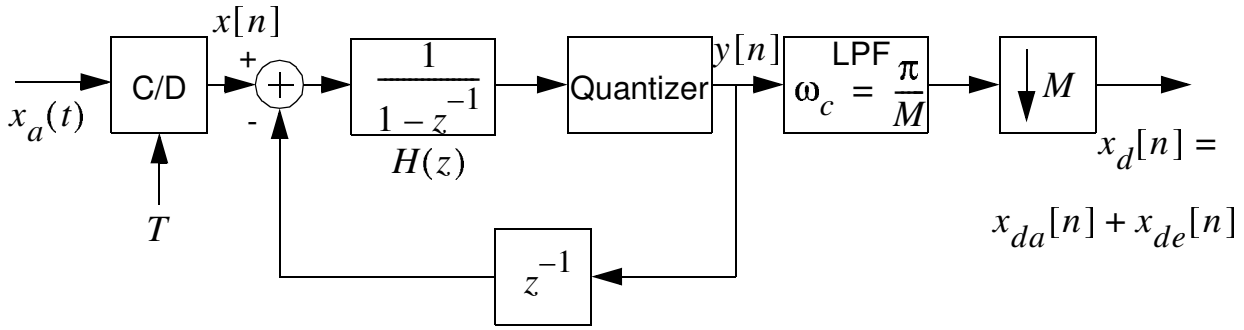
- A generic *Delta-Sigma* modulator or just $\Delta\Sigma$ A/D is shown in the following figure



A general $\Delta\Sigma$ A/D

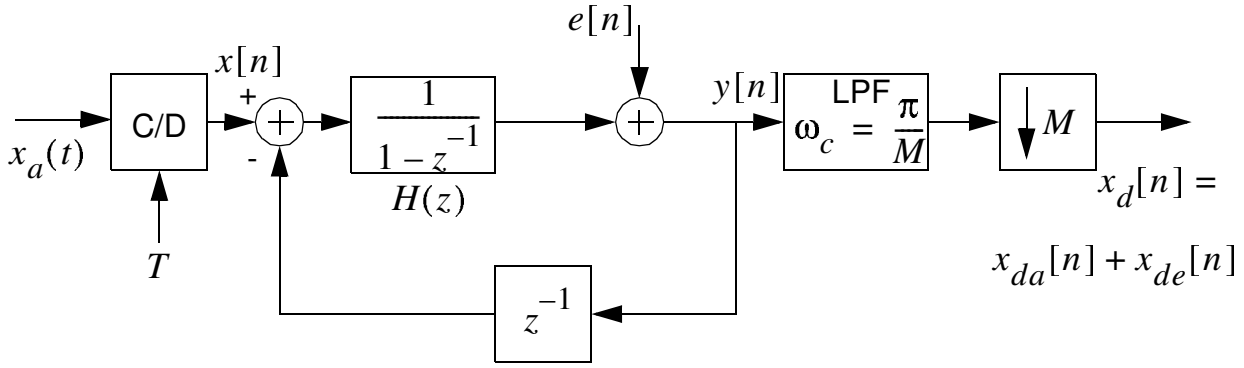
- The complete design is implemented in integrated circuit form

- The A/D converter is typically a 1-bit quantizer or simply a comparator
- For modeling purposes a discrete-time equivalent can be constructed as shown below



Discrete-time model of a 1st-order $\Delta\Sigma$ A/D

- Note that the switched capacitor integrator is modeled using a discrete-time accumulator
- To model the quantization noise performance we again use a linear quantization noise model



First-order $\Delta\Sigma$ A/D with linear noise model

- The A/D output is due to a transfer function that carries $x[n]$ to $y[n]$ and $e[n]$ to $y[n]$
- Mathematically we can write

$$Y(z) = \underbrace{X(z)H_x(z)}_{Y_x(z)} + \underbrace{E(z)H_e(z)}_{\hat{E}(z)}$$

- To find $H_x(z)$ note that

$$Y_x(z) = \left(X(z) - Y_x(z)z^{-1}\right)\left(\frac{1}{1-z^{-1}}\right)$$

or

$$H_x(z) = \frac{Y_x(z)}{X(z)} = \frac{\frac{1}{1-z^{-1}}}{1+\frac{z^{-1}}{1-z^{-1}}} = 1$$

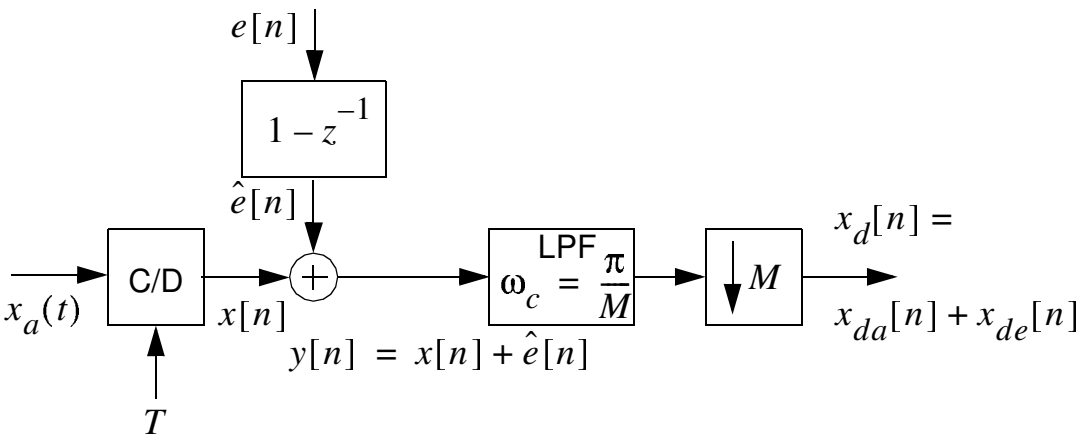
- To find $H_e(z)$ note that

$$\hat{E}(z) = E(z) - \hat{E}(z)\frac{z^{-1}}{1-z^{-1}}$$

or

$$H_e(z) = \frac{\hat{E}(z)}{E(z)} = \frac{1}{1+\frac{z^{-1}}{1-z^{-1}}} = 1 - z^{-1}$$

- The final computational form of the $\Delta\Sigma$ A/D is shown below



- The signal remains unchanged from earlier analysis, that is

$$y_x[n] = x[n]$$

and

$$P_{da} = \mathcal{E}\{x_{da}^2[n]\} = \mathcal{E}\{x^2[n]\} = \mathcal{E}\{x_a^2(t)\}$$

- The filter $H_e(z)$ modifies the quantization noise spectrum so that

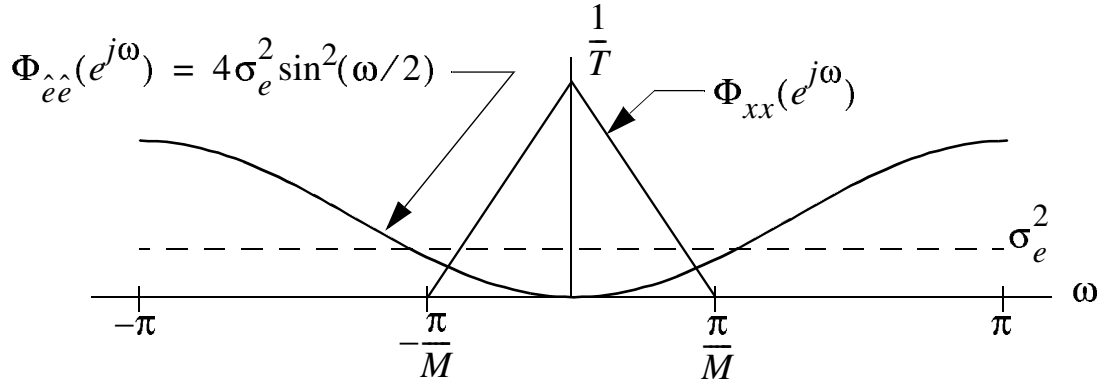
$$\Phi_{\hat{e}\hat{e}}(e^{j\omega}) = \sigma_e^2 |H_e(e^{j\omega})|^2 = \sigma_e^2 [2\sin(\omega/2)]^2$$

