Chapter 2

Useful Discrete-Time Techniques for Digital Communications (Rice ch2)

Contents

.

2.1 Introduction

In this chapter we first review some basic digital signal processing (DSP) concepts and then explore multirate DSP as a useful technique in digital communications.

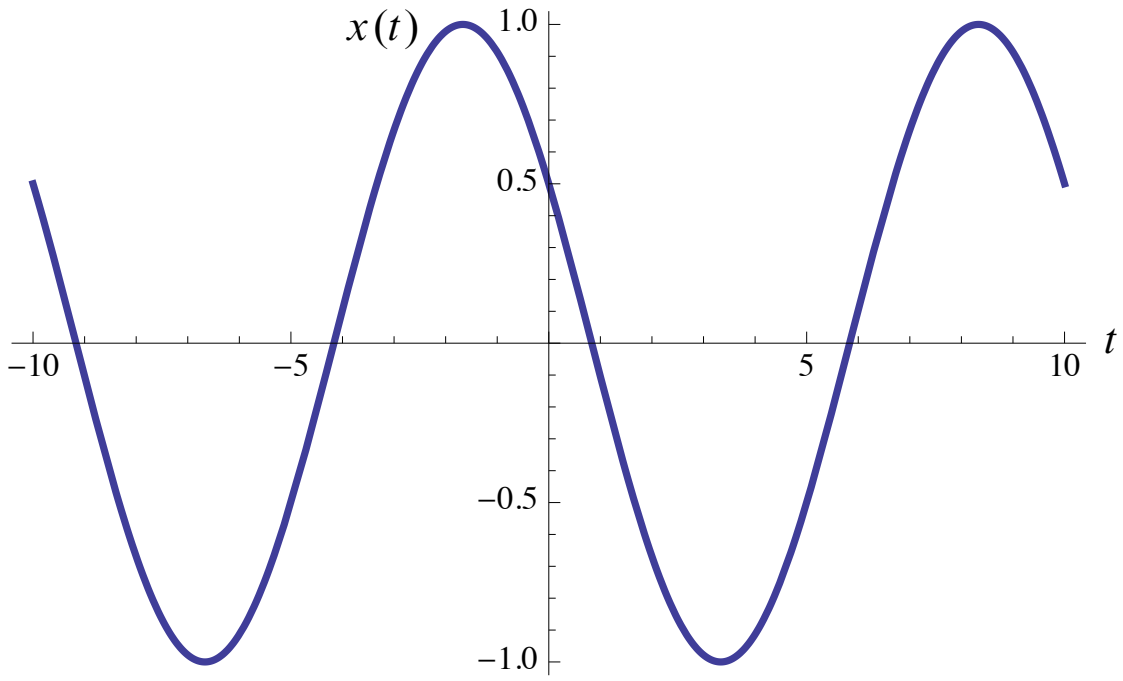
2.2 Review of the Basics

The two aspects of discrete-time signal processing are signals and systems. The review presented here follows the notation established in ECE 5650, *Modern DSP*, and is that of the Oppenheim and Schafer text[1], except ω is used as the continuous-time frequency variable in rad/s and Ω as the discrete-time variable in rad/sample. This is then consistent with the notation of Rice.

2.2.1 Continuous-Time Signals

A function of the continuous-time variable t, typically denoted $x(t) = x_a(t) = x_c(t)$, where the a and c are used to emphasize that the waveform is *analog* or *continuous* in time.

[1] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*, third edition, Prentice-Hall, New Jersey, 2010.



A continuous-time signal plot.

- We can further classify $x(t)$ as being *deterministic* or *random* and either an *energy* signal or a *power* signal
- Random signals will be dealt with in Chapter 3 of the notes
- For $x(t)$ deterministic the energy (Joules), assuming a one ohm impedance level, is

$$E_x = \lim_{T\to\infty} \int_{-T}^{T} |x(t)|^2 \, dt$$

- When the energy is infinite, the signal may be a power signal with power (Watts)

$$P_x = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} |x(t)|^2 \, dt$$

- Sometimes we will be interested in the *time-averaged* autocorrelation function (more in Chapter 3)

$$\text{Energy Signals:} \quad r_x(\tau) = \lim_{T\to\infty} \int_{-T}^{T} x(t)x^*(t-\tau)\,dt$$

$$\text{Power Signals:} \quad \phi_x(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t)x^*(t-\tau)\,dt$$

- The signal $x(t)$ is periodic is we can write

$$\text{Periodic:} \Leftrightarrow \; x(t) = x(t+T_0)$$

for all values of t, where T_0 is the period when it is the smallest value making this true

  - As long as the energy in one period is finite, a periodic signal is a power signal and we can calculate P_x using

$$P_x = \frac{1}{T_0} \int_{T_0} |x(t)|^2\,dt$$

where the integration is over any one period time interval

- The fundamental frequency of this periodic signal is $f_0 = 1/T_0$ Hz or $\omega_0 = 2\pi/T_0$ rads/s

- For $x(t)$ periodic we also represent it via a Fourier series

$$x(t) = \sum_{n=-\infty}^{\infty} X_n e^{jn\omega_o t}$$

$$\text{where} \quad X_n = \frac{1}{T_0} \int_{t_0}^{t_0+T_0} x(t)e^{-jn\omega_0 t}\,dt$$

for any t_0

Special Sequences

- The *impulse* and *unit-step* singularity functions are useful in describing signals of interest

$$\text{Impulse Function:} \quad \int_{-\infty}^{\infty} x(t)\delta(t)\,dt = x(0)$$

$$\text{Unit Step:} \quad u(t) = \int_{-\infty}^{t} \delta(\tau)\,d\tau$$

  - Note that operationally $\delta(t - t_0)$ appears as a sampling function when inside the integral
  - It has unit area and is zero everywhere except at $t = t_0$

- The unit-step may also be defined as

$$u(t) = \begin{cases} 0, & t < 0 \\ \text{undefined}, & t = 0 \\ 1, & t > 0 \end{cases}$$

- Complex and real sinusoids

$$x_1(t) = e^{j(\omega_0 t + \phi)} = \cos(\omega_0 t + \phi) + j\,\sin(\omega_0 t + \phi)$$
$$x_2(t) = \cos(\omega_0 t + \phi)$$

2.2.2 Discrete-Time Signals

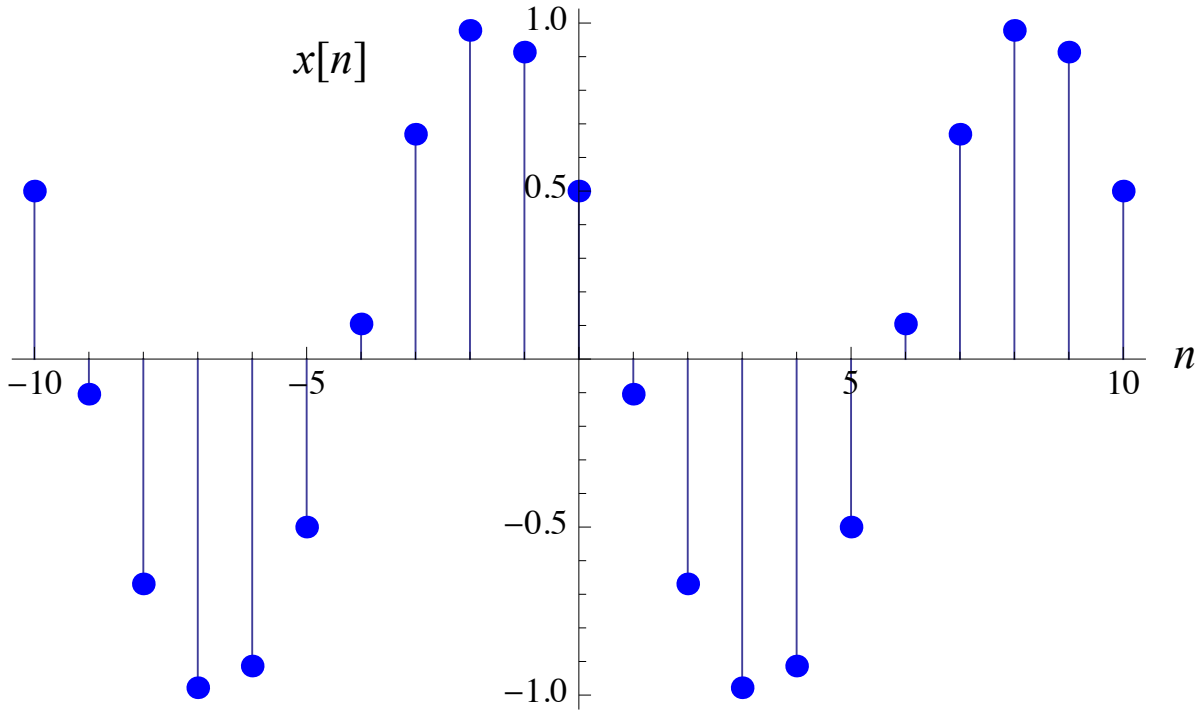
- A one-dimensional discrete-time signal is a real or complex sequence, denoted as $x[n] = x(n)$, where n is an integer valued independent variable

- We frequently assume that $x[n]$ is obtained by uniform sampling a continuous-time or analog signal $x_a(t)$, i.e.,

$$x[n] = x_a(nT_s)$$

where T_s is the sampling period and $f_s = 1/T_s$ is the sampling rate in samples per second

- An analog-to-digital converter is always used to obtain $x[n]$



A discrete-time signal *stem* plot.

- As in the continuous-time case, we also can classify signals as energy or power, except here the notion of Joules and Watts is

more vague, since the signal/sequence resides on a computer

$$E_x = \lim_{N\to\infty} \sum_{n=-N}^{N} |x[n]|^2$$

$$P_x = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |x[n]|^2$$

- A discrete-time signal is periodic for the smallest integer N_0 such that

$$\text{Periodic} \;\Leftrightarrow\; x[n] = x[n+N_0]$$

- The fundamental frequency can be defined as $\Omega_0 = 2\pi/N_0$ rad/sample
- Note that a sampled periodic signal $x[n] = x(nT_s) = x(nT_s + T_0)$ is not necessarily a periodic sequence. why?

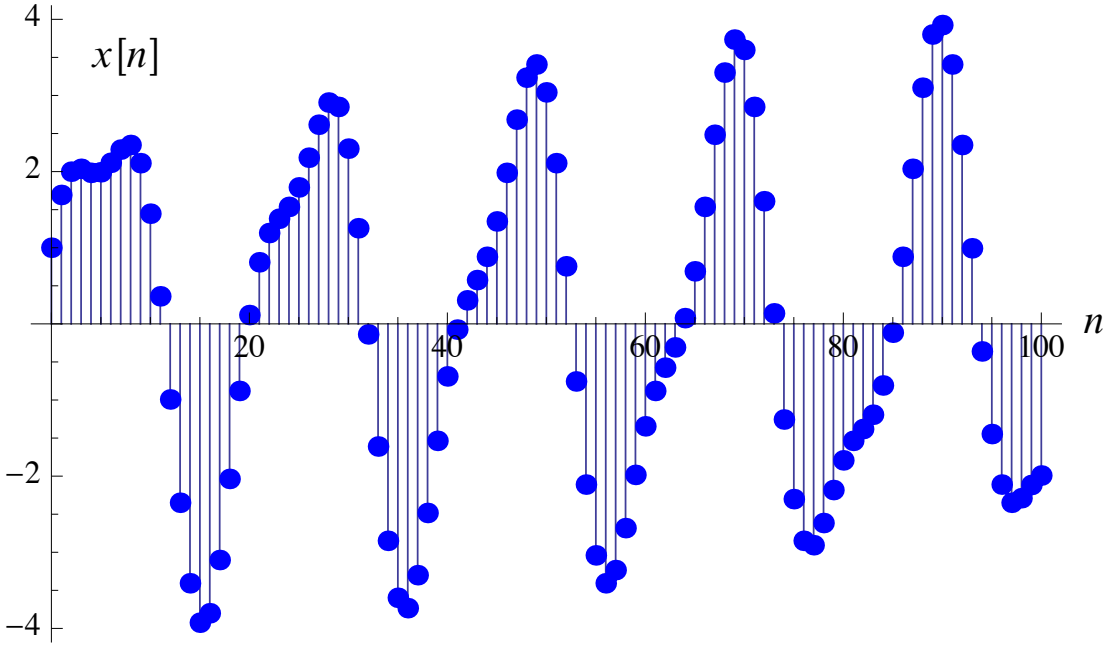
Example 2.1: Power in the Sum of Two Sinusoidal Sequences

- Consider

$$x[n] = \cos(2\pi n/10) + 3\sin(2\pi n/21)$$

- We wish to calculate the signal power, P_x experimentally and theoretically
- Notice that the signal does not appear to be periodic (actually it is the LCM$[10, 21] = 210$) or at least the period is rather large

```
SeqPlot[Table[Cos[2 π n / 10] + 3 Sin[2 π n / 21], {n, 0, 100}], 0]
```



Sum of two sinusoids sequence plot (Mathematica).

- In this case we use Mathematica, but MATLAB could also be used
- For continuous-time sinusoids we know that power in each sinusoid is $A^2/2$, so we expect that in the discrete-time case the results are similar, thus we expect that theoretically

$$P_x = \frac{1^2 + 3^3}{2} = \frac{10}{2} = 5$$

- The experimental calculation we can not let $N \to \infty$, so we must be content with a finite, but large value
- Knowing that the period is 210 we expect that using one period will be the most efficient, but general this my not always be possible, so we may favor using a much larger value for N and take an approximate answer

```
x = Table[Cos[2 π n / 10] + 3 Sin[2 π n / 21], {n, 0, 10 000}];
```

Sequence length is 10,001 points

```
Table[1/N Sum[x[[n]]^2, {n, 1, N}], {N, {10, 100, 210, 1000, 5000}}] // N
{3.96031, 4.94021, 5., 4.98223, 4.99781}
```

Return numerical

Power calculations using $N = 10, 100, 210, 1000, 5000$.

- We will study further in Chapter 3 of the notes how tools such as Mathematica and MATLAB have supplied statiscal functions that perform the equivalent of an average power calculation, i.e., `Variance[ ]` or `var( )`

```
Variance[x] // N

5.02392
```

Power calculations using `Variance[ ]` on 10,001 points.

Special Sequences

- The *unit sample* or impulse sequence

$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & \text{otherwise} \end{cases}$$

  - An sequence can be expressed as a linear combination of time-shifted impulses, i.e.,

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

    Note: Time shifting, $\delta[n-k]$, moves the delta to location $n = k$

- The *unit step* sequence is denoted

$$u[n] = \begin{cases} 1, & n \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

  - The unit step and unit sample sequences are related through discrete-time integral/differential type relationships

$$u[n] = \sum_{k=-\infty}^{n} \delta[k]$$
$$\delta[n] = u[n] - u[n-1]$$

- Complex and real sinusoids

$$x_1[n] = e^{j(\Omega_0 n + \phi)} = \cos(\Omega_0 n + \phi) + j \sin(\Omega_0 n + \phi)$$
$$x_2[n] = \cos(\Omega_0 n + \phi)$$

- Note that it may be that $\omega_0 t$ when sampled with period T or rate $f_s = 1/T$ results in

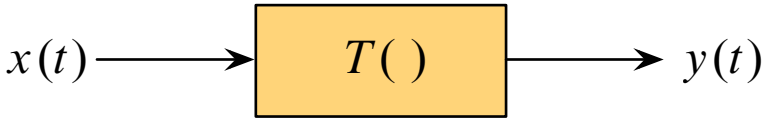
$$\omega_0 \cdot nT = \omega_o T \cdot n$$

  which implies that

$$\omega_0 T = \Omega_0 \text{ or } \omega_o = \Omega_0 / T = \Omega_0 f_s$$

2.2.3 Continuous-Time Systems

Continuous-time systems map the input $x(t)$ to the output $y(t)$



A continuous-time system transforming $x(t)$ to $y(t)$.

- In this course we are for the most part interested in *linear time invariant* (LTI) systems
- In terms of signal processing electronics, circuits composed of resistors, capacitors, inductors, and active elements (amplifiers) that are operating in their linear region, LTI operator $L[\,]$ implies that

$$\begin{aligned} y(t) = L\big[x(t)\big] &\Rightarrow y(t-t_0) = L\big[x(t-t_0)\big] \\ y_i(t) = L\big[x_i(t)\big] &\Rightarrow a y_1(t) + b y_2(t) = \\ &\qquad L\big[a x_1(t) + b x_2(t)\big] \end{aligned}$$

  for any t_0, any $x_1(t)$ and $x_2(t)$, and any a_1 and a_2
- We also need nonlinear systems to perform signal multiplication and take advantage of (near) instantaneous nonlinearities, i.e.,

$$y(t) = x_1(t) \cdot x_2(t)$$

$$y(t) = \sum_{n=0}^{N} a_n x^n(t)$$

- Filter operations are critical to operation of all wireless devices, and fortunately can be considered LTI in most applications

- **Definition**: When we input $\delta(t)$ to continuous-time LTI filter H, at rest, the output is $h(t)$, defined as the *impulse response*

- It can be shown that for an LTI system with impulse response $h(t)$, the output is given by the convolution integral

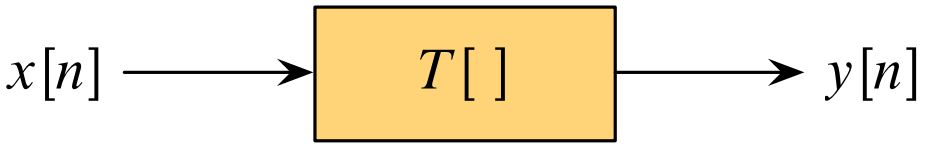
$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\lambda)h(t-\lambda)\,d\lambda = \int_{-\infty}^{\infty} x(t-\lambda)h(\lambda)\,d\lambda$$

- Lumped element LTI systems can be described by a linear constant coefficient differential equation, i.e.,

$$\sum_{n=0}^{N} a_n \frac{d^{(n)}y(t)}{dt^n} = \sum_{n=0}^{M} b_n \frac{dx^{(n)}(t)}{dt^n}$$

2.2.4 Discrete-Time Systems

Similarly, a discrete-time system maps the input sequence to the output sequence according to some rule set.



A discrete-time system transforming $x[n]$ to $y[n]$.

