Section 1

Digital Filter Introduction

Subsection 1

Introduction

Digital Filter Applications

- A filter is a system that selectively changes characteristics of an input signal. Common examples of filtering operations include:
  - Noise suppression - Many useful signals are usually contaminated by noise. For example, RF communications signals, physiological signals such as ECG and image signals received by image sensors.
  - Bandwidth limiting - Used, for example, to prevent aliasing and to prevent interference with neighbouring channels in communication applications, such as commercial radio and television.
  - Enhancement of certain frequency ranges - Examples of this application include graphics equalizers in audio systems and edge enhancement in images (amplifying high frequency components).
  - Removal or attenuation of specific frequencies - Examples include blocking DC or 60 Hz interference in a signal.
  - Special operations - Such as differentiation, integration and Hilbert transform.

Types of Digital Filters

- There are two major types of linear time-invariant (LTI) digital filters and they are usually specified by their z-domain transfer function, which is usually rational and causal,

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{a_0 + a_1 z^{-1} + \cdots + a_N z^{-N}} \tag{1}$$

where $a_0 = 1$.

FIR Filters

- Digital filters where $N = 0$ in $H(z)$ are called finite impulse response (FIR) filters with transfer function,

$$H(z) = b_0 + b_1 z^{-1} + \cdots + b_M z^{-M} \tag{2}$$

- The impulse response, $h[n]$, is the inverse z-transform of equation (2),

$$h[n] = b_0 \delta[n] + b_1 \delta[n-1] + \cdots + b_M \delta[n-M]$$

which is also represented as $[b_0, b_1, \cdots, b_M]$.

- The impulse response is finite length, thus the name finite impulse response.

Types of Digital Filters

- The input, $x[n]$, and output, $y[n]$, signals of an FIR filter are related by the convolution sum

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

which for FIR filters is also the difference equation for the system.

IIR Filters

- Digital filters, where $N \geq 1$ in (1), are called infinite impulse response (IIR) filters.
- The impulse response, $h[n]$, could be found using a partial fraction decomposition of $H(z)$, but this is a relatively long procedure and the result is not too useful. Though it would demonstrate that $h[n]$ has infinite length.

Types of Digital Filters

- The input, $x[n]$, and output, $y[n]$, signals of an IIR filter are related by the convolution sum (similar to the FIR difference equation),

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

- Though, in practice, it is not possible to compute this expression, since the sum is infinite.
- Instead a recursive form is used to define the input-output relationship.
- This recursive difference equation can be directly generated from $H(z) = Y(x)/X(z)$ (equation (1)),

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k]$$

, where a_k and b_k are coefficients of the filter.

Example 1: IIR Filter

An IIR filter (known as a comb filter) is defined by the following coefficient vectors

$$\begin{aligned} b &= [1, 0, 0, 0.2^3] \\ a &= [1, 0, 0, 0, 0, 0.98^5] \end{aligned}$$

1. Determine the difference equation of the filter.
   Answer: $y[n] = x[n] + 0.2^3 x[n-3] - 0.98^5 y[n-5]$
2. Determine an expression for the impulse response of the filter.
   Answer: $h[n] = \delta[n] + 0.2^3 \delta[n-3] - 0.98^5 h[n-5]$
3. Calculate the first 11 elements of $h[n]$.
   Answer: $h[n] = [1, 0, 0, 0.2^3, 0, -0.98^5, 0, 0, -0.98^5 * 0.2^3, 0, (-0.98^5)^2]$
4. Plot the first 50 elements of $h[n]$ using MATLAB and compare with part a).
   Answer:

```
b = [ 1 0 0 0.2^3];
a = [ 1 0 0 0 0 0.98^5];
h= filter(b,a,[1 zeros(1,49)]);
stem([0:49],h)
ylabel('Impulse Response'), xlabel('n (samples)')
```

Example 1: IIR Filter

5. Plot the frequency response of the filter using MATLAB. Plot the magnitude (dB) and phase vs ω on one figure and the linear magnitude vs f on another figure.

   Answer (first figure):

```
b = [ 1 0 0 0.2^3];
a = [ 1 0 0 0 0 0.98^5];
figure(1)
freqz(b,a)
```

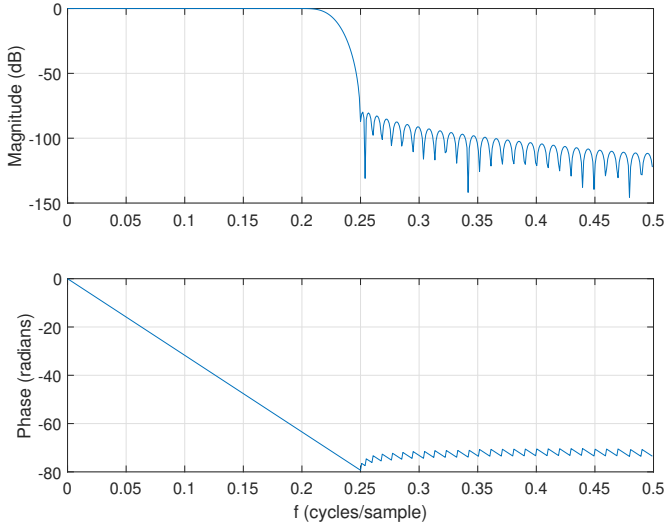
   Answer (second figure):

```
b = [ 1 0 0 0.2^3];
a = [ 1 0 0 0 0 0.98^5];
[H,w] = freqz(b,a);
figure(2)
plot(w/(2*pi),abs(H))
ylabel('Magnitude')
xlabel('f (cycles/sample)')
```

Subsection 2

Distortionless Filters

What is a Distortionless Filter?

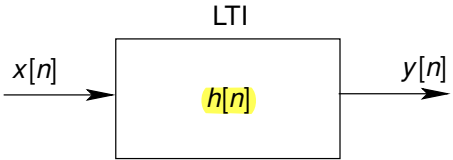


A Distortionless Filter Is

- A filter is distortionless in the passband, if the output signal of the filter has the same shape as the input signal.
- This type of filter can only amplitude scale and/or delay the signal in the passband. In terms of the frequency response of the filter, these conditions are
  - the magnitude response must have a constant gain across the passband,
  - the phase response must be linear (or equivalently, the group delay must be constant) in the passband.

Example 2: Linear Phase System

In this example, we want to examine the effect of a filter's phase response on the shape of a signal waveform. The filter of interest is a linear time-invariant (LTI) system, where $h[n]$ is the impulse response of the filter.



- Define the input $x[n]$ as the sum of two or more sinusoids

$$x[n] = \sum_{k=1}^{K} x_k[n]$$

where $x_k[n] = \sin(\omega_k n)$.

Example 2: Linear Phase System

- Assume there are only two sinusoids in the sum,

$$x[n] = x_1[n] + x_2[n] = \sin(\omega_1 n) + \sin(\omega_2 n),$$

where $\omega_k = 2\pi f_k$, and $f_1 = 1/80$ cycles/sample and f_2 is the next odd harmonic, ie. $f_2 = 3/80$ cycles/sample.

- The output, $y[n]$, can be determined by recalling (see O&S 3rd ed, example 2.15, page 42) that if $h[n]$ is real, the output is

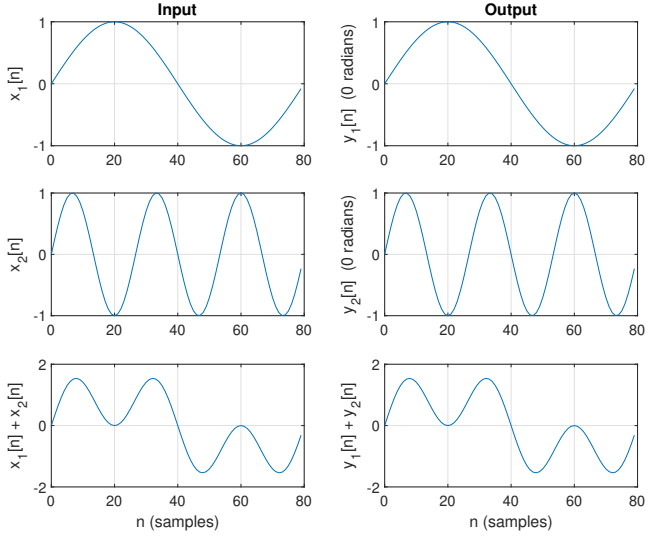
$$y[n] = |H(e^{j\omega_1})| \sin(\omega_1 n + \angle H(e^{j\omega_1}))$$

if the input is

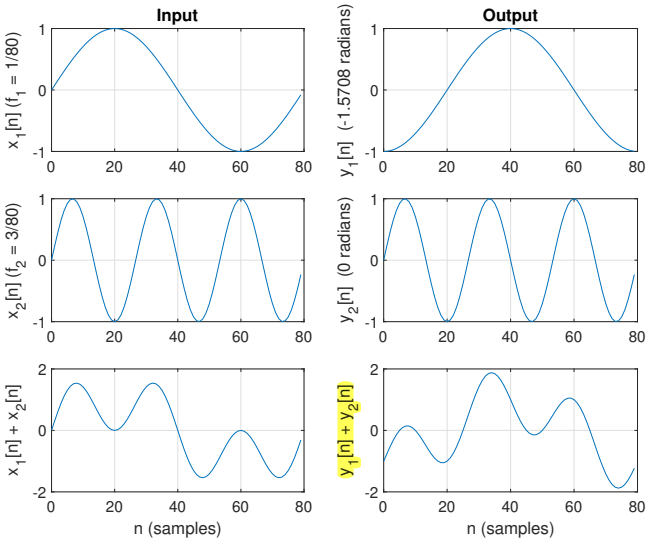
$$x[n] = \sin(\omega_1 n)$$

- Only interested in phase, so set $|H(e^{j\omega})| = 1$.
- User defined MATLAB function used: lp_demo(phase,num_sinusoids)

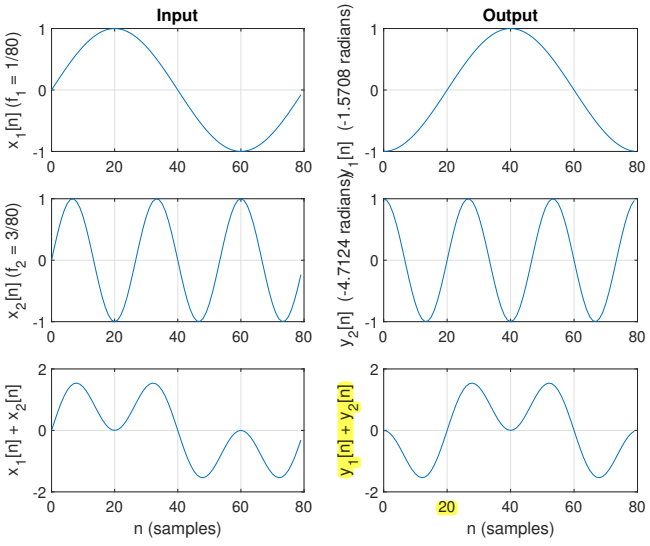
Example 2: Figure Generated with lp_demo([0,0],2)



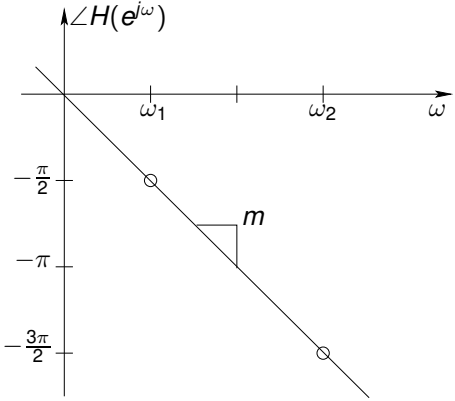
Example 2: Figure Generated with lp_demo([-π/2,0],2)



Example 2: Figure Generated with lp_demo([-π/2,-3π/2],2)



Example 2: Plotting the Phase Response



What is the group delay, τ_g, for the linear phase response?

Example 2: Constant Group Delay in the Passband

- In the previous slide, the group delay was calculated as constant value of 20 samples.
- A constant group delay is a pure delay of signals in the passband.
- For this example, the phase shifts of each sinusoid in the passband for a pure delay of 20 samples can be calculated as

$$\begin{aligned} y[n] &= x[n-20] \\ &= x_1[n-20] + x_2[n-20] \\ &= \sin(\omega_1(n-20)) + \sin(\omega_2(n-20)) \\ &= \sin(2\pi n/80 - \pi/2) + \sin(2\pi n 3/80 - 3\pi/2) \end{aligned}$$

- This demonstrates that a 20 sample delay of the input signal requires $y_1[n]$ to have a phase shift of $-\pi/2$, and $y_2[n]$ to have a phase shift of $-3\pi/2$.

Example 2: Frequency Response

- Recall, for this example, $y[n] = x[n - 20]$, and taking the z-transform gives

$$Y(x) = X(z)z^{-20}$$

- Rearrange to generate the transfer function

$$H(z) = \frac{Y(z)}{X(z)} = z^{-20}$$

- The frequency response is determined by substituting $z = e^{j\omega}$:

$$H(e^{j\omega}) = |H(e^{j\omega})|e^{j\theta_M(\omega)} = e^{-j20\omega}$$

- This can also be put in the form

$$H(e^{j\omega}) = e^{-j\tau_g\omega}$$

where, for a linear phase response, τ_g is a constant, which is the negative of the slope of the phase response. τ_g is referred to as the group delay.

Example 2: Linear Phase System Summary

- A filter that has a linear phase response and constant gain results in a pure delay of the input signal. In this example, $y[n] = x[n - 20]$.
- If the gain is one, the frequency response of a linear phase filter is

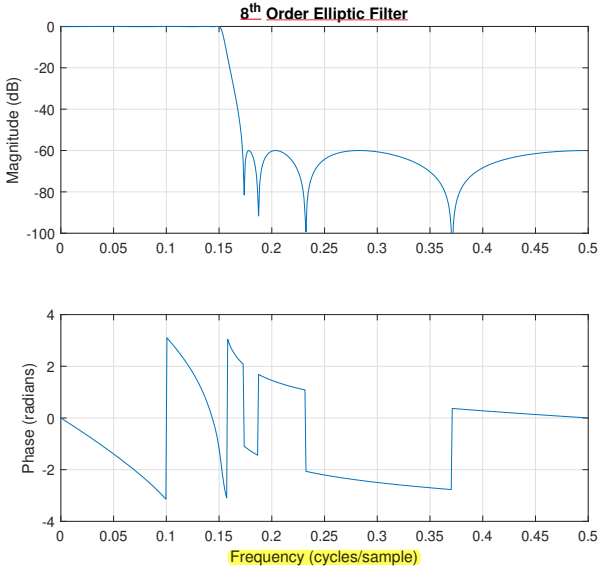
$$H(e^{j\omega}) = e^{-j\tau_g \omega}$$

  where τ_g is the negative of the slope of the phase response and is referred to as the group delay.
- In this example, $\tau_g = 20$ samples.
- Linear phase is important for applications that depend on the shape of the signal, such as communications systems.
- A linear phase response is relatively easy to interpret, but you probably can imagine that a non-linear phase response is more difficult to interpret. This will be explored in the next section.

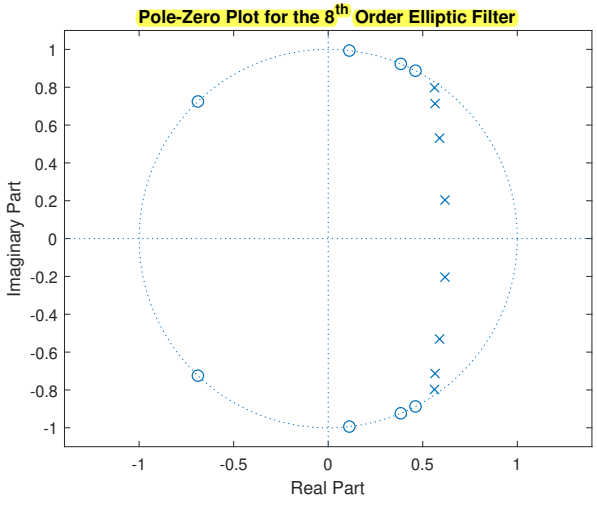
Subsection 3

Generating a Continuous Phase Response

Frequency Response of an 8th Order Elliptic Filter



Pole Zero Plot for the 8th Order Elliptic Filter



Causes of Phase Response Discontinuities

- There are two types of discontinuities, one results in a jump of π radians and the other a jump of 2π radians.
- The π discontinuity results from the magnitude response being defined as non-negative, and it can be eliminated by using the amplitude response, $A(\omega)$, which can take on negative values, instead of the magnitude response, $|H(e^{j\omega})|$, which only can have non-negative values.
- The 2π discontinuity results from wrapping the phase, which is commonly done by software tools, such as MATLAB, to keep the phase in a range of $-\pi$ to π, and it can be eliminated by simply unwrapping the phase (in MATLAB the command is `unwrap`).
- The frequency response, specified in terms of the amplitude response and the continuous phase response is

$$H(e^{j\omega}) = A(\omega)e^{j\theta_A(\omega)}$$

Example 3: System with a π Discontinuity

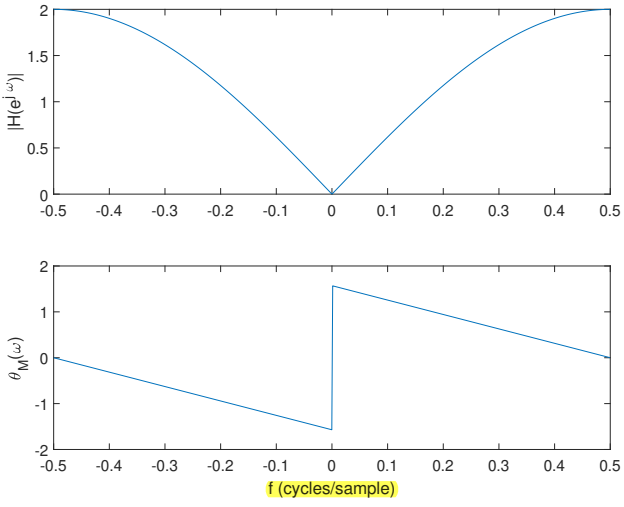
Given the system, $y[n] = x[n] - x[n-1]$, plot the linear magnitude response, $|H(e^{j\omega})|$, and phase response, $\theta_M(\omega)$, using MATLAB.

- **Solution**: Using the impulse response, $h[n] = \delta[n] - \delta[n-1]$, the MATLAB code that will generate a plot of the frequency response is:

```
h=[1 -1]; % system impulse response
[H,w]=freqz(h,1,[-pi:.01:pi]);
subplot(2,1,1)
plot(w/(2*pi),abs(H))
ylabel('|H(e^{j \omega})|')
subplot(2,1,2)
plot(w/(2*pi), angle(H))
ylabel('\theta_M(\omega)')
xlabel('f (cycles/sample)')
```

- Note that the ylabel in this code uses latex notation to generate the exponent and the greek symbol.

Example 3: System with a π Discontinuity



Example 3: System with a π Discontinuity

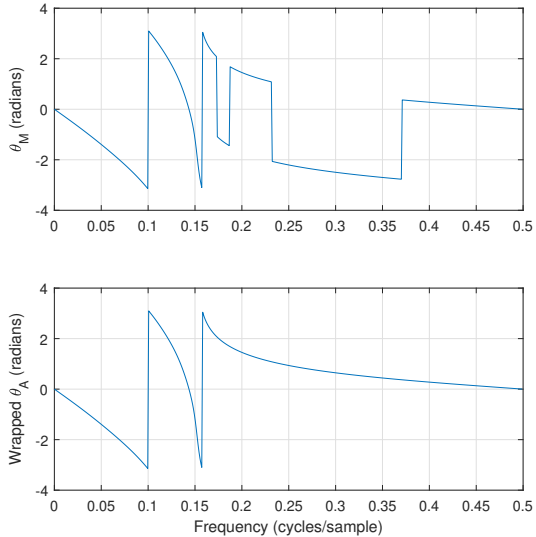
Given the system, $y[n] = x[n] - x[n-1]$, generate expressions for the two forms of the frequency response:

$$H(e^{j\omega}) = |H(e^{j\omega})|e^{j\theta_M(\omega)}$$

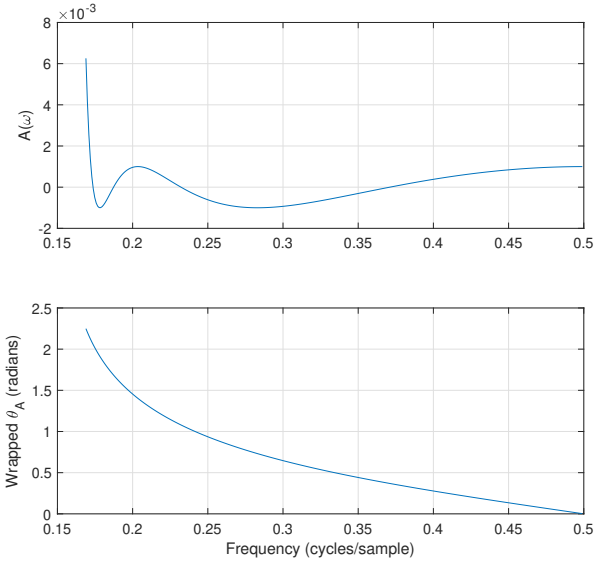
and

$$H(e^{j\omega}) = A(\omega)e^{j\theta_A(\omega)}$$

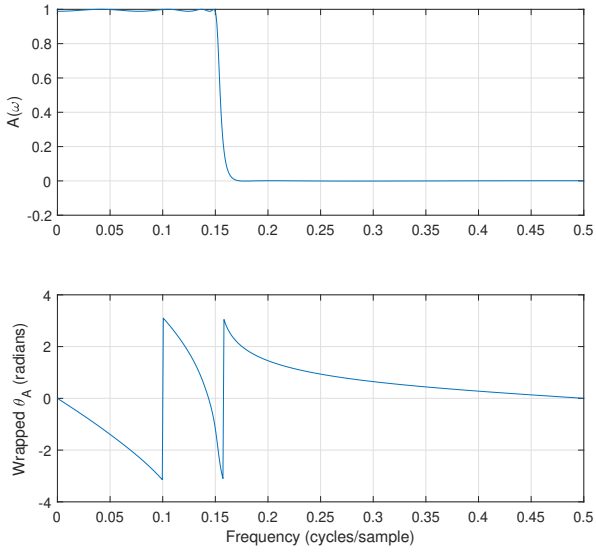
Phase Response with π Shifts Removed



Zoomed Amplitude and Phase Responses with π Shifts Removed



Amplitude and Phase Responses with π Shifts Removed



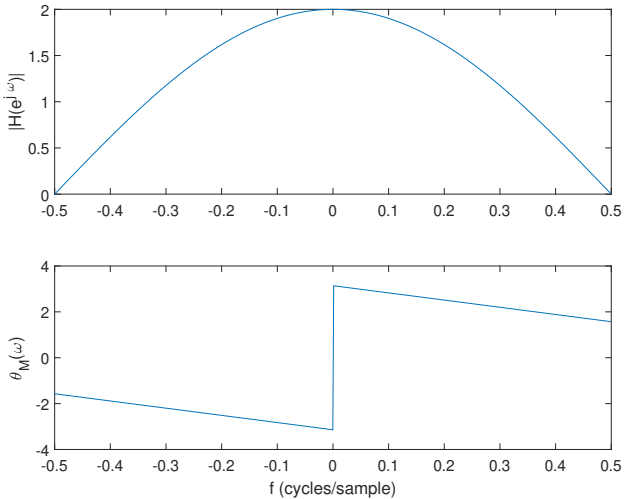
Example 4: System with a 2π Discontinuity

Given the system, $y[n] = -x[n] - x[n-1]$, plot the linear magnitude response, $|H(e^{j\omega})|$, and phase response, $\theta_M(\omega)$, using MATLAB.