- The noise power contained in $\hat{e}[n]$ is

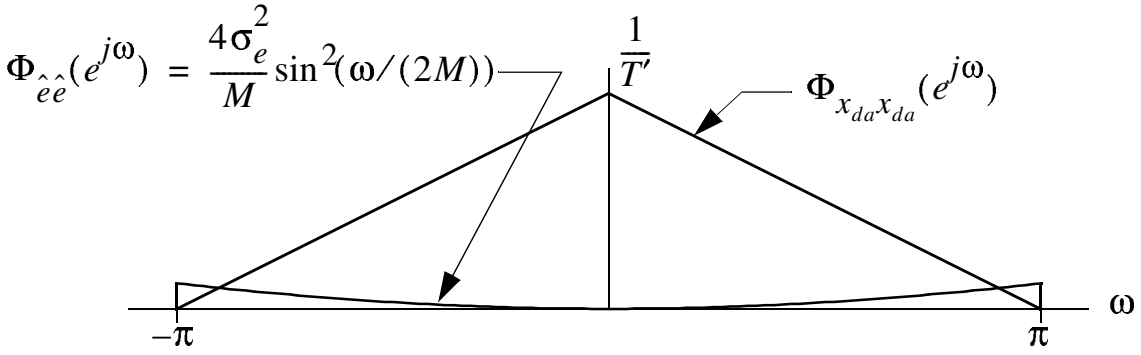
$$\begin{aligned}\mathcal{E}\{\hat{e}[n]\} &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \sigma_e^2 [2\sin(\omega/2)]^2 \, d\omega \\ &= \frac{4\sigma_e^2}{2\pi}\int_{-\pi}^{\pi} \sin^2(\omega/2)\, d\omega \\ &= \frac{4\sigma_e^2}{2\pi}\int_{-\pi}^{\pi} \left[\frac{1}{2} - \frac{1}{2}\cos(\omega)\right] d\omega = 2\sigma_e^2\end{aligned}$$

which is twice that of $e[n]$!

- The total noise power may be doubled, but in the frequency band $[-\pi/M, \pi/M]$ the power spectral density is much smaller than without noise shaping



- Assuming as before an ideal lowpass filter in the downsampler, the composite power spectral density of $x_d[n]$ is as shown below



Shaped noise and signal spectrum following downsampling

- The quantization noise power is

$$\begin{aligned} P_{de} &= \mathcal{E}\{y_{de}[n]\} = \frac{1}{2\pi}\int_{-\pi}^{\pi}\Phi_{x_{de}x_{de}}(e^{j\omega})\,d\omega \\ &= \frac{1}{2\pi}\frac{1}{12M}\left(\frac{X_m}{2^B}\right)^2\int_{-\pi}^{\pi}\left(2\sin\left(\frac{\omega}{2M}\right)\right)^2 d\omega \\ &= \frac{1}{12M}\left(\frac{X_m}{2^B}\right)^2\left[2-\frac{2M}{\pi}\sin(\pi/M)\right] \end{aligned}$$

- Assuming that M is large we can approximate $\sin(u) \approx u$ so

$$\begin{aligned} P_{de} &\approx \frac{1}{2\pi}\frac{1}{12M}\left(\frac{X_m}{2^B}\right)^2\int_{-\pi}^{\pi}\left(2\frac{\omega}{2M}\right)^2 d\omega \\ &= \frac{1}{2\pi}\frac{1}{12M}\left(\frac{X_m}{2^B}\right)^2\left(\frac{\omega^3}{3M^3}\Big|_{-\pi}^{\pi}\right) \\ &= \frac{1}{36}\left(\frac{X_m}{2^B}\right)^2\frac{\pi^2}{M^3} \end{aligned}$$

- Using the approximation we see that doubling M is equivalent to a gain of 1.5 bits
- By solving for B in the expressions for P_{de} we can calculate the savings in bits when using combinations of oversampling and noise shaping compared to $M = 1$ and flat noise
- For the case of no noise shaping, but oversampling, the bit savings is given by

$$\Delta B_0(M) = \frac{1}{2}\log_2\left[M\right]$$

- Considering the exact expression for P_{de} with first-order noise shaping, the bit savings is

$$\Delta B_1(M) = \frac{1}{2}\log_2\left[\frac{M}{2\left\{1 - \frac{M}{\pi}\sin(\pi/M)\right\}}\right]$$

- Considering the approximate expression for P_{de} with first-order noise shaping, the bit savings is

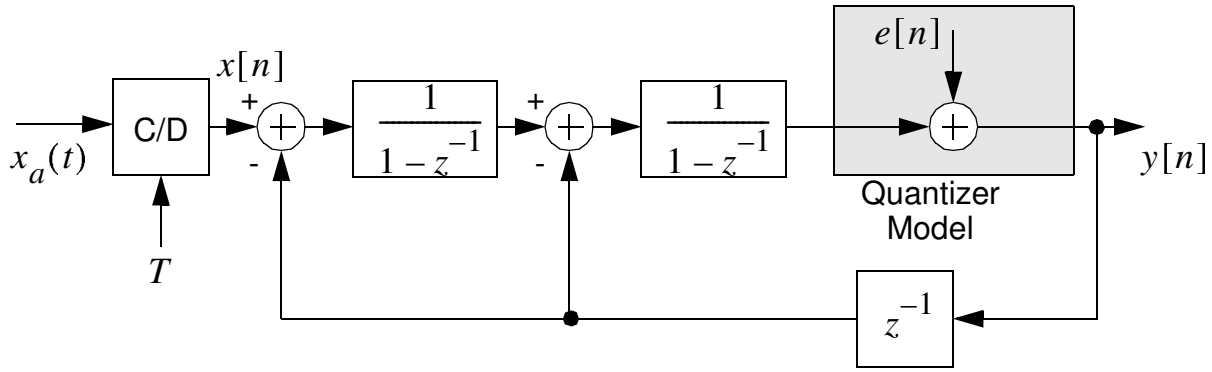
$$\Delta \tilde{B}_1(M) = \frac{1}{2}\log_2\left[\frac{3M^3}{\pi^2}\right]$$

- The bit savings expressions are compared in the following table for $M = 4, \ldots, 64$

M	Direct Quant. $\Delta B_0(M)$	Noise Shape $\Delta B_1(M)$	Noise Shape approx. $\Delta \tilde{B}_1(M)$
4	1	2.16	2.14
8	1.5	3.65	3.64
16	2	5.14	5.14
32	2.5	6.64	6.64
64	3.5	8.14	8.14

Higher-Order Noise Shaping

- To further improve the bit savings a higher-order noise shaping scheme can be employed
- A second-order noise shaping $\Delta\Sigma$ A/D is shown below



Second-order noise shaping $\Delta\Sigma$-A/D

- The design is such that $H_x(z) = 1$ and

$$H_e(z) = (1 - z^{-1})^2$$

so,

$$\Phi_{\hat{e}\hat{e}}(e^{j\omega}) = \sigma_e^2\left[2\sin(\omega/2)\right]^4$$

- When p-stages are employed the quantization noise power spectral density becomes

$$\Phi_{\hat{e}\hat{e}}(e^{j\omega}) = \sigma_e^2\left[2\sin(\omega/2)\right]^{2p}$$

- The approximate expression for P_{de} when using the small angle approximation for $\sin(u)$ becomes

$$\begin{aligned} P_{de} &= \frac{1}{2\pi}\frac{1}{12M}\left(\frac{X_m}{2^B}\right)^2 \int_{-\pi}^{\pi}\left[2\left(\frac{\omega}{2M}\right)\right]^{2p} d\omega \\ &= \frac{1}{12(2p+1)}\left(\frac{X_m}{2^B}\right)^2 \frac{\pi^{2p}}{M^{2p+1}} \end{aligned}$$