- Examples:

$$y_1[n] = \sum_{k=0}^{N} \alpha_k x^k[n]$$

$$y_2[n] = (1-\beta) \sum_{k=0}^{N} \beta^k x[n-k],\ 0 < \beta < 1$$

$$y_3[n] = \beta y_3[n-1] + (1-\beta)x[n-1],\ 0 < \beta < 1$$

- **Definition**: For a system initially at rest, the system output $y[n]$ to input $x[n] = \delta[n]$ is again defined as the impulse response; the notation of the impulse response is $h[n] = T\{\delta[n]\}$

Linearity and Time (shift) Invariance

- *Linearity* says that superposition holds, that is $x_1[n] \xrightarrow{L} y_1[n]$ and $x_2[n] \xrightarrow{L} y_2[n]$,

$$\begin{aligned} L\{ax_1[n] + bx_2[n]\} &= aL\{x_1[n]\} + bL\{x_2[n]\} \\ &= ay_1[n] + by_2[n] \end{aligned}$$

  for any constants a and b

- Using linearity we can write that

$$\begin{aligned} y[n] &= L\left\{ \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \right\} \\ &= \sum_{k=-\infty}^{\infty} x[k] L\{\delta[n-k]\} \\ &= \sum_{k=-\infty}^{\infty} x[k] h_k[n] \end{aligned}$$

using the fact that for any sequence $x[n]$ we can always write

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

and using $h_k[n]$ to denote the system response to a delayed by k impulse

- *Time-invariance* states that the system properties do not change with time, so that if $y[n] = T\{x[n]\}$ then for any $x[n]$ and any time offset k, $y_1[n] = T\{x[n-k]\} \overset{\text{also}}{=} y[n-k]$
- Combining the definition of impulse response with the time invariance property results in

$$\begin{aligned} h[n] &= L\{\delta[n]\} \\ \Rightarrow\ h[n-k] &= L\{\delta[n-k]\} \end{aligned}$$

for any value k

- The linearity result that stopped with $L\{\delta[n-k]\} = h_k[n]$ can now be completed

$$\begin{aligned} y[n] &= \sum_{k=-\infty}^{\infty} x[k]h[n-k] = \sum_{k=-\infty}^{\infty} x[n-k]h[k] \\ &= x[n] * h[n] = h[n] * x[n] \end{aligned}$$

which is known as the *convolution sum*

Causality

- A system is causal if the output $y[n]$ up to some time n_0 depends only upon the input $x[n]$ for $n \leq n_0$

- Simply stated, the output depends only on past and present values of the input; no predicting the future

Stability

- The definition of *bounded-input/bounded-output* (BIBO) stability says that the output $|y[n]| \leq B < \infty$ for any bounded input $|x[n]| \leq A < \infty$
- When the system is also linear and time-invariant (LTI), we have the theorem which states BIBO stability as

$$\sum_{n=-\infty}^{\infty} |h[n]| < \infty$$

  where $h[n]$ is the LTI system impulse response

Example 2.2: First-Order LCCDE

- Consider the first-order linear constant coefficient difference equation

$$y[n] = ay[n-1] + bx[n] + cx[n-1] + dx[n+1]$$

- If $d \neq 0$ the system is not causal, as $x[n+1]$ is a future value of the input relative to the output $y[n]$
- Suppose now that $c = d = 0$, then the system impulse response is (why?)

$$h[n] = b \cdot a^n u[n]$$

- Using the LTI system stability theorem, we have

$$\sum_{n=0}^{\infty} |ba^n| = \frac{|b|}{1-a}, \quad |a| < 1$$

  with the aid of the infinite geometric series sum formula

- Hence, the system is stable provided b is finite and $|a| < 1$

2.2.5 Difference Equations and LTI Filters

- LTI systems most frequency encountered are those described by a constant coefficient difference equation (LCCDE) of the form

$$y[n] + \sum_{k=1}^{p} a_k y[n-k] = \sum_{k=0}^{q} b_k x[n-k]$$

- The filter coefficients $\{a_k\}$ and $\{b_k\}$ completely describe the system

- When the LCCDE is rearranged to solve for $y[n]$ in terms of past and present inputs, and past outputs we have

$$y[n] = -\sum_{k=1}^{p} a_k y[n-k] + \sum_{k=0}^{q} b_k x[n-k]$$

- In MATLAB the operation of LCCDE filtering is efficiently done using

```
>> a = [define  the feedback coefficients];
>> b = [define the feedforward coefficients];
>> x = [define the input signal vector]
>> y = filter(b,a,x); % produce the output vector
```

  - Since this filter involves feedback terms, the impulse response is likely infinite (barring pole/zero cancellations), thus the system is termed *infinite impulse response* (IIR)

- When the feedback coefficients are set to zero, we obtain a finite impulse response (FIR) filter of the form

$$y[n] = \sum_{k=0}^{q} b_k x[n-k]$$

- When only the present input is utilized we have an all pole filter of the form

$$y[n] = -\sum_{k=1}^{p} a_k y[n-k] + x[n]$$

2.2.6 Frequency Domain Characterizations

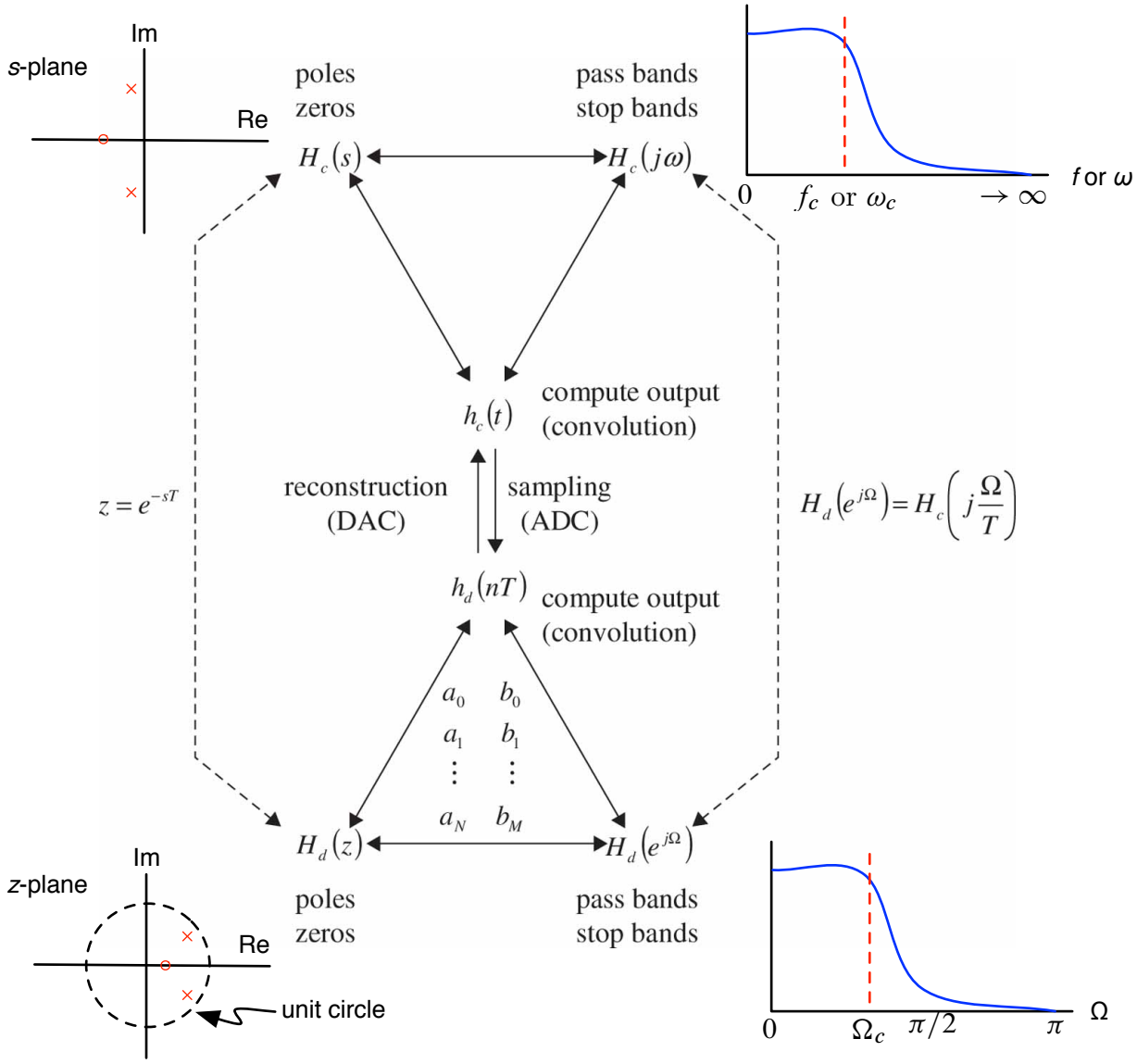
Frequency domain characterization, including spectral analysis, are vital tools for communications modeling. Both continuous-time and discrete-time have three analysis domains: *time*, *complex frequency* (s or z), and *frequency*.

- In the time domain we have the impulse response $h(t)$ or $h[n]$ which relates the input to the output via the convolution integral or convolution sum

- For LTI systems realizable via constant coefficient differential or difference equations, we can transform via the *Laplace* transform to $H(s)$ or via the z-transform to $H(z)$

– In the complex frequency domain we have rational transfer functions with poles and zeros

– The convolution has been replaced by polynomial multiplication and returning to the time domain requires partial fraction expansion and table lookup



Relationships between the three domains and continuous/discrete.

- To characterize the frequency selectivity of an LTI system we use the frequency domain via Fourier transformation of $h(t)$ or

$h[n]$

  - For rational $H(s)$ and $H(z)$ this amounts to $s \rightarrow j\omega = j2\pi f$ or $z \rightarrow e^{j\Omega}$
  - We this the magnitude and phase of $H(j\omega)$ or $H(e^{j\Omega})$
  - For signals $x(t)$ and $x[n]$ we obtain the spectral content by Fourier transforming $x(t) \Leftrightarrow X(j\omega)$ or $x[n] \Leftrightarrow X(e^{j\Omega})$
  - In the discrete-time domain the Fourier transform is know as the *discrete-time Fourier transform* (DTFT)

- When we uniformly sample $x(t)$ with spacing T to produce $x[n]$, the Laplace and z-transforms are related via $z_0 = e^{s_0 T}$
- Similarly the Fourier transforms become related via

$$H_d\left(e^{j\Omega}\right) = H_c\left(j\frac{\Omega}{T}\right)$$

  - The sampling theorem will be reviewed shortly

The Laplace Transform

Formally the two-sided Laplace transform and inverse Laplace transform are defined as

$$X(s) = \int_{-\infty}^{\infty} x(t)e^{-st}\,dt$$
$$x(t) = \frac{1}{j2\pi}\oint X(s)e^{st}\,ds$$

Laplace transform properties.

Property	Signal	Laplace Transform	ROC
	$x(t)$	$X(s)$	R_x
	$y(t)$	$Y(s)$	R_y
Linearity	$ax(t) + by(t)$	$aX(s) + bY(s)$	at least $R_x \cap R_y$
Time Shifting	$x(t - t_0)$	$e^{-st_0}X(s)$	R_x
Time Scaling	$x(at)$	$\frac{1}{\lvert a\rvert}X\left(\frac{s}{a}\right)$	all s for which s/a is in R_x
Conjugation	$x^*(t)$	$X^*(s^*)$	R_x
Convolution	$x(t) * y(t)$	$X(s)Y(s)$	at least $R_x \cap R_y$
Differentiation	$\frac{d}{dt}x(t)$	$sX(s)$	at least R_x
Integration	$\int_{-\infty}^{t} x(\tau)d\tau$	$\frac{1}{s}X(s)$	at least $R_x \cap \{\text{Real}\{s\} > 0\}$
Initial Value Theorem[a]		$\lim_{t\to 0^+} x(t) = \lim_{s\to\infty} sX(s)$	
Final Value Theorem[b]		$\lim_{t\to\infty} x(t) = \lim_{s\to 0} sX(s)$	

[a] The initial value theorem is valid for signals $x(t)$ that satisfy the following conditions: $x(t) = 0$ for $t < 0$ and $x(t)$ contains no impulses or higher-order singularities at $t = 0$.
[b] The final value theorem is valid for signals $x(t)$ that satisfy the following conditions: $x(t) = 0$ for $t < 0$ and $x(t)$ has a finite limit as $t \to \infty$.

- The *region of convergence* (ROC) is important when dealing with the two-sided transform, as it specifies values of s for which the integral converges (like the two-sided z-transform)
- A *contour integral* is formally required to obtain the inverse transform, but in practice we use *table lookup*
- For LTI systems defined by a constant coefficient differential equation, the Laplace transform of both sides allows us to form the *system function* as the ratio $Y(s)/X(s) = H(s)$

$$H(s) = \frac{Y(s)}{X(s)} = \frac{b_M s^M + \cdots + b_1 s + b_0}{a_N s^N + \cdots + a_1 s + a_0} = \frac{B(s)}{A(s)}$$

- The roots of $B(s)$ are the *zeros* of $H(s)$ and the roots of $A(s)$ are the *poles* of $H(s)$
- The inverse Laplace transform of $H(s)$ yields the impulse response $h(t)$

Laplace transform pairs.

$x(t)$	$X(s)$	ROC
$\delta(t)$	1	all s
$u(t)$	$\frac{1}{s}$	Real $\{s\} > 0$
$\frac{t^{n-1}}{(n-1)!}u(t)$	$\frac{1}{s^n}$	Real $\{s\} > 0$
$e^{-at}u(t)$	$\frac{1}{s+a}$	Real $\{s\} > -a$
$\frac{t^{n-1}}{(n-1)!}e^{-at}u(t)$	$\frac{1}{(s+a)^n}$	Real $\{s\} > 0$
$\delta(t-T)$	e^{-sT}	all s
$\cos(\omega_0 t)u(t)$	$\frac{s}{s^2+\omega_0^2}$	Real $\{s\} > 0$
$\sin(\omega_0 t)u(t)$	$\frac{\omega_0}{s^2+\omega_0^2}$	Real $\{s\} > 0$
$e^{-at}\cos(\omega_0 t)u(t)$	$\frac{s+a}{(s+a)^2+\omega_0^2}$	Real $\{s\} > -a$
$e^{-at}\sin(\omega_0 t)u(t)$	$\frac{\omega_0}{(s+a)^2+\omega_0^2}$	Real $\{s\} > -a$
$\frac{d^n\delta(t)}{dt}$	s^n	all s
$\underbrace{u(t) * \cdots * u(t)}_{n \text{ times}}$	$\frac{1}{s^n}$	Real $\{s\} > 0$

- For example, the step response of $H(s) = 1/(s+a)$ is just

$$y_s(t) = \mathcal{L}^{-1}\left[\frac{1}{s+1}\cdot\frac{1}{s+a}\right] = \mathcal{L}^{-1}\left[\frac{a-1}{s+1}\right] + \mathcal{L}^{-1}\left[\frac{1-a}{s+a}\right]$$
$$= (a-1)u(t) + (1-a)e^{-at}u(t)$$

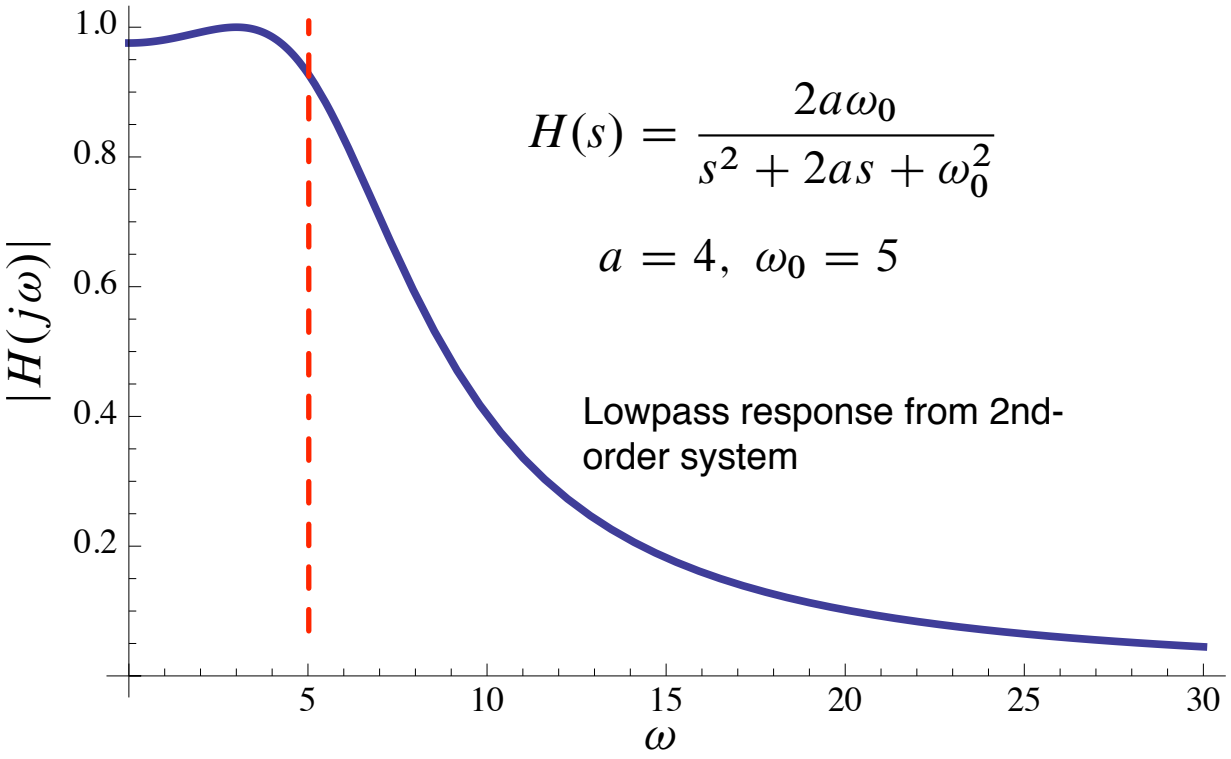
The Continuous-Time Fourier Transform

Communications theory relies heavily on the use of the Fourier transform and in particular spectral analysis. The continuous-time Fourier transform, with frequency variable ω in rad/s, is defined as

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

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

- For $x(t)$ a true energy signal (or impulse response) it also follows that $X(j\omega)$ can be obtained from the corresponding Laplace transform by letting $s \rightarrow j\omega$
- Consider a second-order lowpass system:



Frequency response from $H(s)$.

- In communications systems it is more common to work with the frequency variable $f = \omega/(2\pi)$ in Hz, and hence work with the Fourier transform definition

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}\,dt$$
$$x(t) = \int_{-\infty}^{\infty} X(f)e^{j2\pi ft}\,df$$

- Tables of transform properties and pairs are given below

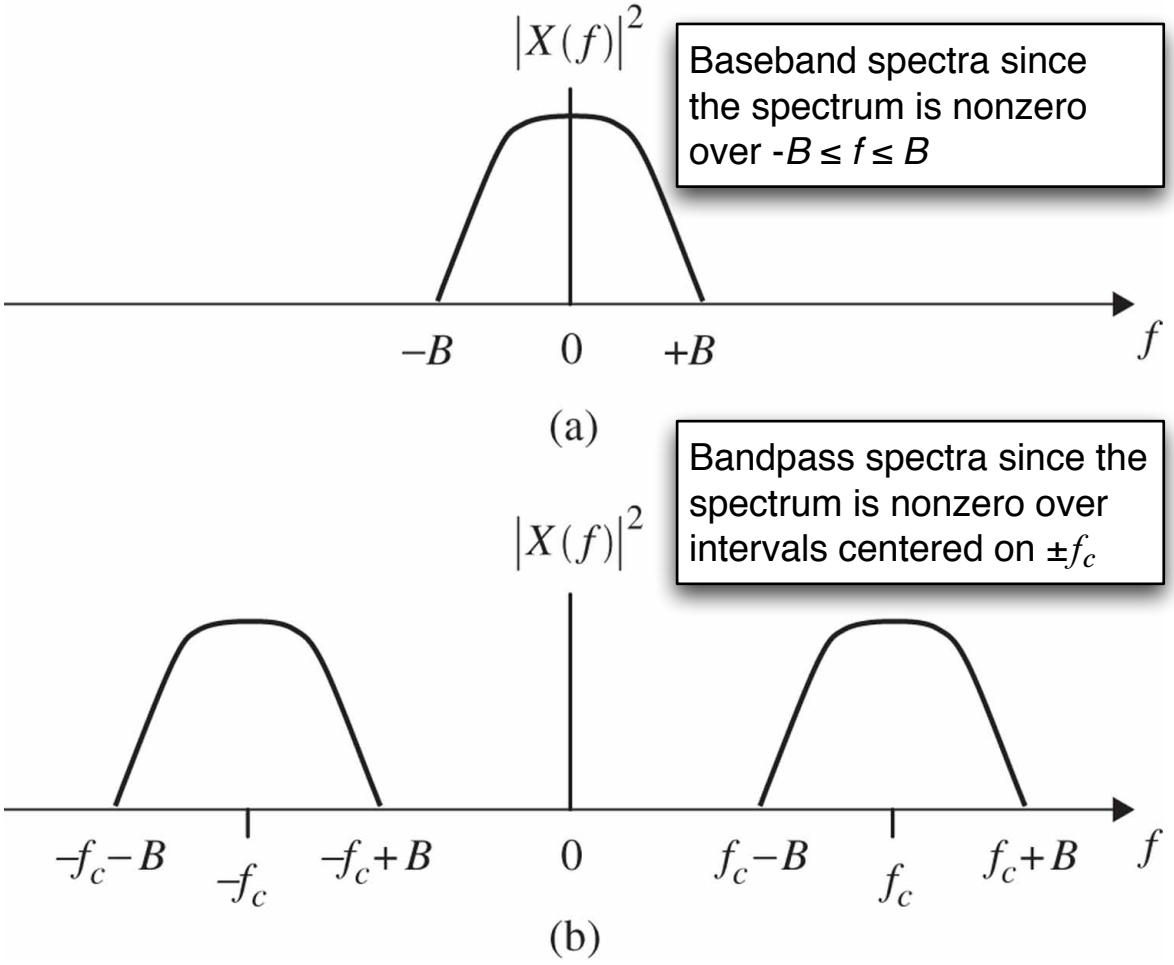
Fourier transform properties in ω and f.

Property	Signal	Fourier Transform in ω	Fourier Transform in f
	$x(t)$	$X(j\omega)$	$X(f)$
	$y(t)$	$Y(j\omega)$	$Y(f)$
Linearity	$ax(t)+by(t)$	$aX(j\omega)+bY(j\omega)$	$aX(f)+bY(f)$
Time Shifting	$x(t-t_0)$	$e^{-j\omega t_0}X(j\omega)$	$e^{-j2\pi ft_0}X(f)$
Time Scaling	$x(at)$	$\frac{1}{\lvert a\rvert}X\left(\frac{j\omega}{a}\right)$	$\frac{1}{\lvert a\rvert}X\left(\frac{f}{a}\right)$
Conjugation	$x^*(t)$	$X^*(-j\omega)$	$X^*(-f)$
Convolution	$x(t)*y(t)$	$X(j\omega)Y(j\omega)$	$X(f)Y(f)$
Differentiation	$\frac{d}{dt}x(t)$	$j\omega X(j\omega)$	$j2\pi fX(f)$
Integration	$\int_{-\infty}^{t} x(\tau)d\tau$	$\frac{1}{j\omega}X(j\omega)+\pi X(0)\delta(\omega)$	$\frac{1}{j2\pi f}X(f)+\frac{1}{2}X(0)\delta(f)$
Frequency Shifting ($\omega_0 = 2\pi f_0$)	$e^{j\omega_0 t}x(t)$	$X(j(\omega-\omega_0))$	$X(f-f_0)$
Multiplication	$x(t)y(t)$	$\frac{1}{2\pi}X(j\omega)*Y(j\omega)$	$X(f)*Y(f)$
Parseval's Theorem	$\int_{-\infty}^{\infty}\lvert x(t)\rvert^2dt = \frac{1}{2\pi}\int_{-\infty}^{\infty}\lvert X(j\omega)\rvert^2d\omega = \int_{-\infty}^{\infty}\lvert X(f)\rvert^2df$		

Fourier transform pairs.