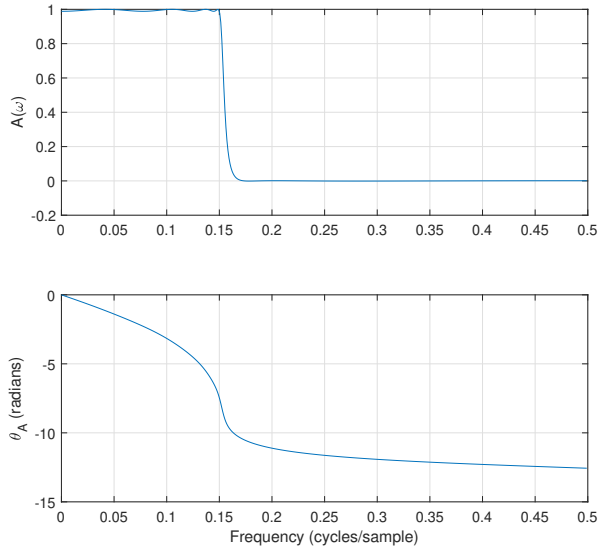
- **Solution**: Using the impulse response, $h[n] = -\delta[n] - \delta[n-1]$, the MATLAB code that will generate a plot of the frequency response is:

```
h=[-1 -1]; % system impulse response
[H,w]=freqz(h,1,[-pi:.01:pi]);
subplot(2,1,1)
plot(w/(2*pi),abs(H))
ylabel('|H(e^{j \omega})|')
subplot(2,1,2)
plot(w/(2*pi), angle(H))
ylabel('\theta_M(\omega)')
xlabel('f (cycles/sample)')
```

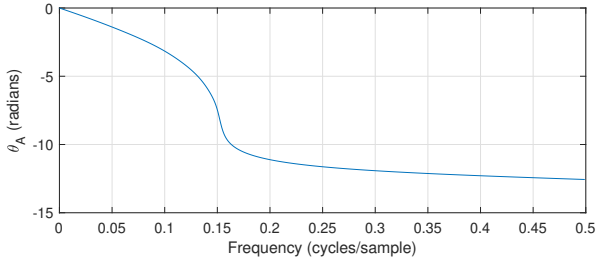
Example 4: System with a 2π Discontinuity

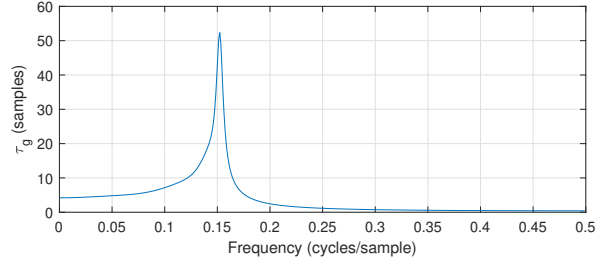


Continuous Phase Response



Group Delay of the 8th Order Elliptic Filter, $\tau_g = -\frac{d\theta_A(\omega)}{d\omega}$





Calculating the Amplitude Response

The DTFT of $h[n]$, $H(e^{j\omega})$ can be represented as

$$H(e^{j\omega}) = A(\omega)e^{j\theta_A(\omega)}.$$

Multiplying both sides by $e^{-j\theta_A(\omega)}$ gives

$$A(\omega) = H(e^{j\omega})e^{-j\theta_A(\omega)}.$$

Thus if $H(e^{j\omega})$ could be determined using MATLAB, $A(\omega)$ could then be calculated.
$H(e^{j\omega})$ could be evaluated at equally spaced frequencies around the unit circle using the DFT (Discrete Fourier Transform).

$$H(e^{j\omega})\big|_{\omega=2\pi k/N} = \underbrace{\sum_{n=0}^{N-1} h[n]e^{-j\omega n}}_{\text{DTFT}}\Bigg|_{\omega=2\pi k/N} = \underbrace{\sum_{n=0}^{N-1} h[n]e^{-j2\pi kn/N}}_{\text{DFT}}$$

Note, the DFT is typically represented as

$$H[k] = \sum_{n=0}^{N-1} h[n]e^{-j2\pi kn/N}$$

Calculating the Amplitude Response

If $h[n]$ has a low number of elements (ie. N is small), then $H[k]$ will not have enough elements to produce a good representation of $H(e^{j\omega})$ in a plot. Thus, typically, $h[n]$ is zero padded to obtain a better plot for $H(e^{j\omega})$. Define the length, L, zero padded signal as

$$h_z[n] = \begin{cases} h[n]; & 0 \leq n \leq N-1 \\ 0; & N \leq n \leq L-1 \end{cases}$$

Thus

$$H_z(e^{j\omega})\big|_{\omega=2\pi k/L} = \sum_{n=0}^{L-1} h_z[n] e^{-j2\pi kn/L} = \sum_{n=0}^{N-1} h_z[n] e^{-j2\pi kn/L}$$

Note that in MATLAB, the DFT is implemented as the fast Fourier transform, fft.
Using the the zero padded sequence, the amplitude response is

$$A(\omega)|_{\omega=2\pi k/L} = H_z(e^{j\omega}) e^{-j\theta_A(\omega)}\Big|_{\omega=2\pi k/L} = \underbrace{H_z[k]}_{\text{DFT}} e^{-j\theta_A(\omega)}\Big|_{\omega=2\pi k/L}$$

Note that the fft calculates the response for $k = 0$ to $L-1$, which is from 0 to $2\pi(L-1)/L$ in radians/sample. Or from 0 to $(L-1)/L$ in cycles/sample. This is one period of the response.

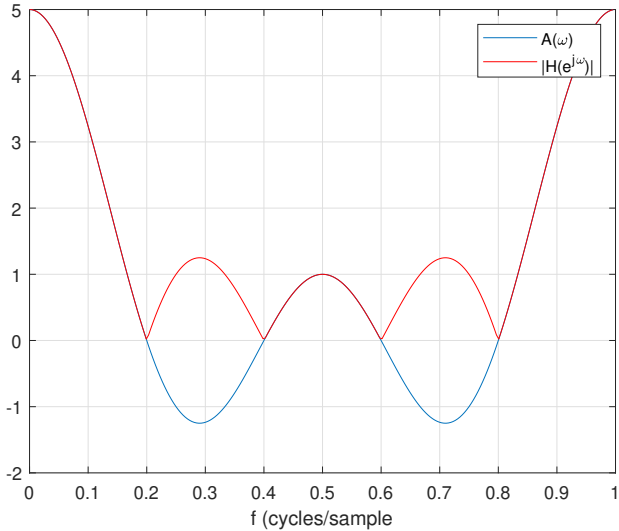
Example 5: Use MATLAB to Calculate $A(\omega)$

Calculate and plot $A(\omega)$ for the system with impulse response $h[n] = [1, 1, 1, 1, 1]$.

- **Solution:** Choose $L = 256$, a power of 2, for the length of the zero padded sequence. $h[n]$ is even symmetric, with a constant group delay of $\tau_g = M/2$, where $M = 4$ is the order of the filter (one less than the length of the filter). The phase response is $\theta_A(\omega) = -\tau_g\omega = -\frac{M}{2}\omega = -2\omega$. The MATLAB solution is

```
h=ones(1,5); L=256; hz=[h, zeros(1,L-length(h))];
Hz=fft(hz); k=0:L-1;
theta_A=-2*2*pi*k/L; W=exp(-j*theta_A);
A=Hz.*W;
A=real(A); %imaginary part is very close to zero
f=k/L; % cycles/sample
clf, plot(f,A); hold
plot(f, abs(Hz),'r')
legend('A(\omega)','|H(e^{j\omega})|')
xlabel('f (cycles/sample'), grid
```

Example 5: Use MATLAB to Calculate $A(\omega)$



Summary: Continuous Phase Response

- There are two types of discontinuities, one results in a jump of π radians and the other a jump of 2π radians.
- The two causes of discontinuities in a phase response are:
  1. The magnitude response is defined as non-negative.
  2. The phase response is typically defined for the range $-\pi$ to π (thus tools, such as MATLAB, wrap the phase).
- The first discontinuity can be eliminated by using the amplitude, $A(\omega)$, and associated continuous phase response, $\theta_A(\omega)$, instead of the magnitude, $|H(e^{j\omega})|$, and associated phase response (which is usually discontinuous), $\theta_M(\omega)$, as given in the following frequency response expressions:

$$\begin{aligned} H(e^{j\omega}) &= |H(e^{j\omega})|e^{j\theta_M(\omega)} \\ &= A(\omega)e^{j\theta_A(\omega)} \end{aligned}$$

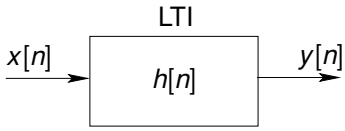
- The difference between these two is that the magnitude, $|H(e^{j\omega})|$, is non-negative and the amplitude, $A(\omega)$, can be negative.
- The second type of discontinuity can be removed by unwrapping the phase (in MATLAB use unwrap).

Subsection 4

Types of Linear Phase Systems

Linear Phase Systems

The frequency response of the output of this system is $Y(e^{j\omega}) = H(e^{j\omega})X(e^{j\omega})$.



Using the continuous phase notation,

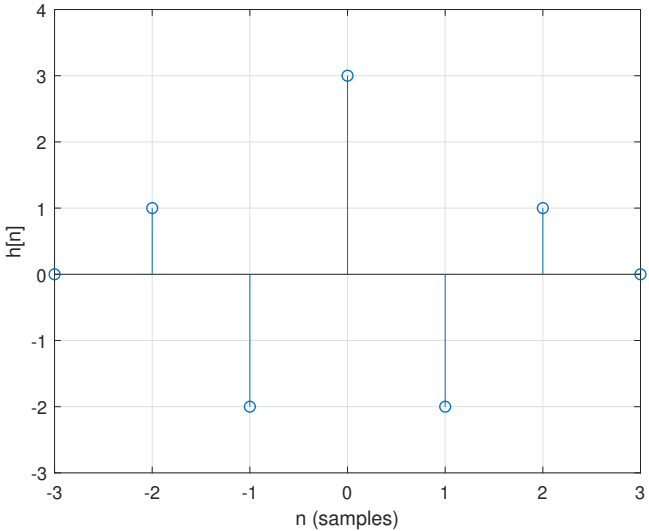
$$\begin{aligned} Y(e^{j\omega}) &= A_Y(\omega)e^{j\theta_Y(\omega)} \\ H(e^{j\omega}) &= A(\omega)e^{j\theta_A(\omega)} \\ X(e^{j\omega}) &= A_X(\omega)e^{j\theta_X(\omega)} \end{aligned}$$

Thus

$$\begin{aligned} A_Y(\omega) &= A(\omega)A_X(\omega) \\ \theta_Y(\omega) &= \theta_A(\omega) + \theta_X(\omega) \end{aligned}$$

Focusing on the phase, if $\theta_A(\omega)$ is nonzero, it will distort the input signal. Thus, ideally a zero phase is desired, $\theta_A(\omega) = 0$. But, practically, this is not possible, as explored in the next slides.

System with a Zero Phase Response



System with a Zero Phase Response

- The impulse response of the system defined in the previous figure is

$$h[n] = \delta[n+2] - 2\delta[n+1] + 3\delta[n] - 2\delta[n-1] + \delta[n-2]$$

- $h[n]$ is symmetrical about $n = 0$ and thus it is an even function.
- An even function is defined as

$$h[n] = h[-n]$$

- Recall that the Fourier transform of an even function is real, thus there is no imaginary term in the frequency response, resulting in the phase response being 0.
- This can be demonstrated by evaluating the frequency response

$$H(e^{j\omega}) = 3 - 4\cos(\omega) + 2\cos(2\omega) = A(\omega)e^{j\theta_A(\omega)}$$

- Thus $\theta_A(\omega) = 0$, and also the group delay, $\tau_g = 0$.

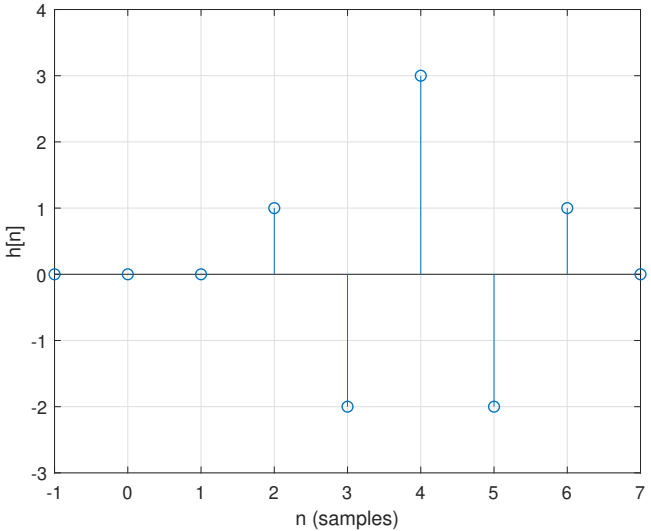
Example 6: Zero Phase System

Derive the frequency response of

$$h[n] = \delta[n+2] + \delta[n+1] + \delta[n] + \delta[n-1] + \delta[n-2]$$

Plot $h[n]$ and use the DTFT to determine the frequency response, $H(e^{j\omega}) = A(\omega)e^{j\theta_A}$.

Even Symmetric Impulse Response



Even Symmetric Impulse Response

- The impulse response of the system defined in the previous figure is

$$h[n] = \delta[n-2] - 2\delta[n-3] + 3\delta[n-4] - 2\delta[n-5] + \delta[n-6]$$

- $h[n]$ is even about $n = 4$ and thus it is referred to as an even symmetric function, which is defined as

$$h[n] = h[M-n]$$

where M is the order of the system ($M = N-1$, where N is the length of $h[n]$). Here M has a value 8.

- An even symmetric $h[n]$ has a linear phase response and an even amplitude response:

$$\theta_A(\omega) = -\frac{M}{2}\omega \quad \text{and} \quad A(\omega) = A(-\omega)$$

- This can be demonstrated by evaluating the frequency response,

$$H(e^{j\omega}) = (3 - 4\cos(\omega) + 2\cos(2\omega))e^{-j4\omega} = A(\omega)e^{j\theta_A(\omega)}$$

- Thus $\theta_A(\omega) = -4\omega$ and $A(\omega) = 3 + 4\cos(\omega) + 2\cos(2\omega)$

Example 7: Even Symmetric System

Determine the frequency response of $h[n] = \delta[n-1] + \delta[n-2] + \delta[n-3] + \delta[n-4] + \delta[n-5]$.

Example 8: Even Symmetric Impulse Response with Odd Order

Given a system $h[n] = \delta[n-2] + \delta[n-3] + \delta[n-4] + \delta[n-5]$, determine

1. the value that $h[n]$ is even symmetric about,
2. the system frequency response in the form of an amplitude response, $A(\omega)$ and phase response, $\theta_A(\omega)$.

Summary: Even Symmetric $h[n]$

A system that has an impulse response that is even symmetric about $M/2$ (M could be even or odd), which is defined by the relationship

$$h[n] = h[M - n],$$

is a linear phase system with a phase response given by

$$\theta_A = -\frac{M}{2}\omega,$$

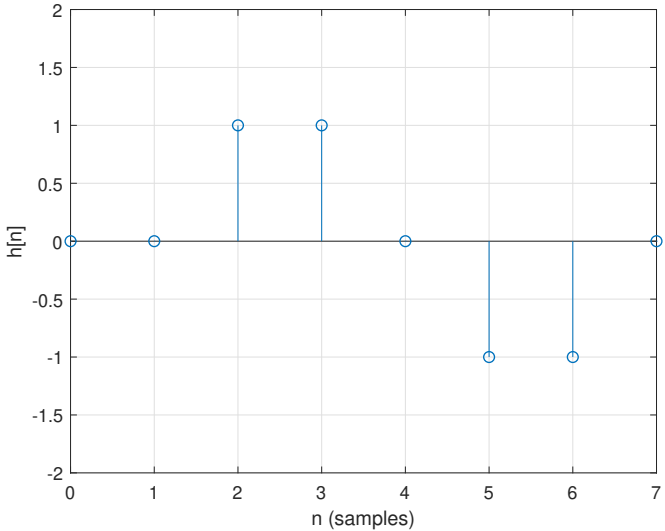
and group delay

$$\tau_g = \frac{M}{2}.$$

The amplitude response for $h[n]$ is an even function,

$$A(\omega) = A(-\omega).$$

Odd Symmetric Impulse Response



Odd Symmetric Impulse Response

- The impulse response of the system defined in the previous figure is

$$h[n] = \delta[n-2] + \delta[n-3] - \delta[n-5] - \delta[n-6]$$

- $h[n]$ is an odd about $n = 4$ and thus it is referred to as an odd symmetric function, which is defined as

$$h[n] = -h[M-n]$$

where M is the order of the system ($M = N-1$, where N is the length of $h[n]$). M is even with value 8.

- An odd symmetric $h[n]$ has a **generalized** linear phase response and an odd amplitude response:

$$\theta_A(\omega) = \frac{\pi}{2} - \frac{M}{2}\omega \text{ and } A(\omega) = -A(-\omega)$$

- This can be demonstrated by determining the frequency response,

$$H(e^{j\omega}) = (2sin(2\omega) + 2sin(\omega))e^{j(\pi/2-4\omega)}$$

- Thus $\theta_A(\omega) = \pi/2 - 4\omega$ and $A(\omega) = 2sin(2\omega) + 2sin(\omega)$

Example: Odd Symmetric Impulse Response with Odd Order

Given a system $h[n] = \delta[n] + \delta[n-1] - \delta[n-2] - \delta[n-3]$, determine the frequency response.

Summary: Odd Symmetric $h[n]$ (Generalized Linear Phase)

A system that has an impulse response that is odd symmetric about $M/2$ (M could be even or odd), which is defined by the relationship

$$h[n] = -h[M - n],$$

is referred to as a generalized linear phase system with a phase response given by

$$\theta_A(\omega) = \frac{\pi}{2} - \frac{M}{2}\omega,$$

and group delay

$$\tau_g = \frac{M}{2}.$$

The amplitude response for $h[n]$ is an odd function,

$$A(\omega) = -A(-\omega).$$

Section 2

Basics of FIR Filters

Subsection 1

FIR Filter Introduction

FIR Filter Characteristics

FIR filters have a number of desirable characteristics when compared to IIR filter,

- FIR filters have linear phase or generalized linear phase, and thus they do not distort the input signal shape,
- FIR filters are always stable,
- There are a number of versatile methods for designing FIR filters.

On the other hand,

- FIR filters are higher order and thus have a more complex implementation compared to IIR, thus if a linear phase response is not required, IIR filters are more efficient.
- The group delay for FIR filters can be large, resulting in a large delay between input and output.

FIR Filter Types

Practical FIR filters are designed to have linear phase and there are four types of linear phase filters:

$$H(e^{j\omega}) = A(\omega)e^{j(\beta - \frac{M}{2}\omega)}$$

Type	**1**	**2**	**3**	**4**
Order (M)	even	odd	even	odd
$h[n]$	even symmetric	even symmetric	odd symmetric	odd symmetric
$A(\omega)$	even	even	odd	odd
$A(\omega)$ period	2π	4π	2π	4π
β	0	0	$\pi/2$	$\pi/2$
$H(e^{j0})$	arbitrary	arbitrary	0	0
$H(e^{j\pi})$	arbitrary	0	0	arbitrary
Common Uses	LP, HP, BP, BS, Mulitband	LP, BP	Differentiator, Hilbert Transform	Differentiator, Hilbert Transform

Note: Type 3 could be used for bandpass filters and Type 4 could be used for highpass and bandpass, but they are not commonly used for these types of filters, since they have generalized linear phase, not linear phase.

Type 1 Filter Frequency Response

Type 1 filters have even order, M, phase term $\beta = 0$, and a even symmetric impulse response

$$h[n] = h[M - n]$$

The frequency response is derived using the DTFT

$$\begin{aligned} H(e^{j\omega}) &= \sum_{n=0}^{M} h[n] e^{-j\omega n} \\ &= e^{-j\omega M/2} \sum_{n=0}^{M} h[n] e^{j\omega(M/2-n)} \\ &= e^{-j\omega M/2} \left[h[\tfrac{M}{2}] + \sum_{n=0}^{M/2-1} h[n] e^{j\omega(M/2-n)} + \sum_{n=M/2+1}^{M} h[n] e^{j\omega(M/2-n)} \right] \end{aligned}$$

Take the third term in the above equation and substitute $l = M - n$. Replace n for l. Substitute $h[n] = h[M - n]$. Replace the third term with the new expression, to give,

$$H(e^{j\omega}) = e^{-j\omega M/2} \left[h[\frac{M}{2}] + \sum_{n=0}^{M/2-1} h[n] e^{j\omega(M/2-n)} + \sum_{n=0}^{M/2-1} h[n] e^{-j\omega(M/2-n)} \right]$$

Type 1 Filter Frequency Response

Using Euler's and substituting $l = M/2 - n$

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

This can be put in the form

$$H(e^{j\omega}) = e^{-j\omega M/2}\sum_{k=0}^{M/2} a[k]\cos(\omega k)$$

where

$$a[k] = \begin{cases} h[M/2]; & k = 0 \\ 2h[M/2 - k]; & k = 1, 2, \cdots M/2 \end{cases}$$

Type 1 Filter Frequency Response

- The frequency response of a type 1 FIR filter is

$$A(\omega) = \sum_{k=0}^{M/2} a[k] \cos(\omega k) \quad \text{and} \quad \theta_A(\omega) = -\omega \frac{M}{2}$$

where

$$a[k] = \begin{cases} h[M/2]; & k = 0 \\ 2h[M/2 - k]; & k = 1, 2, \cdots M/2 \end{cases}$$

- $A(\omega)$ is even about $\omega = 0$ and $\omega = \pi$, since the cosine terms are even at these ω locations.
- $A(\omega)$ is also periodic with a period of 2π, since the fundamental period of the cosine terms is 2π.

Example 9: Type 1 System

Given

$$H(z) = 1 + 0.4z^{-1} - 0.4z^{-2} + 1.2z^{-3} - 0.4z^{-4} + 0.4z^{-5} + z^{-6},$$

is this filter a Type 1 filter, and if yes, what is the frequency response?

Type 2 Filter Frequency Response

Type 2 filters have odd order. They have a symmetric impulse response,

$$h[n] = h[M - n],$$

and thus the phase term $\beta = 0$. The frequency response can derived using an approach similar to the Type 1 derivation, to give

$$H(e^{j\omega}) = e^{-j\omega M/2} \sum_{k=1}^{(M+1)/2} b[k] \cos(\omega(k - 1/2)$$

where

$$b[k] = 2h[(M + 1)/2 - k]; \quad k = 1, 2, \cdots (M + 1)/2$$

Type 2 Filter Frequency Response

Using the results of the previous slide, the frequency response of a Type 2 FIR filter is

$$A(\omega) = \sum_{k=1}^{(M+1)/2} b[k] \cos(\omega(k - 1/2))$$

$$\theta_A(\omega) = -\omega \frac{M}{2}$$

where

$$b[k] = 2h[(M+1)/2 - k]; \quad k = 1, 2, \cdots (M+1)/2$$

- $A(\omega)$ is even about $\omega = 0$ and odd about $\omega = \pi$, since the $\cos(\omega(k - 1/2))$ terms are even about $\omega = 0$ and they are odd about $\omega = \pi$.
- $A(\omega)$ is also periodic with a period of 4π, since the fundamental period of $\cos(\omega/2)$ $(k = 1)$ is 4π.

Type 3 Filter Frequency Response

Type 3 filters have even order. They have a odd symmetric impulse response,

$$h[n] = -h[M-n],$$

and thus the phase term $\beta = \pi/2$. The frequency response can derived using an approach similar to the Type 1 derivation, to give

$$H(e^{j\omega}) = e^{\pi/2 - j\omega M/2} \sum_{k=1}^{M/2} c[k] \sin(\omega k)$$

where

$$c[k] = 2h[M/2-k]; \quad k = 1, 2, \cdots M/2$$

Type 3 Filter Frequency Response

Using the results of the previous slide, the frequency response of a Type 3 FIR filter is

$$A(\omega) = \sum_{k=1}^{M/2} c[k] \sin(\omega k)$$

$$\theta_A(\omega) = \frac{\pi}{2} - \frac{M}{2}\omega$$

where

$$c[k] = 2h[M/2 - k]; \quad k = 1, 2, \cdots M/2$$

- $A(\omega)$ is odd about $\omega = 0$ and $\omega = \pi$, since $\sin(\omega k)$ is odd about these ω locations.
- $A(\omega)$ is also periodic with a period of 2π, since the fundamental period of $\sin(\omega k)$ ($k = 1$) is 2π.