- Using this approximate expression for P_{de} the bit savings compared with no noise shaping ($p = 0$) and no oversampling ($M = 1$) is

$$\Delta \tilde{B}_p(M) = \frac{1}{2}\log_2\left[\frac{(2p+1)M^{2p+1}}{\pi^{2p}}\right]$$

- Quantizer bit savings as a function of noise shaping filter order, p, and oversampling ratio, M, is given in the following table

Filter Order	Oversampling Factor M				
p	4	8	16	32	64
0	1.00	1.50	2.00	2.50	3.00
1	2.14	3.64	5.14	6.64	8.14
2	2.86	5.36	7.86	10.36	12.86
3	3.45	6.95	10.45	13.95	17.45
4	3.98	8.48	12.98	17.48	21.98
5	4.47	9.97	15.47	20.97	26.47

- Note that the numbers in the above table differ slightly from those in the text (*why?*)

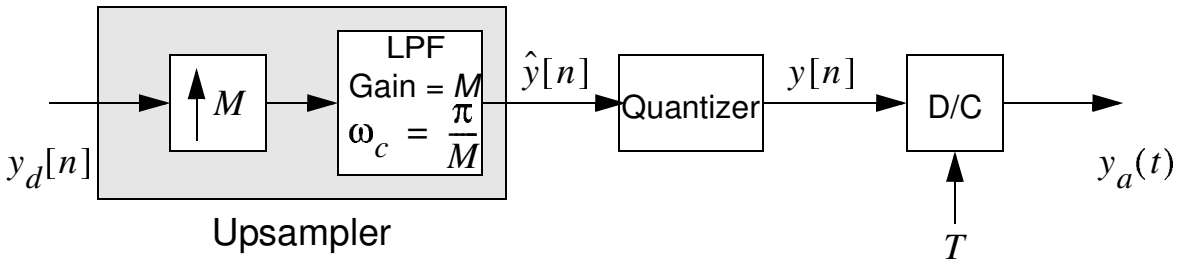
- Suppose that we want to achieve $\geq$ 16-bit accuracy with just a 1-bit quantizer; Choosing $M = 64$ and $p = 3$ equates to 18.45 equivalent bits

  - The text authors point out that large values of p can be difficult to deal with
  - The filter structure known as *multistage noise shaping* or *MASH* is an alternative explored in Text problem 4.62

4.13.3 Oversampling and Noise Shaping in D/A Conversion

In the D/A conversion process similar issues exist with regard to realizability of analog filters, here reconstruction, and the use of oversampling to reduce the number of quantization bits.

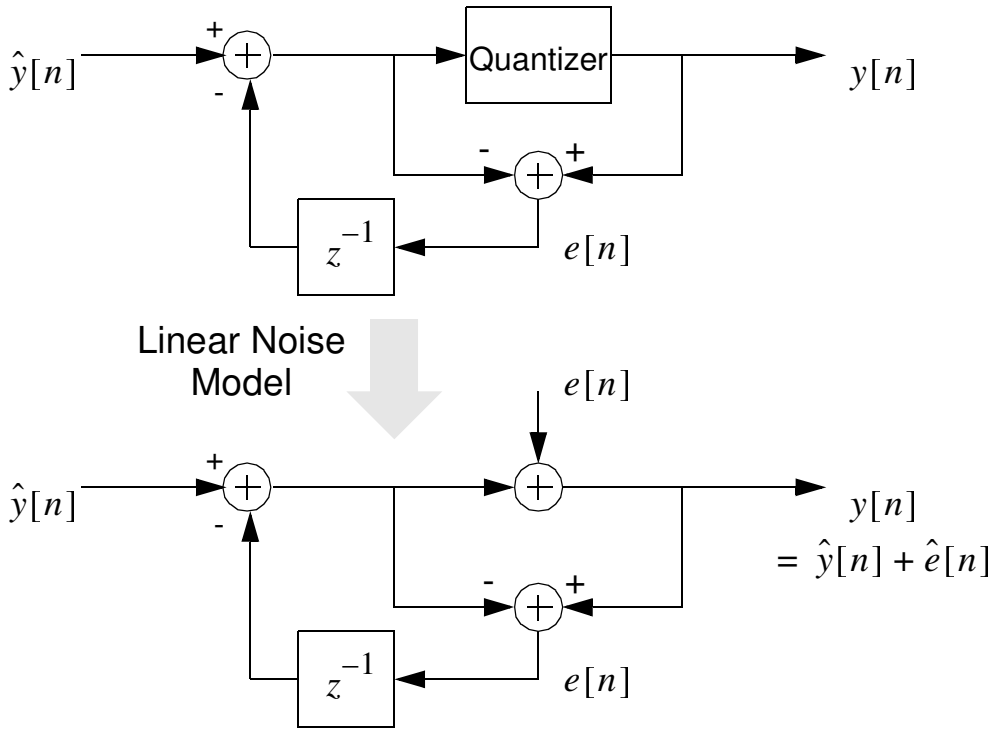
- The basic idea is to oversample a signal just prior to the D/A converter, but also choose the D/A to have fewer bits (hence requantize the higher sample rate signal)



Upsampler followed by requantization to motivate $\Delta\Sigma$-D/A

- For this scheme to work the quantization noise must be shifted outside the signal band

- A first-order noise shaping filter and the linear noise model equivalent is shown below



First-order noise shaping $\Delta\Sigma$-D/A and linear noise equivalent

- The signal path transfer function is such that $y[n] = \hat{y}[n]$
- The transfer function relating $\hat{e}[n]$ to $e[n]$ is