$x(t)$	$X(j\omega)$	$X(f)$
$\delta(t)$	1	1
$\delta(t-t_0)$	$e^{-j\omega t_0}$	$e^{-j2\pi f t_0}$
$u(t)$	$\frac{1}{j\omega}+\pi\delta(\omega)$	$\frac{1}{j2\pi f}+\frac{1}{2}\delta(f)$
$\frac{t^{n-1}}{(n-1)!}e^{-at}u(t)$	$\frac{1}{(a+j\omega)^n}$	$\frac{1}{(a+j2\pi f)^n}$
$\sum_{k=-\infty}^{\infty} a_k e^{jk\omega_0 t}$ $(\omega_0=2\pi f_0)$	$2\pi\sum_{n=-\infty}^{\infty} a_k\delta(\omega-k\omega_0)$	$\sum_{n=-\infty}^{\infty} a_k\delta(f-kf_0)$
$\sum_{k=-\infty}^{\infty}\delta(t-kT)$	$\frac{2\pi}{T}\sum_{n=-\infty}^{\infty}\delta\left(\omega-\frac{2\pi n}{T}\right)$	$\frac{1}{T}\sum_{n=-\infty}^{\infty}\delta\left(f-\frac{n}{T}\right)$
1	$2\pi\delta(\omega)$	$\delta(f)$
$\begin{cases}1 & \lvert t\rvert<T\\ 0 & \lvert t\rvert>T\end{cases}$	$2T\frac{\sin(\omega T)}{\omega T}$	$2T\frac{\sin(2\pi fT)}{2\pi fT}$
$e^{j\omega_0 t}$ $(\omega_0=2\pi f_0)$	$2\pi\delta(\omega-\omega_0)$	$\delta(f-f_0)$
$\cos(\omega_0 t)$ $(\omega_0=2\pi f_0)$	$\pi\delta(\omega-\omega_0)+\pi\delta(\omega+\omega_0)$	$\frac{1}{2}\delta(f-f_0)+\frac{1}{2}\delta(f+f_0)$
$\sin(\omega_0 t)$ $(\omega_0=2\pi f_0)$	$\frac{\pi}{j}\delta(\omega-\omega_0)-\frac{\pi}{j}\delta(\omega+\omega_0)$	$\frac{1}{j2}\delta(f-f_0)-\frac{1}{j2}\delta(f+f_0)$
$\exp\{-a\lvert t\rvert\}$	$\frac{2a}{a^2+\omega^2}$	$\frac{2a}{a^2+(2\pi f)^2}$
$\exp\left\{-\pi t^2\right\}$	$\exp\left\{-\pi\left(\frac{\omega}{2\pi}\right)^2\right\}$	$\exp\left\{-\pi f^2\right\}$

- As we study digital modulation schemes, we will be considering both *baseband* and *bandpass* signals



Baseband and bandpass spectra.

2.2.7 The z-Transform

- The z-transform generalizes the DTFT by considering the entire complex plane for the transform domain

- The two-sided z-transform is defined as

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n}$$

where z is a complex variable

- Note that when we write $z = re^{j\omega}$, and then set $r = 1$ we have

$$X(e^{j\omega}) = X(z)\Big|_{z=e^{j\omega}} = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n},$$

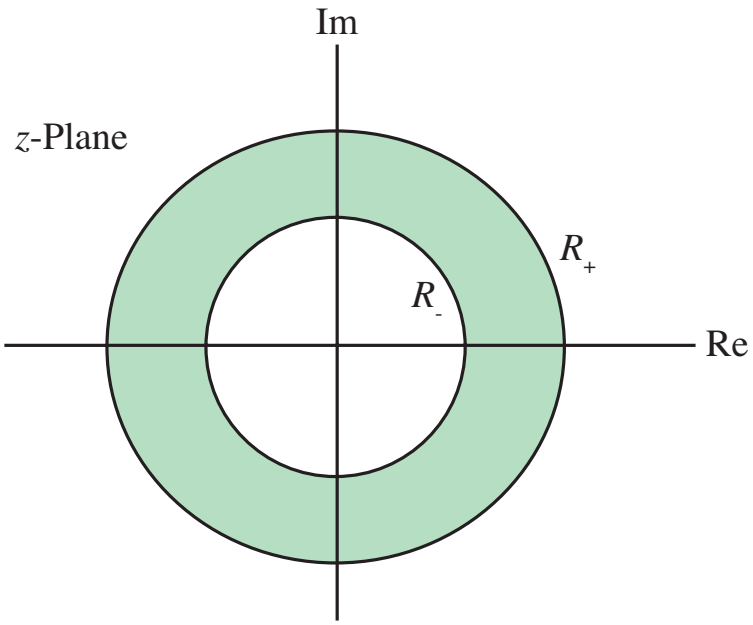
  thus the DTFT is obtained by evaluating the z-transform around the unit circle

- The z-transform does not in general converge over the entire z-plane

- The region where it does converge is called the *region of convergence* (ROC)

- Since convergence is in terms of the absolute sum, the ROC is in general and annulus of the form

$$R_- < |z| < R_+$$



Annular ROC

- If $x[n]$ is a finite length sequence (finite support), then

$$X(z) = \sum_{n=N_1}^{N_2} x[n]z^{-n}$$

  - For $[N_1, N_2]$ straddling $n = 0$, $X(z)$ will contain both positive and negative powers of z, thus making the ROC exclude $z = 0$ and $z = \infty$

- If $x[n]$ is right-sided of the form $a^n u[n]$, we find that

$$X(z) = \sum_{n=0}^{\infty} a^n \, z^{-n} = \frac{1}{1 - az^{-1}}, \quad |z| < |a|$$

- On the other hand if $x[n]$ is left-sided of the form $-a^n u[-1 - n]$, we find that

$$X(z) = -\sum_{n=-\infty}^{-1} a^n z^{-n} = \frac{1}{1 - az^{-1}}, \quad |z| > |a|$$

  - Note that the z-transform of both sequences is identical with the exception of the ROC

- There are many z-transform theorems and properties, for example the convolution theorem

$$y[n] = x[n] * h[n] \overset{\mathcal{Z}}{\longleftrightarrow} X(z)H(z) = Y(z)$$

  with $R_Y = R_X \cap R_H$ unless there is a pole-zero cancellation

A Short Table of ZT Theorems

Property	Signal	z-transform	ROC
	$x(n)$	$X(z)$	R_x
	$y(n)$	$Y(z)$	R_y
Linearity	$ax(n) + by(n)$	$aX(z) + bY(z)$	at least $R_x \cap R_y$
Time Shifting	$x(n - n_0)$	$z^{-n_0}X(z)$	R_x
Time Scaling (Upsampling)	$\begin{cases} x(n/K) & n \text{ is a multiple of K} \\ 0 & \text{otherwise} \end{cases}$	$X\left(z^K\right)$	$R_x^{1/K}$
Conjugation	$x^*(n)$	$X^*\left(z^*\right)$	R_x
Convolution	$x(n) * y(n)$	$X(z)Y(z)$	at least $R_x \cap R_y$
First Difference	$x(n) - x(n-1)$	$(1 - z^{-1})X(z)$	at least $R_x \cap \{\|z\| > 0\}$
Accumulation	$\sum_{k=-\infty}^{n} x(k)$	$\frac{1}{1 - z^{-1}}X(z)$	at least $R_x \cap \{\|z\| > 1\}$
Initial Value Theorem	$x(0) = \lim_{z\to\infty} X(z)$		

A Short Table of ZT Pairs

Signal	Transform	ROC
$\delta(n)$	1	all z
$u(n)$	$\frac{1}{1 - z^{-1}}$	$\|z\| > 1$
$a^n u(n)$	$\frac{1}{1 - az^{-1}}$	$\|z\| > \|a\|$
$na^n u(n)$	$\frac{az^{-1}}{\left(1 - az^{-1}\right)^2}$	$\|z\| > \|a\|$
$\delta(n - m)$	z^{-m}	all z[†]
$\cos(\Omega_0 n)\, u(n)$	$\frac{1 - \cos(\Omega_0)z^{-1}}{1 - [2\cos(\Omega_0)]z^{-1} + z^{-2}}$	$\|z\| > 1$
$\sin(\Omega_0 n)\, u(n)$	$\frac{\sin(\Omega_0)z^{-1}}{1 - [2\cos(\Omega_0)]z^{-1} + z^{-2}}$	$\|z\| > 1$
$r^n \cos(\Omega_0 n)\, u(n)$	$\frac{1 - r\cos(\Omega_0)z^{-1}}{1 - [2r\cos(\Omega_0)]z^{-1} + z^{-2}}$	$\|z\| > r$
$r^n \sin(\Omega_0 n)\, u(n)$	$\frac{r\sin(\Omega_0)z^{-1}}{1 - [2r\cos(\Omega_0)]z^{-1} + z^{-2}}$	$\|z\| > r$

[†]except 0 if $m > 0$ or ∞ if $m > 0$

- When we consider the impulse response of an LTI filter, the z-transform produces the *system function*

$$H(z) = \sum_{n=-\infty}^{\infty} h[n]z^n$$

- For an FIR filter of order M and coefficient set $\{b_k\}$, the system function is a polynomial in z^{-1} of the form

$$H(z) = \sum_{k=0}^{M} b_k z^{-k} = b_0 \prod_{k=1}^{M} \left(1 - z_k z^{-1}\right),$$

  where the roots, denoted by the $\{z_k\}$ are the *zeros* of the filter, thus an FIR filter is composed of only zeros

  - If we factor $H(z)$ into positive powers of z we find that there are also M poles at $z = 0$

- For an IIR filter, composed of feed forward and feedback terms in the LCCDE, the system function is obtained by z-transforming both sides of the difference equation using the delay theorem

$$\mathcal{Z}\left\{\sum_{k=0}^{N} a_k y[n-k]\right\} = \mathcal{Z}\left\{\sum_{k=0}^{M} b_k x[n-k]\right\}$$

$$Y(z)\sum_{k=0}^{N} a_k z^{-k} = X(z)\sum_{k=0}^{M} b_k z^{-k}$$

  which becomes

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}} = b_0 \frac{\prod_{k=1}^{M} \left(1 - z_k z^{-1}\right)}{\prod_{k=1}^{N} \left(1 - p_k z^{-1}\right)}$$

Note: We have assumed that $a_0 = 1$

- The roots in the denominator are the *poles* of $H(z)$
- A causal IIR filter must have $|p_k| < 1$ for stability
- The filter order is N nonzero poles and M nonzero zeros
- Even though $N \neq M$ in general, the number of poles and zeros in $H(z)$ will be equal, with balanced achieved via poles or zeros at $z = 0$
- If $M = 0$ we have an *all-pole* filter which has N zeros at $z = 0$

- When working with the z-transform, and later the discrete-time Fourier transform, geometric series sum formulas and related forms, are useful:

Sum	Condition
$\sum_{k=N_1}^{N_2} a^k = \frac{a^{N_1} - a^{N_2+1}}{1-a}$	none
$\sum_{k=0}^{n} a^k = \frac{1-a^{n+1}}{1-a}$	none
$\sum_{k=0}^{n} k a^k = \frac{a\left[1-(n+1)a^n + n a^{n+1}\right]}{(1-a)^2}$	none
$\sum_{n=0}^{N-1} k^2 = \frac{1}{6} N(N-1)(2N-1)$	none
$\sum_{k=0}^{\infty} a^k = \frac{1}{1-a}$	$\|a\| < 1$
$\sum_{k=0}^{\infty} k a^k = \frac{a}{(1-a)^2}$	$\|a\| < 1$

Example 2.3: Find $y[n]$ given $x[n]$ and $H(z)$

The input to an LTI system is given by

$$x[n] = 4\left(\frac{1}{2}\right)^n u[n]$$

The system function is

$$H(z) = \frac{1+\frac{1}{3}z^{-1}}{1-\frac{1}{4}z^{-1}-\frac{3}{8}z^{-2}};$$

Denominator roots analysis:

$$\frac{\frac{1}{4}\pm\sqrt{\frac{1}{16}+\frac{3}{2}}}{2} = \frac{1}{8}\pm\frac{5}{8} \Rightarrow \text{roots} = -\frac{1}{2}, \frac{3}{4}$$

Find the system output $y[n]$ using z-transform techniques.

- The denominator of $H(z)$ has roots -1/2 and 3/4
- The z-transform of $x[n]$ is $4/(1-(1/2)z^{-1})$
- Writing out the product $Y(z) = X(z)H(z)$, we have

$$Y(z) = \frac{4\left(1+\frac{1}{3}z^{-1}\right)}{\left(1-\frac{1}{2}z^{-1}\right)\left(1+\frac{1}{2}z^{-1}\right)\left(1-\frac{3}{4}z^{-1}\right)} = \frac{A_1}{1-\frac{1}{2}z^{-1}} + \frac{A_2}{1+\frac{1}{2}z^{-1}} + \frac{A_3}{1-\frac{3}{4}z^{-1}}$$

$$A_1 = \left.\frac{4\left(1+\frac{1}{3}z^{-1}\right)}{\left(1+\frac{1}{2}z^{-1}\right)\left(1-\frac{3}{4}z^{-1}\right)}\right|_{z^{-1}=2} = \frac{4\left(1+\frac{2}{3}\right)}{(1+1)\left(1-\frac{3}{2}\right)} = \frac{4\cdot\frac{5}{3}}{2\cdot-\frac{1}{2}} = -\frac{20}{3} = -6.667$$

$$A_2 = \left.\frac{4\left(1+\frac{1}{3}z^{-1}\right)}{\left(1-\frac{1}{2}z^{-1}\right)\left(1-\frac{3}{4}z^{-1}\right)}\right|_{z^{-1}=-2} = \frac{4\left(1-\frac{2}{3}\right)}{(1+1)\left(1+\frac{3}{2}\right)} = \frac{4\cdot\frac{1}{3}}{2\cdot\frac{5}{2}} = \frac{4}{15} = 0.2667$$

$$A_3 = \left.\frac{4\left(1+\frac{1}{3}z^{-1}\right)}{\left(1-\frac{1}{2}z^{-1}\right)\left(1+\frac{1}{2}z^{-1}\right)}\right|_{z^{-1}=\frac{4}{3}} = \frac{4\left(1+\frac{4}{9}\right)}{\left(1-\frac{2}{3}\right)\left(1+\frac{2}{3}\right)} = \frac{4\cdot\frac{13}{9}}{\frac{1}{3}\cdot\frac{5}{3}} = \frac{52}{5} = 10.4$$

$$Y(z) = \frac{-20/3}{1-\frac{1}{2}z^{-1}} + \frac{4/15}{1+\frac{1}{2}z^{-1}} + \frac{52/5}{1-\frac{3}{4}z^{-1}}$$

- Use the transform pair $a^n u[n] \Leftrightarrow 1/(1-az^{-1})$ and linearity to finally obtain

$$y[n] = -\frac{20}{3}\left(\frac{1}{2}\right)^n u[n] + \frac{4}{15}\left(-\frac{1}{2}\right)^n u[n] + \frac{52}{5}\left(\frac{3}{4}\right)^n u[n]$$

The Discrete-Time Fourier Transform

- The frequency domain view of discrete-time signals and systems is obtained via the *discrete-time Fourier transform* (DTFT)
- The DTFT of signal $x[n]$ produces a complex valued continuous function of Ω (remember we will keep $\omega = 2\pi f$ as the continuous-time frequency variable for this course)

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

  referred to as the *signal spectrum*
- The DTFT exists if

$$\sum_{n=-\infty}^{\infty} |x[n]| < \infty,$$

  which is the definition of absolute summability
- The inverse DTFT (IDTFT) is obtain via integration of $X(e^{j\Omega})$ over and 2π interval

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

- Certain signals of interest are not absolutely summable, but a DTFT can be defined if we allow impulse functions in the frequency domain, i.e., for $x[n] \overset{\mathcal{F}}{\longleftrightarrow} X(e^{j\Omega})$

$$e^{jn\Omega_0} \overset{\mathcal{F}}{\longleftrightarrow} 2\pi\delta(\Omega - \Omega_0)$$

$$A\cos(\Omega_0 n) \overset{\mathcal{F}}{\longleftrightarrow} \frac{A}{2}\left[\delta(\Omega - \Omega_0) + \delta(\Omega + \Omega_0)\right]$$

- The DTFT is a periodic function of Ω, since $e^{j(\Omega+2\pi k)} = e^{j\Omega}$ for any k, it follows that

$$X(e^{j(\Omega+2\pi k)}) = X(e^{j\Omega})$$

- For $x[n]$ a real sequence the DTFT has the symmetry property

$$\begin{aligned} X(e^{j\Omega}) &= X^*(e^{j\Omega}) \\ \Rightarrow\ |X(e^{j\Omega})| &= |X(e^{-j\Omega})| \text{ magnitude even in } \Omega \\ \angle X(e^{j\Omega}) &= -\angle X(e^{-j\Omega}) \text{ phase odd in } \Omega \end{aligned}$$

- When the signal $x[n]$ is replaced by the impulse response of an LTI system, the resulting DTFT has the special interpretation as system/filter *frequency response*

$$H(e^{j\Omega}) = \sum_{n=-\infty}^{\infty} h[n]e^{-j\Omega n}$$

  - Note that

$$|H(e^{j\Omega})| = \left| \sum_{n=-\infty}^{\infty} h[n]e^{-j\Omega n} \right| \leq \sum_{n=-\infty}^{\infty} |h[n]| < \infty$$

    if the DTFT exists and also if the system is BIBO stable

- One of the more significant DTFT properties is the *convolution theorem*, which states

$$y[n] = x[n] * h[n] \overset{\mathcal{F}}{\longleftrightarrow} X(e^{j\Omega})H(e^{j\Omega}) = Y(e^{j\Omega})$$

  - This result in particular shows us that a filter modifies the spectrum of a signal via multiplication by the filter frequency response

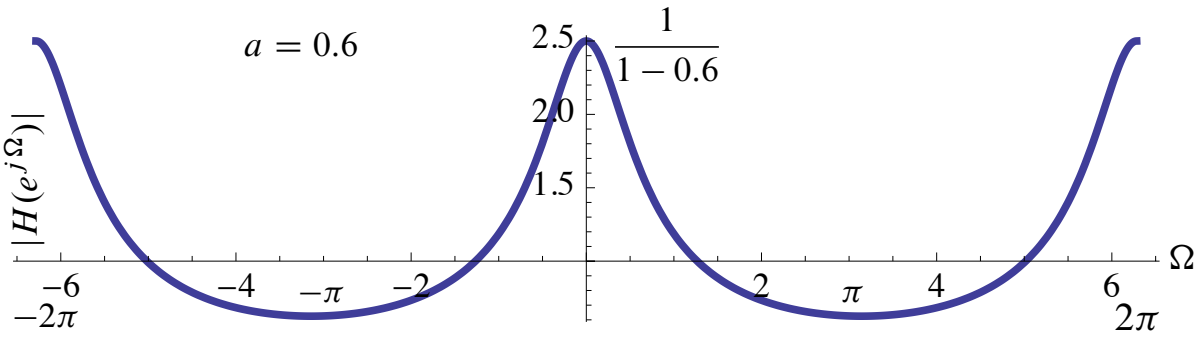
Short Tables of DTFT Theorems/Properties/Pairs

Property	Signal	DTFT
	$x(n)$	$X\left(e^{j\Omega}\right)$
	$y(n)$	$Y\left(e^{j\Omega}\right)$
Linearity	$ax(n)+by(n)$	$aX\left(e^{j\Omega}\right)+bY\left(e^{j\Omega}\right)$
Time Shifting	$x(n-n_0)$	$e^{-j\Omega n_0}X\left(e^{j\Omega}\right)$
Time Scaling (Upsampling)	$\begin{cases} x(n/K) & n \text{ is a multiple of K} \\ 0 & \text{otherwise} \end{cases}$	$X\left(e^{jK\Omega}\right)$
Conjugation	$x^*(n)$	$X^*\left(e^{-j\Omega}\right)$
Convolution	$x(n)*y(n)$	$X\left(e^{j\Omega}\right)Y\left(e^{j\Omega}\right)$
Multiplication	$x(n)y(n)$	$\frac{1}{2\pi}X\left(e^{j\Omega}\right)*Y\left(e^{j\Omega}\right)$
First Difference	$x(n)-x(n-1)$	$\left(1-e^{-j\Omega}\right)X\left(e^{j\Omega}\right)$
Accumulation	$\sum_{k=-\infty}^{n} x(k)$	$\frac{1}{1-e^{-j\Omega}}X\left(e^{j\Omega}\right)$
Parseval's Relation	$\sum_{n=-\infty}^{\infty}\lvert x(n)\rvert^2=\frac{1}{2\pi}\int_{2\pi}\left\lvert X\left(e^{j\Omega}\right)\right\rvert^2 d\Omega$	