Type 4 Filter Frequency Response

Type 4 filters have odd order. They have a odd symmetric impulse response,

$$h[n] = -h[M-n],$$

and thus the phase term $\beta = \pi/2$. The frequency response can derived using an approach similar to the Type 1 derivation, to give

$$H(e^{j\omega}) = e^{j(\pi/2-\omega M/2)} \sum_{k=1}^{(M+1)/2} d[k] \sin(\omega(k-1/2))$$

where

$$d[k] = 2h[(M+1)/2-k]; \quad k = 1, 2, \cdots (M+1)/2$$

Type 4 Filter Frequency Response

Using the results of the previous slide, the frequency response of a Type 4 FIR filter is

$$\begin{aligned} A(\omega) &= \sum_{k=1}^{(M+1)/2} d[k] \sin(\omega(k-1/2)) \\ \theta_A(\omega) &= \frac{pi}{2} - \omega\frac{M}{2} \end{aligned}$$

where

$$d[k] = 2h[(M+1)/2 - k]; \quad k = 1, 2, \cdots (M+1)/2$$

- $A(\omega)$ is odd about $\omega = 0$ and even about $\omega = \pi$, since the $\sin(\omega(k-1/2))$ terms are even about $\omega = 0$ and they are odd about $\omega = \pi$.
- $A(\omega)$ is also periodic with a period of 4π, since the fundamental period of $\sin(\omega/2)$ ($k = 1$) is 4π.

Filter Type Defined Zero Locations

The frequency response of a Type 2 FIR filter always has a zero at $\omega = \pi$. This can be demonstrated by example.

First note that a Type 2 filter is even symmetric with odd order. One such system is $h[n] = [h_0, h_1, h_2, h_2, h_1, h_0]$. Taking the z-transform of $h[n]$ gives

$$H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + h_2 z^{-3} + h_1 z^{-4} + h_0 z^{-5}$$

Evaluating $H(z)$ at $z = -1$ gives

$$H(-1) = h_0 - h_1 + h_2 - h_2 + h_1 - h_0 = 0$$

Thus

$$H(e^{j\pi}) = H(-1) = 0$$

Similarly, rules can be derived for Type 3 and 4 filters

Type	Defined Zeros
1	none
2	$\omega = \pi$
3	$\omega = 0, \pi$
4	$\omega = 0$

Summary of FIR Filter Types

$$H(e^{j\omega}) = A(\omega)e^{j(\beta - \frac{M}{2}\omega)}$$

Type	**1**	**2**	**3**	**4**
Order (M)	even	odd	even	odd
$h[n]$	even symmetric	even symmetric	odd symmetric	odd symmetric
$A(\omega)$	even (about $\omega = 0$)	even	odd (about $\omega = 0$)	odd
$A(\omega)$*period*	2π	4π	2π	4π
β	0	0	$\pi/2$	$\pi/2$
$H(e^{j0})$	arbitrary	arbitrary	0	0
$H(e^{j\pi})$	arbitrary	0	0	arbitrary
Common Uses	LP, HP, BP, BS, Mulitband	LP, BP	Differentiator, Hilbert Transform	Differentiator, Hilbert Transform

FIR Filter Zero Locations

The zero locations for the all-zero linear phase FIR filters are not arbitrary. The zero locations are constrained and possible locations are derived as follows:
The symmetry condition for linear phase filters is

$$h[n] = \pm h[M - n]$$

and in the z-domain

$$H(z) = \pm z^{-M} H(\frac{1}{z}).$$

If z_0 is a zero of $H(z)$, $H(z_0) = 0$ and $H(z_0^*) = 0$, since roots of real coefficient polynomials are complex conjugate pairs, then

$$\begin{aligned} H(z_0) &= \pm z_0^{-M} H(\tfrac{1}{z_0}) = 0 \\ H(z_0^*) &= \pm (z_0^*)^{-M} H(\tfrac{1}{z_0^*}) = 0 \end{aligned}$$

and thus

$$H(\frac{1}{z_0}) = H(\frac{1}{z_0^*}) = 0$$

FIR Filter Zero Locations

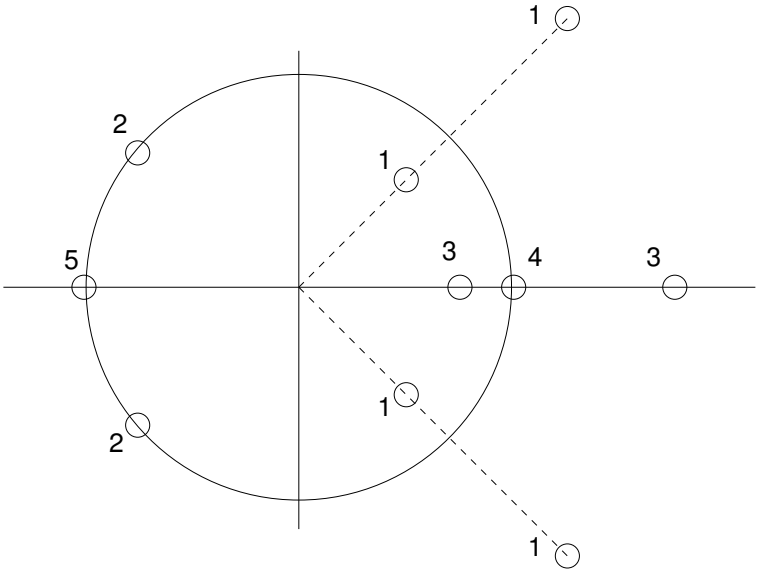
$$H(\frac{1}{z_0}) = H(\frac{1}{z_0^*}) = 0$$

indicates that if z_0 is a zero of a real valued linear phase filter, then so are z_0^*, $1/z_0$ and $1/z_0^*$.
It follows that

1. generic zeros of a linear phase filter exist in sets of 4 ($z_0 = re^{j\phi}$,$z_0^* = re^{-j\phi}$, $1/z_0 = r^{-1}e^{-j\phi}$ and $1/z_0^* = r^{-1}e^{j\phi}$)
2. zeros on the unit circle exist in sets of 2 ($z_0 = e^{\pm j\phi}$, $z_0 \neq \pm 1$).
3. zeros on the real line exist in sets of 2 ($z_0 = r, 1/r$).
4. a zero at $z_0 = 1$ has no other conditional zero locations.
5. a zero at $z_0 = -1$ has no other conditional zero locations.

The figure on the following slide is a plot of the above 5 possible zero locations.

FIR Filter Zero Locations



Subsection 2

Impulse Response Truncation

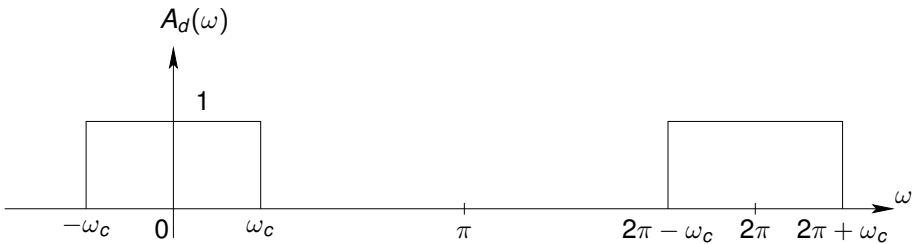
Introduction to the IRT Design Method

- FIR filter design usually involves calculating values for $h[n]$ that meet a desired $H(e^{j\omega})$ specification.
- There are a number of different FIR design methods that will be explored, starting with the simplest method, called impulse response truncation (IRT).
- The IRT method is introduced by first considering a version that involves delaying the impulse response.
- This delay IRT method involves the following steps:
  - Specify the desired ideal filter (with zero phase response),
  - Generate the infinite impulse response using the Inverse DTFT (IDTFT),
  - Truncate the impulse response.
  - Delay (shift) the impulse response to make it causal.
- The delay IRT method is demonstrated in the example on the following slide.

Example 10: LPF Design using the Delay IRT Method

Design a lowpass filter with cutoff frequency, ω_c, and order M.

- **Solution:** The ideal frequency response for this lowpass filter is (assume a zero phase response)



- Recall from the table summarizing the four filter types on slide 70, only types 1 and 2 can be lowpass filters. Choose Type 1 for this example.
- Take the IDTFT of $A_d(\omega)$ to calculate the non-causal impulse response, $h_{nc}[n]$ (recall the frequency response is zero phase).

Example 10: LPF Design using the Simplified IRT Method

$$h_{nc}[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} A_d(\omega)e^{j\omega n}d\omega = \frac{1}{2\pi}\int_{-\pi}^{\pi} \underbrace{A_d(\omega)}_{even}(\cos(\omega n) + j\underbrace{\sin(\omega n)}_{odd})d\omega$$

- The above expression can be simplified by noting that an even function times an odd function is odd and the integral of an odd function is 0.

$$h_{nc}[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} A_d(\omega)\cos(\omega n)d\omega = \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} \cos(\omega n)d\omega$$

- Evaluating the integral gives

$$h_{nc}[n] = \begin{cases} \frac{1}{2\pi}2\omega_c; & n = 0 \\ \frac{1}{2\pi n}\sin(\omega n)\Big|_{-\omega_c}^{\omega_c}; & \text{otherwise} \end{cases}$$

- and simplifying

$$h_{nc}[n] = \begin{cases} \frac{\omega_c}{\pi}; & n = 0 \\ \frac{\omega_c}{\pi}\frac{\sin(\omega_c n)}{\omega_c n}; & \text{otherwise} \end{cases}$$

Example 10: LPF Design using the Delay IRT Method

- Truncate and delay $h_{nc}[n]$ to make it causal, call this new sequence $h[n]$.

$$h[n] = h_{nc}\left[n - \frac{M}{2}\right] = \begin{cases} \frac{\omega_c}{\pi}; & n = 0 \\ \frac{\omega_c}{\pi}\frac{\sin(\omega_c(n-\frac{M}{2}))}{\omega_c(n-\frac{M}{2})}; & n = 1, 2, \cdots, M \end{cases}$$

- Using $\omega_c = 2\pi f_c$,

$$h[n] = \begin{cases} 2f_c; & n = 0 \\ 2f_c\frac{\sin(2\pi f_c(n-\frac{M}{2}))}{2\pi f_c(n-\frac{M}{2})}; & n = 1, 2, \cdots, M \end{cases}$$

- Using $\mathrm{sinc}(x) = \sin(\pi x)/(\pi x)$

$$h[n] = 2f_c\mathrm{sinc}\left(2f_c\left(n - \frac{M}{2}\right)\right); \quad n = 0, 1, 2, \cdots, M$$

IRT Procedure

- In the previous example, the impulse response for the non-causal system, $h_{nc}[n]$, was generated and then it was shifted by $\frac{M}{2}$ to make it causal ($h[n] = h_{nc}\left[n - \frac{M}{2}\right]$). This restricts M to an even number, since a discrete-time sequence can only be shifted an integer number of samples.
- An alternative approach, which avoids the even number constraint for M, is to incorporate a linear phase term into the frequency response,

$$A(\omega)e^{-j\omega M/2}$$

- After taking the IDTFT, the resulting impulse response, once truncated to length $N = M + 1$, will be causal.
- This is implemented in the impulse response truncation filter design procedure listed on the next slide.

IRT Procedure

1. Specify the desired ideal amplitude response $A_d(\omega)$, ie. LP, HP, etc.
2. Choose the phase characteristics: integer delay (even M) or fractional delay (odd M) and initial phase $\beta = 0$ (for even symmetric $h[n]$) or $\pi/2$ (for odd symmetric $h[n]$).
3. Choose the filter order M (even for Type 1 or 3, odd for Type 2 or 4).
4. The resulting ideal desired frequency response is

$$H_d(e^{j\omega}) = A_d(\omega)e^{j(\mu\frac{\pi}{2} - \frac{M}{2}\omega)}$$

   where $\mu = 0$ for even symmetric $h[n]$ and $\mu = 1$ for odd symmetric $h[n]$.
5. Compute the impulse response of the ideal filter using the IDTFT,

$$h_d[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} A_d(\omega)e^{j(\mu\frac{\pi}{2} - \omega M/2)}e^{j\omega n}d\omega.$$

6. Truncate the impulse response

$$h[n] = \begin{cases} h_d[n]; & 0 \le n \le M \\ 0; & \text{otherwise} \end{cases}$$

IRT: Lowpass, Highpass and Bandpass Filters

- A common form for the impulse response can be derived for the three types of filters: lowpass, highpass and bandpass.
- Define the desired amplitude response

$$A_d(\omega) = \begin{cases} 1; & \omega_1 \leq |\omega| \leq \omega_2 \\ 0; & \text{otherwise} \end{cases}$$

- For $A_d(\omega)$,

$$\begin{array}{lcl} \text{if } \omega_1 = 0 & \rightarrow & \text{a LPF} \\ \text{if } \omega_2 = \pi & \rightarrow & \text{a HPF} \\ \text{otherwise} & \rightarrow & \text{a BPF} \end{array}$$

- Referring to the table on slide 70, only the Type 1 filter can be a LP, HP or BP.
- A Type 1 filter has $\beta = 0$, thus $\theta_A(\omega) = -\omega M/2$, where M is the filter order. Thus the desire impulse response is

$$H_d(e^{j\omega}) = \begin{cases} e^{-j\omega M/2}; & \omega_1 \leq |\omega| \leq \omega_2 \\ 0; & \text{otherwise} \end{cases}$$

IRT: Lowpass, Highpass and Bandpass Filters

- Taking the IDTFT

$$\begin{aligned}
h_d[n] &= \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega})e^{j\omega n}d\omega \\
&= \frac{1}{2\pi}\int_{-\omega_2}^{-\omega_1} e^{-j\omega M/2}e^{j\omega n}d\omega + \frac{1}{2\pi}\int_{\omega_1}^{\omega_2} e^{-j\omega M/2}e^{j\omega n}d\omega \\
&= \frac{1}{2\pi}\int_{-\omega_2}^{-\omega_1} e^{j\omega(n-M/2)}d\omega + \frac{1}{2\pi}\int_{\omega_1}^{\omega_2} e^{j\omega(n-M/2)}d\omega \\
&= \frac{1}{2\pi j(n-M/2)}\left(e^{-j\omega_1(n-M/2)} - e^{-j\omega_2(n-M/2)}\right) + \frac{1}{2\pi j(n-M/2)}\left(e^{j\omega_2(n-M/2)} - e^{j\omega_1(n-M/2)}\right) \\
&= \frac{1}{2\pi j(n-M/2)}\left(e^{j\omega_2(n-M/2)} - e^{-j\omega_2(n-M/2)}\right) - \frac{1}{2\pi j(n-M/2)}\left(e^{j\omega_1(n-M/2)} - e^{-j\omega_1(n-M/2)}\right)
\end{aligned}$$

- Using Euler's

$$h_d[n] = \frac{\sin(\omega_2(n-M/2))}{\pi(n-M/2)} - \frac{\sin(\omega_1(n-M/2))}{\pi(n-M/2)}$$

- Putting this in terms of f_1 and f_2, and using $\mathrm{sinc}(x) = \sin(\pi x)/(\pi x)$ gives,

$$h_d[n] = 2f_2\mathrm{sinc}(2f_2(n-M/2)) - 2f_1\mathrm{sinc}(2f_1(n-M/2))$$

IRT: Lowpass, Highpass and Bandpass Filters

- Truncate to a length of $N = M + 1$, to give the BPF

$$h[n] = 2f_2\text{sinc}(2f_2(n - M/2)) - 2f_1\text{sinc}(2f_1(n - M/2)) \;\; n = 0, 1, \cdots, M$$

- For a LPF, with $f_2 = f_c$ and $f_1 = 0$,

$$h[n] = 2f_c\text{sinc}(2f_c(n - M/2))$$

- For a HPF, with $f_2 = f_c$ and $f_1 = 1/2$,

$$h[n] = \text{sinc}(n - M/2) - 2f_c\text{sinc}(2f_c(n - M/2)) \;\; n = 0, 1, \cdots, M$$

- For the HPF, what is $\text{sinc}(n - M/2) \;\; n = 0, 1, \cdots, M$ equivalent to? (note M is even for a Type 1 filter)

Example 11: IRT: Bandpass Filter

Design a bandpass filter, using the impulse response truncation technique, with cutoff frequencies of

$$\omega_1 = 0.2\pi, \omega_2 = 0.6\pi$$

and an order of 40. Plot the impulse response on one figure window, the amplitude response (linear) and phase response on a second figure window and the magnitude response (linear) and associated phase response (with phase in the range $-\pi$ to π) on a third figure window. Plot the frequency response plots as a function of frequency in cycles/samples for the range -0.5 to 0.5.

- See bandpass_filter_example.m

Multiband Filters

- A multiband filter is a superposition of bandpass filters. If the desired amplitude response has K bands, where the cutoff frequencies for the k^{th} band are ω_{1k} and ω_{2k} and the band gain is C_k, then

$$A_d(\omega) = \sum_{k=1}^{K} A_{d,k}(\omega)$$

where

$$A_{d,k}(\omega) = \begin{cases} C_k; & \omega_{1k} \leq |\omega| \leq \omega_{2k} \\ 0; & \text{otherwise} \end{cases}$$

- Using the impulse response truncation technique and the bandpass impulse response expression on slide 83, the impulse response for the general multiband filter (K bands) is

$$h[n] = \sum_{k=1}^{K} C_k[2f_{2k}\text{sinc}(2f_{2k}(n - M/2)) - 2f_{1k}\text{sinc}(2f_{1k}(n - M/2))]$$

- A bandstop filter is a special case of a multiband filter with two bands, having $C_1 = C_2 = 1$, $f_{11} = 0$, and $f_{22} = 1/2$.

Example 12: IRT: Two Band Filter

Design a two band filter, using the impulse response truncation technique, with $M = 80, f_{11} = 0.1, f_{21} = 0.2, f_{12} = 0.35, f_{22} = 0.4, C_1 = 1, C_2 = 0.5$. Plot the impulse response on one figure window and the amplitude response on a second figure window for f from -0.5 to 0.5 cycles/sample. Use the MATLAB function fft to generate the amplitude response.

- See two_band_filter_example.m

How Good is the IRT Method

- IRT filters are optimal in the sense of minimizing the integral of the square error,

$$\epsilon = \frac{1}{2\pi}\int_{-\pi}^{\pi} |H_d(e^{j\omega}) - H(e^{j\omega})|^2 d\omega$$

- If $h[n], 0 \leq n \leq M$ are the coefficients of a causal FIR filter of order M, then using Parseval's theorem,

$$\epsilon = \sum_{n=-\infty}^{\infty} (h_d[n] - h[n])^2 = \sum_{n=-\infty}^{-1} h_d^2[n] + \sum_{n=M+1}^{\infty} h_d^2[n] + \sum_{n=0}^{M} (h_d[n] - h[n])^2$$

- Only the third term is a function of $h[n]$. This term is non-negative and ϵ is minimized if this term is 0, which occurs if

$$h[n] = h_d[n] \quad 0 \leq n \leq M$$

which is the criterion for the impulse response truncation method.

Table 2.2 Oppenheim and Schafer 3rd Edition, Page 58

1. $ax[n] + by[n]$	$aX(e^{j\omega}) + bY(e^{j\omega})$
2. $x[n - n_d]$ (n_d an integer)	$e^{-j\omega n_d} X(e^{j\omega})$
3. $e^{j\omega_0 n} x[n]$	$X(e^{j(\omega-\omega_0)})$
4. $x[-n]$	$X(e^{-j\omega})$ $X^*(e^{j\omega})$ if $x[n]$ real.
5. $nx[n]$	$j\frac{dX(e^{j\omega})}{d\omega}$
6. $x[n] * y[n]$	$X(e^{j\omega})Y(e^{j\omega})$
7. $x[n]y[n]$	$\frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\theta})Y(e^{j(\omega-\theta)})d\theta$

Parseval's theorem:

8. $$\sum_{n=-\infty}^{\infty} |x[n]|^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} |X(e^{j\omega})|^2 d\omega$$

9. $$\sum_{n=-\infty}^{\infty} x[n]y^*[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})Y^*(e^{j\omega})d\omega$$

How Good is the IRT Method

- But, even though this technique is optimal in the sense of the integral of the square error, it is not considered a good technique, because of the ripple in the frequency response near discontinuity locations in $H_d(e^{j\omega})$.
- As M increases, ϵ decreases, but the magnitude of the ripple does not decrease to 0, it approaches a finite number. This is known as Gibbs phenomenon.

Example 13: Gibbs Phenomenon

Compare the amplitude responses for two impulse response truncation lowpass filters. One with the order $M = 10$ and the other with $M = 40$.Set the cutoff to be 0.25 cycles/sample.
Use the MATLAB function fft in the process of generating the amplitude response. Plot both responses on one plot as a function of frequency in cycles/samples for the range -0.5 to 0.5.

IRT Method Limitations

- In the previous demonstration, the passband and stopband ripple is approximately the same for $M = 10$ and $M = 40$, with peak values close to 0.09.
- For the IRT method, this value remains approximately constant, regardless of the filter order.
- This phenomenon is named after J.W. Gibbs.
- Practically, the IRT method is suitable only for filters whose ripple is not less than 0.09 or

$$A_p = 20 \log_{10}(1 + 0.09) = 0.75 \text{ dB}$$

$$A_s = -20 \log_{10}(0.09) = 20.9; \text{dB}$$

  where A_p is the passband ripple in dB and A_s is the stopband attenuation in dB.
- Other design techniques can be used to mitigate the Gibbs phenomenon. For example, the windows FIR design approach, which will be covered later.

Subsection 3

Filter Transformations

Lowpass to Highpass Transformation

- A lowpass filter can be converted to a highpass filter using simple techniques that can be implemented in real time.
- The two techniques that will be examined are:
  - Technique 1: Subtract a lowpass filter from an allpass filter.
  - Technique 2: Modulation theorem.

Technique 1: Lowpass Subtracted from a Allpass Filter

Allpass Filter

- An allpass filter has a constant gain (usually 1) magnitude response at all frequencies, ie.

$$|H(e^{j\omega})| = 1$$

- There are no restrictions on the phase response.
- An allpass filter implemented as an FIR filter is simply

$$H(e^{j\omega}) = A(\omega)e^{j\theta_A(\omega)}$$

where $A(\omega) = 1$ and $\theta_A(\omega) = -\alpha\omega$.

Technique 1: Lowpass Subtracted from a Allpass Filter

- A highpass filter, with frequency response $H_{HP}(e^{j\omega})$, can be generated from a lowpass filter, with frequency response $H_{LP}(e^{j\omega})$, by subtracting the lowpass response from an allpass filter frequency response that has the same phase response as $H_{LP}(e^{j\omega})$,

$$H_{HP}(e^{j\omega}) = \underbrace{e^{j\theta_A(\omega)}}_{\text{allpass}} - \underbrace{A(\omega)e^{j\theta_A(\omega)}}_{H_{LP}(e^{j\omega})} = (1 - A(\omega))e^{j\theta_A(\omega)}$$

- For a Type 1 FIR linear phase filter, $\theta_A(\omega) = -\frac{M}{2}\omega$, and

$$H_{HP}(e^{j\omega}) = (1 - A(\omega))e^{-j\frac{M}{2}\omega}$$

- Converting this expression to its z-transform gives,

$$H_{HP}(z) = z^{-M/2} - H_{LP}(z)$$

- Taking the inverse z-transform gives

$$h_{HP}[n] = \delta[n - M/2] - h_{LP}[n]$$

- The cutoff frequency of the lowpass filter occurs at approximately $A(\omega) = 0.5$ (ie. the -6 dB point in $|H(e^{j\omega})|_{\text{dB}}$), thus the highpass filter will have the same cutoff frequency as the lowpass filter.

Example 14: Convert LPF to HPF

Design a lowpass filter, using the impulse response truncation technique, with a cutoff frequency of $f_c = 0.1$ and an order of 30. Convert this filter to a highpass filter with a cutoff frequency of $f_c = 0.1$.

Technique 2: Modulation Theorem

Modulation Theorem

- The modulation theorem is defined as

$$h[n]x[n] \xrightarrow{\text{DTFT}} \frac{1}{2\pi} H(e^{j\omega}) \circledast X(e^{j\omega})$$

  where $\circledast$ represents circular (periodic) convolution.
- If $x[n] = \cos(\omega_0 n)$, with $\omega_0 = \pi$, specify

$$h_{HP}[n] = h_{LP}[n] \cos(\pi n)$$

- In the frequency domain, the circular convolution between the lowpass frequency response and the impulses due to the cosine will result in a highpass filter response.
- $h_{HP}[n]$ can be simplified,

$$h_{HP}[n] = h_{LP}[n] 0.5(e^{j\pi n} + e^{-j\pi n}) = h_{LP}[n] e^{j\pi n} = h_{LP}[n](-1)^n$$

Technique 2: Modulation Theorem

- Taking the DTFT

$$H_{HP}(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h_{LP}[n]e^{j\pi n}e^{-j\omega n} = \sum_{n=-\infty}^{\infty} h_{LP}[n]e^{-j(\omega-\pi)n} = H_{LP}(e^{j(\omega-\pi)})$$

- Thus $H_{HP}(e^{j\omega})$ is generated by shifting $H_{LP}(e^{j\omega})$ to the right by π.
- Comparing $H_{LP}(e^{j\omega})$ and $H_{HP}(e^{j\omega})$, it should be apparent that given the cutoff frequency of the lowpass filter is ω_c^{LP}, the cutoff frequency of the highpass filter will be

$$\omega_c^{HP} = \pi - \omega_c^{LP}$$

Example 15: Convert LPF to HPF Using Technique 2

Design a lowpass filter, using the impulse response truncation technique, with a cutoff frequency of $f_c = 0.1$ and an order of 30. Convert this filter, using technique 2 based on the modulation theorem, to a highpass filter. What is the cutoff frequency of the highpass filter?