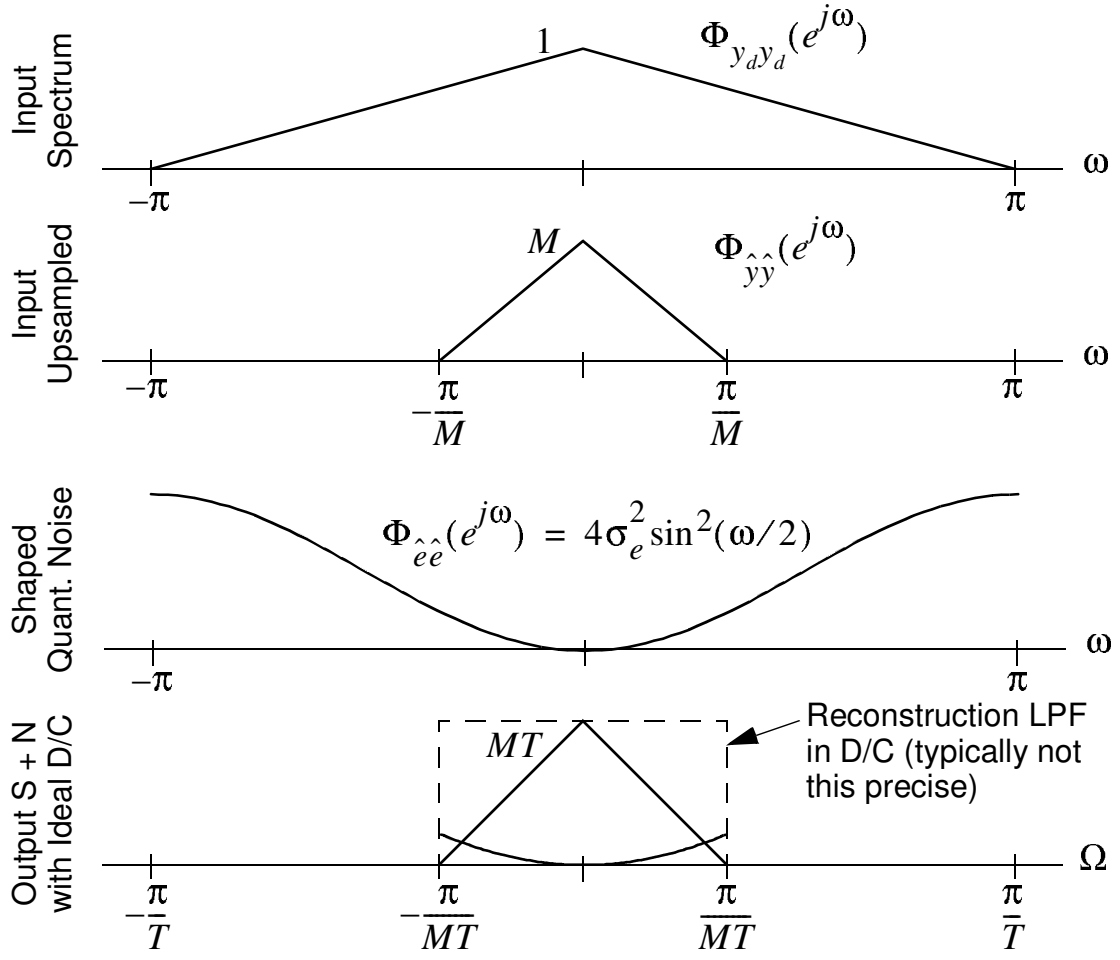
$$\hat{E}(z) = E(z) - E(z)z^{-1}$$

or

$$H_e(z) = \frac{\hat{E}(z)}{E(z)} = 1 - z^{-1}$$

- The quantization noise spectrum is given by

$$\Phi_{\hat{e}\hat{e}}(e^{j\omega}) = \sigma_e^2\left[2\sin(\omega/2)\right]^2$$



Signal and noise spectra in the first-order $\Delta\Sigma$-D/A

- In the above figure it is assumed that the D/C has an ideal reconstruction filter
- This requirement must be relaxed in a practical $\Delta\Sigma$-D/A
- A softer reconstruction filter will allow more noise to pass, but a multistage noise shaping filter can be employed to counter this loss

- As in the A/D case an order-p noise shaping filter produces a spectrum of the form

$$\Phi_{\hat{e}\hat{e}}(e^{j\omega}) = \sigma_e^2 \left[2\sin(\omega/2)\right]^{2p},$$

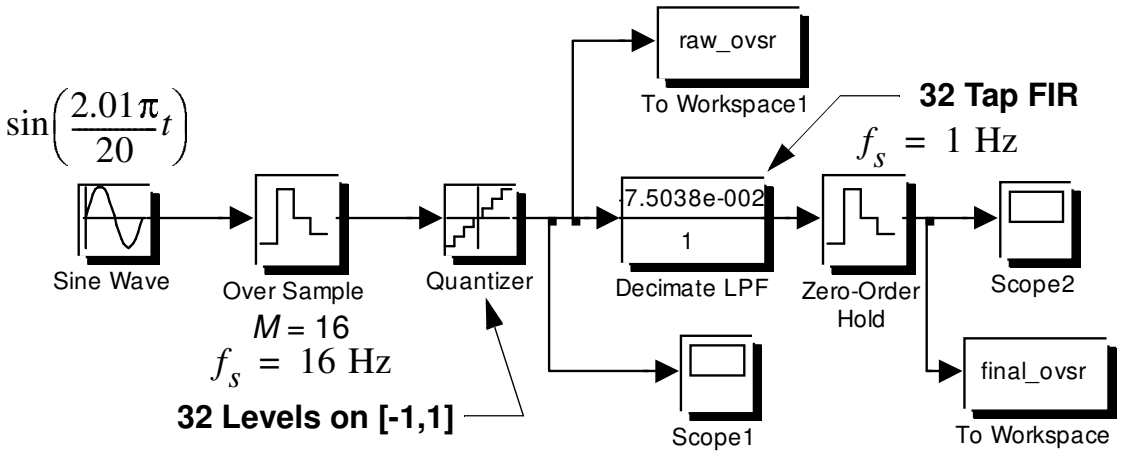
thus allowing simple reconstruction filter designs

Example 4.18: Simulink $\Delta\Sigma$ Simulation

In this example MATLAB *Simulink* is used to investigate the quantization noise properties of oversampling with and without noise shaping. The MATLAB Simulink environment, which is now included as part of the MathWorks Student edition, allows for block diagram based simulation of continuous, discrete, and mixed systems.

Oversampled A/D

- The block diagram constructed is that of the basic oversampled based A/D conversion



Simulink Oversampled A/D having $M = 16$

- In this block diagram we have a continuous-time sinusoid of the form $\sin[(2.01\pi/20)t]$ driving a zero-order hold block which sets the sampling rate at $T = 1/16$ s or $f_s = 16$ Hz

- The quantizer block has quantization steps at intervals of $1/16$ so the interval $[-1, 1]$ contains 32 levels (actually 33 since zero is also included)

- Sampling rate conversion is performed by a 32-tap FIR low-pass filter with $\omega_c = \pi/16$

- Decimation by $M = 16$ is performed by another zero-order hold block which samples with $T = 1$s

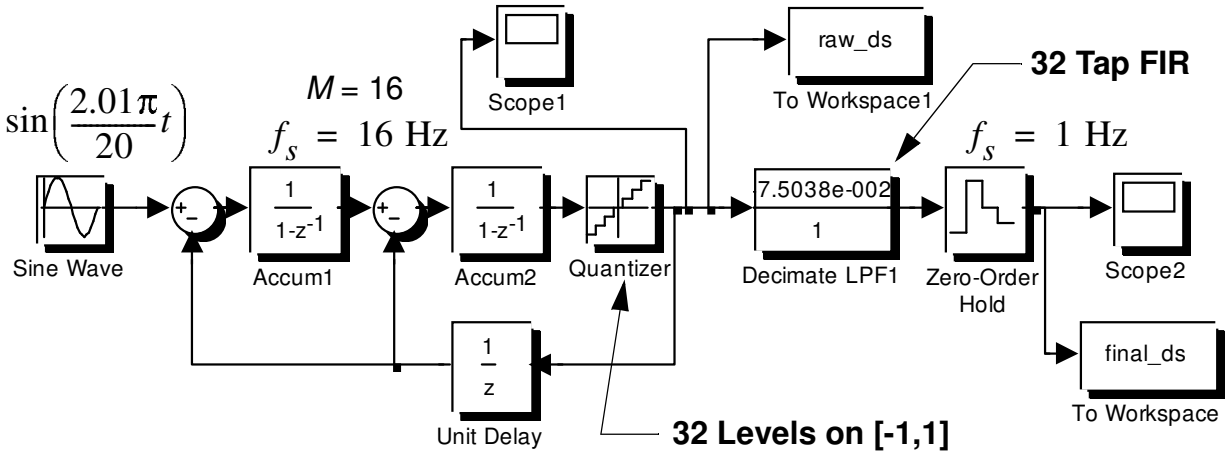
- Samples are collected in the MATLAB workspace from the quantizer output and the decimator output

- Spectral analysis is then performed using `psd()`

$p = 2$ $\Delta\Sigma$ A/D

- For comparison purposes a $\Delta\Sigma$ A/D block diagram is configured using a second-order ($p = 2$) shaping filter

- The remaining block diagram elements are the same as in the straight oversampled A/D