Example 2.4: $h[n]=a^n u[n]$

- Taking the Fourier transform

$$H(e^{j\Omega})=\sum_{n=0}^{\infty}a^n e^{-j\Omega n}=\sum_{n=0}^{\infty}\left(ae^{-j\Omega}\right)^n=\frac{1}{1-ae^{-j\Omega}}$$



Useful Fourier Transform Pairs

Signal	DTFT
$x(n)$	$X\left(e^{j\Omega}\right)$
$\delta(n)$	1
$\delta(n-n_0)$	$e^{-j\Omega n_0}$
$u(n)$	$\frac{1}{1-e^{-j\Omega}}+\pi\sum_{l=-\infty}^{\infty}\delta\left(\Omega-2\pi l\right)$
$a^n u(n),\ \lvert a\rvert<1$	$\frac{1}{1-ae^{-j\Omega}}$
$\sum_{k=k_0}^{k_0+N-1} c_k e^{jk(2\pi/N)n}$	$2\pi\sum_{k=-\infty}^{\infty} c_k\delta\left(\Omega-\frac{2\pi k}{N}\right)$
$\sum_{k=-\infty}^{\infty}\delta(n-kN)$	$\frac{2\pi}{N}\sum_{l=-\infty}^{\infty}\delta\left(\Omega-\frac{2\pi l}{N}\right)$
1	$2\pi\sum_{l=-\infty}^{\infty}\delta\left(\Omega-2\pi l\right)$
$\begin{cases}1 & \lvert n\rvert\le N\\ 0 & \lvert n\rvert>N\end{cases}$	$\frac{\sin\left(\Omega\left[N+\frac{1}{2}\right]\right)}{\sin\left(\frac{\Omega}{2}\right)}$
$e^{j\Omega_0 n}$	$2\pi\sum_{l=-\infty}^{\infty}\delta\left(\Omega-\Omega_0-2\pi l\right)$
$\cos(\Omega_0 n)$	$\pi\sum_{l=-\infty}^{\infty}\left\{\delta(\Omega-\Omega_0-2\pi l)+\delta(\Omega+\Omega_0-2\pi l)\right\}$
$\sin(\Omega_0 n)$	$\frac{\pi}{j}\sum_{l=-\infty}^{\infty}\left\{\delta(\Omega-\Omega_0-2\pi l)-\delta(\Omega+\Omega_0-2\pi l)\right\}$

2.2.8 Filter Flowgraphs

- The LCCDE describes mathematically how to calculate the output $y[n]$ from the input $x[n]$ and using past values of the input and output

- Hardware/firmware/software implementations require that a particular filter topology be chosen to implement the LCCDE

- For both IIR and FIR filters there are standard forms such as

  1. Direct form I and II
  2. Transposed direct form I and II
  3. Parallel and cascade
  4. Lattice (text Chapter 6)

- The direct form structures follow directly from the LCCDE

$$\begin{aligned} y[n] &= -\sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k] \\ &= -\sum_{k=1}^{N} a_k y[n-k] + w[n] \end{aligned}$$

  where

$$w[n] = \sum_{k=0}^{M} b_k x[n-k]$$

- In the z-domain we can view this as $H(z) = H_{\text{zeros}}(z)H_{\text{poles}}(z)$

$$H(z) = \underbrace{\frac{1}{1 + \sum_{k=1}^{N} a_k z^{-k}}}_{H_{\text{poles}}(z)} \times \underbrace{\sum_{k=0}^{M} b_k z^{-k}}_{H_{\text{zeros}}(z)}$$

- Due to linearity we can filter with the poles first and the zeros second, i.e., $H_{\text{poles}}(z)H_{\text{zeros}}(z)$, thus

$$w[n] = -\sum_{k=1}^{N} a_k w[n-k] + x[n]$$

$$y[n] = \sum_{k=0}^{M} b_k w[n-k]$$

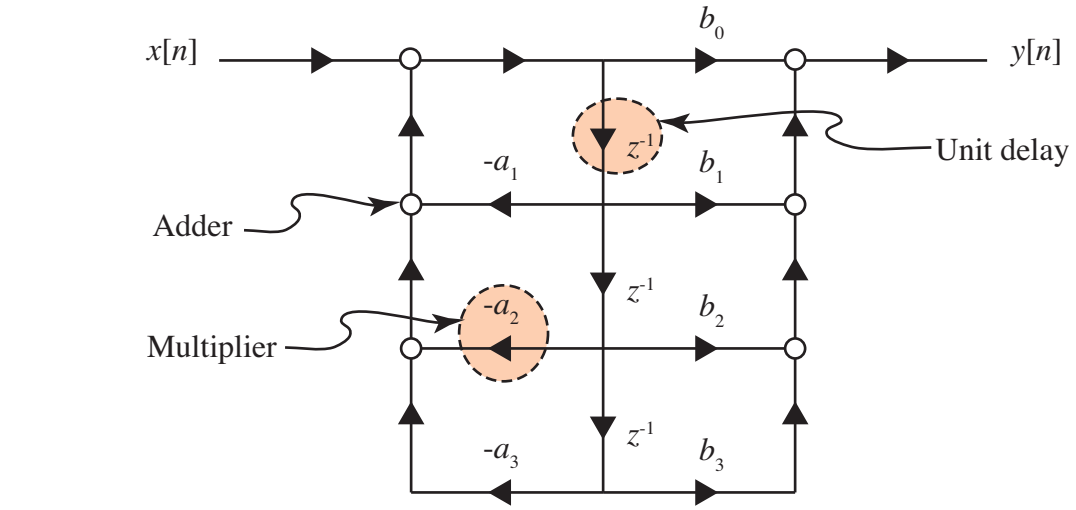
  which are the steps for a direct form II IIR filter and result in

$$H(z) = \underbrace{\sum_{k=0}^{M} b_k z^{-k}}_{H_{\text{zeros}}(z)} \times \underbrace{\frac{1}{1 + \sum_{k=1}^{N} a_k z^{-k}}}_{H_{\text{poles}}(z)}$$

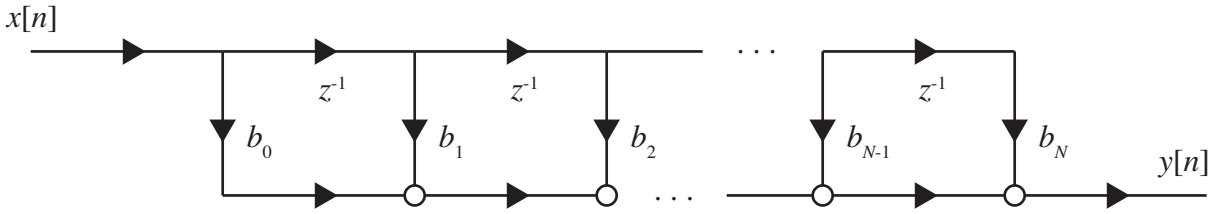
- Given infinite precision mathematics all of the topologies produce identical outputs

- The lattice filters of chapter 6 have particular relationship to the linear prediction and minimum mean square error estimation, and come in pole-zero, all-zero, and all-zero forms

- Signal flow graphs (SFGs) or alternatively block diagrams are convenient to depict the implementation topology



(a) Third-order direct form II IIR



(b) *N*th-order direct form FIR

2.2.9 The DFT and FFT

- The last topic in this general DSP review is the *discrete Fourier transform* (DFT)
- The DFT is a computationally tractable way of transforming a finite length sequence into the frequency domain and back again
- The DFT is also a very useful modeling and analysis tool

- The DFT/IDFT pair is defined for sequence $x[n]$ having support over $0 \leq n \leq N-1$ as

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}, \quad 0 \leq k \leq N-1$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N}, \quad 0 \leq n \leq N-1$$

- A vitally important interpretation of the DFT is that is is a sampled version of the DTFT, that is it produced uniformly sampled frequency values on $[0, 2\pi)$

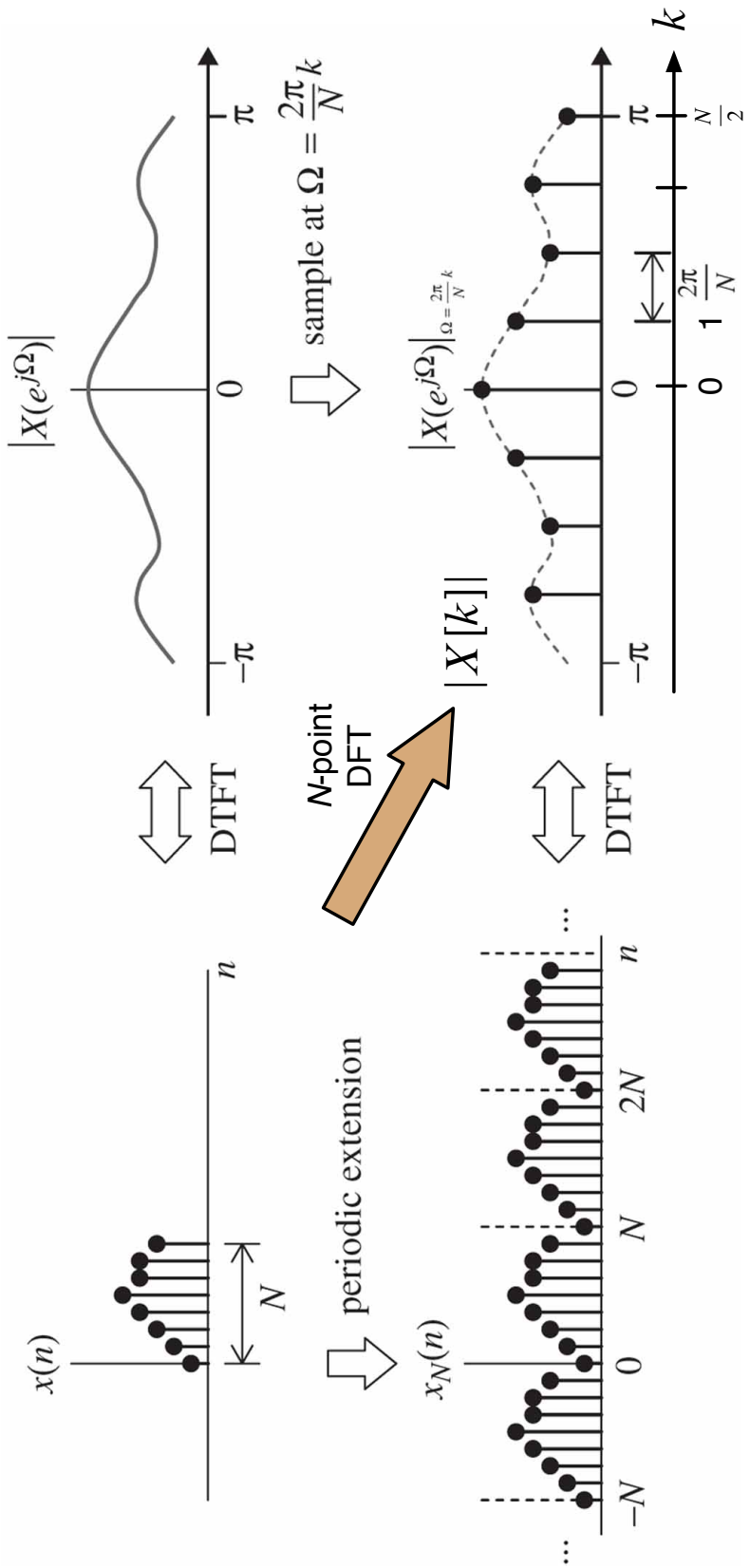
$$X[k] = X(e^{j\Omega})\big|_{\Omega=2\pi k/N}, \quad 0 \leq k \leq N-1$$

- Transform domain processing can be performed using the DFT and IDFT once one understands circular convolution, which is the result of multiplying IDFT$\{X[n]H[k]\}$ in lieu of $x[n] * h[n]$

- The first operation produces the circular convolution of $x[n]$ and $[h]$, while the second produces the more familiar linear convolution

- To make them equivalent we can zero pad the sequences so the aliasing produced by circular convolution can be mitigated

- The motivation for transform domain processing is the fact that the DFT can be efficiently calculated using *Fast Fourier Transform* (FFT) algorithms, e.g., the radix-2 algorithm requires $N/2 \log_2(N)$ complex multiplies, while the raw DFT requires N^2 complex multiplies; N must be a power of 2



DFT and DTFT relationships

2.2.10 The Relationship Between Discrete- and Continuous-Time Systems

The Sampling Theorem

Sampling theory is developed by considering the spectrum of a continuous-time signal, $x_c(t)$, following ideal sampling, i.e.,

$$x_p(t) = x_c(t) \times \underbrace{\sum_{n=-\infty}^{\infty} \delta(t - nT)}_{p(t)} = \sum_{n=-\infty}^{\infty} x_c(nT)\delta(t - nT).$$

- We are interested in $X_p(j\omega) = \mathcal{F}\{x_p(t)\}$
- The Fourier transform of the ideal sampling waveform is

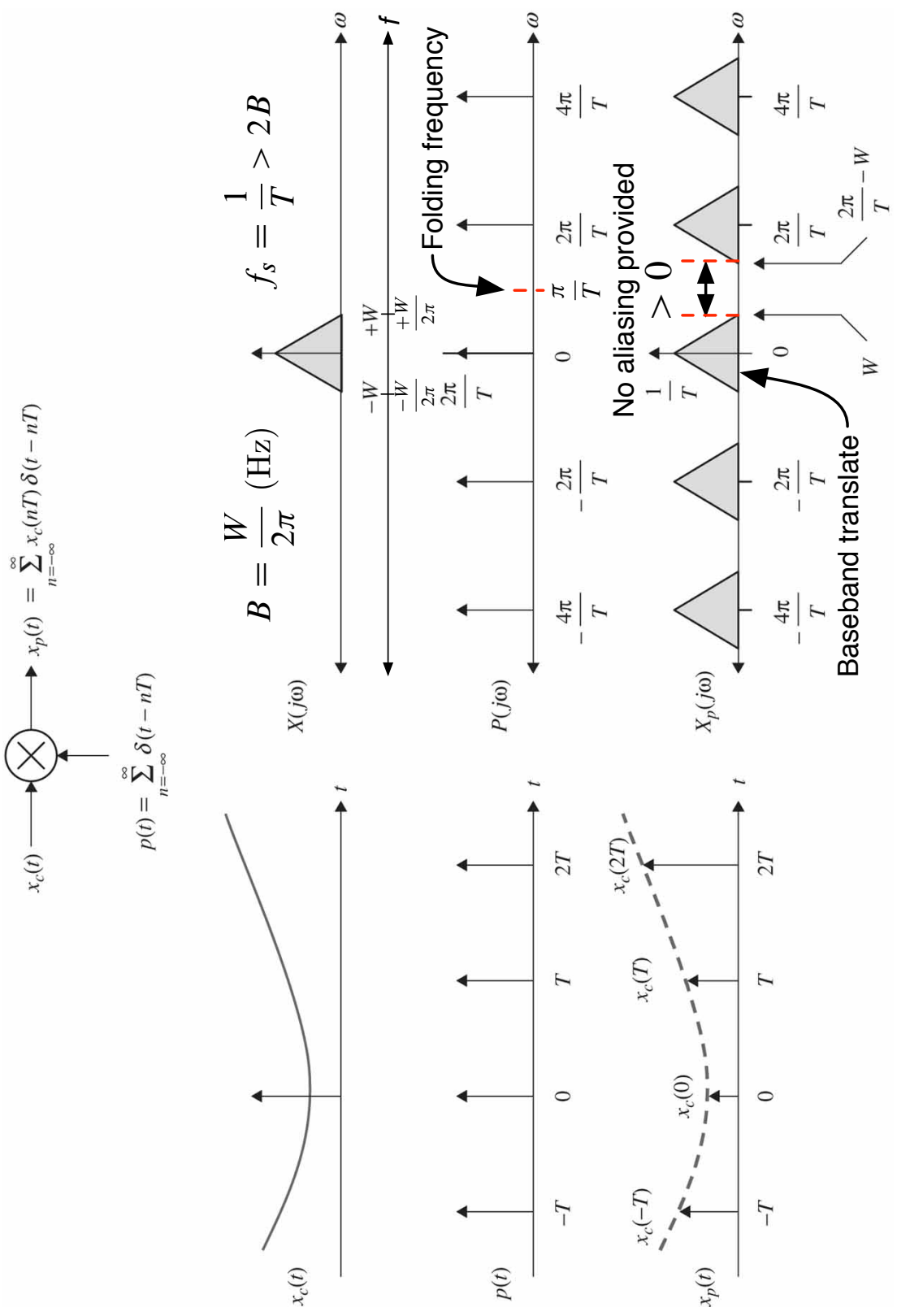
$$P(j\omega) = \mathcal{F}\{p(t)\} = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta\left(\omega - k\frac{2\pi}{T}\right),$$

  and using the multiplication theorem we have that

$$\begin{aligned} X_p(j\omega) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} X_c(j\omega) P(j(\omega - \theta)\, d\theta \\ &= T \sum_{k=-\infty}^{\infty} X_c\left(\omega - k\frac{2\pi}{T}\right) \end{aligned}$$

- For $x_c(t)$ bandlimited such that $X_c(j\omega) = 0$ for $\omega > W$ rad/s, the *sampling theorem* states that $x_c(t)$ can be recovered from $x_p(t)$ provided

$$\frac{2\pi}{T} > 2W \quad \text{or} \quad f_s = \frac{1}{T} > 2 \cdot \frac{W}{2\pi} = 2 \cdot B \text{ (Hz)}$$



Sampling theorem graphical example

- Notice from the figure above, that recovery of $x_c(t)$ is possible since spectral *translates* do not overlap (no *aliasing*)

$$\left(\frac{2\pi}{T} - W\right) - W > 0 \quad \text{or} \quad \frac{2\pi}{T} > 2W$$

- To recover $x_c(t)$ from the continuous-time signal $x_p(t)$, we need an analog filter that can interpolate the correct signal values between the samples

- The passband of this filter needs to extend to W rad/s, can have a transition band over $W < |\omega| < 2\pi/T - W$, and have stopband $|\omega| > 2\pi/T - W$

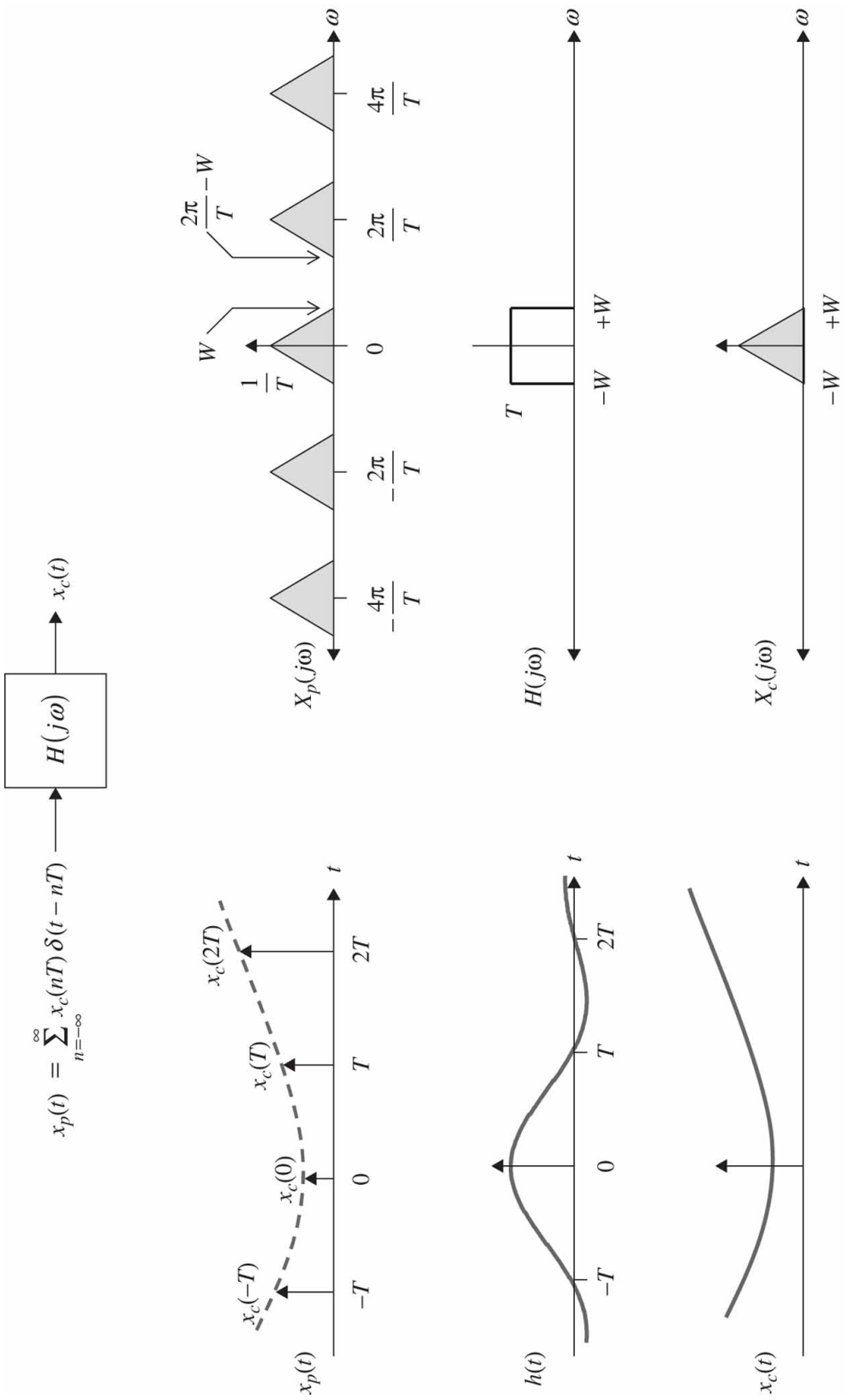
- One such filter is an ideal lowpass, having cutoff frequency $\Omega_c = W$, with impulse response

$$h(t) = \frac{WT}{\pi}\frac{\sin(Wt)}{Wt}, \; -\infty < t < \infty$$

- Using the samples $x_p[n] = x_p(nT) \overset{\text{also}}{=} x_c(nT)$, we have

$$x_c(t) = x_p(t) * h(t) = \sum_{n=-\infty}^{\infty} x_p[n]h(t - nT)$$

- Since $h(t)$ is noncausal, we have to find a more practical filter and live with the imperfections