Section 3

FIR Filter Design

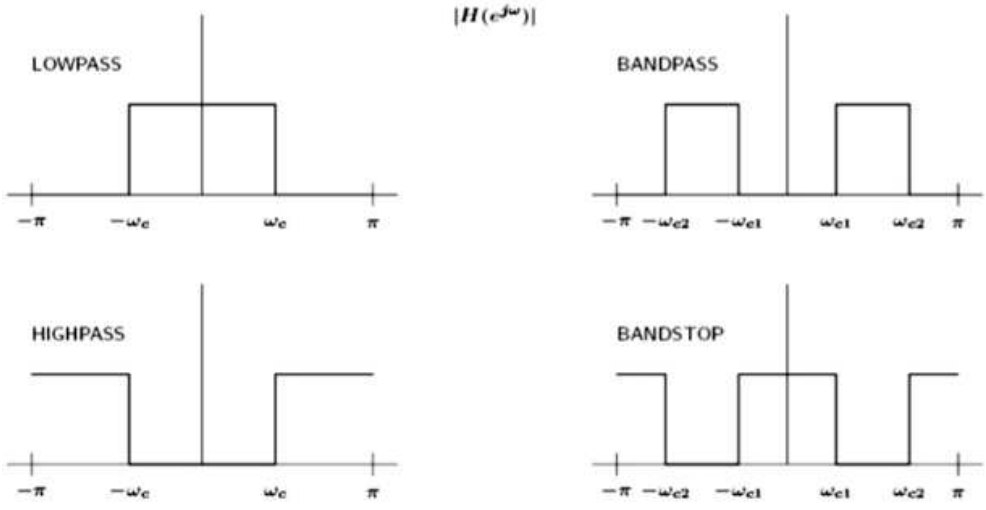
Subsection 1

Filter Specifications

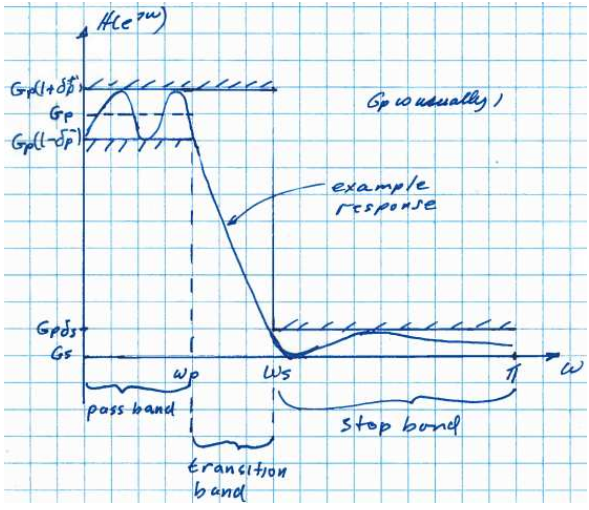
Introduction

- You have already been using filters and working with some filter specifications.
- Here we are gong to formally define the notation for filter specifications used in these slides (similar to Oppenheim and Schafer, Discrete-Time Signal Processing, 3rd Ed.) and also the notation used in MATLAB.
- The first step in designing a filter is to specify the performance requirements in the frequency domain. This includes specifying which frequency bands in the filter should pass the signal and which bands should attenuate the signal, along with the associated gain and attenuation parameters.
- There are four basic filter types: low-pass, high-pass, band-pass and band-stop. An example of the ideal frequency responses are shown on the next slide.
- These ideal frequency responses are impossible to implement in practise.
- The magnitude response of practical filters vary continuously as a function of frequency, and the response is never exactly one in the pass bands or zero in the stop bands for a range of frequencies.

Ideal Filters



Lowpass Filter Specification



See Figure 7.1, Oppenheim and Schafer, 3rd Ed.

Lowpass Filter Specification

In the previous figure

- ω_p is the passband corner frequency or passband edge frequency.
- ω_s is the stopband corner frequency or stopband edge frequency.
- $\omega_p \leq \omega \leq \omega_s$ is the filter transition band.
- G_p is the desired or nominal passband gain, it is typically 1.
- G_s is the nominal stopband gain, it is typically 0.
- $G_p\delta_p^+$ is the positive tolerance of the magnitude response in the passband.
- $G_p\delta_p^-$ is the negative tolerance of the magnitude response in the passband.
- $G_p\delta_s$ is the tolerance of the magnitude response in the stopband.

These filter specifications can vary depending on the application and historical basis. There is a difference in specifications for IIR and FIR filters and this will be discussed in the next few slides.

IIR Lowpass Filter Specification

It is common when designing IIR filters to use

$$\delta_p^+ = 0 \quad \delta_p^- = \delta_p$$

Using this simplification for IIR filters:

- The nominal, or in this case the maximum passband gain is G_p, which is typically 1,
- The passband ripple or passband tolerance is $G_p\delta_p$,
- The passband ripple in dB is

$$A_p = 20\log_{10}\left(\frac{G_p}{G_p(1-\delta_p)}\right) = -20\log_{10}(1-\delta_p)\ \text{dB}$$

.

- The stopband attenuation or maximum stopband gain (which is $-A_s$) in dB is

$$A_s = -20\log_{10}\left(\frac{G_p\delta_s}{G_p}\right) = -20\log_{10}(\delta_s)$$

FIR Lowpass Filter Specification

It is common when designing FIR filters to use $\delta_p = \delta_p^+ = \delta_p^-$ Using this simplification for FIR filters:

- The passband ripple or passband tolerance is $G_p\delta_p$,
- The passband ripple in dB is

$$A_p = 20\log_{10}\left(\frac{G_p(1+\delta_p)}{G_p}\right) = 20\log_{10}(1+\delta_p) \text{ dB}$$

- In MATLAB, the passband ripple in dB (peak to peak) is specified as

$$A_{pM} = 20\log_{10}\left(\frac{G_p(1+\delta_p)}{G_p(1-\delta_p)}\right) = 20\log_{10}\left(\frac{1+\delta_p}{1-\delta_p}\right) \text{ dB}$$

- And for the MATLAB case, calculating δ_p from the expression for A_{pM} gives

$$\delta_p = \frac{1-10^{-A_{pM}/20}}{1+10^{-A_{pM}/20}}$$

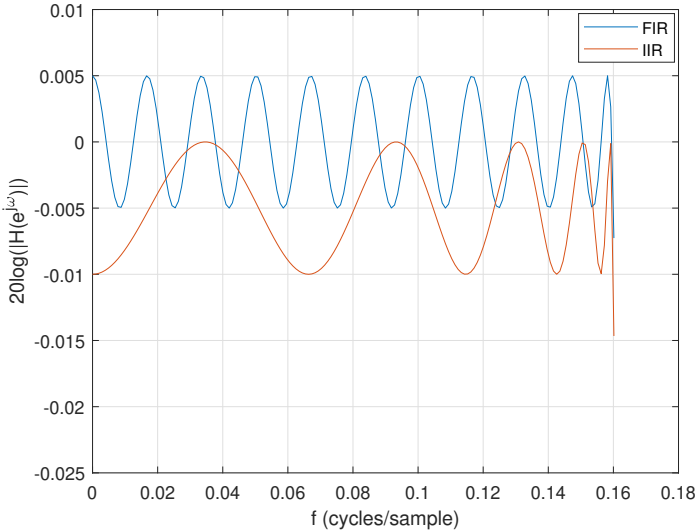
- Similar to the IIR filter, the stopband attenuation or maximum stopband gain (which is $-A_s$) in dB is $A_s = -20\log_{10}(\delta_s)$.

Filter Specification Used in MATLAB

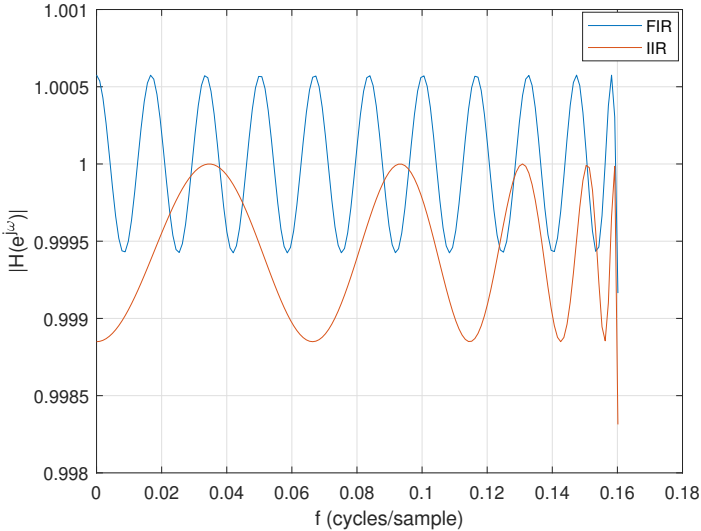
- This MATLAB code generates a plot an FIR and IIR filter for passband ripple comparison.

```
M   = 120; % order of FIR filter
fp  = 0.16; % passband edge frequency
ApM  = 0.01; % peak to peak ripple, dB
As = 80; % stopband attenuation, dB
dp  = (1 - 10^(-ApM/20))/(1 + 10^(-ApM/20)); % passband ripple (peak)
ds = 10^(-As/20); % stopband ripple
bfir = firceqrip(M,2*fp,[dp ds],'passedge'); % FIR filter coefficients
M = 10; % IIR filter order
[b,a]= ellip(M,ApM,As,2*fp); % IIR filter coefficients
% Plot Magnitude Responses (both linear and dB)
[Hfir,wfir]=freqz(bfir); [Hiir,wiir]=freqz(b,a);
figure, plot(wfir/(2*pi),abs(Hfir)); hold on
plot(wiir/(2*pi),abs(Hiir)); grid
figure, plot(wfir/(2*pi),20*log10(abs(Hfir))); hold on
plot(wiir/(2*pi),20*log10(abs(Hiir))); grid
```

Filter Specification Used in MATLAB

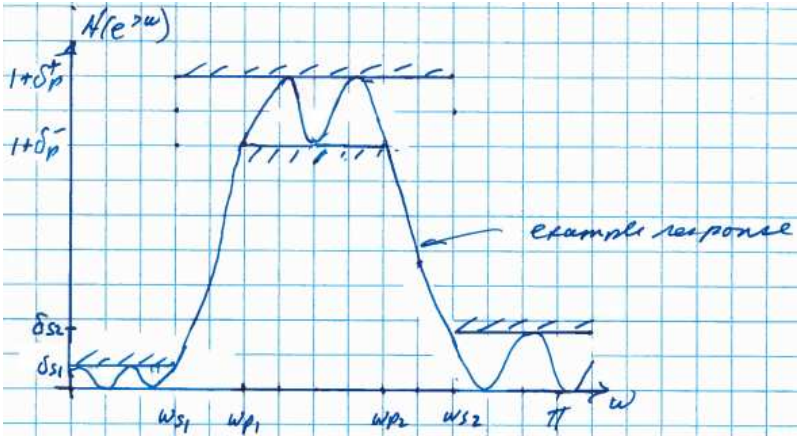


Filter Specification Used in MATLAB

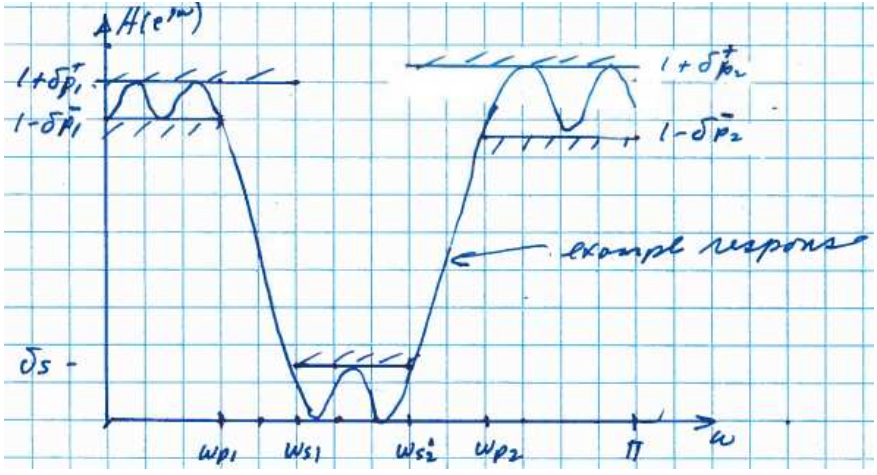


Bandpass Filter Specification

Here there are two stopbands: $A_{s1} = -20\log_{10}(\delta_{s1}), \quad A_{s2} = -20\log_{10}(\delta_{s2})$



Bandstop Filter Specification



FIR Filter Specification Summary

The linear parameters are (assuming $G_p = 1$ and $G_s = 0$):

- δ_p - the passband ripple or passband tolerance
- δ_s - the stopband ripple or stopband tolerance

The FIR filter parameters in dB are:

- The passband ripple and stopband attenuation in dB are

$$A_p = 20\log_{10}(1+\delta_p) \quad \text{and} \quad A_s = -20\log_{10}(\delta_s)$$

- The MATLAB (peak to peak) passband ripple in dB is

$$A_{pM} = 20\log_{10}\left(\frac{1+\delta_p}{1-\delta_p}\right)$$

Calculating the linear parameter δ_p from A_{pM}:

$$\delta_p = \frac{1-10^{-A_{pM}/20}}{1+10^{-A_{pM}/20}}$$

Subsection 2

Design of FIR Filters by Windowing

Introduction

- Recall the impulse response truncation technique (IRT), which had issues related to Gibbs oscillations.
- These oscillations limited the stopband attenuation to 21 dB and the passband ripple to 0.75 dB.
- These values can be improved upon using a technique involving windows to reduce the ripple.
- There are a number of well known windows that can be selected. The best choice depends on the application.
- But first, the Gibbs phenomenon will be reviewed in a MATLAB demonstration.

Example 16: Gibbs Oscillations

This demo generates and plots the amplitude response for an impulse response truncation technique lowpass filter with cutoff frequency, $f_c = 0.2$ cycles/sample, for filter orders $M = 20, 40, 80$ and 160, as subplots on a figure window. It also plots the magnitude of the amplitude response squared (which is equivalent to the squared magnitude response) in dB on a separate figure window.

- See gibb.m

What is a Window?

- A window, $w[n]$, is a finite length time sequence that is used to weight the desired impulse response, $h_d[n]$, to produce the FIR filter impulse response, $h[n] = h_d[n]w[n]$.
- You have already been introduced to an FIR window design technique in the IRT method.
- Recall in the IRT method, $h_d[n]$ was generated and then it was truncated to a length of $M + 1$.
- This truncation is equivalent to multiplying $h_d[n]$ by a rectangular window with an amplitude of one.
- Thus the FIR window design technique could be viewed as generating $h_d[n]$ using the IRT method and then multiplying $h_d[n]$ by the desired window.
- This section introduces the common windows that are used.
- The FIR windows design method is summarized in the next slide.

FIR Window Design Method

1. Select a filter type.
2. Specify $A_d(e^{j\omega})$ for the chosen filter type.
3. Determine the desired impulse response, $h_d[n]$, that minimizes the integral of the squared error of the magnitude response for a filter of length $M+1$ (impulse response truncation technique).
4. Determine a symmetric window, $w[n] = w[M-n]$, that will achieve the desired filter specifications.
5. Multiply the impulse response by the window.

Windowing in the Frequency Domain

- An expressions for the frequency response of a filter as a function of the window can be determined using the DTFT modulation (multiplication) property:

$$x_1[n]x_2[n] \xrightarrow{\text{DTFT}} \frac{1}{2\pi} X_1(e^{j\omega}) \circledast X_2(e^{j\omega})$$

where $\circledast$ represents periodic (circular) convolution (see equation 2.166, page 61, Oppenheim and Schafer, 3rd Ed).

- Let $h[n] = h_d[n]w[n]$, then

$$H(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\theta})W(e^{j(\omega-\theta)})d\theta = \frac{1}{2\pi}\int_{-\pi}^{\pi} A_d(e^{j\theta})e^{j(\beta-\alpha\theta)}A_W(e^{j(\omega-\theta)})e^{-j\alpha(\omega-\theta)}d\theta$$

and simplifying

$$H(e^{j\omega}) = A(e^{j\omega})e^{j\theta_A} = \underbrace{\frac{1}{2\pi}\int_{-\pi}^{\pi} A_d(e^{j\theta})A_W(e^{j(\omega-\theta)})d\theta}_{\text{Periodic Convolution}}\, e^{j(\beta-\alpha\omega)}$$

where $\alpha = M/2$, and this is valid for M even or odd.

Properties of Window Functions

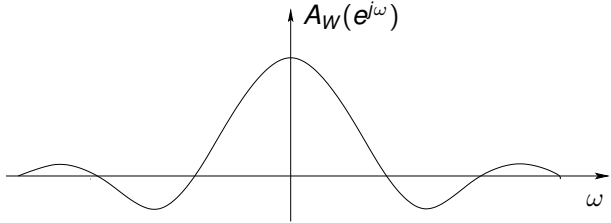
1. $w[n] = 0$ for $n < 0$ and $n > M$.
2. $w[n]$ is symmetric, $w[n] = w[M - n]$.
3. $A_W(e^{j\omega}) = A_W(e^{-j\omega})$, ie. even about $\omega = 0$.
4. $A_W(e^{j(\pi+\omega)}) = A_W(e^{-j(\pi-\omega)})$ for M even; ie. even about π, periodic with a period of 2π.
5. $A_W(e^{j(\pi+\omega)}) = -A_W(e^{-j(\pi-\omega)})$ for M odd; ie. odd about π, periodic with a period of 4π.
6. The window, $w[n]$, is chosen, such that

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} A_W(e^{j\omega})d\omega = w[M/2] = 1 \quad \text{for } M \text{ even, or}$$

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} A_W(e^{j\omega})d\omega = 1 \quad \text{for } M \text{ odd}$$

Form of Window Amplitude Response

- $A_W(e^{j\omega})$, for most windows, has the form



- The area under the lobes define the ripple in the passband and stopband, though the area under the lobes does not depend on M.
- Also, the width of the lobes are proportional to $1/M$ and the height of the lobes are proportional to $M + 1$.

Window Functions

- The properties, such as the stopband attenuation, passband ripple and transition bandwidth, of the frequency response of a filter, $H(e^{j\omega})$, depend on the ideal desired frequency response, $H_d(e^{j\omega})$, and the chosen window.
- A window is characterized by two primary parameters:
  - The width of the main lobe.
  - The magnitude of the highest side lobe relative to the main lobe.
- There are a large number of possible window functions. Initially five will be consider in detail (the same ones that Oppenheim and Schafer focus on): rectangular, Bartlett, Hann, Hamming and Blackman.
- Specifications for these windows are given on the next slide.
- Note that when using a windows design method, δ_p and δ_s, must be equal, so the worst case value is used for both.

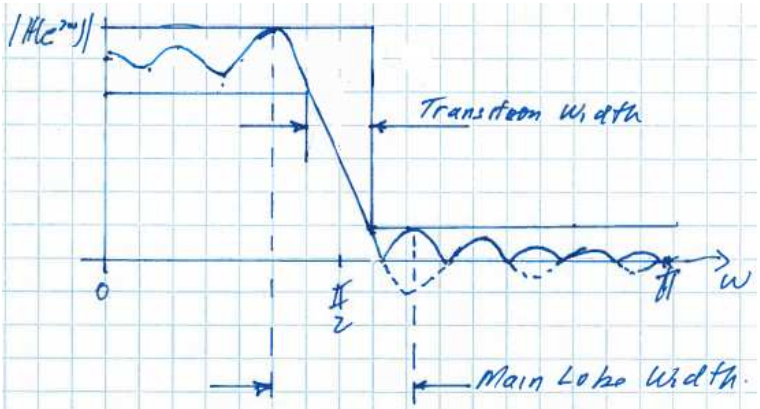
Window Specifications

The specifications for the five windows are summarized in this table (similar to Table 7.2 in Oppenheim and Schafer, 3rd Ed.):

Window	**Relative Side Lobe Attenuation**	**≈Main Lobe Width**	A_s **(dB)**	β **(Equiv. Kaiser Window)**	$(\omega_s - \omega_p)$ **Transition Width (Equiv. Kaiser Window)**
Rectangular	-13 dB	$4\pi/(M+1)$	21 dB	0	$1.81\pi/M$
Bartlett	-25 dB	$8\pi/M$	25 dB	1.33	$2.37\pi/M$
Hann	-31 dB	$8\pi/M$	44 dB	3.86	$5.01\pi/M$
Hamming	-41 dB	$8\pi/M$	53 dB	4.86	$6.27\pi/M$
Blackman	-57 dB	$12\pi/M$	74 dB	7.04	$9.19\pi/M$

Note: In the above table, the transition widths are specified in terms of an equivalent Kaiser window. For a given shape factor, β, the Kaiser window is approximately the same as the given window, so the Kaiser window transition width is used for the given window.

Mainlobe Width vs Transition Width



Note: The cutoff frequency is $\omega_c = \pi/2$, this is defined as the half gain point, or $20\log_{10}(1/2) = -6.02$ dB. This notation is used in MATLAB and Oppenheim and Schafer.

Window Definitions

- **Rectangular**

$$w[n] = \begin{cases} 1; & 0 \le n \le M \\ 0; & \text{otherwise} \end{cases}$$

- **Bartlett**

$$w[n] = \begin{cases} \frac{2n}{M}; & 0 \le n \le \frac{M}{2} \\ 2 - \frac{2n}{M}; & \frac{M}{2} \le n \le M \\ 0; & \text{otherwise} \end{cases}$$

The Bartlett window is a triangular function for even order and a triangular function with a flat top for odd order.

- **Hann(Hanning), Hamming, Blackman (generalized cosine windows)**

$$w[n] = a_1 - a_2 \cos\left(\frac{2\pi n}{M}\right) + a_3 \cos\left(\frac{4\pi n}{M}\right) \quad 0 \le n \le M$$

where for Hann, $a_1 = 0.5, a_2 = 0.5, a_3 = 0$, for Hamming, $a_1 = 0.54, a_2 = 0.46, a_3 = 0$, and for Blackman, $a_1 = 0.42, a_2 = 0.5, a_3 = 0.08$.

Example 18: Lowpass Filter Design Using the Window Method

Design a lowpass filter using a window method, where the

- passband gain is 1,
- passband ripple, δ_p, is 0.01
- stopband ripple, δ_s, is 0.1
- passband corner frequency is $\omega_p = \pi/8$,
- stopband corner frequency is $\omega_s = \pi/4$.

See lpf_hann_example.m

Filter Design Using a Kaiser Window

- A Kaiser window is defined as

$$w[n] = \begin{cases} I_0(\beta\sqrt{1-(\frac{2n}{M}-1)^2}); & 0 \le n \le M \\ 0; & \text{otherwise} \end{cases}$$

  where $I_0(\cdot)$ is the modified Bessel function of the first kind.
- The Kaiser window parameter, β, controls the shape of the window.
- For example, if $\beta = 0$, the Kaiser window is rectangular. If $\beta = 4.86$, the Kaiser window is very similar to the Hamming window.
- Kaiser, the namesake of this window, developed expressions for β and M.

$$\beta = \begin{cases} 0.1102(A-8.7);; & A > 50 \\ 0.5842(A-21)^{0.4} + 0.07886(A-21);; & 21 \le A \le 50 \\ 0; & A < 21 \end{cases}$$

$$M = \frac{A-8}{2.285(\omega_s - \omega_p)}$$

  where $A = -20\log_{10}\delta$.