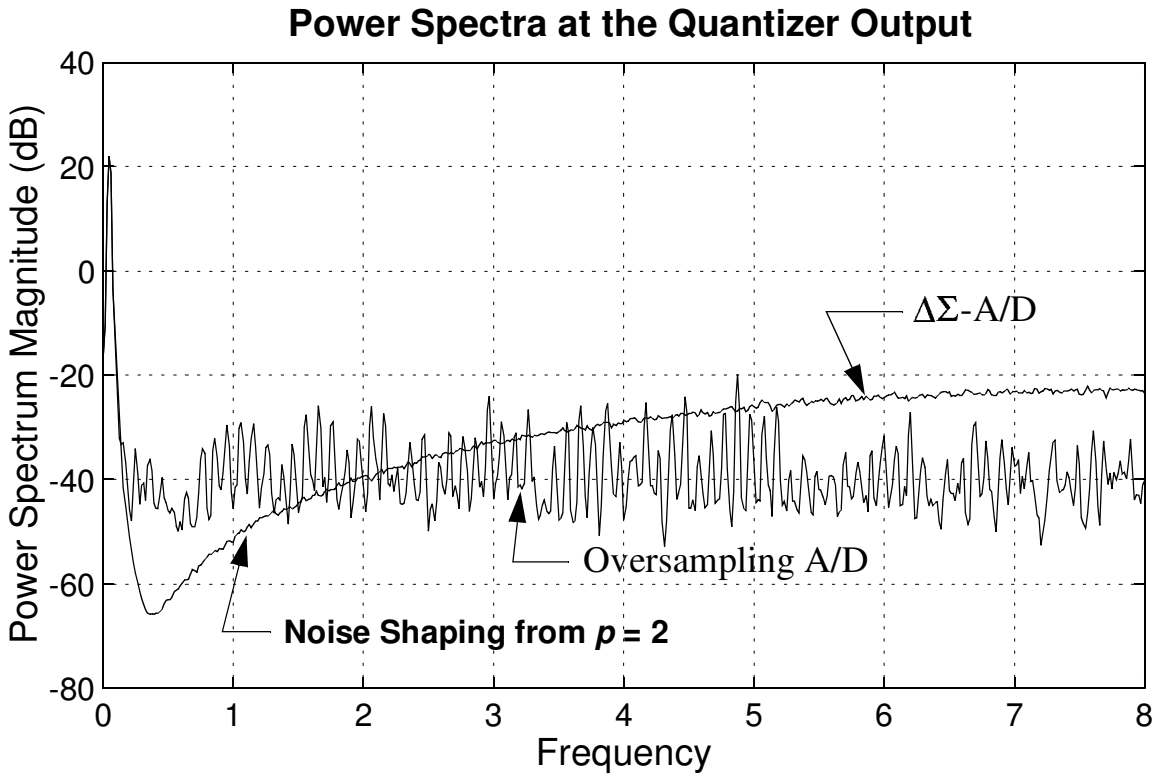


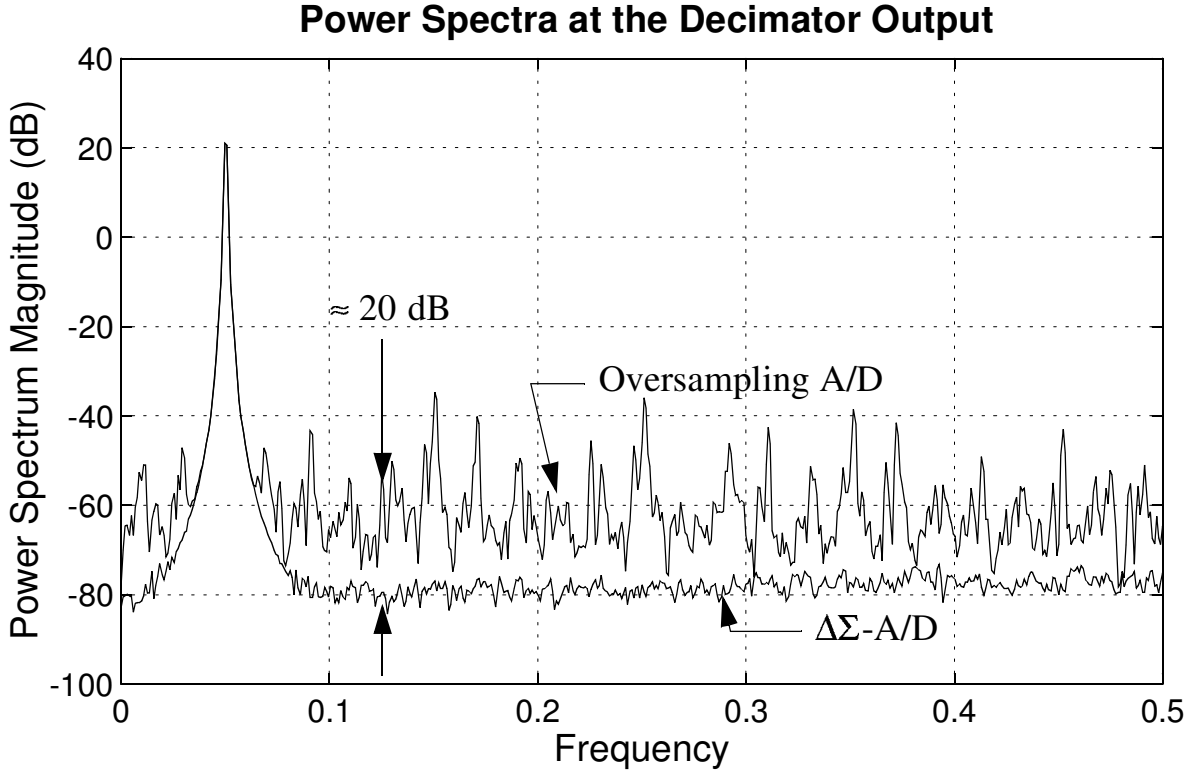
Simulink $\Delta\Sigma$ A/D having $M = 16$ and $p = 2$

- Simulation runs of length 10,000 seconds are run with each system

- Simulink scope blocks are used to initially validate waveform correctness

- The following figures compare the power spectrum differences between the two systems as viewed from the quantizer output and the decimator output





- The quantization noise floor drops by about 20 dB in the $\Delta\Sigma$-A/D compared with the oversampled A/D

- The theoretical reduction in total noise power should be the result of gaining an equivalent of 7.86 bits or $6.02 \times 7.86 = 47$ dB
- More work is needed to verify this

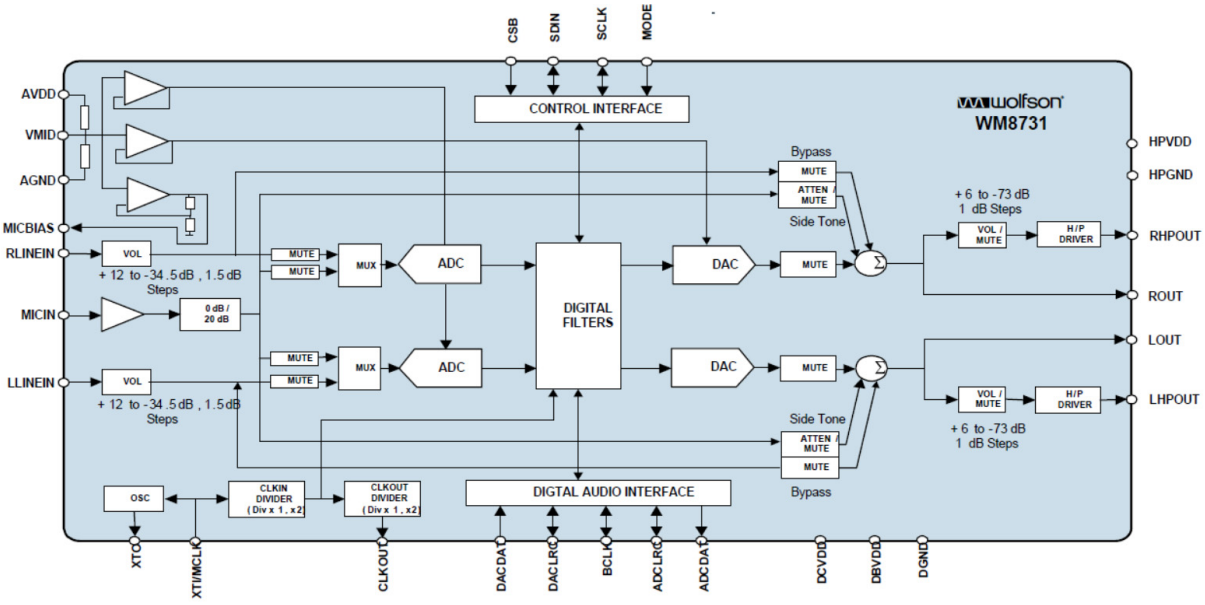
Example 4.19: Practical Analog Interface Circuits

In this example we look at two extremes in contemporary ADC/DAC electronic subsystems:

- At the low end a Wolfson (now part of Cirrus Logic) $\Sigma\Delta$ audio processor for use in portable electronics
- At the high end a very powerful RF sampling ADC from Texas Instruments (TI) introduced Summer 2016.