Reconstruction graphical example

Relating $X_c(j\omega)$ to $X_d(e^{j\Omega})$

- When we obtain the sequence $x_d[n]$ via uniform sampling of $x_c(t)$, i.e., $x_d[n] = x_c(nT)$, $X_c(j\omega)$ and $X(e^{j\Omega})$ are then related

- The Fourier transform of the intermediate sampled signal $x_p(t)$ can be obtained term-by-term as

$$\begin{aligned} X_p(j\omega) &= \int_{-\infty}^{\infty} x_c(nT)\delta(t-nT)e^{-j\omega t}\,dt \\ &= \sum_{n=-\infty}^{\infty} x_c(nT)e^{-j\omega nT} \overset{\text{also}}{=} \sum_{n=-\infty}^{\infty} X_c\left(\omega - n\frac{2\pi}{T}\right) \end{aligned}$$

- Now also note that by definition of the DTFT,

$$X_d(e^{j\Omega}) = \sum_{n=-\infty}^{\infty} x_d[n]e^{-j\Omega} = \sum_{n=-\infty}^{\infty} x_c(nT)e^{-j\Omega},$$

  so realizing that $\Omega = \omega T$ or $\omega = \Omega/T$ we have

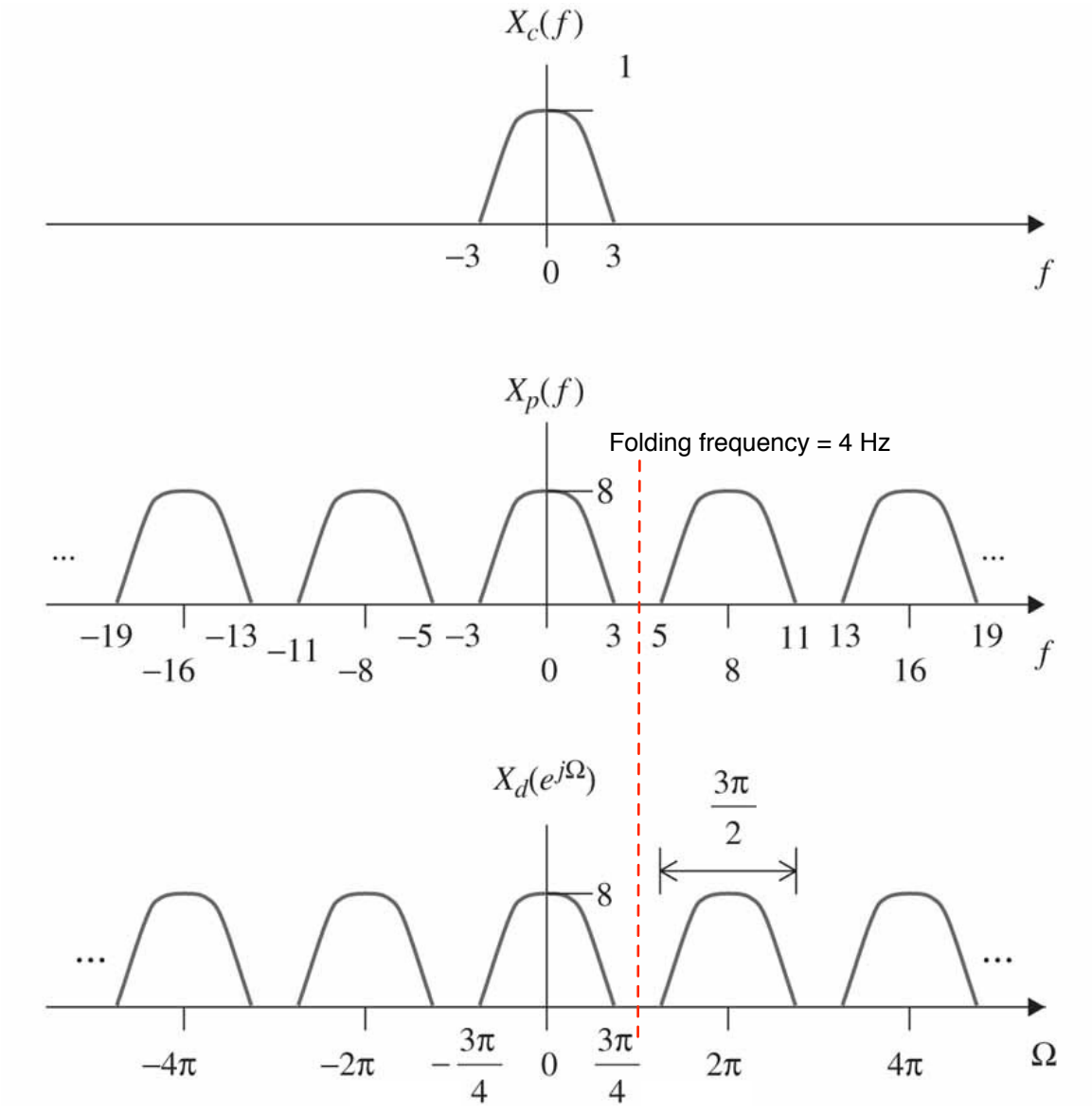
$$X_d(e^{j\Omega}) = X_p\left(j\frac{\Omega}{T}\right) = \frac{1}{T}\sum_{n=-\infty}^{\infty} X_c\left(j\left(\frac{\Omega}{T} - k\frac{2\pi}{T}\right)\right)$$

- Note in particular if the sampling theorem is satisfied, then

$$X_d(e^{j\Omega}) = X_c\left(j\frac{\Omega}{T}\right),\ -\pi < \Omega < \pi$$

Example 2.5: Bandband Signal with $B = 3$ Hz

- Suppose that $x_c(t)$ is a baseband signal having bandwidth $B = 3$ Hz and we sample the signal using $f_s = 1/T = 8$ Hz
- The two-sided baseband bandwidth in the discrete-time domain is $2 \times 3 \cdot 2\pi/8 = 3\pi/2$ rad/sample

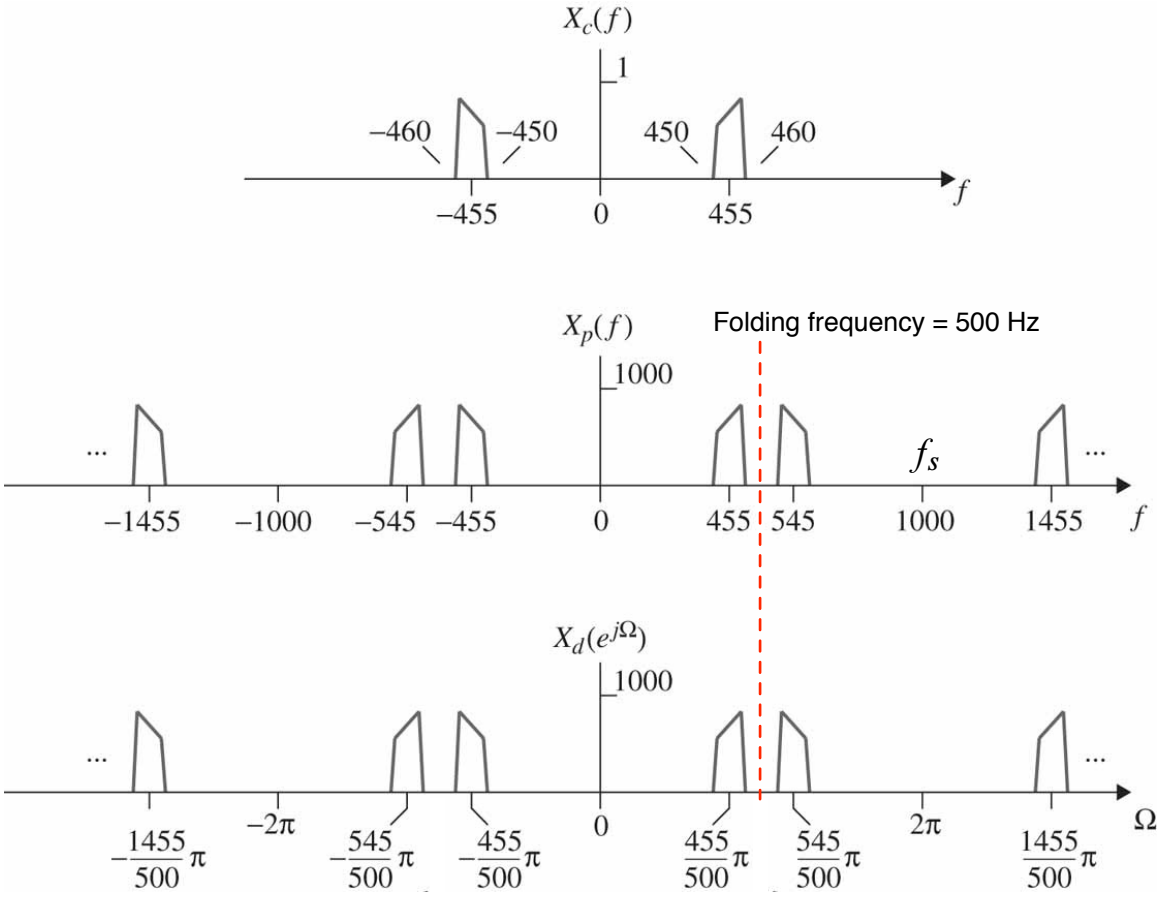


Spectra of a sampled baseband signal

Example 2.6: Bandpass Signal with $f_l = 450$ Hz and $f_u = 460$ Hz

- Suppose that $x_c(t)$ is a bandpass signal having bandwidth $B = 10$ Hz centered on 455 Hz, and we sample the signal using $f_s = 1/T = 1000$ Hz (note $1000 > 2 \times 460 = 920$)

- The two-sided baseband bandwidth in the discrete-time domain is $2 \times 3 \cdot 2\pi/8 = 3\pi/2$ rad/sample



Spectra of a sampled bandpass signal

Bandpass Sampling

- In DSP-based implementations of communication systems, we often deal with bandpass signals
- There is a bandpass version of the sampling theorem[2] which states that for a real bandpass signal occupying the band $[f_l, f_u]$,

$$\frac{2f_u}{n} \leq f_s \leq \frac{2f_l}{n-1}$$

where the integer n is given by

$$1 \leq n \leq \left\lfloor \frac{f_u}{B} \right\rfloor$$

and $B = f_u - f_l$

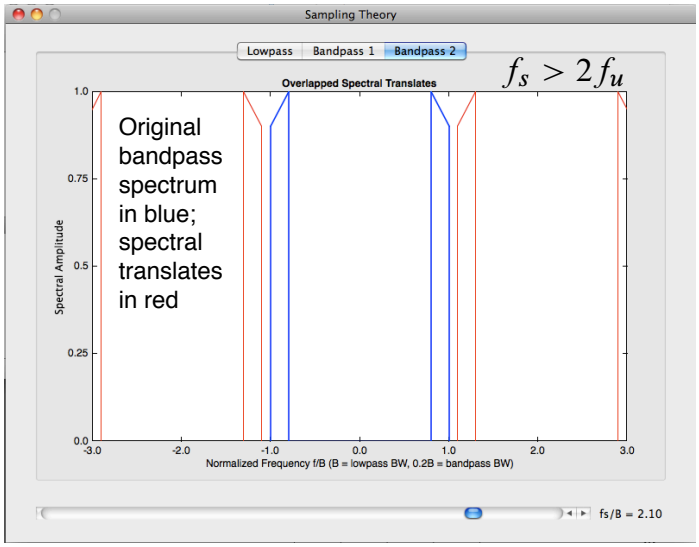
  - Note that when $n = 1$ we have the lowpass sampling theorem and $f_s > 2f_u$
  - Note also that $\lfloor f_u/B \rfloor$ is the greatest integer less than or equal to f_u/B

- Assuming that the bandpass signal is symmetrical about its center, here denoted as $f_c = (f_l + f_u)/2$, we may also want to constrain the center frequency of the lowest set of translates to $\pm f_s/4$, i.e., choose

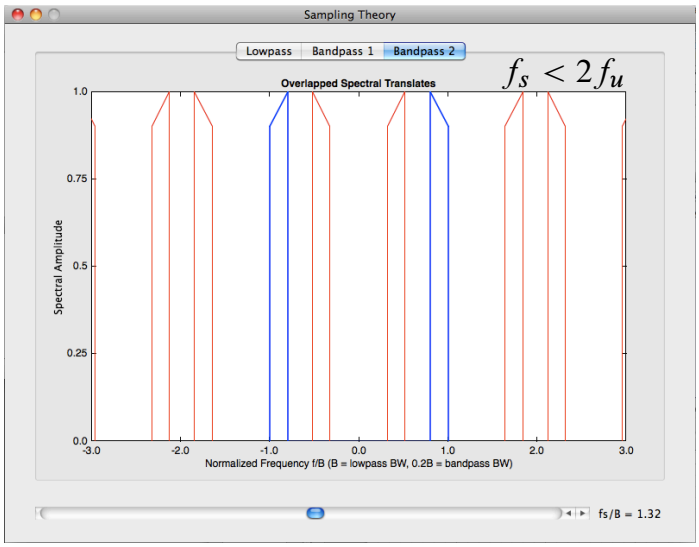
$$|f_c - Mf_s| = f_s/4$$

where M is a positive integer and f_s must lie in the valid intervals defined above

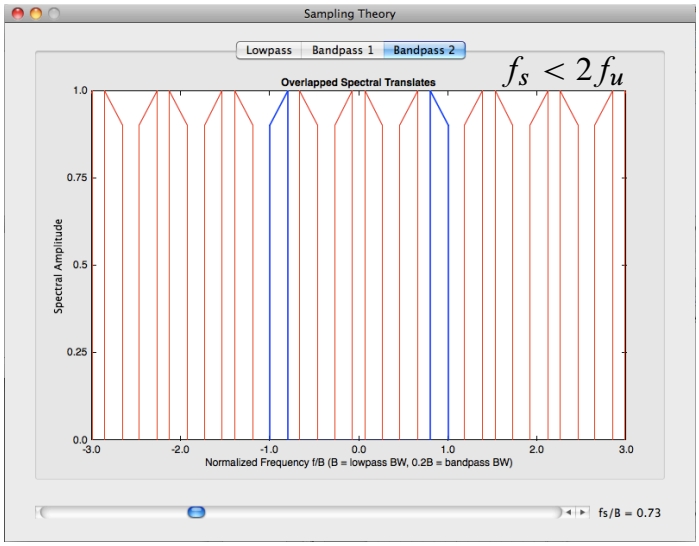
[2] R. Vaughan, N. Scott, and D. White, "The Theory of Bandpass Sampling," *IEEE Transactions on Signal Processing*, vol. 39, no. 9, pp. 1973–1984, September 1991.



Standard Lowpass sampling theory



Under sample by a small amount, yet avoid aliasing



Under sample by a larger amount. Translated are packed tighter.

Bandpass sampling animation screen shots

- A MATLAB function was written which provides a list of the f_s frequency intervals and a list of the specific f_s values which result in $f_c = f_s/4$

```
function [BP_bands fso4] = find_BP_bands(fl,fu)
% [BP_bands fso4] = find_BP_bands(fl,fu)
%=========================================================================
%        fl = Lower bandpass frequency
%        fu = Upper bandpass frequency
%
%  BP_bands = The valid fs intervals for bandpass sampling
%      fso4 = The valid fs values resulting in fc = fs/4
%             Note: We assume that the passband is symmetrical
%
% Mark Wickert, August 2010


% Assume bandpass is symmetrical with regard to fc calulations
fc = (fl + fu)/2;

% Find all the usable frequency bands
n_max = floor(fu/(fu-fl));
BP_bands = zeros(n_max,2);
for k=1:n_max
    BP_bands(k,:) = [2*fu/k 2*fl/(k-1)];
end

% Find the frequency bands that result in fc = fs/4
fso4_maybe = zeros(2*n_max,1);
for k=1:n_max
   fso4_maybe(2*k-1) = 4*fc/(4*k+1);
   fso4_maybe(2*k) = 4*fc/(4*k-1);
end

% See which of the fc = fs/4 frequencies actually work
fso4 = [];
for n=1:length(fso4_maybe)
    for m=1:n_max
       if (fso4_maybe(n) >= BP_bands(m,1)) && (fso4_maybe(n) <= BP_bands(m,2))
           fso4 = [fso4 fso4_maybe(n)];
       end
    end
end
fso4 = fso4';
fso4 = sort(fso4);
```