Example 19: Lowpass Filter Design Using a Kaiser Window

Design a LPF using a Kaiser window, with a peak passband ripple of 0.001, $\omega_p = 0.4\pi$ and $\omega_s = 0.5\pi$.

- See kaiser_example.m for the MATLAB implementation.

Example 20: Design a Type 1 HPF

Design a Type 1 highpass filter with $\delta_s = 0.0002, \delta_p = 0.001, \omega_p = 0.8\pi$ and $\omega_s = 0.7\pi$, using a windows design and a Kaiser window design.

- See blackman_kaiser_example.m for the MATLAB implementation.

Filter Design if $G_p \neq 1$ and/or $G_s \neq 0$

- In a filter design, if $G_p \neq 1$ and/or $G_s \neq 0$, then the ripple specification must be modified to account for G_p or G_s changes before the filter is designed.
- For example, if $G_p = 2$, then if the ripple specification is not modified, the ripple will be larger in the final design by a factor of 2. To account for the increased gain the ripple would have to be scaled by a factor of 2.

Scaling the Ripple

- If the desired passband gain is G_p and the desired stopband gain is G_s, then

$$\delta_n = \frac{\delta}{G_p - G_s}$$

where δ is the desired peak ripple for the filter and δ_n is the normalized peak ripple that is used to design the filter.

Example 21: Lowpass Filter with a Gain of 2

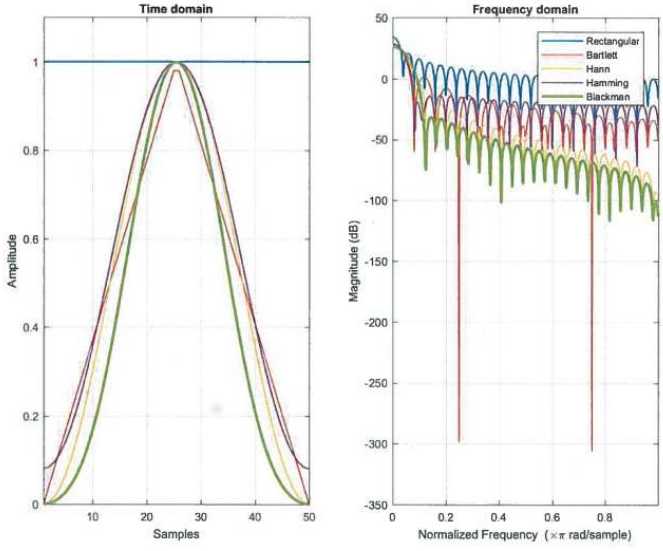
Design a lowpass filter using a Kaiser window, with $G_p = 2, G_s = 0, \delta = 0.01, \omega_p = 0.3\pi$ and $\omega_s = 0.4\pi$.

- See lowpass_filter_kaiser_example_gp2.m for the MATLAB implementation.

MATLAB windowDesigner

- Various windows can be specified and examined using the MATLAB app, windowDesigner (wintool is used in older versions of MATLAB).
- There are a number of parameters that can be viewed or set in windowDesigner:
  - Leakage factor - ratio of the power in the side lobes to the total window power.
  - Relative side lobe attenuation - the maximum side lobe amplitude relative to the maximum main lobe amplitude.
  - Sampling - Symmetric vs periodic reflects how the window is calculated. Use symmetric for filter design. Periodic is used for spectral analysis.
- A plot of the time domain and frequency domain representations of five windows, with $M = 49$ is shown on the next slide.

windowDesigner Plots for $M = 49$



Subsection 3

Least Squares Design of FIR Filters

Introduction

- One of the major disadvantages of the various window methods for FIR filter design is that δ_s and δ_p must be equal.
- Though, often the specification of δ_s is more stringent that δ_p (ie. $\delta_s < \delta_p$), resulting in unnecessarily high accuracy in the passband. This results in a higher order filter than is actually needed.
- In the least squares design technique, δ_s and δ_p can be indirectly specified separately.

Least Squares FIR Design

- The least squares design involves choosing the coefficients that minimize the integral of the square of the weighted frequency response error,

$$\epsilon^2 = \int_0^\pi E^2(\omega) d\omega$$

where

$$E(\omega) = W_t(\omega)(A_d(\omega) - A(\omega))$$

and $W_t(\omega)$ is a weight function, which reflects the relative importance of the error between $A_d(\omega)$ and $A(\omega)$ at a given ω.

- For example, for a weight function for a lowpass filter, one possibility is to assign δ_p^{-1} to all frequencies in the passband and assign δ_s^{-1} to all frequencies in the stopband and a zero weight for the transition band.

Least Squares FIR Design

- In the least squares filter design, the order M of the filter is assumed known and so is the type of linear phase filter.
- For example, for the type 1 filter, from slide 61,

$$A(\omega) = \sum_{k=0}^{\frac{M}{2}} a[k] \cos \omega k$$

where

$$a[k] = \begin{cases} h[\frac{M}{2}]; & k = 0, \\ 2h[\frac{M}{2} - k]; & k = 1, 2, \cdots, \frac{M}{2} \end{cases}$$

Thus

$$E(\omega) = W_t(\omega)(A_d(\omega) - \sum_{k=0}^{\frac{M}{2}} a[k] \cos \omega k)$$

- Designing the FIR filter involves determining the coefficients $a[k]$ (ie. the filter impulse response coefficients) that minimizes the expression for ϵ^2. This is done numerically.

Least Squares FIR Design in MATLAB

The basic form of the least squares filter design function, from the MATLAB documentation, is `b = firls(n,f,a,w)`, Using the notation in this class or notation that does not conflict with this class, this function can be written as

```
b = firls(M,fb,a,wght)
```

where

- `b` is a vector containing the `M+1` calculated impulse response coefficients.
- `M` is the order of the desired FIR filter
- `fb` is a vector of the corner frequencies of the stop/pass bands. `fb` has two entries for each band. The band frequencies specified for ω from 0 to π. The units of `fb` are radians/sample divided by π (or cycles/sample multiplied by 2). This results in `fb` having values between 0 and 1 (this is because MATLAB normalizes the analog frequencies by $\frac{F_s}{2}$ instead of F_s), and the first element must be a 0 and the last element must be a 1.
- For example, for a pass band from 0 to 0.2π and a stop band from 0.5π to π, the band frequency vector is `fb = [0, 0.2, 0.5, 1];`.

Least Squares FIR Design in MATLAB

Continuing the definition of

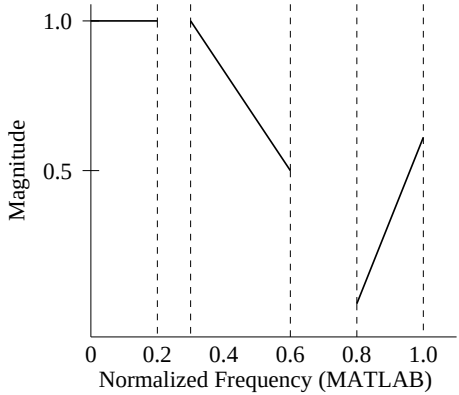
```
b = firls(M,fb,a,wght)
```

- `a` is a vector specifying the desired gain at the frequency points defined by `fb`. The gain does not have to be flat for each band. The gain is specified at the corners of each band and the gain across the band is a straight line joining the gains at the corners. `fb` and `a` must have the same length and this length must be an even number.
- For example, `a = [1, 0.8, 0, 0];`, results in the first band gain starting at 1 and ending at 0.8 and the gain of the second band being zero.
- `wght` specifies the relative ripple in the bands. It is a vector having one element for each band, thus it is half the length of `fb`. `wght` weights the error.
- For example, a relative weight of 10 will cause the ripple to be 1/10 that of the reference band (`wght = [1, 10];`) Or you can assign the inverse of the peak ripple desired for each band, for example, for a lowpass filter `wght=[1/deltap, 1/deltas]`.

Least Squares FIR Design in MATLAB

For example, the desired frequency magnitude response is given in the figure for the vectors

```
fb = [0, 0.2, 0.3, 0.6, 0.8, 1.0];
a = [1, 1.0, 1.0, 0.5, 0.1, 0.6];
wght = [1/delta1, 1/delta2, 1/delta3];
```



Example 22: Least Squares FIR Filter Design

Design a lowpass filter using a least squares design and compare it with a Kaiser window design. The filter specifications are $\delta_p = 0.1, \delta_s = 0.01, \omega_p = 0.2\pi$ and $\omega_s = 0.3\pi$.

- Determining the order of the filter is a trial an error process involving selecting the order to obtain the desired stop band attenuation. The resulting chosen filter has an order of M=32.
- See firls_design_example.m for the MATLAB implementation.
- Note: The least squares designed filter is compared with a filter designed using a Kaiser window. The Kaiser filter has an order of M = 45. This is a significant difference in length of filters, which primarily results from allowing the pass band ripple to be different and larger than the stop band ripple.

Subsection 4

Equiripple (Parks-McClellan) Design of FIR Filters

Introduction

- The least squares design approach focused on minimizing the integral of the square of the weighted frequency response error over all frequency domain bands.
- The equiripple design approach uses a better criterion, which involves the minimization of the maximum error in each band.
- This criterion results in an equiripple filter, one whose amplitude response oscillates uniformly between the ripple bounds for each band.
- This technique is optimum in the sense of minimizing the maximum magnitude of the ripple in each of the bands for a given filter order. The bands between transitions are not considered in the optimization.
- The equiripple design approach is also referred to as the Parks-McClellan approach or the optimum approximation method.
- This type of filter is also called a minmax filter, since the algorithm tries to minimize the maximum error between the desired and actual frequency responses.

Equiripple (Parks-McClellan) Filter Design

- Similar to the least squares technique, the weighted frequency response error is defined as

$$E(\omega) = W_t(\omega)(A_d(\omega) - A(\omega))$$

- The goal of the equiripple design algorithm involves determining the impulse response that minimizes the maximum absolute weighted error $|E(\omega)|$ of the entire set of bands.
- This can be expressed as minimizing ϵ over all coefficients of the impulse response, where

$$\epsilon = \max_{\text{all bands}} |E(\omega)|$$

Equiripple (Parks-McClellan) Filter Design

- For example, for a type 1 filter, the actual frequency response (from slide 61) is

$$A(\omega) = \sum_{k=0}^{\frac{M}{2}} a[k] \cos \omega k$$

where

$$a[k] = \begin{cases} h[\frac{M}{2}]; & k = 0, \\ 2h[\frac{M}{2} - k]; & k = 1, 2, \cdots, \frac{M}{2} \end{cases}$$

to give

$$E(\omega) = W_t(\omega)(A_d(\omega) - \sum_{k=0}^{\frac{M}{2}} a[k] \cos \omega k)$$

- Thus, designing the equipple FIR filter involves determining the coefficients, $a[k]$ (ie. the filter impulse response coefficients) that minimizes ϵ.
- An advantage of this approach, similar to the least squares design, is that the pass band ripple does not have to be the same as the stop band ripple.

Equiripple (Parks-McClellan) Design in MATLAB

- The equiripple design algorithm is implemented in MATLAB using the `firpm` function, where `pm` stands for Parks-McClellan.
- The `firpm` function implements the Parks-McClellan algorithm, which uses Chebyshev approximation theory to develop a transfer function with equiripple passband and stopband magnitude responses.
- The basic form of the function from the MATLAB documentation is `b = firpm(n,f,a,w)`, Using the notation in this class or notation that does not conflict with this class, this function can be written as `b = firpm(M,fb,a,wght)`.
- The variables in `b = firpm(M,fb,a,wght)` are the same as those used for the least squares method as describe in slides 138 to 140.

Estimating the Filter Order for the Parks-McClellan Method

The order of the filter is needed to use this technique and the order can be estimated using emperically derived expressions developed by Bellanger, Kaiser and Harris. The approximation derived by Bellanger is

$$M \approx \frac{2}{3}\left(\log_{10}\left(\frac{1}{10\delta_p\delta_s}\right)\right)\left(\frac{2\pi}{|\omega_s - \omega_p|}\right)$$

The Kaiser approximation is

$$M \approx \frac{-20\log_{10}(\sqrt{\delta_p\delta_s}) - 13}{2.324|\omega_s - \omega_p|}$$

The Harris approximation is

$$M \approx \frac{-20\log_{10}(\delta_s)}{22|f_s - f_p|}$$

Estimating the Filter Order for the Parks-McClellan Method

- The order for the equiripple filter can also be estimated by the MATLAB function, `firpmord`.
- MATLAB's basic form of the function, as specified in their documentation, is `n = firpmord(f,a,dev,fs)`.
- Using the notation in this class (or notation that does not conflict with this class), this function is written as

```
M = firpmord(ford,aord,dev,Fsord)
```

where

- `ford` is `fb` with the first and last elements removed. `firpmord` assumes they are 0 and 1.
- `aord` is a vector giving the amplitudes of each band in `ford`. It has a length of one half of the length of `fb`.
- `dev` is the peak ripple in linear units. It is a vector with one element for each frequency band. It has the same number of elements as `aord`.
- `Fsord` is the sampling frequency in Hz. If not included it has a default value of 2. Though, if specified as 1, `ford` can use the non-MATLAB normalized frequency notation. The actual sampling frequency can also be specified.

`firpmord` can Generate the `firpm` Parameters

- The MATLAB function, `firpmord`, can be used to estimate the parameters needed for `firpm`.
- MATLAB's basic form of the function is

```
[n,fo,ao,w] = firpmord(f,a,dev,fs)
```

.

- Using the notation in this class (or notation that does not conflict with this class), this function is written as

```
[M,fb,a,wght]=firpmord(ford,aord,dev,Fsord)
```

where `ford`, `aord`, `dev`, `Fsord` are defined in the last slide.

Example 23: LPF Design Using the Equiripple (Parks-McClellan) Method

Design a lowpass filter using a equiripple design. The filter specifications are $\delta_p = 0.02, \delta_s = 0.01, \omega_p = \pi/4$ and $\omega_s = 3\pi/8, G_p = 1, G_s = 0$.
Divide the example into two parts:

1. Generate the filter order using 4 techniques: Bellanger's, Kaiser's and Harris's equations and `firpmord`. Select the highest order and then design the filter by specifying `fb,a` and `wght` and then using `firpm`.
2. Design the filter by first using `firpmord` to generate `fb,a,wght` and the filter order `M`, then use `firpm`.

- See lpf_equiripple_example.m for the MATLAB implementation.

Example 24: Multiband Filter Design Using the Equiripple (Parks-McClellan) Method

Design a multiband filter using a equiripple design. The filter specifications are $\omega_{p1} = \pi/4, \omega_{s1} = 3\pi/8, \omega_{s2} = 5\pi/8, \omega_{p2} = 11\pi/16, \delta_{p1} = 0.02, \delta_s = 0.01,$ $\delta_{p2} = 0.02, G_{p1} = 1, G_s = 0, G_{p2} = 0.3.$
Divide the example into two parts:

1. Generate the filter order using 4 techniques: Bellanger's, Kaiser's and Harris's equations and `firpmord`. Select the highest order and then design the filter by specifying `fb,a` and `wght` and then using `firpm`.
2. Design the filter by first using `firpmord` to generate `fb,a,wght` and the filter order `M`, then use `firpm`.

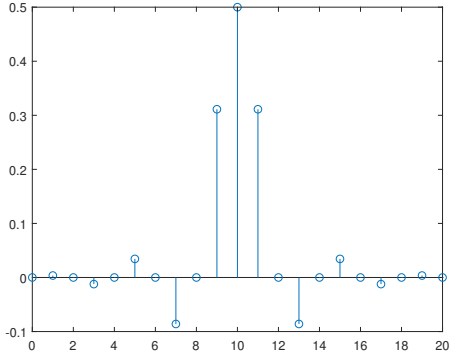
- See multiband_equiripple_example.m for the MATLAB implementation.

Subsection 5

FIR Nyquist Filter Design

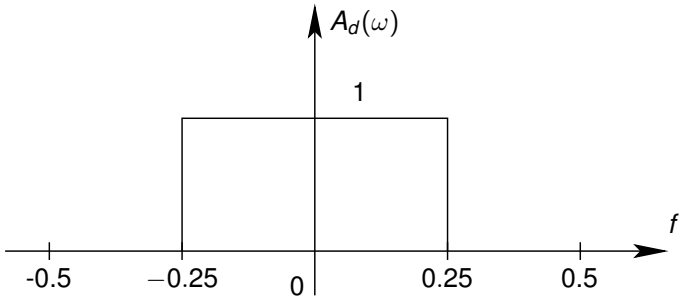
Introduction

- Nyquist filters (also called K^{th} band filters) have periodic zero values for every K^{th} coefficient, except for the center coefficient.
- These filters are used in multirate applications, such as as interpolation and decimation.
- They also find applications in pulse shaping filters for communications systems.
- An example of the impulse response of a K^{th} band filter, with $K = 2$ is



Halfband Filters

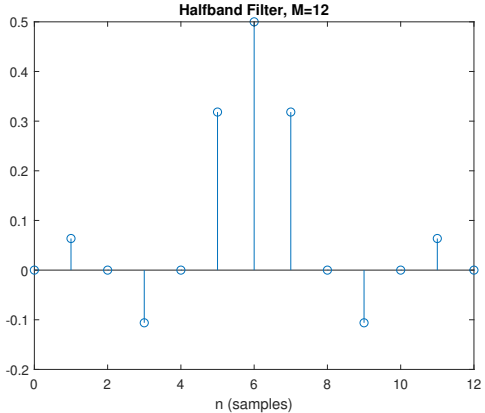
- A halfband filter is a K^{th} band filter where $K = 2$. Every second coefficient in these filters is zero, making for a more efficient implementation.
- But, note that every second coefficient will be zero only if the order of the halfband filter is even.
- An ideal halfband filter is a lowpass filter with a cutoff frequency at the half-way point, $\omega_c = \pi/2$ ($f_c = 1/4$).



Basic Halfband Filter

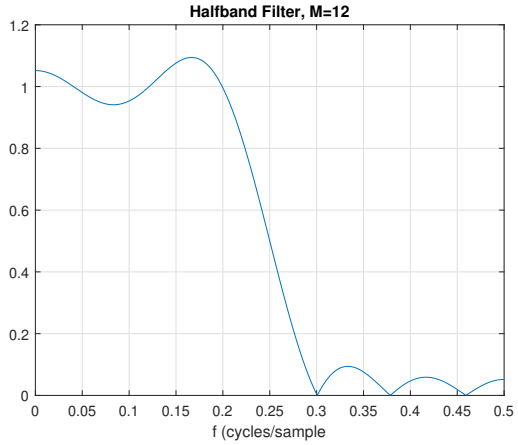
- A basic halfband filter can be designed using the IRT technique (rectangular window).
- The impulse response for such a filter is

$$h[n] = 2f_c\text{sinc}(2f_c(n - M/2)) = 0.5\text{sinc}(0.5(n - M/2)) \quad n = 0, 1, \cdots, M$$



Basic Halfband Filter Frequency Response

- The frequency response for the basic halfband filter of the previous slide is shown below.
- It has a cutoff frequency of 0.25 cycles/sample, as indicated by the half gain location.



Example 25: Window Design of a Halfband Filter

Design a halfband filter using a Kaiser window, with a ripple of 0.001 and a transition width of 0.1 cycles/sample.

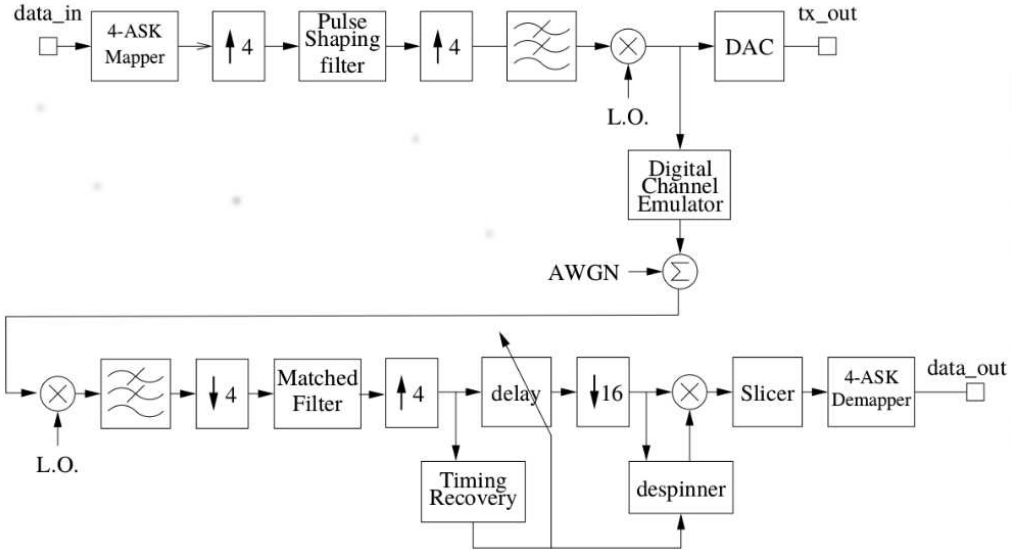
- See halfband_kaiser.m for the MATLAB implementation.

Example 26: Parks-McClellan Design of a Halfband Filter

Design a halfband filter using the Parks-McClellan method, with a ripple of 0.001 and a transition width of 0.1 cycles/sample.

- See halfband_equiripple.m for the MATLAB implementation.

EE465 QAM System



Ideal Continuous-Time Raised Cosine

- Raised cosine filters are Nyquist (K^{th} band) filters. Every K^{th} coefficient of these filters is zero.
- They are used extensively in communications as pulse shaping/matched filters.
- They are called raised cosine because their frequency response has a raised cosine shape.
- In continuous time, the frequency response is

$$H_{rc}(F) = \begin{cases} 1; & 0 \le |F| \le (1-\beta)F_c \\ \frac{1}{2}\left(1+\cos\left[\frac{\pi}{2\beta}\left(\frac{|F|}{F_c}-(1-\beta)\right)\right]\right); & (1-\beta)F_c \le |F| \le (1+\beta)F_c \\ 0; & |F| \ge (1+\beta)F_c \end{cases}$$

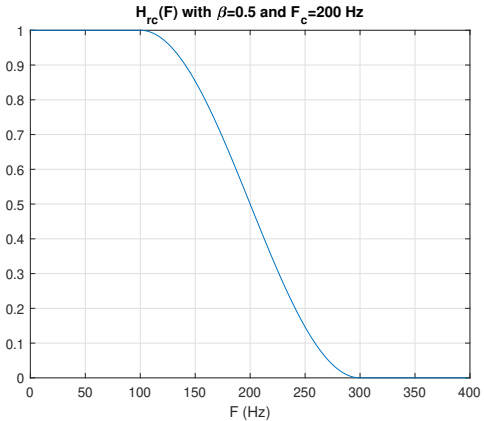
  where the cutoff frequency is $F_c = 1/(2T_s)$, T_s is the symbol time and $0 \le \beta \le 1$ is the roll-off factor.
- In the time domain,

$$h_{rc}(t) = \begin{cases} \frac{\pi}{4T_s}\text{sinc}\left(\frac{1}{2\beta}\right); & t = \pm\frac{T_s}{2\beta} \\ \frac{1}{T_s}\text{sinc}\left(\frac{t}{T_s}\right)\frac{\cos\left(\frac{\pi\beta t}{T_s}\right)}{1-\left(\frac{2\beta t}{T_s}\right)^2}; & \text{otherwise} \end{cases}$$

- Note that the time between sidelobe zero crossing in $h_{rc}(t)$ is T_s.

Ideal Raised Cosine Frequency Response

- The following figure is a plot of the ideal frequency response for a raised cosine pulse. Note the raised cosine has a gain of 1/2 at the cutoff frequency.



Excess Bandwidth

- As indicated in the earlier $H_{rc}(F)$ expression, the absolute bandwidth of a raised cosine pulse is

$$B_{abs} = (1+\beta)F_c = \frac{1+\beta}{2T_s}$$

  where $0 \leq \beta \leq 1$.
- Using this expression, the minimum bandwidth is F_c or $1/(2T_s)$, when $\beta = 0$ (frequency response is rectangular), and the maximum bandwidth is $2F_c$ or $1/T_s$, when $\beta = 1$.
- One convention is to call β the excess bandwidth.
- For example, a 0 % excess bandwidth indicates the raised cosine has the minimum bandwidth allowed, $1/2T_s$, and a 100 % excess bandwidth indicates the raised cosine has the maximum bandwidth allowed, $1/T_s$.