Wolfson 8731 Audio Codec

WM8731
Portable Internet Audio CODEC with Headphone Driver and Programmable Sample Rates



- Key attributes of the WM8731

FEATURES

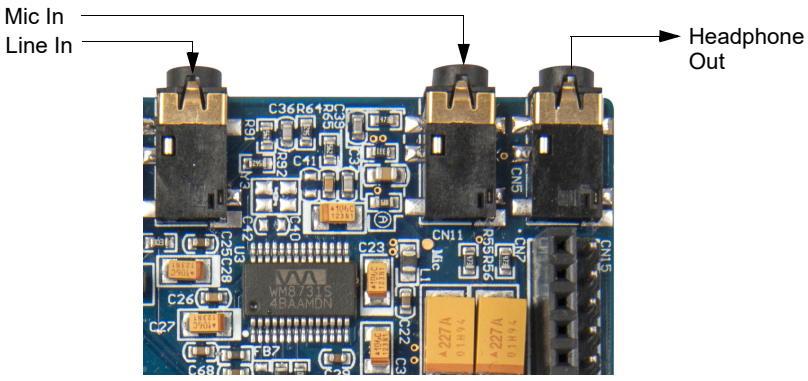
- Highly Efficient Headphone Driver
- Audio Performance
  - ADC SNR 90dB ('A' weighted) at 3.3V, 85dB at 1.8V
  - DAC SNR 100dB ('A' weighted) at 3.3V, 95dB at 1.8V
- Low Power
  - Playback only 22mW, 8mW ('L' Variant)
  - Analogue Pass Through 12mW, 3.5mW ('L' variant)
  - 1.42 – 3.6V Digital Supply Operation
  - 2.7 – 3.6V Analogue Supply Operation
  - 1.8 – 3.6V Analogue Supply Operation ('L' Variant)
- ADC and DAC Sampling Frequency: 8kHz – 96kHz
- Selectable ADC High Pass Filter
- 2 or 3-Wire MPU Serial Control Interface
- Programmable Audio Data Interface Modes
  - I^2S, Left, Right Justified or DSP
  - 16/20/24/32 bit Word Lengths
  - Master or Slave Clocking Mode
- Microphone Input and Electret Bias with Side Tone Mixer
- Available in 28-lead SSOP or 28-lead QFN package

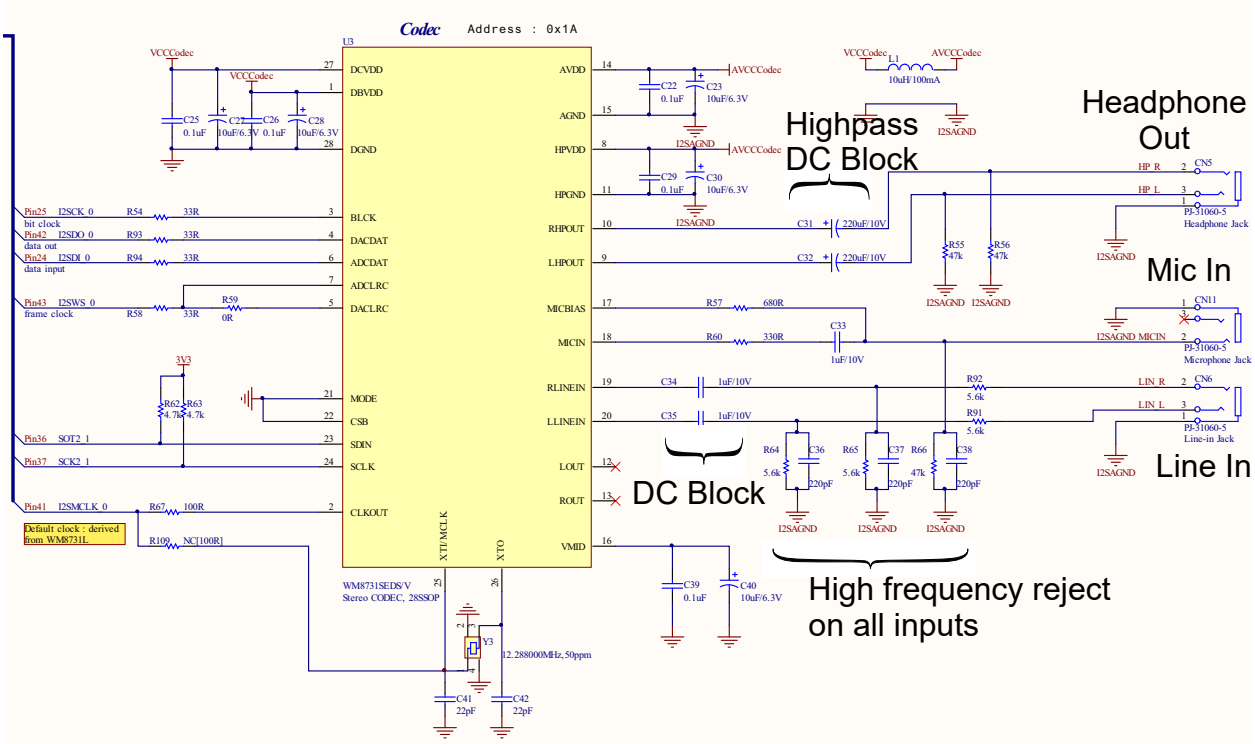
APPLICATIONS

- Portable MP3 Players and Recorders
- CD and Minidisc Recorders

- This audio codec is used on the Cypress FM4, which is currently (Spring 2016/Fall 2016) used for ECE 4680 and ECE 4655/5655

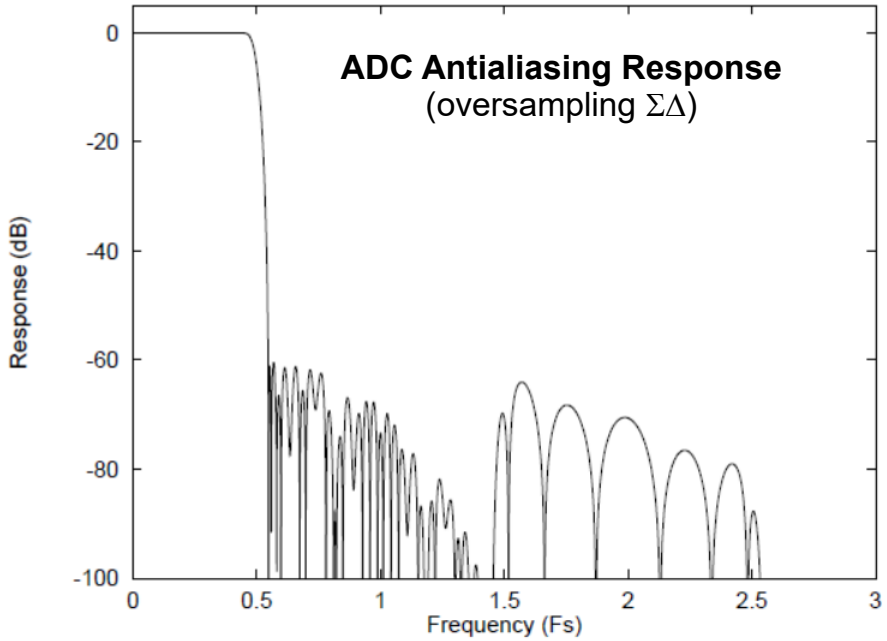
- The board level analog I/O and detailed schematic are shown below