Example 2.7: Revisit $f_c = 455,\ B = 10$ Bandpass Signal

```
>> [BP_bands fso4] = find_BP_bands(450,460);
>> % Format output matrices for display tables inside function using:
% for n=1:n_max
%     ds = sprintf('n=%2d: [%6.2f, %6.2f] Hz',n,BP_bands(n,:));
%     disp(ds)
% end
% for n=1:length(fso4)
%     ds = sprintf('n=%2d: fs = %6.2f Hz',n,fso4(n));
%     disp(ds)
% end
```

fs Intervals (`BP_band`) for fl = 4, fu = 460, B = 10, & fc = 455

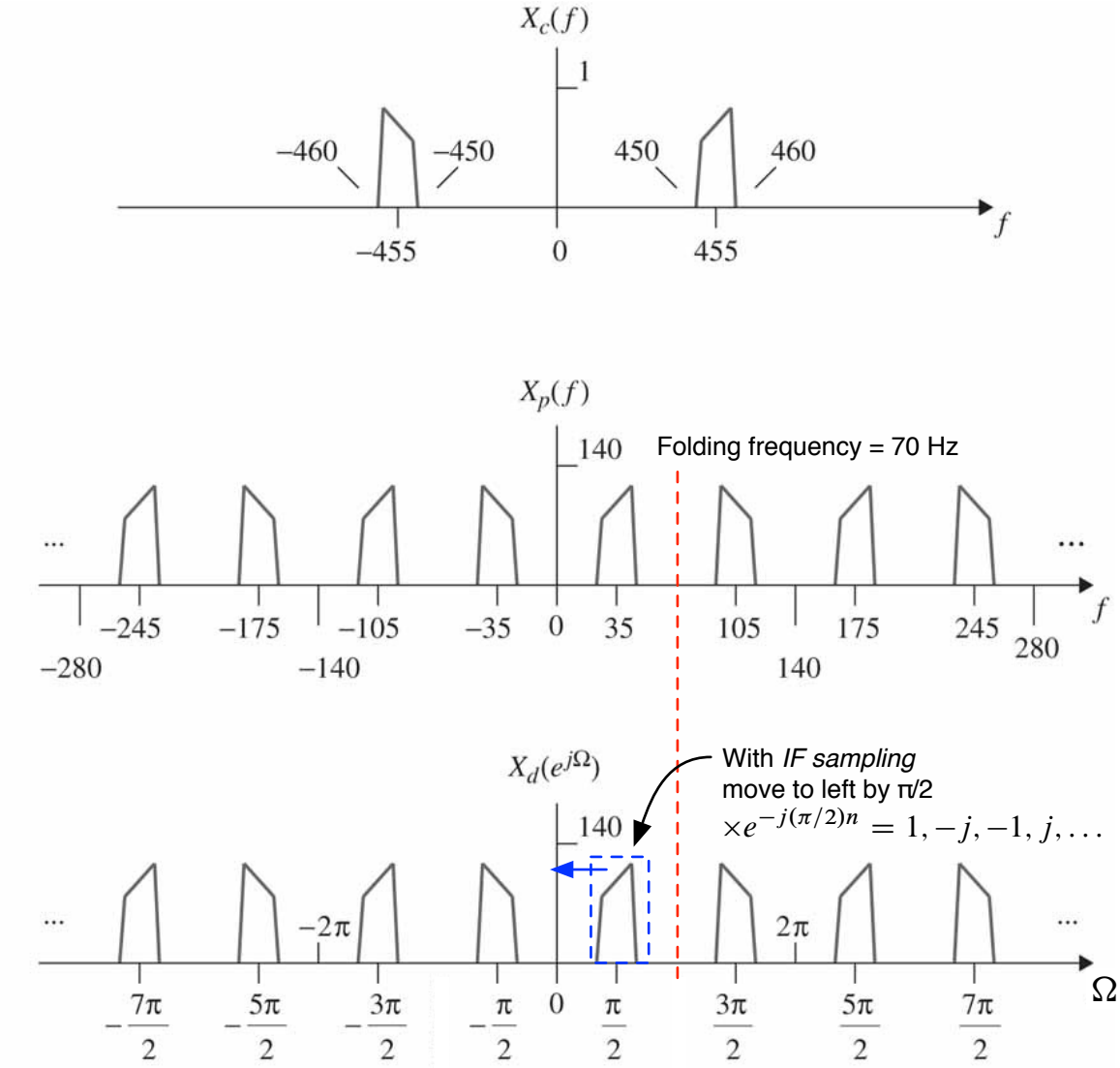
n= 1: [920.00, Inf] Hz	n=17: [54.12, 56.25] Hz	n=33: [27.88, 28.12] Hz
n= 2: [460.00, 900.00] Hz	n=18: [51.11, 52.94] Hz	n=34: [27.06, 27.27] Hz
n= 3: [306.67, 450.00] Hz	n=19: [48.42, 50.00] Hz	n=35: [26.29, 26.47] Hz
n= 4: [230.00, 300.00] Hz	n=20: [46.00, 47.37] Hz	n=36: [25.56, 25.71] Hz
n= 5: [184.00, 225.00] Hz	n=21: [43.81, 45.00] Hz	n=37: [24.86, 25.00] Hz
n= 6: [153.33, 180.00] Hz	n=22: [41.82, 42.86] Hz	n=38: [24.21, 24.32] Hz
n= 7: [131.43, 150.00] Hz	n=23: [40.00, 40.91] Hz	n=39: [23.59, 23.68] Hz
n= 8: [115.00, 128.57] Hz	n=24: [38.33, 39.13] Hz	n=40: [23.00, 23.08] Hz
n= 9: [102.22, 112.50] Hz	n=25: [36.80, 37.50] Hz	n=41: [22.44, 22.50] Hz
n=10: [92.00, 100.00] Hz	n=26: [35.38, 36.00] Hz	n=42: [21.90, 21.95] Hz
n=11: [83.64, 90.00] Hz	n=27: [34.07, 34.62] Hz	n=43: [21.40, 21.43] Hz
n=12: [76.67, 81.82] Hz	n=28: [32.86, 33.33] Hz	n=44: [20.91, 20.93] Hz
n=13: [70.77, 75.00] Hz	n=29: [31.72, 32.14] Hz	n=45: [20.44, 20.45] Hz
n=14: [65.71, 69.23] Hz	n=30: [30.67, 31.03] Hz	n=46: [20.00, 20.00] Hz
n=15: [61.33, 64.29] Hz	n=31: [29.68, 30.00] Hz	
n=16: [57.50, 60.00] Hz	n=32: [28.75, 29.03] Hz	

fs (`fso4`) such that fc = fs/4

n= 1: fs = 20.00 Hz	n=16: fs = 29.84 Hz	n=31: fs = 58.71 Hz
n= 2: fs = 20.45 Hz	n=17: fs = 30.85 Hz	n=32: fs = 62.76 Hz
n= 3: fs = 20.92 Hz	n=18: fs = 31.93 Hz	n=33: fs = 67.41 Hz
n= 4: fs = 21.41 Hz	n=19: fs = 33.09 Hz	n=34: fs = 72.80 Hz
n= 5: fs = 21.93 Hz	n=20: fs = 34.34 Hz	n=35: fs = 79.13 Hz
n= 6: fs = 22.47 Hz	n=21: fs = 35.69 Hz	n=36: fs = 86.67 Hz
n= 7: fs = 23.04 Hz	n=22: fs = 37.14 Hz	n=37: fs = 95.79 Hz
n= 8: fs = 23.64 Hz	n=23: fs = 38.72 Hz	n=38: fs = 107.06 Hz
n= 9: fs = 24.27 Hz	n=24: fs = 40.44 Hz	n=39: fs = 121.33 Hz
n=10: fs = 24.93 Hz	n=25: fs = 42.33 Hz	n=40: fs = 140.00 Hz
n=11: fs = 25.63 Hz	n=26: fs = 44.39 Hz	n=41: fs = 165.45 Hz
n=12: fs = 26.38 Hz	n=27: fs = 46.67 Hz	n=42: fs = 202.22 Hz
n=13: fs = 27.16 Hz	n=28: fs = 49.19 Hz	n=43: fs = 260.00 Hz
n=14: fs = 28.00 Hz	n=29: fs = 52.00 Hz	n=44: fs = 364.00 Hz
n=15: fs = 28.89 Hz	n=30: fs = 55.15 Hz	n=45: fs = 606.67 Hz

Useable frequency bands for f_s and f_s values giving $f_c = f_s/4$

- The $n = 7$ band contains $f_s = 140$ Hz, which is also yields $f_c = f_s/4 = 35$ Hz.

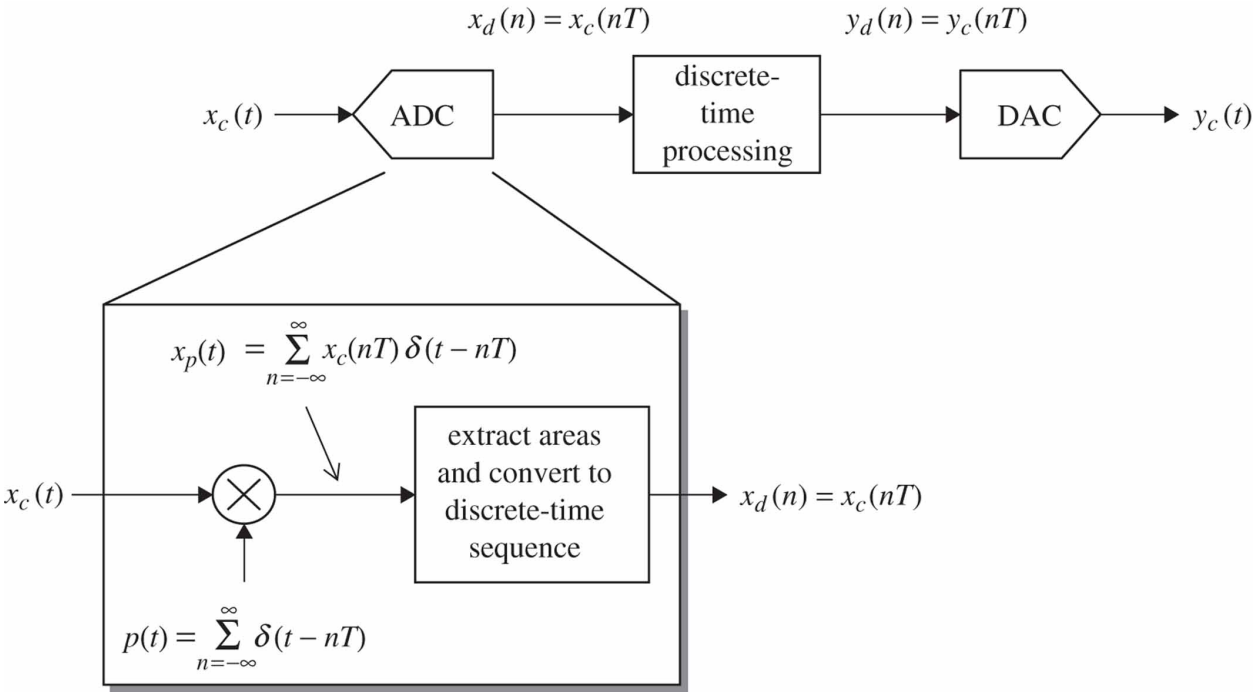


Spectra of undersampled bandpass signal for $f_s = 140$ Hz

- There are many other valid solutions, but $f_s = 140$ Hz provides large *guard bands*
- As n becomes large, the f_s become very narrow and are thus impractical

Discrete-Time Processing of Continuous-Time Signals

- End-to-end processing of continuous-time signals can be accomplished using DSP by including an analog-to-digital converter (ADC) with a digital-to-analog converter (DAC) at each end



DSP means to operate on a continuous-time signal

- If the discrete-time processing block is a digital filter, $H_d(e^{j\Omega})$, and there is anti-aliasing and reconstruction filtering in the ADC and DAC respectively, the effective continuous-time frequency response is

$$H_c(j\omega) = \begin{cases} H_d(e^{j\omega T}), & |\omega| \le \pi/T \\ 0, & \text{otherwise} \end{cases}$$

2.2.11 Multirate Signal Processing

- Multirate signal processing and modern communication systems go hand-in-hand
- The key topics to be reviewed in this section are:*Downsampling*, *upsampling*, and *polyphase filtering*
- Besides the coverage in Rice, the book by Harris[3] is particularly good
- These techniques allow for processing efficiency
- Understanding these operations from the frequency domain is particularly important

Impulse-Train Sampling

- To gain insight into downsampling and upsampling, we first consider the related operation known as *impulse-train sampling*
- Given discrete-time signal $x[n]$ we may wish to resample this signal with an impulse train of the form

$$p[n] = \sum_{k=-\infty}^{\infty} \delta[n - kN)$$

where N is the integer sample spacing

[3]F.J. Harris, *Mukltirate Signal Processing for Communication Systems*, Prentice Hall, New Jersey, 2004

- Through the use of the Poisson sum formula

$$P(e^{j\Omega}) = \frac{2\pi}{N} \sum_{k=-\infty}^{\infty} \delta\left(\Omega - k\frac{2\pi}{N}\right)$$

- We can now obtain the DTFT of the signal

$$x_p[n] = x[n]p[n] = \begin{cases} x[n], & n = \text{integer} \times N \\ 0, & \text{otherwise} \end{cases}$$

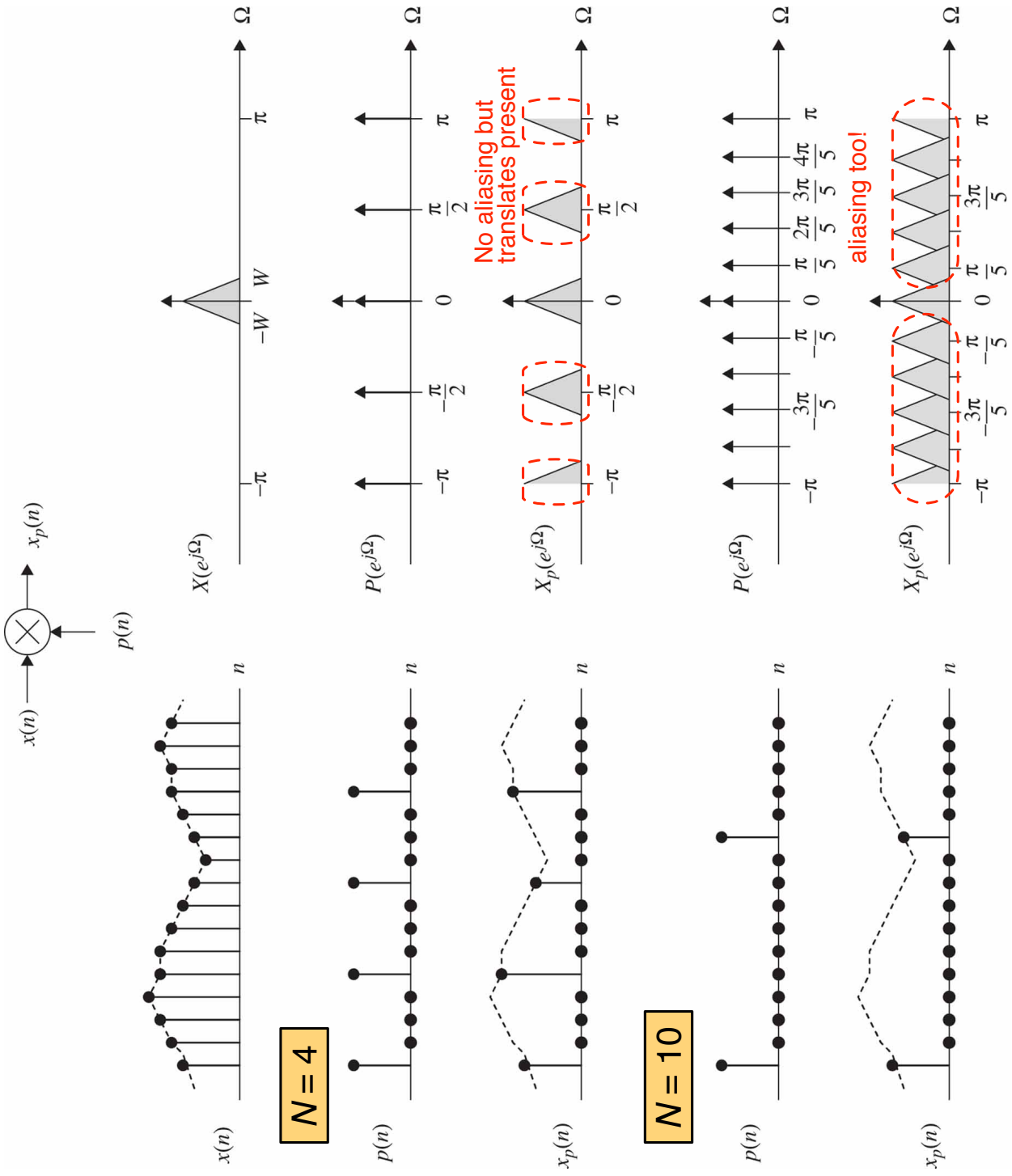
- We use the frequency domain convolution formula which states that

$$\begin{aligned} X_p(e^{j\Omega}) &= \frac{1}{2\pi} \int_0^{2\pi} P\left(e^{j\theta}\right) X\left(e^{j(\Omega-\theta)}\right) d\theta \\ &= \frac{1}{N} \sum_{k=-\infty}^{\infty} \int_0^{2\pi} \delta\left(\theta - k\frac{2\pi}{N}\right) X\left(e^{j(\Omega-\theta)}\right) d\theta \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X\left(e^{j(\Omega - k2\pi/N)}\right) \end{aligned}$$

- We see that taking every Nth sample creates N translates of $X(e^{j\Omega})$ spaced over the $[-\pi, \pi]$ interval

- Depending upon the bandwidth of $x[n]$ aliasing may or may not occur, but the added translates are always present

- To avoid aliasing choose $N < \pi / W$



Impulse-train sampling with $N = 4$ and 10

Downsampling

Downsampling or *decimation* by N reduces the effective sampling rate by N, i.e., only every Nth sample is retained

$$x_D[n] = x[nN]$$

- In the frequency domain (DTFT) we can write

$$X_D(e^{j\Omega}) = \sum_{n=-\infty}^{\infty} x_D[m]e^{-j\Omega n} = \sum_{m=-\infty}^{\infty} x(mN)e^{-j\Omega m}$$

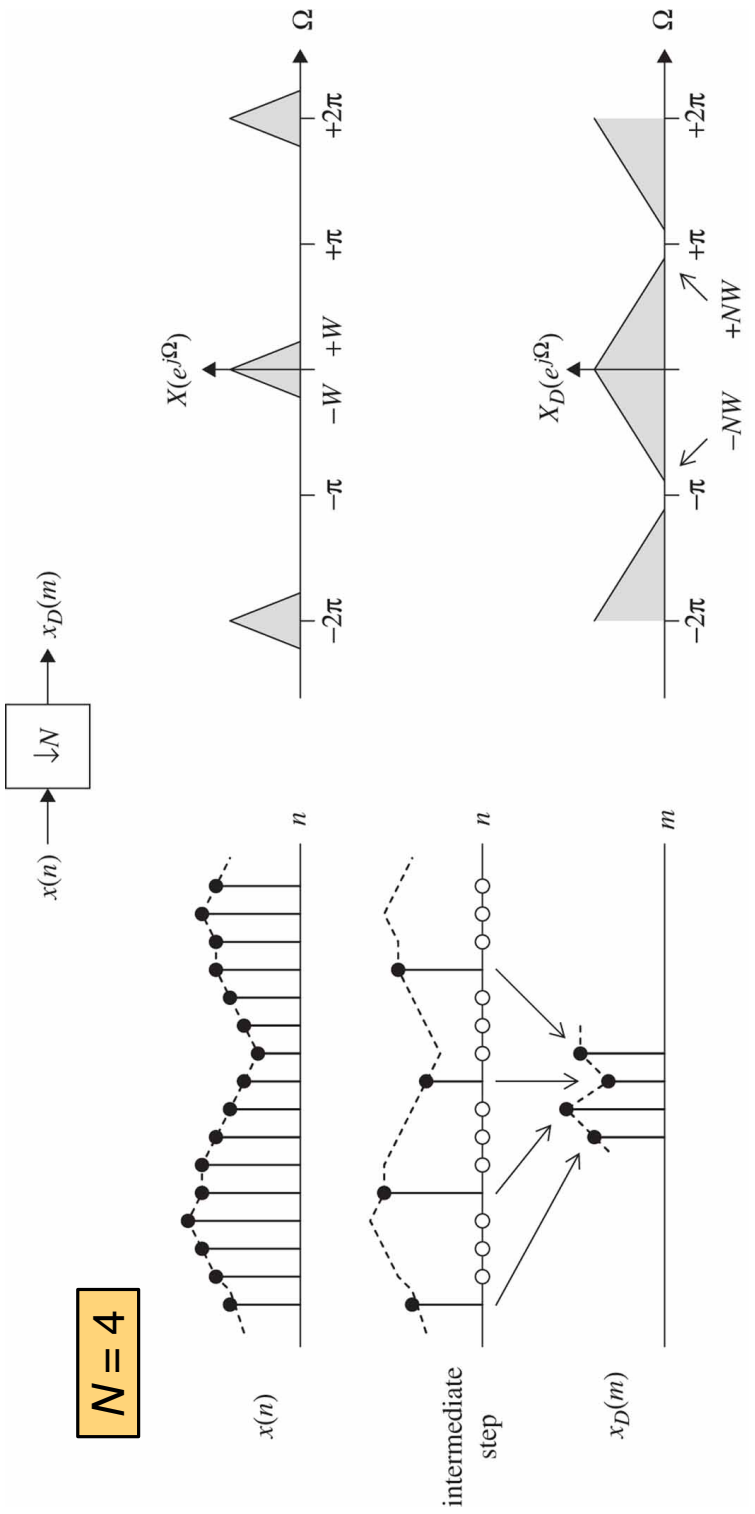
- Let $n = mN$ or $m = n/N$ in the last sum

$$\begin{aligned} X_D(e^{j\Omega}) &= \sum_{n=\text{integer}\times N} x[n]e^{-j\Omega n/N} \\ &= \frac{1}{N}\sum_{k=0}^{N-1} X\left(e^{j(\Omega-2\pi k)/N}\right) \end{aligned}$$

  where the last line follows from the impulse-train sampling DTFT and $\Omega \to \Omega/N$

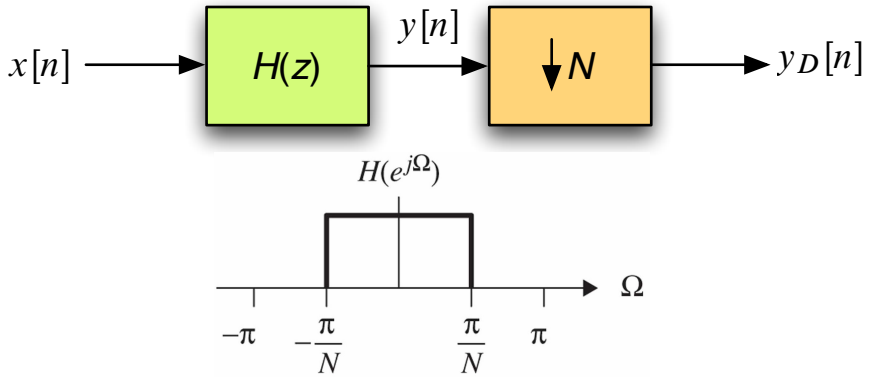
- We now have:

  1. N shifted by $2\pi k/N$, $k = 0, 1, \ldots, N-1$, and Ω-axis stretched by N copies of $X(e^{j\Omega})$
  2. Each copy scaled by $1/N$
  3. In the end we still have $X_D(e^{j\Omega})$ periodic on 2π by virtue of the $\Omega - 2\pi k$ argument



Downsampling by 4

- To avoid aliasing a downsampler is usually preceded by a digital lowpass antialiasing filter
  - MATLAB `y=downsample(x,N,p)` or `y=decimate(x,N)`



Lowpass prefilter with $\Omega_c = \pi/N$ and downsample by N

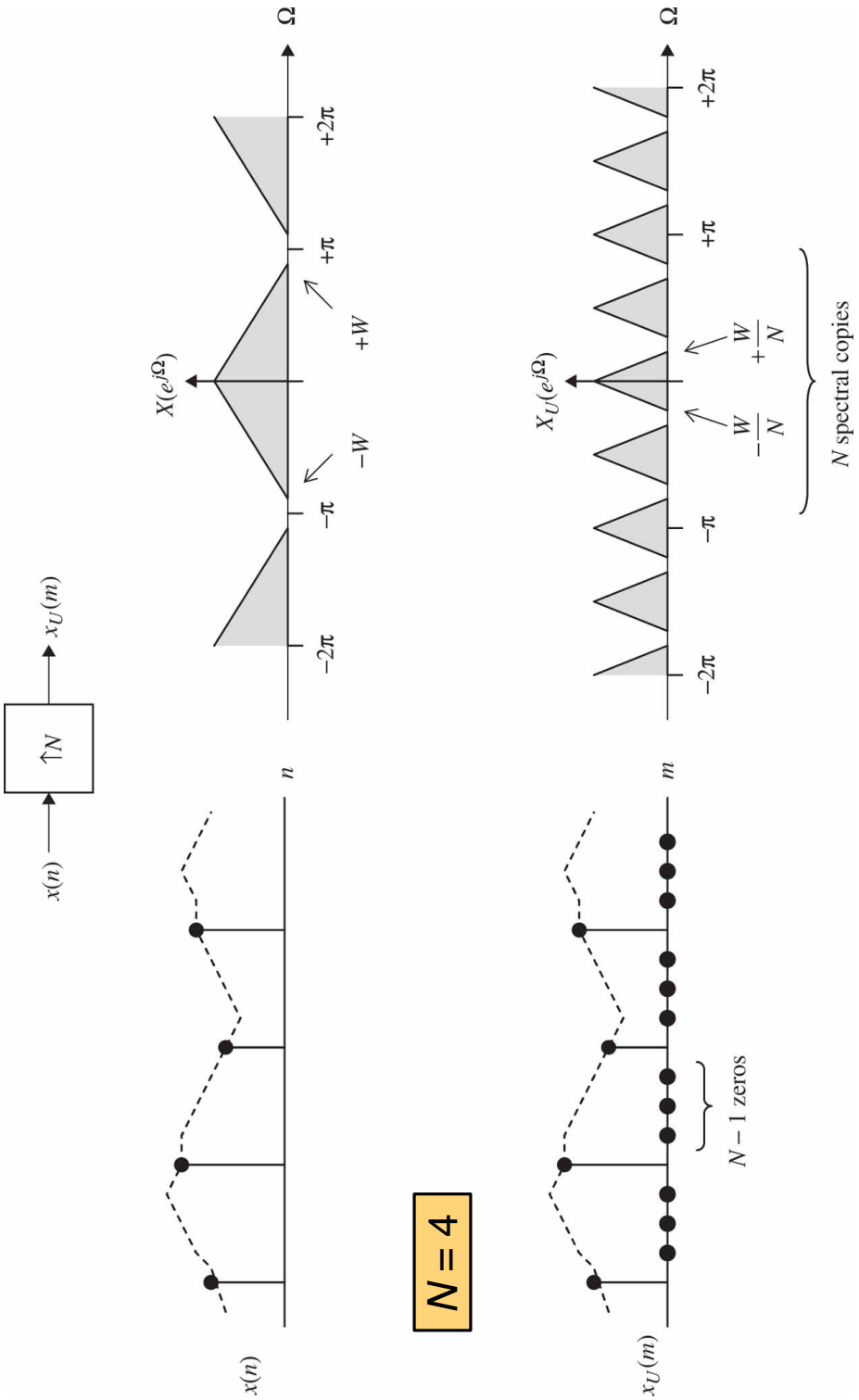
Upsampling

Upsampling is the reverse of downsampling, that is we increase the effective sampling rate by N. The input/output relationship is

$$x_U[n] = \begin{cases} x(n/N), & n = \text{integer} \times N \\ 0, & \text{otherwise} \end{cases}$$

- The upsampling operation stuffs $N-1$ zero samples between each sample of the input
- In the frequency domain (DTFT) we can write that

$$\begin{aligned} X_U(e^{j\Omega}) &= \sum_{n=-\infty}^{\infty} \left(\sum_{k=-\infty}^{\infty} x[k]\delta[n-kN] \right) e^{-j\Omega N} \\ &= \sum_{k=-\infty}^{\infty} x[k]e^{-j\Omega Nk} = X(e^{j\Omega N}) \end{aligned}$$



Upsampling by 4

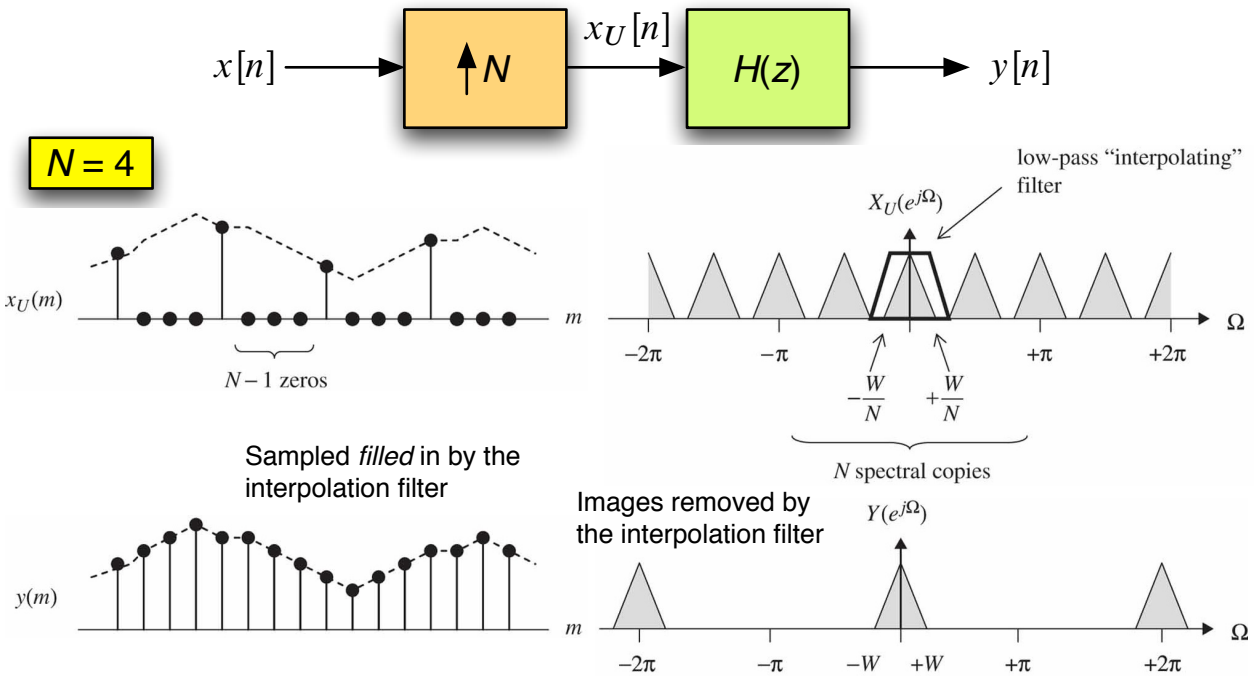
- In words we have that

  1. We compress the frequency axis of $X(e^{j\Omega})$ by N and scale by $1/N$
  2. Place $N-1$ copies of the shifted by $2\pi k/N$ version of (1) on the $[0, 2\pi)$ interval as $k = 1, 2, \ldots, N-1$

- In practice a lowpass interpolation filter follows the upsampler to fill in the zeros sample signal values

- When the two operations are combined we have an *interpolate* by N operation

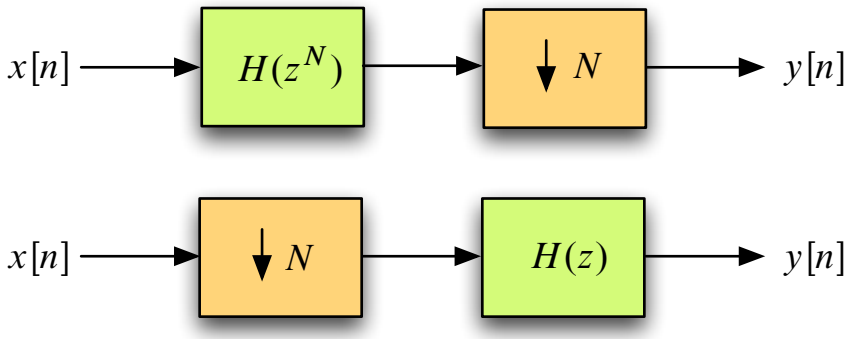
  – MATLAB `y=upsample(x,N)` or `y=interpolate(x,N)`



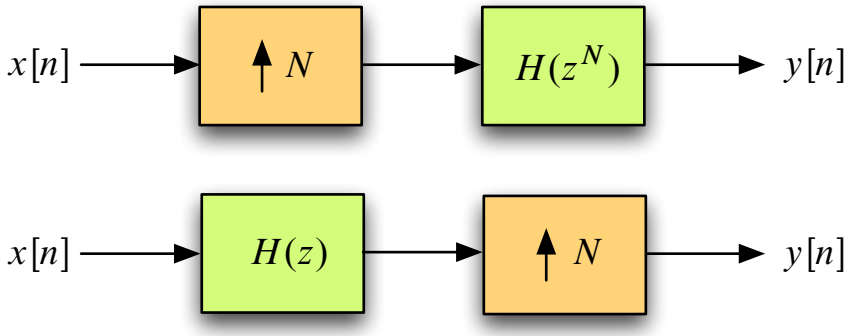
Interpolate by N (upsample followed by lowpass filter)

The Noble Identities

- We have seen that with both a downsampler and an upsampler, there is an associated filtering operation
- When the filter before the downsampler or the filter after the upsampler involves a polynomial in z^N, then the *Nobel* identities may be invoked



(a) Downsampling Identity



(b) Upsampling Identity

The Nobel identities

- In both cases the filtering operation can be moved the low-rate clock side of the operation
- Linear phase FIR filters, where the number of taps is divisible by N find application here

Polyphase Filter Banks

- Moving beyond the Nobel identities, another multirate processing trick is the *polyphase* filterbank
- Consider a causal LTI $H(z)$ of the form

$$\begin{aligned} H(z) &= \sum_{k=0}^{M} h[k]z^{-k} \\ &= h[0] + h[1]z^{-1} + h[2]z^{-2} + \cdots + h[M]z^{-M} \end{aligned}$$

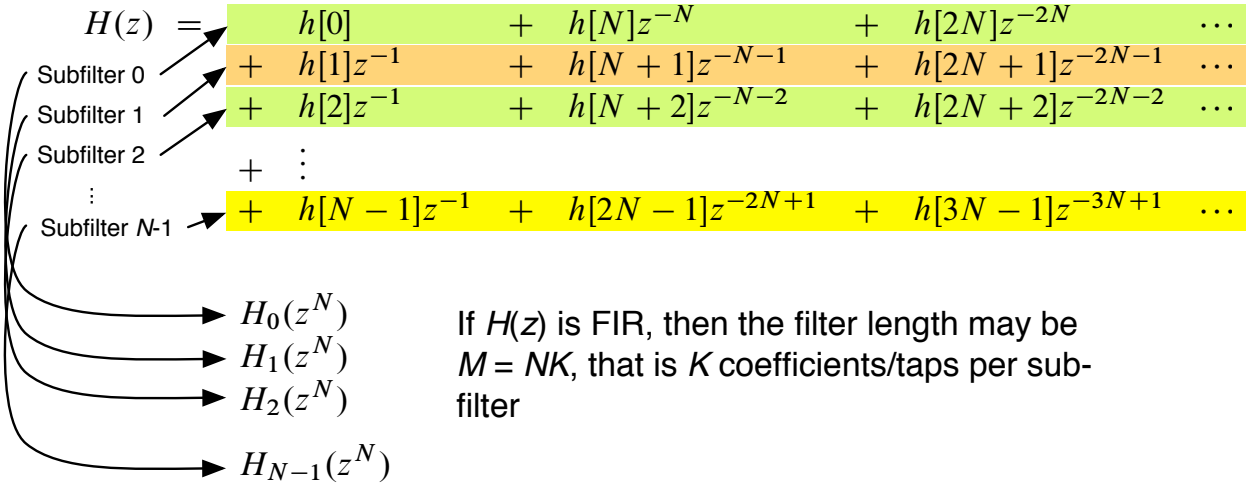
- The filter output via the convolution sum is

$$y[n] = \sum_{k=0}^{M} h[k]x[n-k]$$

  and the downsampled by N output is

$$y_D[n] = y[nN] = \sum_{k=0}^{M} h[k]x[nN-k]$$

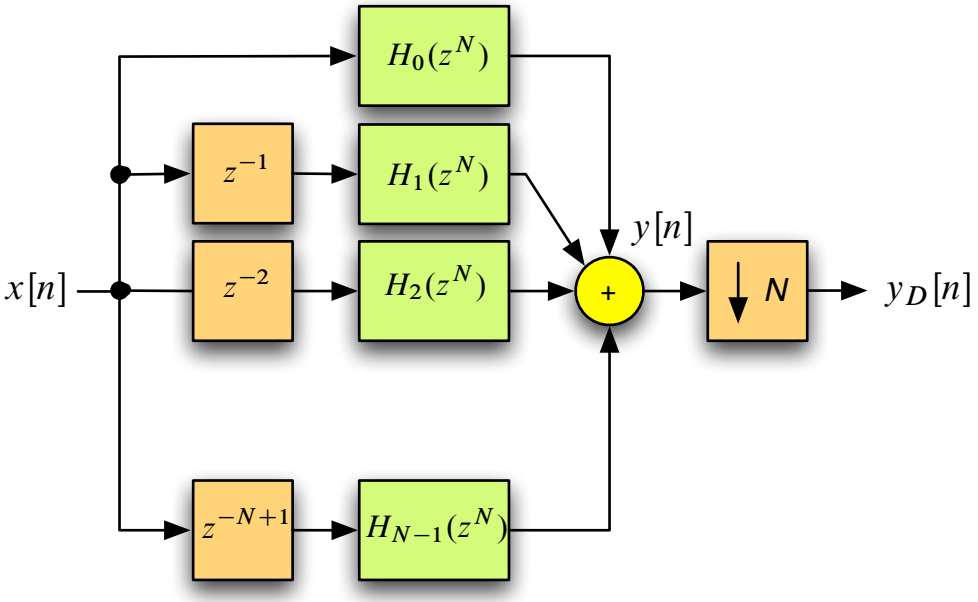
- Note that with downsampling only every Nth filter output need be calculated; $N-1$ of the outputs do not have to be calculated
- For any given filter output calculation only every Nth filter coefficient/tap is needed, the other coefficients are needed in-turn, for other outputs
- To see this more clearly, we can write $H(z)$ in a row-wise decomposition, where each row contains the filter coefficients needed for a particular output, i.e.,



If $H(z)$ is FIR, then the filter length may be $M = NK$, that is K coefficients/taps per sub-filter

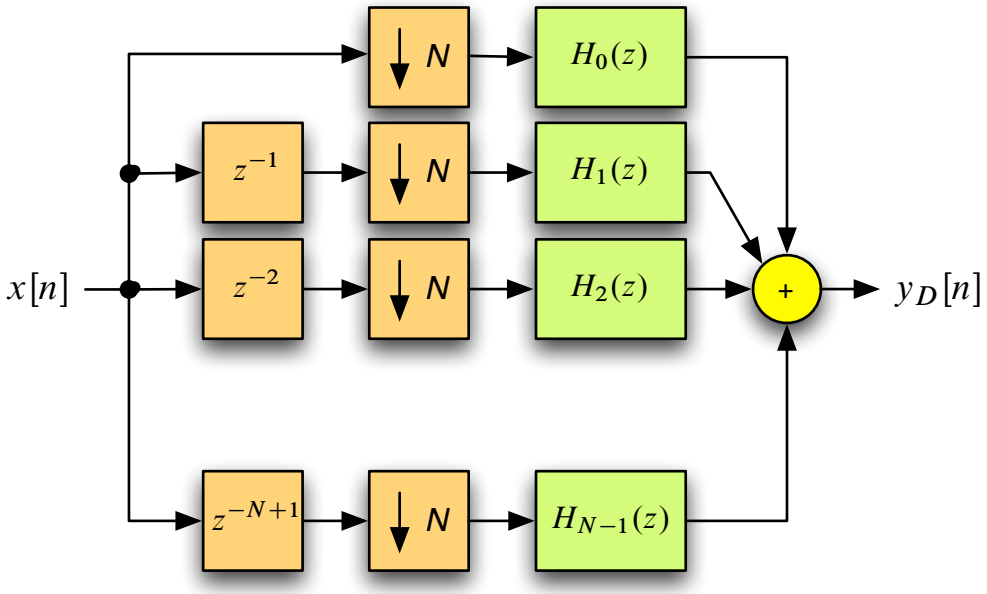
- Note that each filter is driven with a different offset or phase
- The sub filters can be combined to form $H(z)$ by including appropriate offsets

$$H(z) = H_0(z^N) + z^{-1}H_1(z^N) + \cdots + z^{-N+1}H_{N-1}(z^N)$$

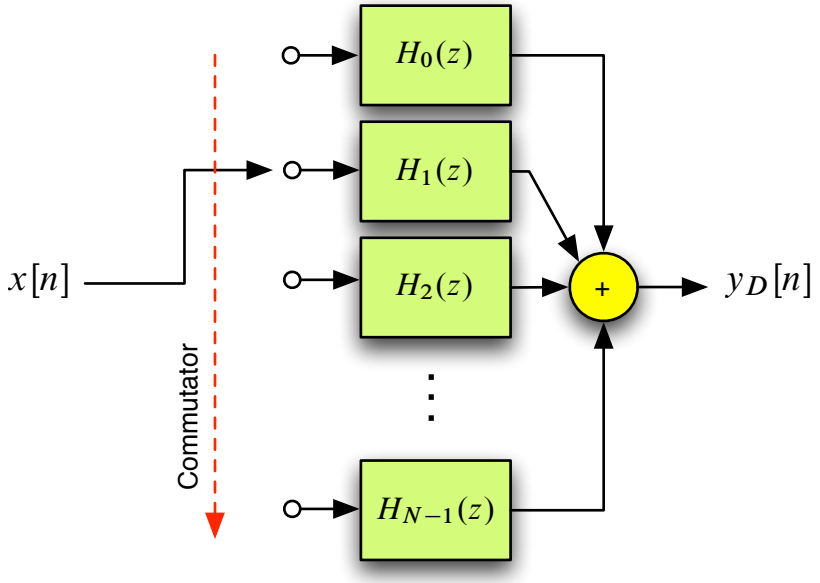


First step in polyphase downsample decomposition

- We can now invoke the downsampling identity and then a commutator



Polyphase decomposition with downsample identity



Polyphase downsample decomposition with commutator

- A polyphase upsampler can analyzed and constructed in like fashion
- The sub-filters are identical, with the filtering operations being the lower clock side, or input, of the upsampler

- Starting from the upsampler output we can write

$$y[n] = \sum_{k=0}^{M} h[k] x_U[n-k]$$

- Since $N-1$ zeros are inserted into $x_U[n]$ we know that not all of the multiplications in the above convolution sum are needed