Ideal Discrete-Time Raised Cosine

- The discrete-time raised cosine frequency response can be determined by normalizing the continuous-time raised cosine frequency response given in slide 160. This is done by dividing the continuous-time frequencies by the sampling frequency, F_s, to give:

$$H_{rc}(f) = \begin{cases} 1; & 0 \le |f| \le (1-\beta)f_c \\ \frac{1}{2}\left(1 + \cos\left[\frac{\pi}{2\beta}\left(\frac{|f|}{f_c} - (1-\beta)\right)\right]\right); & (1-\beta)f_c \le |f| \le (1+\beta)f_c \\ 0; & |f| \ge (1+\beta)f_c \end{cases}$$

  where f and f_c have units cycles/sample.

- This can be simplified to

$$H_{rc}(f) = \begin{cases} 1; & 0 \le |f| \le (1-\beta)f_c \\ \frac{1}{2} + \frac{1}{2}\cos\left[\frac{\pi(|f| - f_c(1-\beta))}{2\beta f_c}\right]; & (1-\beta)f_c \le |f| \le (1+\beta)f_c \\ 0; & |f| \ge (1+\beta)f_c \end{cases}$$

Ideal Discrete-Time Raised Cosine

- The discrete-time version of the raised cosine impulse response can be obtained exactly using the impulse invariance technique, since $H_{rc}(F)$ in slide 160 is bandlimited.
- Using the impulse invariance technique, the discrete-time impulse response is

$$h_{rc}[n] = T h_{rc}(t)|_{t=nT}$$

where T is the sampling time. Thus

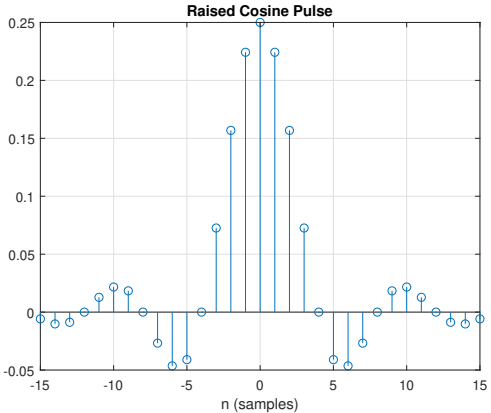
$$h_{rc}[n] = \begin{cases} \frac{\pi T}{4T_s}\text{sinc}\left(\frac{1}{2\beta}\right); & n = \pm\frac{T_s}{2\beta T} \\ \frac{T}{T_s}\text{sinc}\left(\frac{nT}{T_s}\right)\frac{\cos\left(\frac{\pi\beta nT}{T_s}\right)}{1-\left(\frac{2\beta nT}{T_s}\right)^2}; & \text{otherwise} \end{cases}$$

- In communications, it is common for the ratio T_s/T to be defined as $N_{sps} = T_s/T$, where N_{sps} is the number of samples per symbol (typically N_{sps} is 4 or 8). Substituting N_{sps} gives

$$h_{rc}[n] = \begin{cases} \frac{\pi}{4N_{sps}}\text{sinc}\left(\frac{1}{2\beta}\right); & n = \pm\frac{N_{sps}}{2\beta} \\ \frac{1}{N_{sps}}\text{sinc}\left(\frac{n}{N_{sps}}\right)\frac{\cos\left(\frac{\pi\beta n}{N_{sps}}\right)}{1-\left(\frac{2\beta n}{N_{sps}}\right)^2}; & \text{otherwise} \end{cases}$$

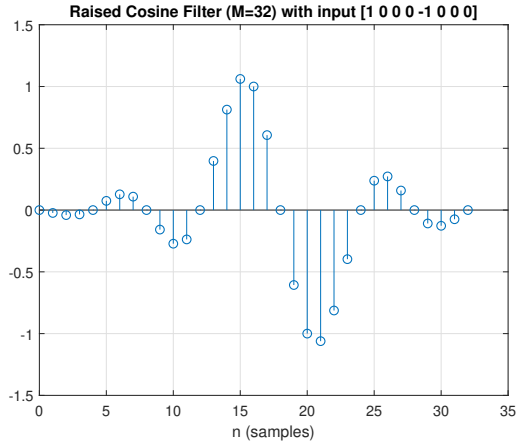
Ideal Discrete-Time Raised Cosine Pulse

- The following figure is a plot of a raised cosine pulse. Here $\beta = 0.25$ and $N_{sps} = 4$ have been used.
- Note that the ideal raised cosine pulse has infinite length, since it is bandlimited.



Raised Cosine Pulse Shaping and Intersymbol Interference

- A pulse shaping filter with a raised cosine spectrum can be used in communications to avoid intersymbol interference (ISI).
- For example, a raised cosine filter ($N_{sps} = 4$) input of [1 0 0 0 -1 0 0 0], produces the output



Ideal Continuous-Time Square Root Raised Cosine

- Raised cosine filters are typically split into two filters, one at the transmitter and one at the receiver, ie. $H_{rc}(F) = \sqrt{H_{rc}(F)}\sqrt{H_{rc}(F)}$.
- $H_{rrc}(F) = \sqrt{H_{rc}(F)}$ is called the square root raised cosine filter or the root raised cosine filter.
- In continuous time, the frequency response is

$$H_{rrc}(F) = \begin{cases} 1; & 0 \le |F| \le (1-\beta)F_c \\ \cos\left[\frac{\pi}{4\beta}\left(\frac{|F|}{F_c} - (1-\beta)\right)\right]; & (1-\beta)F_c \le |F| \le (1+\beta)F_c \\ 0; & |F| \ge (1+\beta)F_c \end{cases}$$

  where the cutoff frequency is $F_c = 1/(2T_s)$, T_s is the symbol time and $0 \le \beta \le 1$ is the roll-off.
- In the time domain,

$$h_{rrc}(t) = \begin{cases} \frac{1}{T_s} + \frac{\beta}{T_s}\left(\frac{4}{\pi} - 1\right); & t = 0 \\ -\frac{\beta}{T_s}\left[\frac{2}{\pi}\cos\left(\frac{\pi}{4\beta}(1+\beta)\right) - \cos\left(\frac{\pi}{4\beta}(1-\beta)\right)\right]; & t = \pm\frac{T_s}{4\beta} \\ \frac{1}{T_s}\frac{\frac{4\beta t}{T_s}\cos\left(\frac{\pi(1+\beta)t}{T_s}\right) + \sin\left(\pi(1-\beta)\frac{t}{T_s}\right)}{\frac{\pi t}{T_s}\left[1-\left(\frac{4\beta t}{T_s}\right)^2\right]}; & \text{otherwise} \end{cases}$$

Practical Square Root Raised Cosine

- The square root raised cosine pulse, defined on the previous slide, is of infinite length, since it's frequency response has a finite length, with an absolute bandwidth of $(1+\beta)/2T_s$.
- In a practical implementation, the pulse is truncated to $-(N_{symb}-1)T_s/2 \leq t \leq (N_{symb}-1)T_s/2$, where N_{symb} is the number of symbols in the pulse. In MATLAB N_{symb} is typically referred to as the span.
- The practical square root raised cosine pulse has finite length, thus the frequency response of the pulse must have infinite length and this increased length takes the form of sidelobes.
- A larger value for the span (N_{symb}) results in a greater attenuation of these sidelobes, but the practical stopband attenuation is limited to about 46 dB, depending on β and the length of the pulse.

Ideal Discrete-Time Square Root Raised Cosine

- The discrete-time version of the square root raised cosine impulse response can be obtained exactly using the impulse invariance technique, since $H_{rrc}(F)$ is bandlimited.
- Using the impulse invariance technique, the discrete-time impulse response is

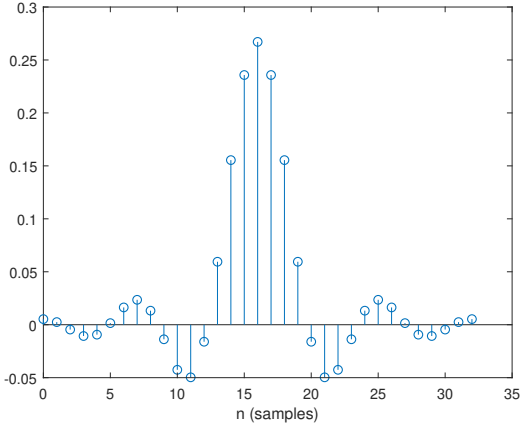
$$h_{rrc}[n] = T h_{rrc}(t)|_{t=nT}$$

where T is the sampling time. Then substituting $N_{sps} = T_s/T$ gives

$$h_{rrc}[n] = \begin{cases} \frac{1}{N_{sps}} + \frac{\beta}{N_{sps}}\left(\frac{4}{\pi} - 1\right); & n = 0 \\ -\frac{\beta}{N_{sps}}\left[\frac{2}{\pi}\cos\left(\frac{\pi}{4\beta}(1+\beta)\right) - \cos\left(\frac{\pi}{4\beta}(1-\beta)\right)\right]; & n = \pm\frac{N_{sps}}{4\beta} \\ \frac{1}{N_{sps}} \frac{\frac{4\beta n}{N_{sps}}\cos\left(\frac{\pi(1+\beta)n}{N_{sps}}\right) + \sin\left(\pi(1-\beta)\frac{n}{N_{sps}}\right)}{\frac{\pi n}{N_{sps}}\left[1-\left(\frac{4\beta n}{N_{sps}}\right)^2\right]}; & \text{otherwise} \end{cases}$$

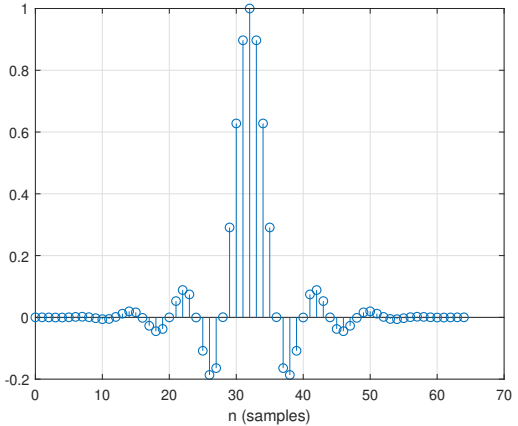
Square Root Raised Cosine Filter Impulse Response

- The following is a plot of a length 33, truncated and shifted root raised cosine pulse for $\beta = 0.25, N_{sps} = 4$ and $M = 32$. Note that this is not a unit energy pulse.



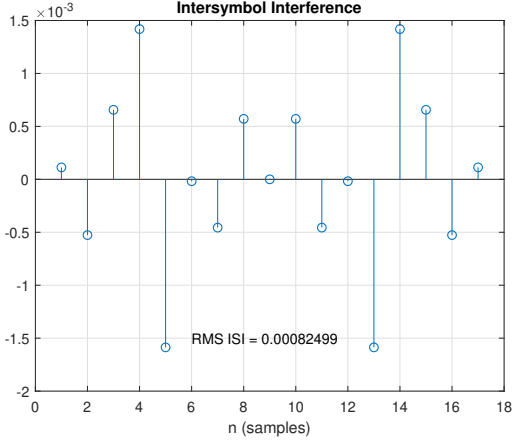
Convolution of Square Root Raised Cosine Filters

- The convolution of two square root raised cosine filters results in a raised cosine response.
- The square root raised cosine pulses that were used in the convolution had unit energy, this results in a peak value of 1 for the raised cosine impulse response.



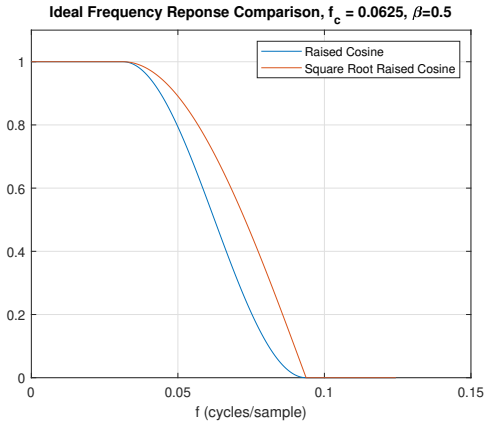
ISI of $h_{rc}[n] = h_{rrc}[n] * h_{rrc}[n]$

- The following is a plot of every $N_{sps} = 4^{th}$ sample of the raised cosine resulting from the convolution $h_{rrc}[n] * h_{rrc}[n]$ (not including the peak value), shown in the previous slide.
- The non-channel induced ISI is a result of the truncation of the ideal square root raised cosine impulse response.



Ideal Frequency Response Comparison

- The following figure is a comparison of the ideal frequency response for a raised cosine and a square root raised cosine filter.
- Note the raised cosine has a gain of 1/2 at the cutoff frequency and the square root raised cosine has a gain of $1/\sqrt{2}$ at f_c.



Transition Width for Ideal Frequency Response

- The transition width for the square root raised cosine can be determined from the ideal frequency response equation on slide 167.
- The transition width is $f_s - f_p$, where the continuous time frequency is normalized by the sampling period T

$$f_p = (1 - \beta)F_c T = (1 - \beta)f_c$$

and

$$f_s = (1 + \beta)F_c T = (1 + \beta)f_c$$

where the cutoff frequency, $F_c = 1/(2T_s)$ is scaled by T to give

$$f_c = TF_c = T/(2T_s) = 1/(2N_{sps})$$

- For the figure on the previous page, you can see that both the raised cosine and square root raised cosine have the same f_p and f_s, where $f_p = (1 - 0.5)0.0625 = 0.03125$ and $f_s = (1 + 0.5)0.0625 = 0.09375$.

Improving the Stopband Attenuation

- The truncated square root raised cosine filter has sidelobes that are rather high, approximately 24 to 46 dB below the passband gain depending on β and the length of the filter.
- Typically, the desired out of band attenuation is on the order of 60 to 80 dB.
- Another window could be applied to the square root raised cosine impulse response to improve the stopband attenuation, but this leads to significant increases in the non-channel ISI levels, which would negatively impact a communications system .
- One method to improve the stopband attenuation, while maintaining the -3 dB gain point at f_c, is to use firpm to design a square root Nyquist filter as proposed in the paper (on the EE461 website):

  F. Harris, C. Dick, S. Seshagiri, and K. Moerder, "An Improved Square-Root Nyquist Shaping Filter," Proc. of the Software-Defined Radio Forum, 2005.

Square Root Nyquist Filter Using firpm

- A square root Nyquist filter with better stopband attenuation can be designed using the Parks-McClellan (equiripple) method, which is implemented in MATLAB as the function firpm.
- The filter gain at the cutoff frequency should be $1/\sqrt{2}$ (-3 dB gain). A zero width frequency interval is used to achieve this gain specification in firpm.
- The approach taken is to specify three frequency points in the transition band of the filter and then use firpm to generate an impulse response that fits these points.
- The three points are f_p with a gain of one, f_c with a gain of $1/\sqrt{2}$ and f_s with a gain of zero.
- The MATLAB code used to generate the impulse response is:

```
beta=0.2; Nsps=4; span=25;
M=span*Nsps; %order of the filter
fc=1/(2*Nsps); fp=(1-beta)*fc; fs=(1+beta)*fc;
fb = [0 fp fc fc fs .5]*2;
a = [1 1 1/sqrt(2) 1/sqrt(2) 0 0];
wght = [2.4535 1 1];
h=firpm(M,fb,a,wght);
```

Choosing an Order for the Square Root Nyquist Filter

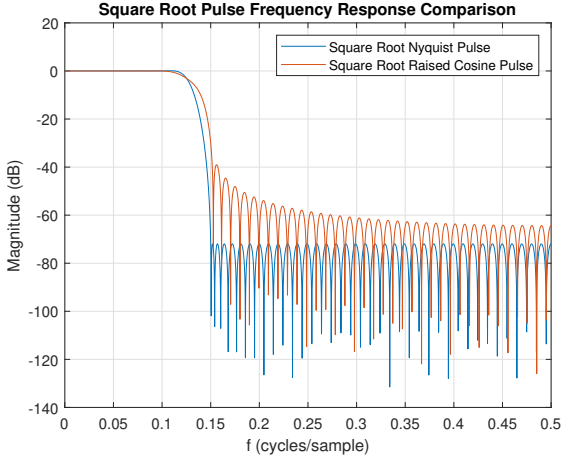
- Choosing the order for the algorithm on the previous slide is an iterative process, though `firpmord` could be used to choose an initial estimate.
- For example, for the previous slide the order was chosen by selecting a desired stopband attenuation of 70 dB and using the following MATLAB code:

```
ford=fb(2:end-1);
aord=[1 1/sqrt(2) 0];
sb=10^(-70/20);
dev=[sb 0.02 sb];
M=firpmord(ford,aord,dev)
```

- The second element of `dev` is the deviation of the zero width frequency interval about the gain $1/\sqrt{2}$. The actual deviation in the designed filter is much better than this value.
- Executing this code generated an order of 100, which is implemented in the previous slide by setting `span` (or `Nsymb`) to be 25, since `Nsps` is set to be 4.

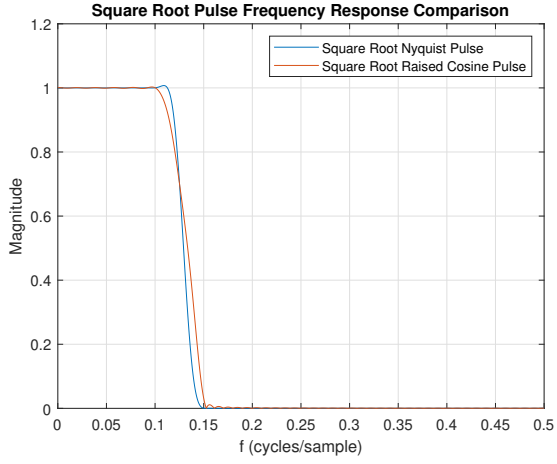
Square Root Pulse Frequency Response Comparison

- The following figure is a comparison of the frequency response of the square root raised cosine pulse with the frequency response of the square root Nyquist pulse determined using `firpm`. The order is the same for both cases.



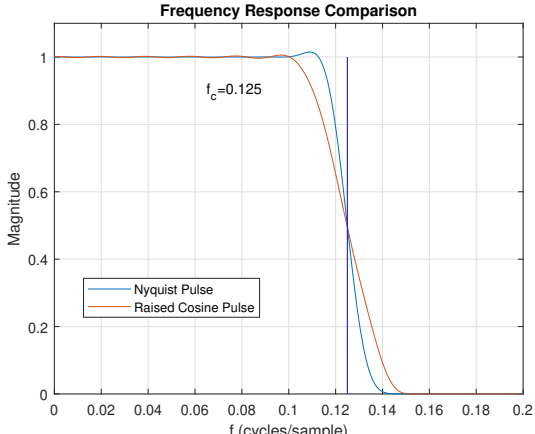
Square Root Pulse Frequency Response Comparison

- This figure is the linear plot of the figure on the last slide. Observe a spectral bump at the edge of the passband. This can be removed using approaches described in the paper listed on slide 175. These approaches will also improve the ISI for the figure on slide 182.



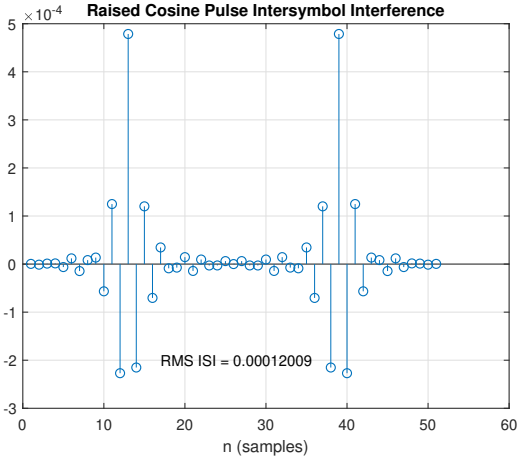
Raised Cosine - Nyquist Pulse Frequency Response Comparison

- The following figure is a frequency response comparison of the raised cosine pulse with the Nyquist pulse, both derived by taking the convolution of the respective impulses responses used to generated the frequency responses on the previous slide (or by squaring the frequency responses on the previous slide).



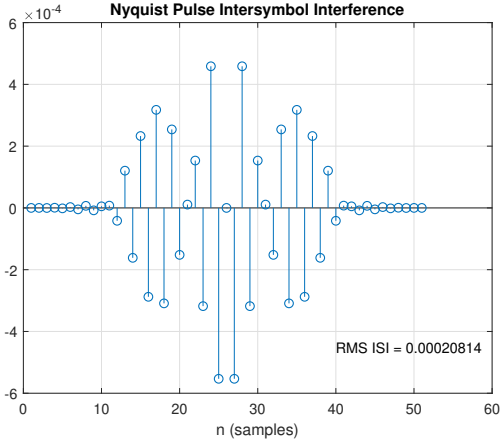
Raised Cosine Pulse Intersymbol Interference

- The following is a plot of every $N_{sps} = 4^{th}$ sample of the raised cosine resulting from the convolution $h_{rrc}[n] * h_{rrc}[n]$ (not including the peak value) ($\beta = 0.2, M = 100$)



Nyquist Pulse Intersymbol Interference

- The following is a plot of every $N_{sps} = 4^{th}$ sample of the Nyquist pulse resulting from the convolution of the square root Nyquist pulse with itself (not including the peak value), which was generated using `firpm`.



MATLAB rcosdesign

- Raised cosine and square root raised cosine filters can be designed in MATLAB using `rcosdesign` (it replaces `firrcos`).
- A raised cosine impulse response is generated using
  `hrc = rcosdesign(beta,Nsymb,Nsps,'normal')`
- A square root raised cosine impulse response is generated using
  `hrrc = rcosdesign(beta,Nsymb,Nsps)`
- The length of the filter is `Nsymb*Nsps + 1` and thus the order of the filter is $M = N_{symb}N_{sps}$.
- If you check the help file (doc rcosdesign). you can see that MATLAB uses `span` for `Nsymb`.
- Both `hrc` and `hrrc` are generated as unit energy pulses (ie. $\sum h_{rrc}^2 = 1$ and $\sum h_{rc}^2 = 1$).
- Thus the nominal passband gain will not be 1, as was defined earlier for the ideal frequency response expressions for the two pulses.
- These pulses need to be scaled to produce a nominal gain of 1, as described on the following two slides.

Scaling the Square Root Raised Cosine Pulse

- `rcosdesign` produces a unit energy square root raised cosine, $h_{rrc}[n]$.
- For a square root raised cosine pulse with unit energy, the convolution, $h_{rrc}[n] * h_{rrc}[n]$ results in raised cosine pulse with a peak value of 1, since for the convolution shift when the pulses completely overlap $\sum(h_{rrc}[n]h_{rrc}[n]) = 1$.
- If a nominal passband gain of 1 for the frequency response is desired, then the pulse can be scaled to produce the same square root raised cosince pulse that would have been produced by the truncated ideal pulse using (see h_{rrc} on slide 169)

$$h_{rrcs} = \frac{h_{rrc}[n]}{\max(h_{rrc}[n])}\left(\frac{1}{N_{sps}} + \frac{\beta}{N_{sps}}(\frac{4}{\pi} - 1)\right)$$

- A simpler approach is to scale by the DC value of the pulse, though this will be slightly different than the above approach, because the pulse is truncated. This will result in $H(e^{j0})$ being set to 1, and it is unlikely the ripple will be centered about the nominal gain of 1 in the passband.

$$h_{rrcs}[n] = \frac{h_{rrc}[n]}{\sum h_{rrc}[n]}$$

Scaling the Raised Cosine Pulse

- `rcosdesign` produces a unit energy raised cosine, $h_{rc}[n]$.
- If a nominal passband gain of 1 for the frequency response is desired, then the raised cosine pulse can be scaled to produce the same pulse that would have been produced by the truncated ideal pulse using (see h_{rc} on slide 164)

$$h_{rcs} = \frac{h_{rc}[n]}{\max(h_{rc}[n])N_{sps}}$$

- A simpler approach is to scale by the DC value of the pulse, though this will be slightly different than the above approach, because the pulse is truncated. This will result in $H(e^{j0})$ being set to 1, and it is unlikely the ripple will be centered about the nominal gain of 1 in the passband.

$$h_{rcs}[n] = \frac{h_{rc}[n]}{\sum h_{rc}[n]}$$

- If a peak value of 1 is desired for the raised cosine pulse then simply normalize the pulse with it's maximum value.

$$h_{rcs} = \frac{h_{rc}[n]}{\max(h_{rc}[n])}$$

Section 4

Multirate Signal Processing

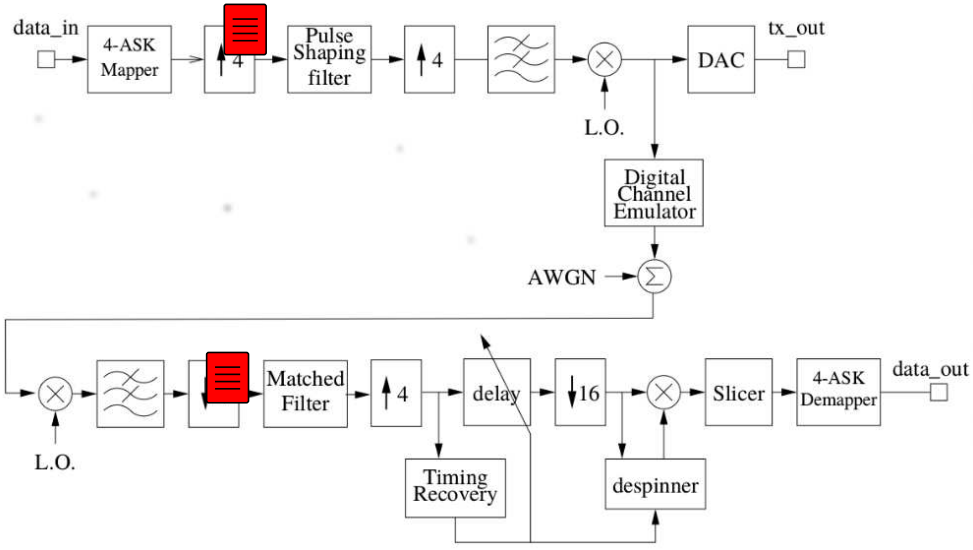
Subsection 1

Upsampling and Downsampling

Introduction

- Multirate signal processing systems are discrete-time systems that use more than one sampling rate.
- An example multirate application is the conversion between digital audio formats, broadcasting is 32 KHz and audio CD is 44.1 KHz.
- The two basic multirate processing operations are upsamping and downsampling, which are special cases of resampling.
- Resampling basically rescales the time axis, which also results in a scaling of the frequency axis.
- The following slides introduce upsampling and downsampling.
- The slides focus on how these operations affect signals in both the time domain and frequency domain.