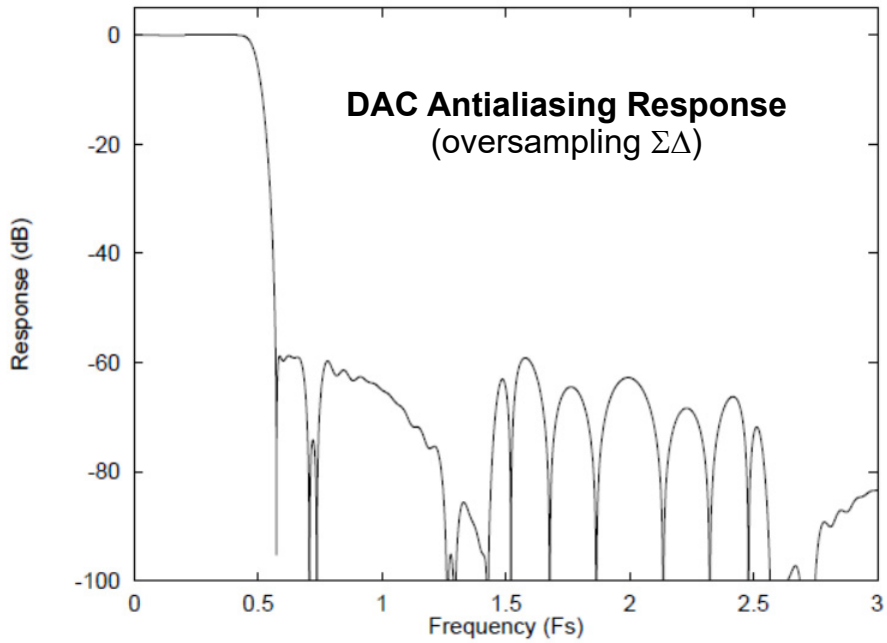




- From the right side of the schematic you can see all inputs and outputs are AC coupled (coupling capacitor present)
- High frequency rejection is provided with RC lowpass filters on all inputs
- Also notice there is only a single channel microphone input, which is typical of devices of this type (not for recording studio applications)

- The higher-order $\Sigma\Delta$ design means that high performance antialising and reconstruction filters are included in the audio signal processing chain



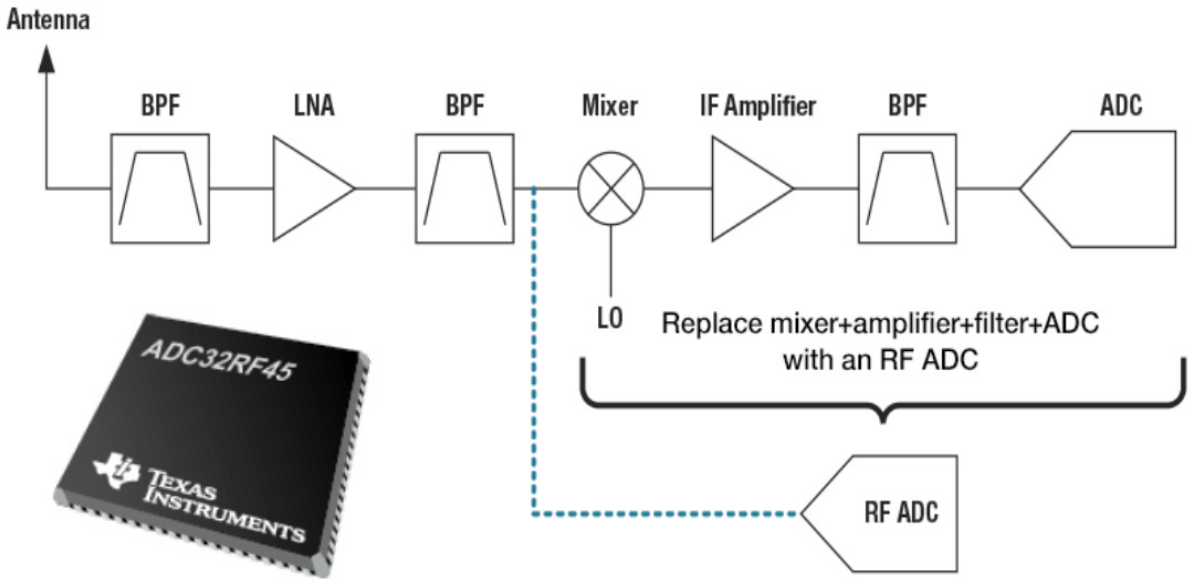


- The above responses provide 60 dB of rejection

RF Sampling ADCs from TI:

- For the past few years TI has been emphasizing RF sampling-based ADCs for use in high performance receiver systems
- The ADC32RF45 is a new model in the family as of Summer 2016
- The underlying philosophy of the RF ADC family is to replace the zero-IF (two ADC's required) and IF sampling based approaches with direct RF sampling

Texas Instruments RF Sampling Analog-to-Digital Converter (ADC)

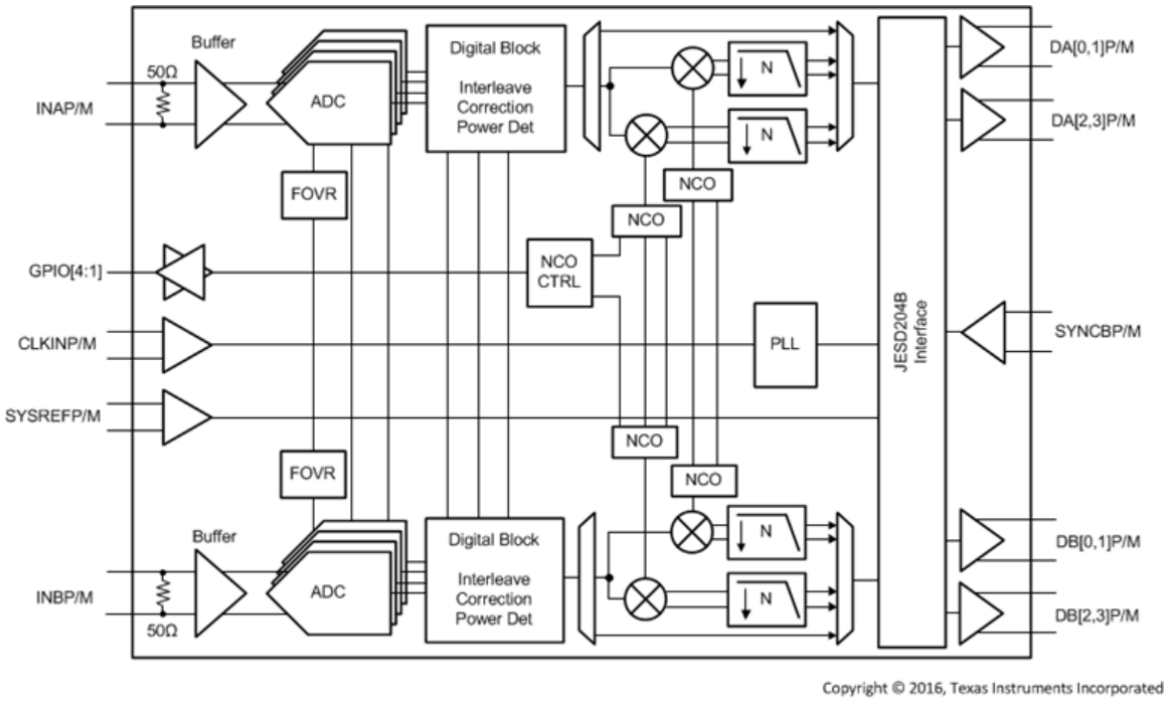


- From the above you see that the RF ADC is not sampling at the antenna as good receiver sensitivity requires an LNA (low-noise amplifier) and BPFs to eliminate strong out-of-band signals

- The function block diagram is shown below
- Notice all the multirate processing blocks and numerically controlled oscillators (NCOs)

SBAS747B – MAY 2016 – REVISED JULY 2016

ADC32RF45 Dual-Channel, 14-Bit, 3.0-GSPS, Analog-to-Digital Converter



- Two channels of RF sampling are provided with one chip
- The outputs for each channel are complex (I/Q) signals centered at zero frequency and downsampled to reduce throughput requirements
- The raw input from the two channels is $2 \cdot 14 \cot 3 \times 10^9 = 84$ Gsps!