- Given that the index n is a multiple of N, the nonzero output values coincide with

$$h[0] \quad h[N] \quad h[2N] \quad \cdots$$

and

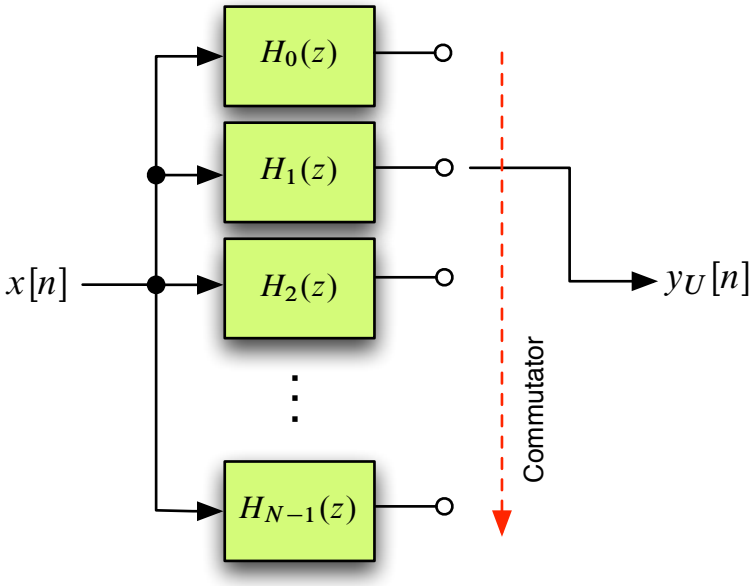
$$y[n] = \sum_{k=0}^{M} h[kN] x[n/N - kN]$$

- If n step forward to $n+1$ we have

$$y[n+1] = \sum_{k=0}^{M} h[kN+1] x[(n-1)/N - kN]$$

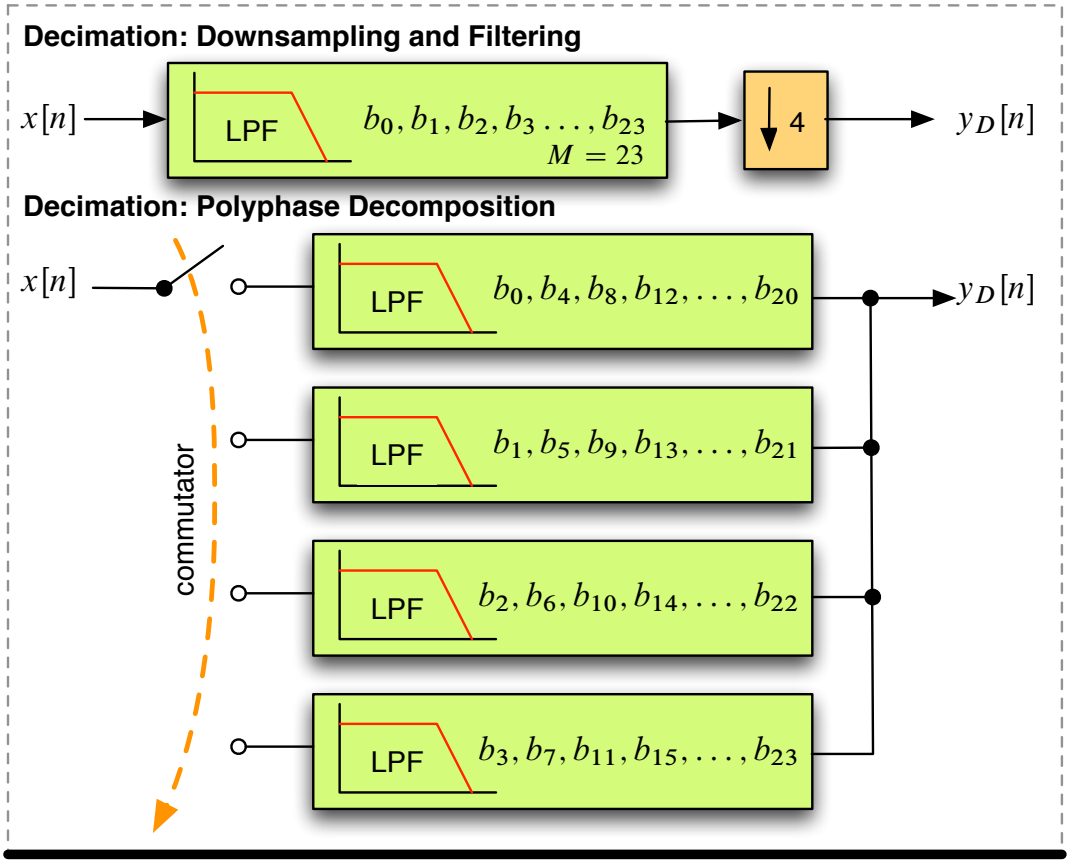
and so on

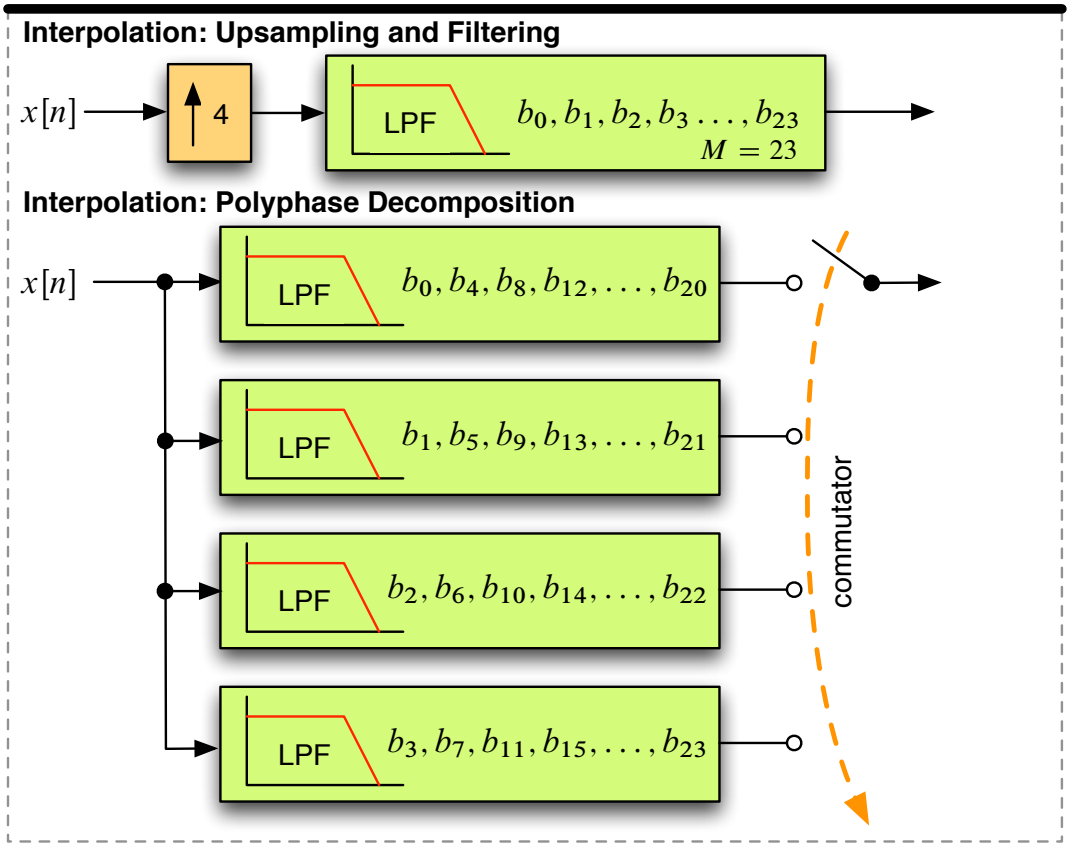
- Finally we see that a filterbank implementation using subfilters is possible, with all of the filtering operations being performed at the low sample rate



Polyphase upsample decomposition

Polyphase Decomposition Summary





2.2.12 Discrete-Time Filter Design Methods

This section of Rice includes details on IIR, FIR filter design. Differentiator and integrator algorithms are also discussed, as these are related to cascade of integrator and comb (CIC) structures that show up in chapters near the end of the text.

See the Rice text for more details.

Example 2.8: Polyphase Interpolator Design

- In this example we will use MATLAB to design an interpolator for $N = 5$
- MATLAB has toolboxes which can make the design of the interpolator very easy

  - With the *Signal Processing Toolbox*™ a wide array of filter design functions are available, including multirate functions
  - By adding the *Filter Design Toolbox*™ to the Signal Processing Toolbox™ multirate filter objects can be created, and in particular polyphase decomposition

- Below are some of the relevant functions available in the Signal Processing Toolbox™

FIR Filter Design

cfirpm	Complex and nonlinear-phase equiripple FIR filter design
fir1	Window-based finite impulse response filter design
fir2	Frequency sampling-based finite impulse response filter design
fircls	Constrained least square, FIR multiband filter design
fircls1	Constrained least square, lowpass and highpass, linear phase, FIR filter design
firls	Least square linear-phase FIR filter design
firpm	Parks-McClellan optimal FIR filter design
firpmord	Parks-McClellan optimal FIR filter order estimation
intfilt	Interpolation FIR filter design
kaiserord	Kaiser window FIR filter design estimation parameters
sgolay	Savitzky-Golay filter design

Communications Filters

firrcos	Raised cosine FIR filter design
gaussfir	Gaussian FIR pulse-shaping filter

Multirate Signal Processing

decimate	Decimation — decrease sampling rate
downsample	Decrease sampling rate by integer factor
interp	Interpolation — increase sampling rate by integer factor
resample	Change sampling rate by rational factor
upfirdn	Upsample, apply FIR filter, and downsample
upsample	Increase sampling rate by integer factor

A sampling of *Signal Processing Toolbox*™ functions

- We can use `intfilt()` to design an $N = 5$ interpolator low-pass filter, $H(z)$ to place in front of an $N = 5$ `upsampler()`
- The parameter `p` controls how many samples are used in the interpolation, and hence the length of the filter
- The parameter `alpha` controls the bandwidth of the stopband regions independent of the passband interpolation

intfilt

Interpolation FIR filter design

Syntax

```
b = intfilt(l,p,alpha)
b = intfilt(l,n,'Lagrange')
```

Description

`b = intfilt(l,p,alpha)` designs a linear phase FIR filter that performs ideal bandlimited interpolation using the nearest `2*p` nonzero samples, when used on a sequence interleaved with `l-1` consecutive zeros every `l` samples. It assumes an original bandlimitedness of alpha times the Nyquist frequency. The returned filter is identical to that used by `interp`. `b` is length `2*l*p-1`

alpha is inversely proportional to the transition bandwidth of the filter and it also affects the bandwith of the don't-care regions in the stopband. Specifying alpha allows you to specify how much of the Nyquist interval your input signal occupies. This is beneficial, particularly for signals to be interpolated, because it allows you to increase the transition bandwidth without affecting the interpolation and results in better stopband attenuation for a given `l` and `p`. If you set alpha to 1, your signal is assumed to occupy the entire Nyquist interval. Setting alpha to less than one allows for don't-care regions in the stopband. For example, if your input occupies half the Nyquist interval, you could set alpha to 0.5.

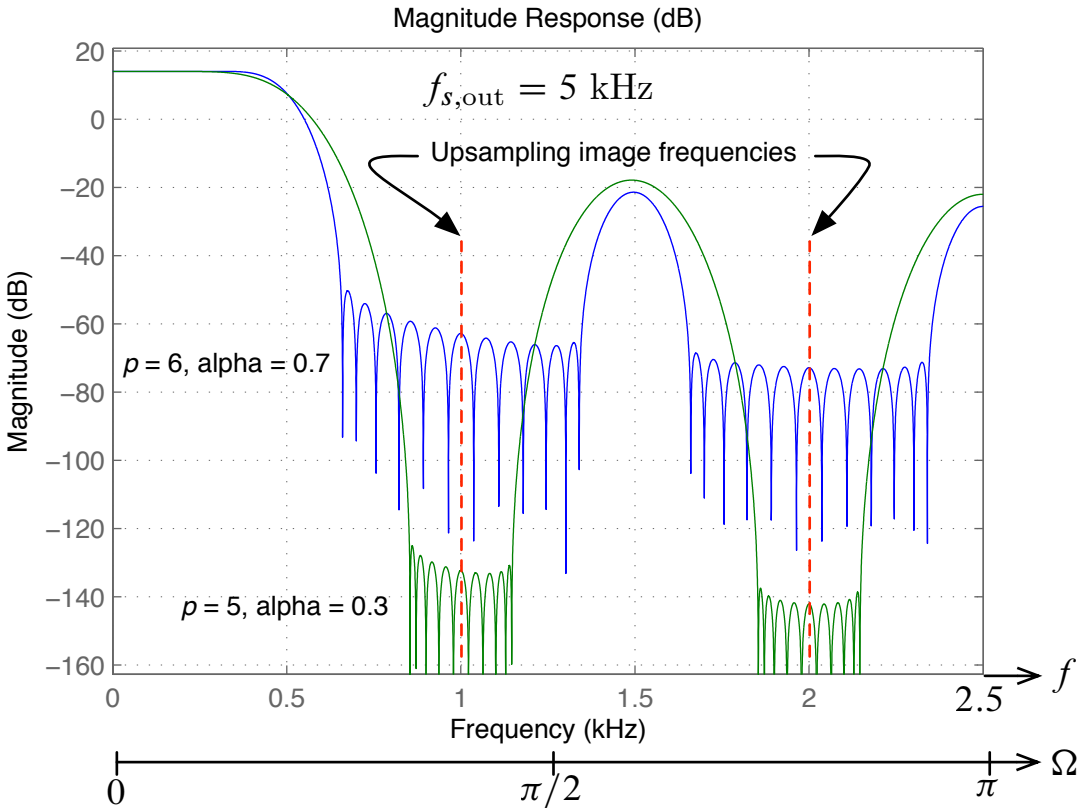
`b = intfilt(l,n,'Lagrange')` designs an FIR filter that performs nth-order Lagrange polynomial interpolation on a sequence interleaved with `l-1` consecutive zeros every `r` samples. `b` has length `(n + 1)*l` for `n` even, and length `(n + 1)*l-1` for `n` odd. If both `n` and `l` are even, the filter designed is not linear phase.

Both types of filters are basically lowpass and have a gain of `l` in the passband..

Design the filter using `intfil()`

- Here we compare two designs, $p = 5$ and $\alpha = 0.3$ versus $p = 6$ and $\alpha = 0.7$, where the resulting filter lengths are 49 and 59, respectively

```
>> bi = intfilt(5,5,0.3);
>> bii = intfilt(5,6,0.7);
>> fvtool(bii,1,bi,1,'fs',5*1000)
```



Interpolation filter frequency response assuming $f_s = 5 \times 1$ kHz

- With the filter now available the complete interpolator (upsampler plus lowpass interpolation filter) can be implemented in two lines of MATLAB code
- Here we will create a two sinusoid signal of 101 samples, create the upsampled signal as `yu` and the respective interpolated with $N = 5$ signals `yi` and `yii`

```
>> n = 0:100;
>> x = cos(2*pi*100/1000*n) + 2*sin(2*pi*10/1000*n);
>> yu = upsample(x,5);
>> yi = filter(bi,1,yu);
>> yii = filter(bii,1,yu);
```

- MATLAB supports object-oriented programming and in particular the Filter Design Toolbox™can crdate filter objects

- The filter objects of interest here are multirate filters, which support a variety of architectures and both floating point and fixed point implementations

Multirate Filters

mfilt.cascade	Cascade filter objects
mfilt.cicdecim	Fixed-point CIC decimator
mfilt.cicinterp	Fixed-point CIC interpolator
mfilt.farrowsrc	Sample rate converter with arbitrary conversion factor
mfilt.fftfirinterp	Overlap-add FIR polyphase interpolator
mfilt.firdecim	Direct-form FIR polyphase decimator
mfilt.firfracdecim	Direct-form FIR polyphase fractional decimator
mfilt.firfracinterp	Direct-form FIR polyphase fractional interpolator
mfilt.firinterp	FIR filter-based interpolator
mfilt.firsrc	Direct-form FIR polyphase sample rate converter
mfilt.firtdecim	Direct-form transposed FIR filter
mfilt.holdinterp	FIR hold interpolator
mfilt.iirdecim	IIR decimator
mfilt.iirinterp	IIR interpolator
mfilt.iirwdfdecim	IIR wave digital filter decimator
mfilt.iirwdfinterp	IIR wave digital filter interpolator
mfilt.linearinterp	Linear interpolator

Relevant multirate *Filter Design Toolbox*™ functions

- For this example we are interested in `mfilt.firinterp` which creates a polyphase FIR interpolator object

- This object will by default create its own FIR interpolation filter, but the user may also pass in a custom filter, say `hi` or `hii` as designed earlier

mfilt.firinterp

FIR filter-based interpolator

Syntax

```
Hm = mfilt.firinterp(L)
Hm = mfilt.firinterp(L,num)
```

Description

`Hm = mfilt.firinterp(L)` returns a FIR polyphase interpolator object `Hm` with an interpolation factor of `L` and gain equal to `L`. `L` defaults to 2 if unspecified.

`Hm = mfilt.firinterp(L,num)` uses the values in the vector `num` as the coefficients of the interpolation filter.

Make this filter a fixed-point or single-precision filter by changing the value of the `Arithmetic` property for the filter `Hm` as follows:

- To change to single-precision filtering, enter

```
set(hm,'arithmetic','single');
```

- To change to fixed-point filtering, enter

```
set(hm,'arithmetic','fixed');
```

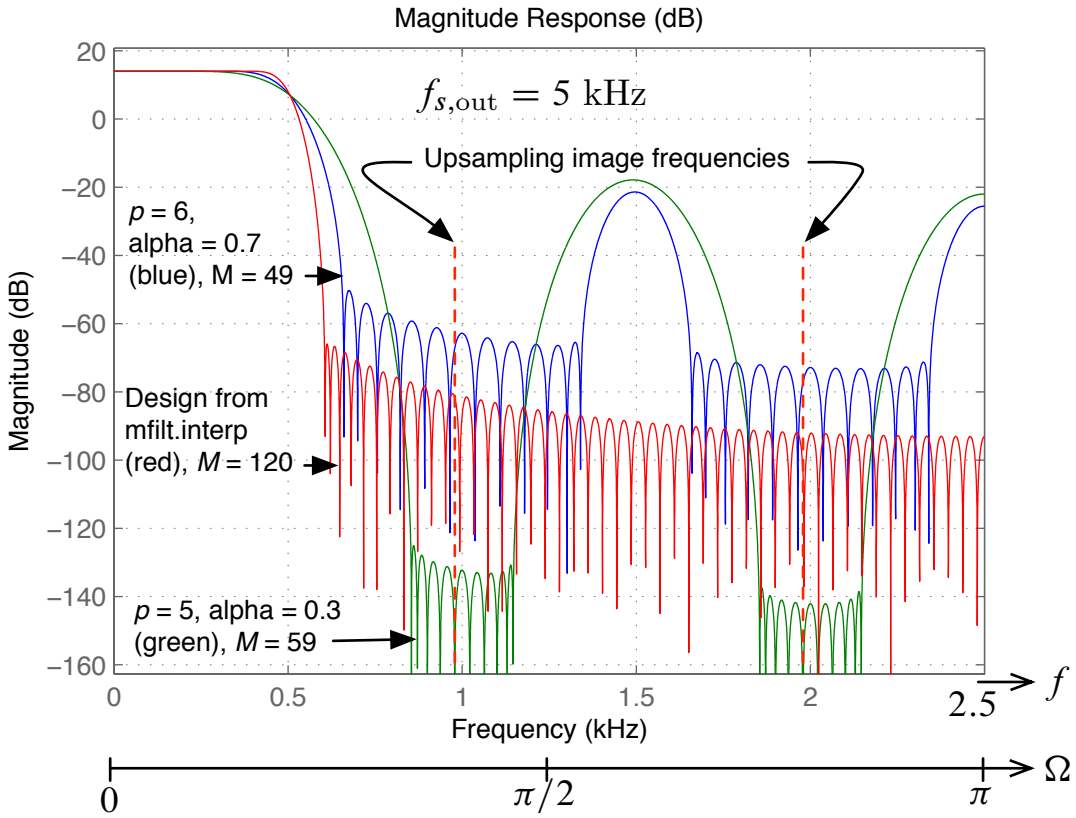
Design a multirate filter object using `mfilt.firinterp`

- Once the filter object is created it can be further manipulated via *set* and *get* methods, to estabilsh how it will behave when used with the `filter()` function
- Upsampling aspect of the object is automatically taken into account when we use `filter()`

```
>> Hm = mfilt.firinterp(5);
>> Hm % See the various fields contained within the Hm object
Hm =

        FilterStructure: 'Direct-Form FIR Polyphase Interpolator'
             Arithmetic: 'double'
              Numerator: [1x120 double]    %<--FIR filter coef. reside here
    InterpolationFactor: 5
       PersistentMemory: false
>> fvtool(bii,1,bi,1,Hm.Numerator,'fs',5*1000) % overlay all 3 filters
>> print -tiff -depsc all_fvtool.eps
```

```
>> % Obtain the frequency response of the default
>> y = filter(Hm,x);
```



Overlay frequency response plot of all three interpolation filters

- At this point we could go on and observe the waveforms `y`, `yi`, and `yii`
- We would find that in all cases the interpolation is very good
- The signals would be delayed by one half the filter length, e.g., 120/2 = 60 samples

.