EE465 QAM System



Upsampling in the Time Domain

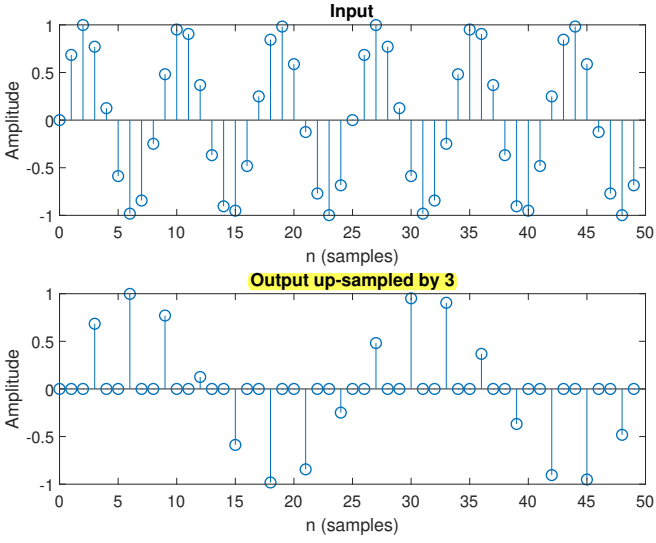
- An upsampling operation resamples the input at a rate L times higher than the present input sampling rate.
- This upsampling operation is performed by inserting $L-1$ zeros between consecutive samples of the input sequence, as defined by

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

- The block diagram representation of the upsampling operation is

- Note that upsampling is a linear, time-varying operation.

Upsampling Time Domain Figure



Upsampling MATLAB Script

The figure on the previous slide was generated using:

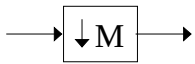
```
Nl=50; f=3/25; L=3;
n = 0:Nl-1;
x = sin(2*pi*f*n);
y=upsample(x,L);
subplot(2,1,1)
stem(n,x);
title('Input');
xlabel('n (samples)');ylabel('Amplitude');
subplot(2,1,2)
stem(n,y(1:length(x)));
title(['Output up-sampled by ', num2str(L)]);
xlabel('n (samples)');ylabel('Amplitude');
```

Downsampling in the Time Domain

- A downsampling operation resamples the input at a rate $1/M$ times lower than the present input sampling rate.
- This downsampling operation is performed by keeping every M^{th} sample of the input sequence and discarding the other $M-1$ in-between samples, as defined by

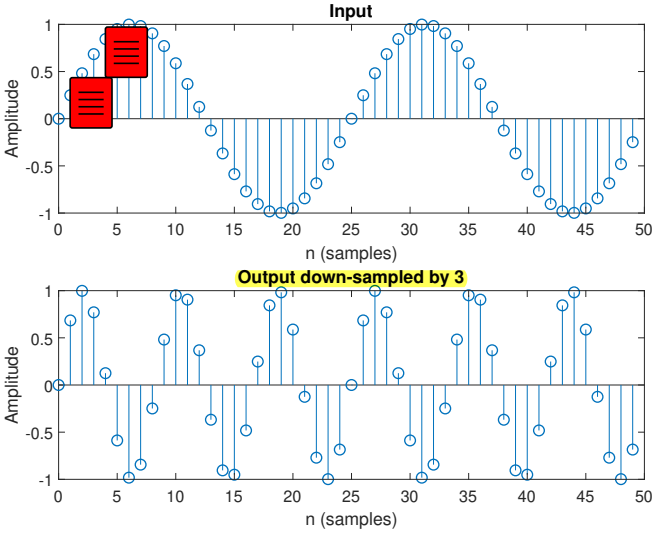
$$x_d[n] = x[nM]$$

- The block diagram representation of the downsampling operation is



- Note that downsampling is a linear, time-varying operation.

Downsampling Time Domain Figure



Downsampling MATLAB Script

The figure on the previous slide was generated using:

```
Nl=50; f=1/25; M=3;
n = 0:Nl-1;
m = 0 : Nl*M-1;
x = sin(2*pi*f*m);
y=downsample(x,M);
subplot(2,1,1)
stem(n, x(1:Nl));
title('Input');
xlabel('n (samples)');ylabel('Amplitude');
subplot(2,1,2)
stem(n,y);
title(['Output down-sampled by ', num2str(M)]);
xlabel('n (samples)');ylabel('Amplitude');
```

Upsampling in the Frequency Domain

- Take the z-transform of the upsampling output assuming $L = 2$

$$\begin{aligned} X_u(z) &= \sum_{n=-\infty}^{\infty} x_u[n] z^{-n} = \sum_{n=-\infty}^{\infty} x[n/2] z^{-n}, \quad n \text{ even} \\ &= \sum_{m=-\infty}^{\infty} x[m] z^{-2m} = X(z^2) \end{aligned}$$

- For a factor of L up-sampler

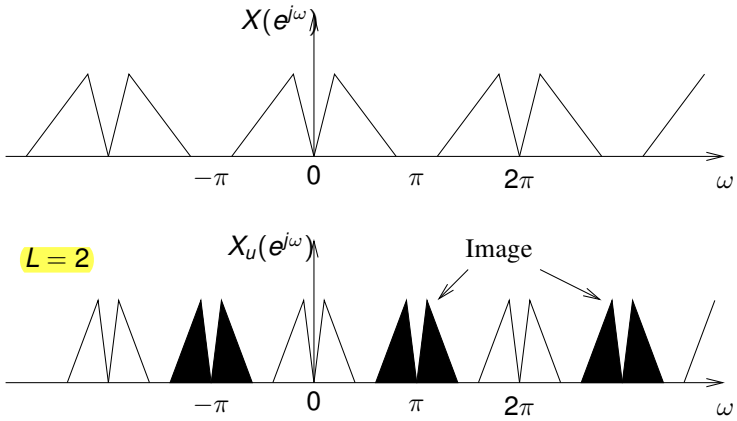
$$X_u(z) = X(z^L)$$

- The frequency response can be determined by evaluating the z-transform on the unit circle. Setting $z = e^{jw}$ gives the DTFT

$$X_u(e^{jw}) = X(e^{jwL})$$

Upsampling Frequency Domain Figure

- This is an example of the effects of upsampling in the frequency domain. The top graph is the input spectrum and the bottom graph is the output spectrum for $L = 2$.



Example 27: Upsampling Frequency Response

A signal, $x[n]$, with a amplitude response $X(e^{j\omega}) = 1$; $-0.25 \leq f \leq 0.25$, is upsampled by 2. Generate a plot of the amplitude response of the upsampled signal for $-0.5 \leq f \leq 0.5$.

Sampling Function

- A useful identity is the closed form expression for the discrete sampling function.

$$c[n] = \begin{cases} 1, & n = 0, \pm M, \pm 2M, \ldots, \\ 0, & \text{otherwise.} \end{cases}$$

$$= \frac{1}{M} \sum_{k=0}^{M-1} e^{-j\frac{2\pi}{M}nk}$$

- Closed form proof: For $n = 0, \pm M, \pm 2M, \ldots,$

$$\frac{1}{M} \sum_{k=0}^{M-1} e^{-j\frac{2\pi}{M}nk} = \frac{1}{M} \sum_{k=0}^{M-1} 1 = 1,$$

since

$$e^{-j\frac{2\pi}{M}Mk} = 1.$$

Sampling Function

- For n other than $0, \pm M, \pm 2M, \ldots$,

$$\frac{1}{M}\sum_{k=0}^{M-1} e^{-j\frac{2\pi}{M}nk} = \frac{1}{M}\left(\frac{1-e^{-j\frac{2\pi}{M}nM}}{1-e^{-j\frac{2\pi}{M}n}}\right)$$

  using the closed form for a summation.

- For the expression in parenthesis, the top exponential is always one and the bottom exponential is never one, thus

$$\frac{1}{M}\sum_{k=0}^{M-1} e^{-j\frac{2\pi}{M}nk} = \frac{1}{M}\left(\frac{1-1}{1-e^{-j\frac{2\pi}{M}n}}\right) = 0,$$

  which completes the proof.

Downsampling in the Frequency Domain

- The downsampling operation is defined as

$$x_d[n] = x[nM]$$

- Define $x_i[n]$, whose values are the same as $x[n]$ at multiples of M and zero at other values of n,

$$x_i[n] = c[n]x[n]$$

where $c[n]$ is the sampling function,

$$\begin{aligned} c[n] &= \begin{cases} 1, & n = 0, \pm M, \pm 2M, \ldots, \\ 0, & \text{otherwise.} \end{cases} \\ &= \frac{1}{M} \sum_{k=0}^{M-1} e^{-j\frac{2\pi}{M}nk} \end{aligned}$$

Downsampling in the Frequency Domain

- Taking the z-transform of $x_d[n] = x[nM] = x_i[nM]$

$$
\begin{aligned}
X_d(z) &= \sum_{n=-\infty}^{\infty} x_i[nM] z^{-n} = \sum_{k=-\infty}^{\infty} x_i[k] z^{-k/M}, \\
&= \sum_{k=-\infty}^{\infty} c[k] x[k] z^{-k/M} \\
&= \sum_{k=-\infty}^{\infty} \left(\frac{1}{M} \sum_{m=0}^{M-1} e^{-j\frac{2\pi}{M}km} \right) x[k] z^{-k/M} \\
&= \frac{1}{M} \sum_{m=0}^{M-1} \left(\sum_{k=-\infty}^{\infty} x[k] e^{-j\frac{2\pi}{M}km} z^{-k/M} \right)
\end{aligned}
$$

Downsampling in the Frequency Domain

- Thus the z-transform of $x_d[n]$ is,

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

- For example, assuming $M = 2$, the down-sampled spectrum is given by

$$\begin{aligned} X_d(z)|_{z=e^{jw}} &= \frac{1}{2}(X(z^{1/2}) + X(-z^{1/2})) \\ X_d(e^{jw}) &= \frac{1}{2}(X(e^{j\omega/2}) + X(-e^{j\omega/2})) \\ &= \frac{1}{2}(X(e^{j\omega/2}) + X(e^{j(\omega-2\pi)/2})) \end{aligned}$$

Downsampling in the Frequency Domain

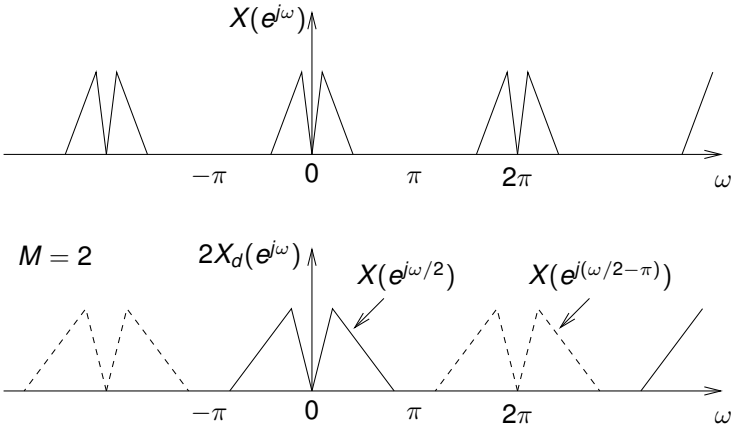
- This expression can be generalized for any M as,

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

- Generating $X_d(e^{jw})$ involves
  1. Drawing M copies of $X(e^{jw})$ with each shifted by $\frac{2\pi m}{M}$ for $m = 0, 1, \cdots, M-1$.
  2. Add all the shifted copies together and scale the amplitude by $\frac{1}{M}$.
  3. Stretch the frequency axis by M (ie. multiply each of the x-axis label values by M).

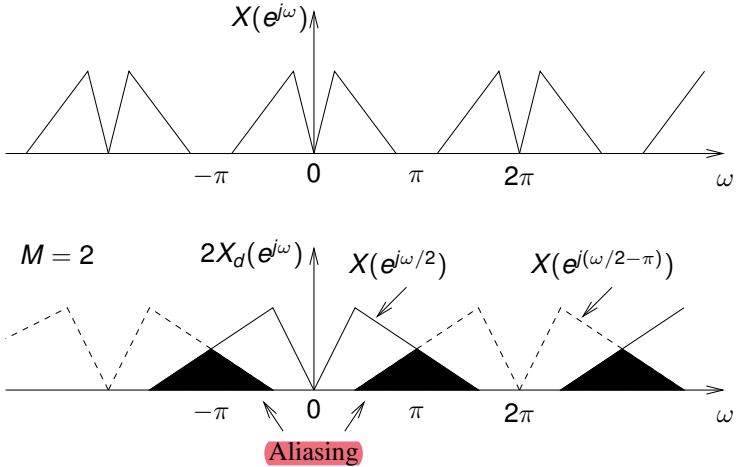
Downsampling Frequency Domain Figure

- This is an example of the effects of downsampling in the frequency domain with no aliasing. The top graph is the input spectrum and the bottom graph is the output spectrum for $M = 2$.



Downsampling with Aliasing Frequency Domain Figure

- This is an example of the effects of downsampling in the frequency domain with aliasing. Here there is aliasing since $X(e^{j\omega})$ is nonzero for $\omega > \pi/2$. The top graph is the input spectrum and the bottom graph is the output spectrum for $M = 2$.



Example 28: Downsampling Frequency Response

A signal, $x[n]$, with a amplitude response $X(e^{j\omega}) = 2; \quad -0.125 \leq f \leq 0.125$, is downsampled by 2. Generate a plot of the amplitude response of the downsampled signal for $-0.5 \leq f \leq 0.5$.

Subsection 2

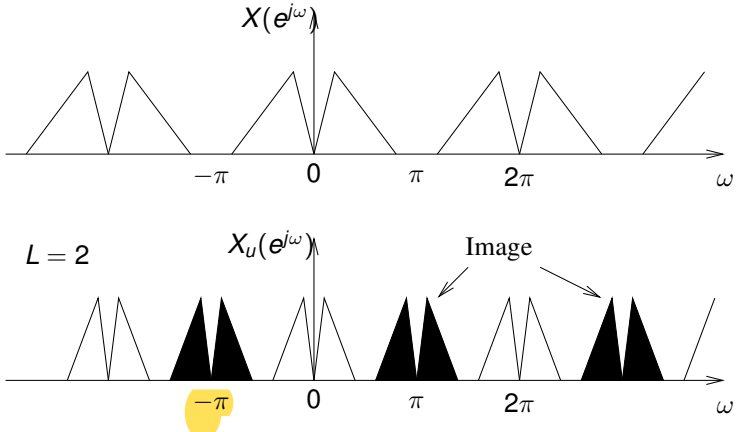
Interpolation and Decimation

Introduction

- Filtering is often used to "fill in" the zeros between each nonzero sample of the signal generated using upsampling.
- This filtering process is referred to as interpolation.
- The combination of an upsampling operation followed by an interpolation filter is typically referred to as an interpolator.
- As previously discussed in the last section, aliasing is possible when using downsampling, thus a lowpass filter is used prior to the downsampling operation to eliminate the possibility of aliasing.
- The combination of the filter followed by the downsampling operation is referred to as a decimation (even though the downsampling operation is typically a factor other than 10).
- This combination is also referred to as a decimator.

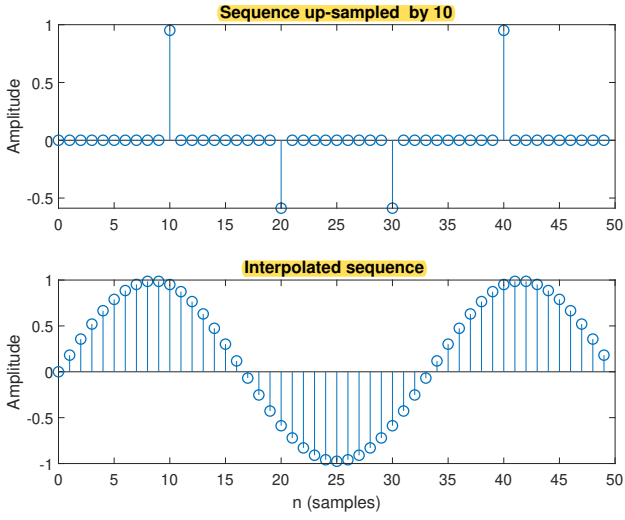
Interpolation

- The up-sampling operation results in periodic repetition of the original spectrum producing unwanted images as shown in this figure. These images must be removed using an interpolation filter.



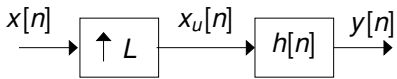
Interpolation

- The interpolation process inserts the missing samples in the up-sampled sequence, as shown here,



Interpolation Filter

- The interpolation filter is just a low-pass filter which is used to remove all of the images leaving the one of interest. A block diagram of the interpolator system is,



- The Nyquist frequency for $x[n]$ is π radians/sample. Upsampling by L compresses the frequency axis, resulting in the $x[n]$ Nyquist frequency moving to $\omega = \frac{\pi}{L}$ in $x_u[n]$.
- Thus, an ideal lowpass filter with a cutoff frequency of $\omega_c = \frac{\pi}{L}$ will remove the images, resulting in an interpolated signal, $y[n]$.

Interpolation Filter

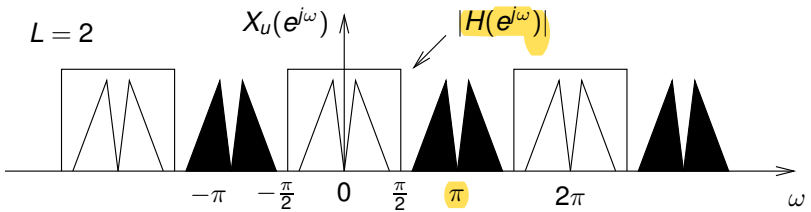
- If $x_u[n]$ is passed through an ideal lowpass filter with a cutoff of π/L, the filter will remove $L-1$ images from the input signal spectrum. The loss in spectrum bandwidth is compensated for by scaling the spectrum by L, implemented as a filter gain of L, as done in Oppenheim and Schafer.
- Define the magnitude of the ideal interpolation filter as

$$|H(e^{j\omega})| = \begin{cases} L, & |\omega| \leq \pi/L, \\ 0, & \pi/L < |\omega| \leq \pi \end{cases}$$

- In a practical filter, the stopband corner frequency will be at $\omega_s = \pi/L$ and the passband corner frequency will be defined to preserve the interpolated signal, while providing a transition band.

Interpolation Filter

- A graphical view of the frequency response of the ideal interpolation filter and the up-sampled signal, for $L = 2$, is given by,



Example 29: Interpolation

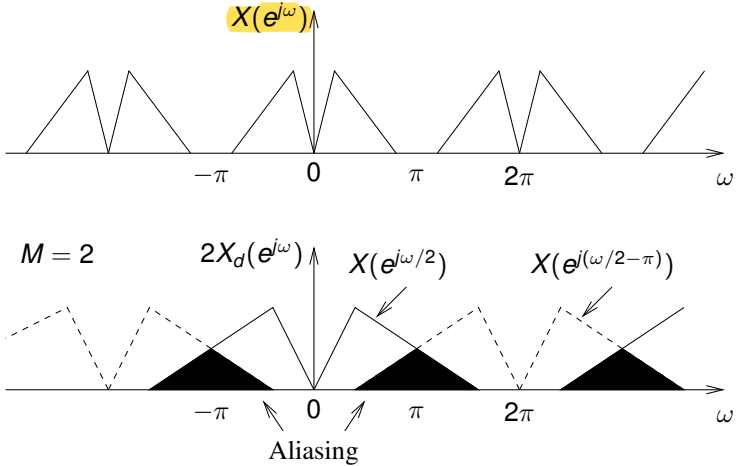
- Generate 2 cycles of a 1/24 cycles/sample cosine. Interpolate this sinusoid by a factor of L=2. For the interpolation filter, use a halfband filter designed using a Kaiser window. The interpolation filter should have a stopband of 60 dB and a transition width of 0.1 cycles/sample.
- See the mfile, interpolate_halfband.m

Example 30: Function Using `interp`

- In this example, a sequence consisting of the sum of two sinusoids is generated and then this sequence is interpolated by a factor of L.
- See the mfile function, interpolation_example.m

Decimation

- The down-sampling process can result in aliasing, as previously demonstrated in the following figure. Thus the sequence must be filtered prior to down-sampling.



Decimation Filter

- The downsampler expands the $x[n]$ spectrum by M, thus any frequencies in $x[n]$ greater than $\frac{\pi}{M}$ will result in aliasing. This is handled with an anti-aliasing filter placed before the downsampler.

Ideal Anti − aliasing filter, $\omega_c = \frac{\pi}{M}$

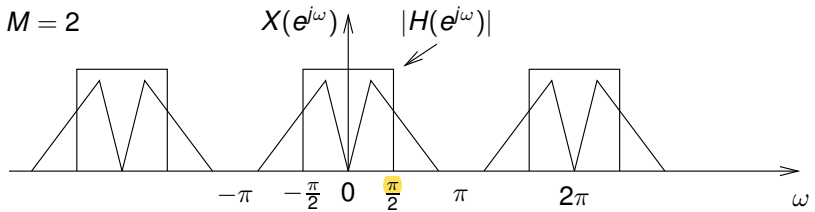
- The required decimation filter is

$$|H(e^{j\omega})| = \begin{cases} 1, & |\omega| \leq \pi/M, \\ 0, & \pi/M < |\omega| \leq \pi \end{cases}$$

- This combination of filter and downsampler is referred to as a decimator.

Decimation Filter

- A graphical view of the frequency response of the ideal decimation filter for $M = 2$, and the original signal that must be filtered, is shown here,



- In a practical filter, the stopband edge will be at $\omega_s = \pi/M$ and the passband edge will be defined to preserve the signal, while providing a transition band.

Example 31: Function Using `decimate`

- In this example, a sequence consisting of the sum of two sinusoids is generated and then this sequence is decimated by a factor of M.
- See the mfile function, decimation_example.m

Subsection 3

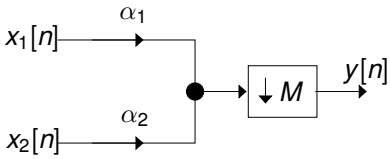
Identities, Re-Sampling and an Efficient Decimator

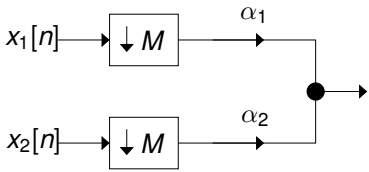
Introduction

- This section introduces six upsampling and downsampling identities that are useful when working with multirate systems.
- Two applications that use the identities to improve the their implementation efficiency are then considered:
  1. Fractional Re-Sampling - which involves changing the sampling rate by a fractional amount.
  2. Efficient Direct-Form Decimator Implementation - which involves using indentity 1 to reduce the computation required.