- ADC32RF45 features and target applications

1 Features

- 14-Bit, Dual-Channel, 3.0-GSPS ADC
- Noise Floor: –155 dBFS/Hz
- RF Input Supports Up to 4.0 GHz
- Aperture Jitter: 70 f_S
- Channel Isolation: 95 dB at f_{IN} = 1.8 GHz
- Spectral Performance (f_{IN} = 900 MHz):
  - SNR: 60.6 dBFS
  - SFDR: 64-dBc HD2, HD3
  - SFDR: 72-dBc Worst Spur
- Spectral Performance (f_{IN} = 1.78 GHz):
  - SNR: 58.0 dBFS
  - SFDR: 63-dBc HD2, HD3
  - SFDR: 77-dBc Worst Spur
- On-Chip Digital Down-Converters:
  - Up to 4 DDCs (Dual-Band Mode)
  - Up to 3 Independent NCOs per DDC
- On-Chip Input Clamp for Overvoltage Protection
- Programmable On-Chip Power Detectors with Alarm Pins for AGC Support
- On-Chip Dither
- On-Chip, 50-Ω Input Termination
- Input Full-Scale: 1.35 V_{PP}
- Support for Multi-Chip Synchronization
- JESD204B Interface:
  - Subclass 1-Based Deterministic Latency
  - 4 Lanes Per ADC at 12.3 Gbps
- Power Supply:
  - 1.9 V (Analog), 1.15 V (Analog), 1.15 V (Digital)
- Power Dissipation: 3.2 W/Ch at 3.0 GSPS
- 72-Pin VQFN Package (10 mm × 10 mm)

2 Applications

- Multi-Band, Multi-Mode 2G, 3G, 4G Cellular Receivers
- Phased Array Radars
- Electronic Warfare
- Cable Infrastructure
- Broadband Wireless
- High-Speed Digitizers
- Software-Defined Radios
- Communications Test Equipment
- Microwave and Millimeter Wave Receivers

4.14 Appendix A: Continuous vs. Discrete WSS Random Processes

Continuous vs Discrete-Time WSS Random Processes The text and notes discussion of random processes is for the most part restricted to discrete-time random processes. In order to work problems involving the sampling of continuous-time random processes, a basic understanding of continuous-time random processes is helpful. This short note is intended to help supply some of the needed information for the special case of wide-sense stationary (WSS) processes.

4.14.1 Continuous-Time Domain Definitions

To begin with consider the basic definitions of auto-correlation and power spectral density of the random process $x_c(t)$.

- The mean is given by

$$m_{x_c} = E\{x_c(t)\}$$

- The autocorrelation function is

$$\phi_{x_c x_c}(\tau) = \{x_c(t+\tau)x_c^*(t)\}$$

- The auto covariance function is

$$\begin{aligned}\gamma_{x_c x_c}(\tau) &= \left\{[x_c(t+\tau) - m_{x_c}][x_c(t) - m_{x_c}]^*\right\} \\ &= \phi_{x_c x_c}(\tau) - |m_{x_c}|^2\end{aligned}$$

- The power spectral density is

$$P_{x_c x_c}(\Omega) = \Phi_{x_c x_c}(j\,\Omega) = \int_{-\infty}^{\infty} \phi_{x_c x_c}(\tau) e^{-j\,\Omega\tau}\, d\,\tau$$

4.14.2 Relationships to Discrete-Time Quantities

The continuous-time quantities of autocorrelation and power spectral density are related to their discrete-time counter-parts via sampling relationships. To begin with we assume that $x[n] = x_c(nT)$, where T is the sample spacing.

- The autocorrelation functions are related as follows:

$$\begin{aligned}\phi_{xx}[m] &= E\{[x[n+m]x^*[n]\} \; = E\{x_c((n+m)T)x_c^*(nT)\} \\ &= \phi_{x_c x_c}(mT)\end{aligned}$$

- To obtain the power spectrum relationship first note that

$$\phi_{xx}[m] = \phi_{x_c x_c}(mT) = \mathcal{F}^{-1}\left\{P_{x_c x_c}(\Omega)\right\}\Big|_{\tau = mT}$$

- Proceed by taking the Fourier transform of both sides

$$\begin{aligned}P_{xx}(\omega) &= \sum_{m=-\infty}^{\infty}\left[\frac{1}{2\pi}\int_{-\infty}^{\infty} P_{x_c x_c}(\Omega)e^{j\Omega\tau}\,d\Omega\right]_{\tau=mT} e^{-j\omega m} \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} P_{x_c x_c}(\Omega)\left[\sum_{m=-\infty}^{\infty} e^{j\Omega mT}e^{-j\omega m}\right]d\Omega\end{aligned}$$

- The inner sum can be written as an impulse train by using the Poisson sum formula

$$\sum_{m=-\infty}^{\infty} e^{j(\Omega-\omega/T)mT} = \frac{2\pi}{T}\sum_{m=-\infty}^{\infty}\delta\left(\Omega - \frac{\omega}{T} + \frac{2\pi m}{T}\right)$$

- Now substitute the impulse train representation into the power spectrum equation to obtain

$$P_{xx}(\omega) = \frac{1}{T}\sum_{m=-\infty}^{\infty}\left[\int_{-\infty}^{\infty} P_{x_c x_c}(\Omega)\delta\left(\Omega - \frac{\omega}{T} + \frac{2\pi m}{T}\right) d\Omega\right]$$
$$= \frac{1}{T}\sum_{m=-\infty}^{\infty} P_{x_c x_c}\left(\frac{\omega}{T} - \frac{2\pi m}{T}\right)$$

4.14.3 Alternate Power Spectrum Derivation

- The power spectral density relationship can also be established in a manner very similar to that found in the notes

$$P_{xx}(\omega) = \sum_{m=-\infty}^{\infty}\phi_{xx}[m]e^{-j\omega m} = \sum_{m=-\infty}^{\infty}\phi_{x_c x_c}(mT)e^{-j\omega m}$$
$$= \left[\mathcal{F}\left\{\phi_{x_c x_c}(\tau)\sum_{m=-\infty}^{\infty}\delta(\tau - mT)\right\}\right]_{\omega=\Omega T}$$
$$= \left[\frac{1}{2\pi}\mathcal{F}\{\phi_{x_c x_c}(\tau)\} * \frac{2\pi}{T}\sum_{m=-\infty}^{\infty}\delta\left(\Omega - \frac{2\pi m}{T}\right)\right]_{\omega=\Omega T}$$
$$= \left[\frac{1}{T}\sum_{m=-\infty}^{\infty} P_{x_c x_c}\left(\Omega - \frac{2\pi m}{T}\right)\right]_{\omega=\Omega T}$$

- Finally,

$$P_{xx}(\omega) = \frac{1}{T}\sum_{m=-\infty}^{\infty} P_{x_c x_c}\left(\frac{\omega - 2\pi m}{T}\right),$$

the same result as obtained earlier

.