Downsampling Identities

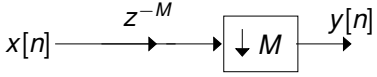
- Identity 1 - The following are equivalent

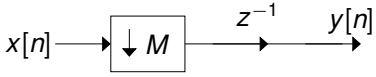




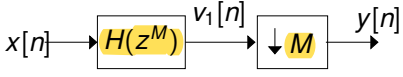
Downsampling Identities

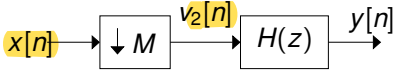
- Identity 2 - The following are equivalent





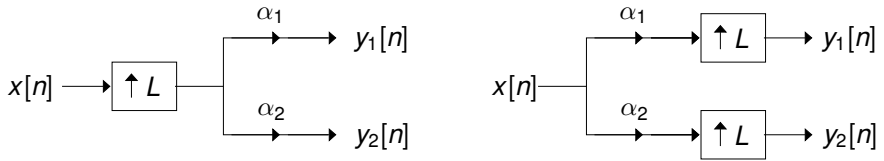
- Identity 3 - The following are equivalent





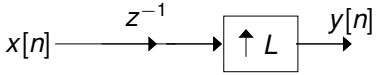
Upsampling Identities

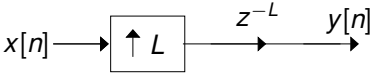
- Identity 4 - The following are equivalent



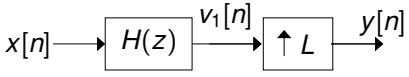
Upsampling Identities

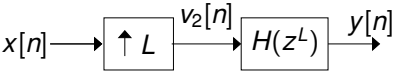
- Identity 5 - The following are equivalent





- Identity 6 - The following are equivalent





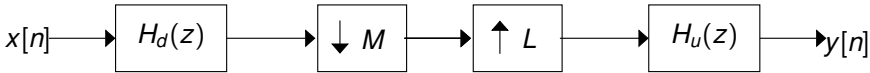
Interconnection of Sampling Blocks

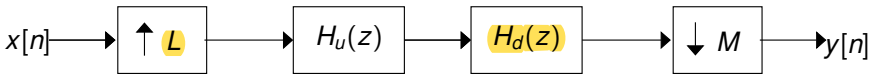
- These two are equivalent only if L and M are relatively prime numbers (greatest common divisor is 1).

- For example, these will work
  - $L = 3, M = 2$
  - $L = 2, M = 5$
- These will not work
  - $L = 2, M = 2$
  - $L = 2, M = 6$

Fractional Sampling Rate Alteration

- Changing the sampling rate by a fractional amount can be achieved by cascading a decimator with an interpolator.
- There are two possible ways to do this, as shown in the figure,

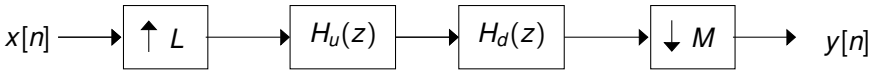


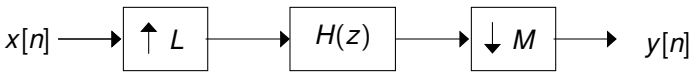


- This cascade is equivalent to a decimator with a decimation factor of $\frac{M}{L}$ $(M > L)$ or an interpolator with an interpolation factor of $\frac{L}{M}$ $(L > M)$.

Fractional Sampling Rate Alteration

- The second cascade connection is more efficient since, the two filters can be combined into one, as shown in this figure,





- The practical filter $H(z)$ has a gain of L and a stopband corner frequency of

$$\omega_s = \min\left(\frac{\pi}{L}, \frac{\pi}{M}\right)$$

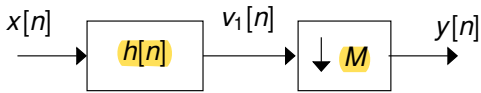
which will discard up-sampler images and perform the anti-aliasing required for the down-sampler.

Example 32: Fractional Re-Sampling

- In this example, a sinusoidal sequence is generated and then this sequence is resampled by a non-integer factor.
- Generate 3 cycles of a 1/24 cycles/sample sine. Resample this signal with a system consisting of a upsampler, followed by a filter, followed by a downsampler, where L=2 and M=3. Design the filter using a Kaiser window. The filter should have a stopband of 80 dB and a transition width of 0.1 cycles/sample.
- See the mfile, fractional_example.m

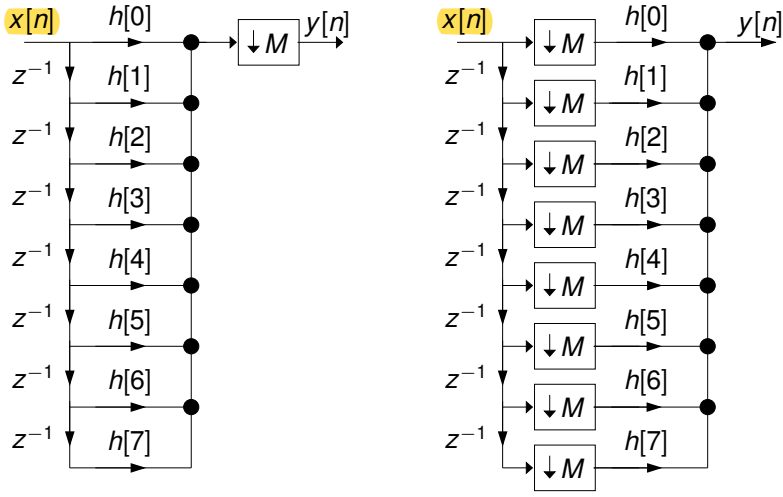
Efficient Decimator Implementations

- The following decimator, which consists of an anti-aliasing filter followed by a down sampler, is not a very efficient implementation.



- For example, if $M = 3$ the downsampler will select every third element in $v_1[n]$, ie. $v_1[0], v_1[3], v_1[6], \ldots$. The other elements, $v_1[1], v_1[2], v_1[4], v_1[5], v_1[7], v_1[8] \ldots$, are discarded. Thus these elements have be calculated unnecessarily.
- Two implementations that eliminate these unnecessary calculations will be presented, an efficient direct-form method and a polyphase implementation.

Efficient Direct-Form Decimator



- The standard direct-form implementation is on the left and the more efficient direct-form implementation is on the right.

Efficient Direct-Form Decimator

- The first identity is used in the conversion of the standard direct-form FIR structure into an efficient direct-form implementation that eliminates the unnecessary calculations.
- This is done by moving the down-sampler inside the direct-form structure as shown on the previous slide.
- In this more efficient structure the number of filter operations is reduced by a factor of M, since the filter operations are now performed at a reduced sampling rate.
- Note that a filter operation consists of a multiplication and an addition.

Subsection 4

Polyphase Decomposition

Introduction

- An efficient method for implementing a decimator can be derived from the polyphase decomposition of an impulse response.
- Polyphase decomposition divides the impulse response, $h[n]$, into M subsequences, $h_k[n]$, with $k = 0, 1, \cdots, M-1$:

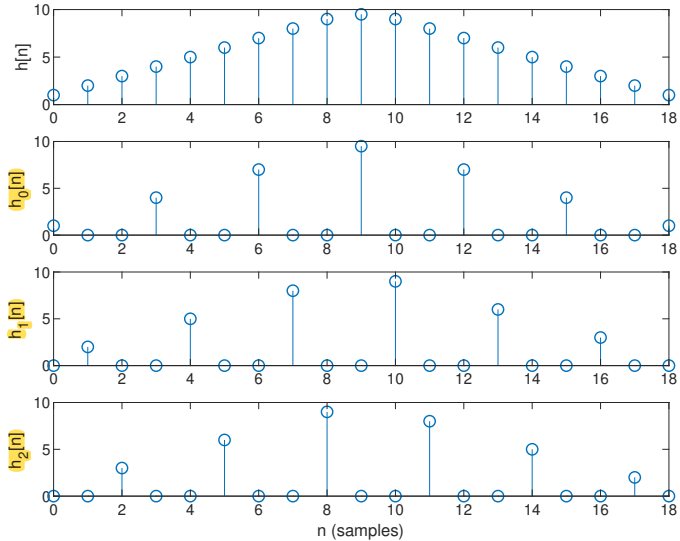
$$h_k[n] = \begin{cases} h[n+k], & n = \text{integer multiple of } M, \\ 0, & \text{otherwise} \end{cases}$$

- $h[n]$ can be constructed from the delayed subsequences as,

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

- An example of this decomposition is given in the figure on the next page for a filter from a decimator with a downsample factor of 3.

Polyphase Decomposition Example ($M = 3$)



Polyphase Filter Structure

- As an example, start with an order 7 FIR filter where the z-transform of the impulse response defined as,

$$\begin{aligned} H(z) &= h[0] + h[1]z^{-1} + h[2]z^{-2} + h[3]z^{-3} + h[4]z^{-4} \\ &+ h[5]z^{-5} + h[6]z^{-6} + h[7]z^{-7} \end{aligned}$$

- Using the polyphase decomposition approach divide $h[n]$ into three parallel filters (three is used because $M = 3$),

$$\begin{aligned} H(z) &= h[0] + h[3]z^{-3} + h[6]z^{-6} \\ &+ z^{-1}(h[1] + h[4]z^{-3} + h[7]z^{-6}) \\ &+ z^{-2}(h[2] + h[5]z^{-3}) \end{aligned}$$

Polyphase Filter Structure

- Define the three terms

$$\begin{aligned} E_0(z) &= h[0] + h[3]z^{-1} + h[6]z^{-2} \\ E_1(z) &= h[1] + h[4]z^{-1} + h[7]z^{-2} \\ E_2(z) &= h[2] + h[5]z^{-1} \end{aligned}$$

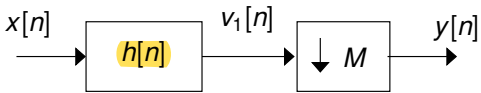
  Each of these is just a filter with the coefficients taken from $h[n]$.

- Using these terms in the last equation on the previous slide gives

$$H(z) = E_0(z^3) + z^{-1}E_1(z^3) + z^{-2}E_2(z^3)$$

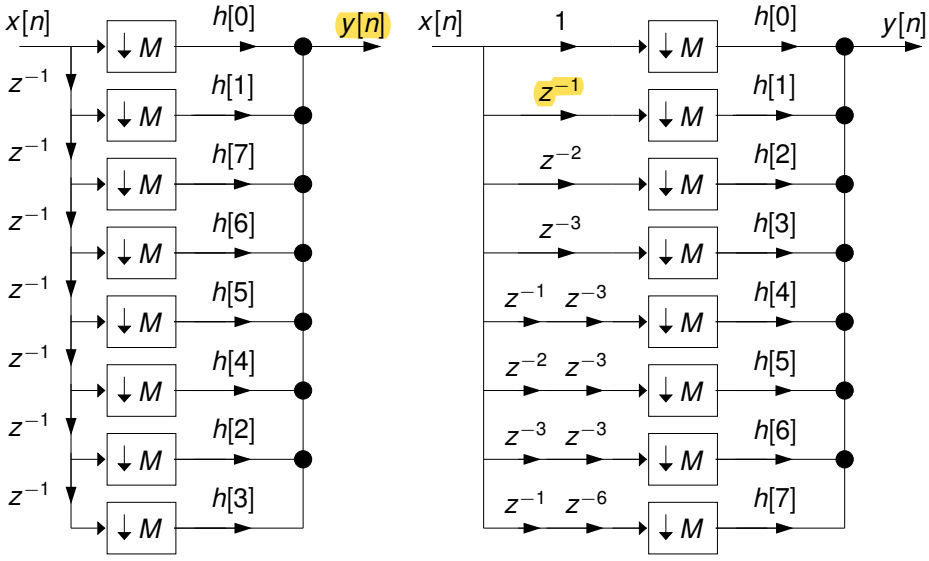
Polyphase Implementation of a Decimator

- A decimator,

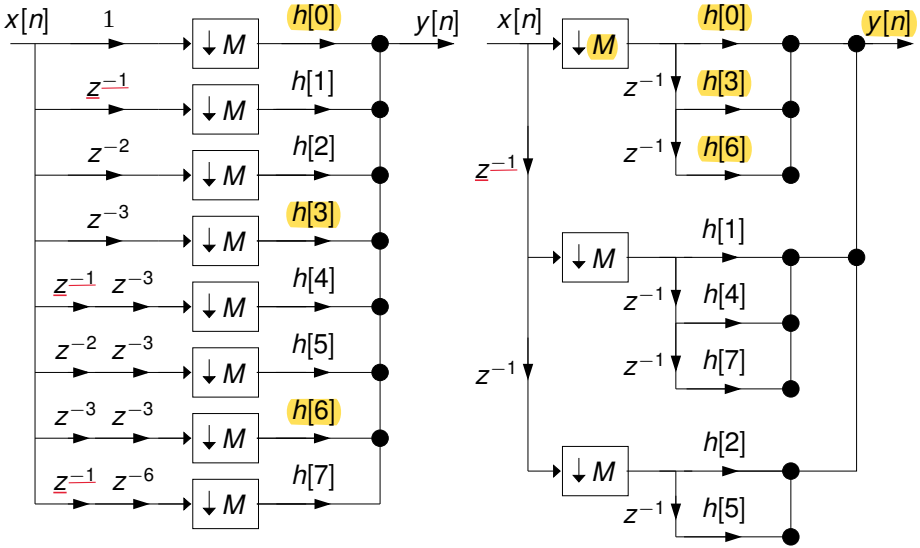


  can be implemented more efficiently with a polyphase filter structure.
- A graphical derivation of this polyphase representation will be presented on the following slides. This derivation starts with the structure derived for the efficient direct-form decimator.
- Throughout this derivation we will assume that $M = 3$.

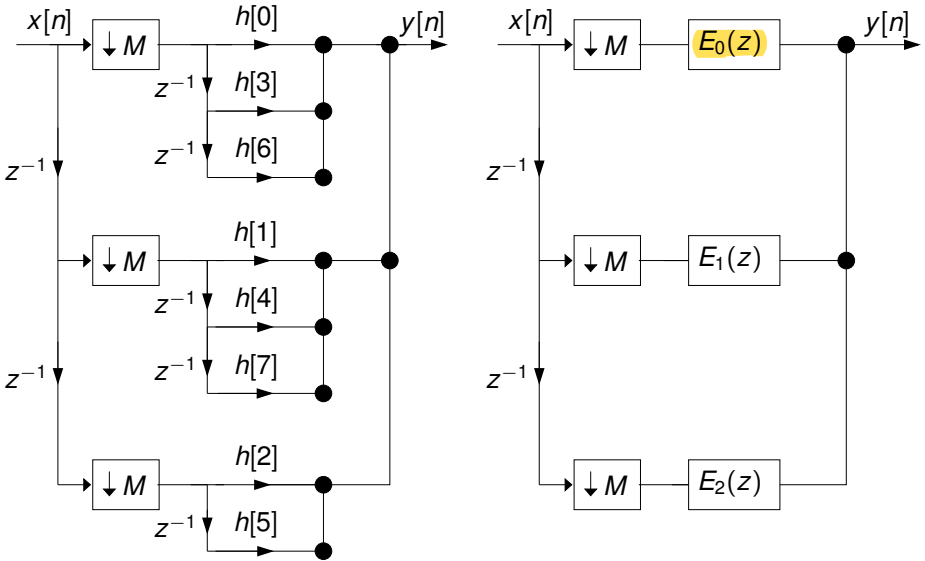
Polyphase Implementation of a Decimator



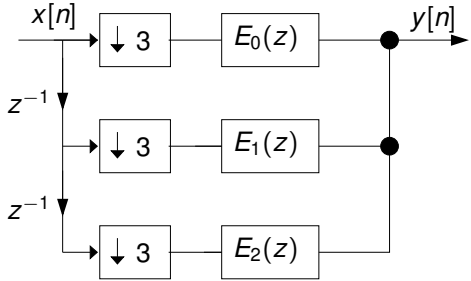
Polyphase Implementation of a Decimator



Polyphase Implementation of a Decimator



Polyphase Implementation of a Decimator



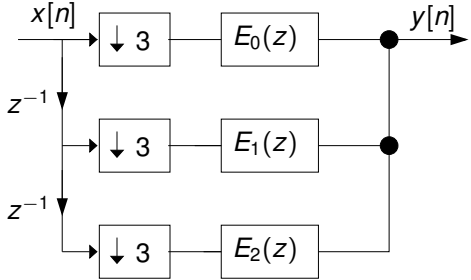
- This implementation is equivalent to using the expression previously derived for the polyphase filter,

$$H(z) = E_0(z^3) + z^{-1}E_1(z^3) + z^{-2}E_2(z^3)$$

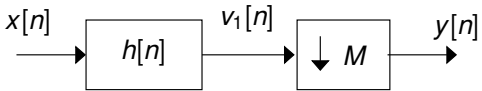
and putting the down-sampler in front of the filter using Identity 3.

Example: Check Polyphase Decimator

- Determine the impulse response for the polyphase decimator (let $x[n] = \delta[n]$)



and compare this response with the impulse response from

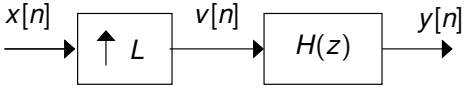


Decimator Computational Comparison

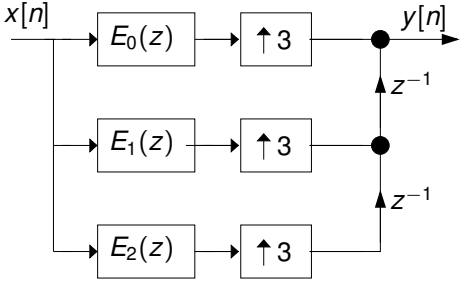
- Standard System
  - $H(z)$ is an M_o^{th} order FIR filter, clocked at 1 sample per unit time.
  - $M_o + 1$ multiplications and M_o additions per unit time.
- Polyphase Implementation
  - Each $E_k(z)$ has order M_o/M (assuming M_o is a multiple of M) and clock rate is 1 per M units of time.
  - Each filter requires $\frac{1}{M}\frac{M_o}{M}$ multiplications and $\frac{1}{M}(\frac{M_o}{M} - 1)$ additions per unit time.
  - M filters, thus need $\frac{M_o}{M}$ multiplications and $(\frac{M_o}{M} - 1) + (M - 1)$ additions.

Polyphase Implementation of an Interpolator

- A standard interpolator is defined as



- Similar to the derivation for the decimator, the polyphase implementation of the interpolator, implementing an 7^{th} order FIR filter with an up-sampling factor of $L = 3$, is given by

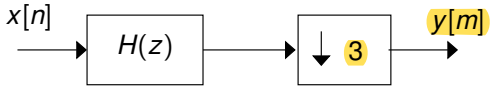


Interpolator Computational Comparison

- Standard System
  - $H(z)$ is an M_o^{th} order FIR filter, the output of the up-sampler is clocked at L samples per unit time.
  - $M_o L$ multiplications and $M_o L - 1$ additions per unit time.
- Polyphase Implementation
  - Each $E_k(z)$ has order M_o/L (assuming M_o is a multiple of L) and clock rate is 1 sample per unit of time.
  - Each filter requires $\frac{M_o}{L}$ multiplications and $(\frac{M_o}{L} - 1)$ additions per unit time.
  - L filters, thus need $L\frac{M_o}{L}$ multiplications and $L(\frac{M_o}{L} - 1)$ additions.

Example 33: Polyphase Decimator with M=3

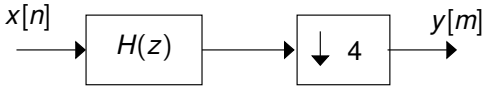
- A system is defined by



where $H(z) = 1 + 2z^{-1} + 3z^{-2} + 2z^{-3} + z^{-4}$. Design an efficient implementation for this system using polyphase decomposition.

Example 34: Polyphase Decimator with M=4

- A system is defined by



where $H(z) = 1 + 2z^{-2} + 2z^{-4} + z^{-8}$. Design an efficient implementation for this system using polyphase decomposition.

Subsection 5

Bandpass Signals and Examples

Common Operations Used with Bandpass Signals

- **Modulation** - One approach to converting a baseband signal $x[n]$ to a bandpass signal is to multiply by a sinusoidal sequence.

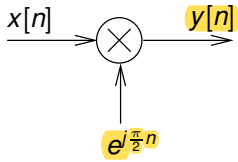
$$\begin{aligned} x[n]\cos(\omega_0 n) &= x[n]\left(\frac{e^{j\omega_0 n}}{2} + \frac{e^{-j\omega_0 n}}{2}\right) \\ &= x[n]\left(\frac{e^{j\omega_0 n}}{2}\right) + x[n]\left(\frac{e^{-j\omega_0 n}}{2}\right) \end{aligned}$$

- **Frequency Shifting** - Multiplying a baseband or bandpass signal, $x[n]$, by a complex exponential shifts the signal in the frequency domain.
  - If the DTFT of $x[n]$ is $X(e^{j\omega})$ then

$$\begin{aligned} x[n]e^{j\omega_0 n} &\longleftrightarrow X(e^{j(\omega-\omega_0)}) \\ x[n]e^{-j\omega_0} &\longleftrightarrow X(e^{j(\omega+\omega_0)}) \end{aligned}$$

Example 35: Frequency Shift

- Generate the DTFT of $y[n]$,



where

$$|X(e^{j\omega})| = \begin{cases} 1, & -0.2 \le f \le 0.2 \\ 0, & \text{otherwise} \end{cases}$$

Example 36: Downsample Bandpass Signals Using a Factor of M=2

1. A real valued sequence $x[n]$ has a DTFT given by

$$|X(e^{j\omega})| = \begin{cases} 40|f|, & f_1 - \frac{1}{20} \leq |f| \leq f_1 + \frac{1}{20} \\ 0, & \text{otherwise} \end{cases}$$

where $f_1 = \frac{1}{4}$. Sketch the DTFT of the sequence obtained by downsampling $x[n]$ by 2. Is there any aliasing?

2. A complex valued sequence $x[n]$ has a DTFT given by

$$|X(e^{j\omega})| = \begin{cases} 40f, & f_1 - \frac{1}{20} \leq f \leq f_1 + \frac{1}{20} \\ 0, & \text{otherwise} \end{cases}$$

where $f_1 = \frac{1}{4}$. Sketch the DTFT of the sequence obtained by downsampling $x[n]$ by 2. Is there any aliasing?

Example 37: Downsample Bandpass Signals Using a Factor of M=4

1. A real valued sequence $x[n]$ has a DTFT given by

$$|X(e^{j\omega})| = \begin{cases} 40|f|, & f_1 - \frac{1}{20} \leq |f| \leq f_1 + \frac{1}{20} \\ 0, & \text{otherwise} \end{cases}$$

   where $f_1 = \frac{1}{4}$. Sketch the DTFT of the sequence obtained by downsampling $x[n]$ by 4. Is there any aliasing?

2. A complex valued sequence $x[n]$ has a DTFT given by

$$|X(e^{j\omega})| = \begin{cases} 40f, & f_1 - \frac{1}{20} \leq f \leq f_1 + \frac{1}{20} \\ 0, & \text{otherwise} \end{cases}$$

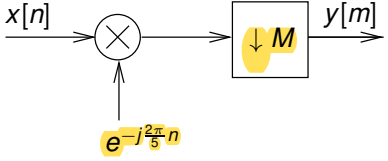
   where $f_1 = \frac{1}{4}$. Sketch the DTFT of the sequence obtained by downsampling $x[n]$ by 4. Is there any aliasing?

Example 38: Downsample Bandpass Signal to Baseband

- A complex valued sequence $x[n]$ has a DTFT given by

$$|X(e^{j\omega})| = \begin{cases} 40f, & f_1 - \frac{1}{20} \leq f \leq f_1 + \frac{1}{20} \\ 0, & \text{otherwise} \end{cases}$$

where $f_1 = \frac{1}{5}$.
This sequence is processed using the following system to produce $y[m]$ (m is used here, since $y[m]$ is at a different sampling rate than $x[n]$)



Is it possible to select a value for M, such that a system that consist of only a factor of M downsampler produces the same output $y[m]$? If so, what is the value of M?