MUS421 Lecture 4
FIR Digital Filter Design

Julius O. Smith III (jos@ccrma.stanford.edu)
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University
Stanford, California 94305

June 27, 2020

Outline

- Ideal Lowpass Filter
- Optimal Least Squares in Time Domain
- Window Method
- Case Study: FIR Hilbert-Transform Design
- Optimal Methods

FIR Digital Filter Design

In a previous lecture, we looked at many windows, and examined their properties. Today, we are going to see how these windows can be used to design Finite Impulse Response (FIR) digital filters.

FFT processors implement long FIR filters more efficiently than any other method (using Overlap-Add).

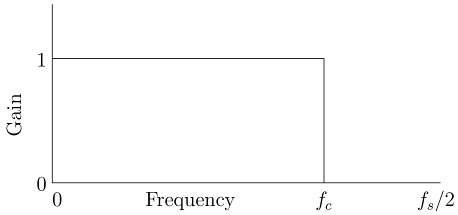
We need flexible ways to design all kinds of FIR filters for use in FFT processors.

See the *FIR Filter Demos*[1] by java@falstad.com

[1] https://www.falstad.com/dfilter/

Ideal Lowpass Filter

Amplitude Response:



- Gain $= 1$ for $f < f_c$
- Gain $= 0$ for $f > f_c$
- $f_c =$ "cut-off frequency" (Hz)
- $f_s =$ "sampling frequency" (Hz)
- Signals and filter assumed *real*

Impulse Response of Ideal LPF

$$\begin{aligned} h_{\text{ideal}}(n) &\triangleq \text{DTFT}_n^{-1}\left(H_{\text{ideal}}\right) \\ &\triangleq \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} e^{j\omega n} d\omega \\ &= \frac{\sin(\omega_c n)}{\pi n} \\ &= 2f_c \text{sinc}(2f_c n), \qquad n \in \mathbb{Z} \end{aligned}$$

Problems:

- Infinitely long
- Non-causal
- Cannot be shifted to make it causal
- Cannot be implemented in practice

Conclusion:

- We must accept some compromise(s) in the design of any practical lowpass filter.
- Managing such trade-offs is the topic of *digital filter design*.

Optimal (but Poor) Least-Squares Impulse Response Design

Let

- $h(n)$ = ideal filter impulse response.
- $\hat{h}(n)$ = length L causal FIR filter (to be designed).

Sum of squared errors:

$$J_2(\hat{h}) \triangleq \sum_{n=-\infty}^{\infty} \left|h(n) - \hat{h}(n)\right|^2 = \sum_{n=0}^{L-1} \left|h(n) - \hat{h}(n)\right|^2 + c_2$$

where $c_2 \triangleq \sum_{n=-\infty}^{-1} \left|h(n)\right|^2 + \sum_{n=L}^{\infty} \left|h(n)\right|^2$ does not depend on $\hat{h}$. Note that $J_2(\hat{h}) \geq c_2$.

Result: *The error is minimized (in the least-squares sense) by simply matching the first L terms in the desired impulse response.*

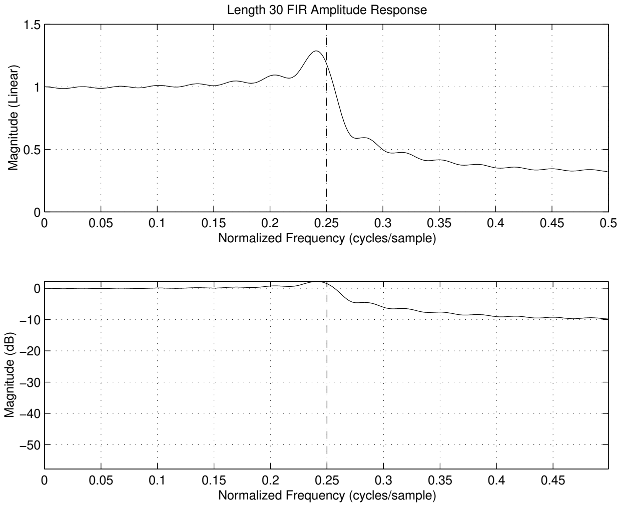
Optimal least-squares FIR filter:

$$\hat{h}(n) \triangleq \begin{cases} h(n), & 0 \leq n \leq L-1 \\ 0, & \text{otherwise} \end{cases}$$

Also optimal under any L_p norm with any error weighting:

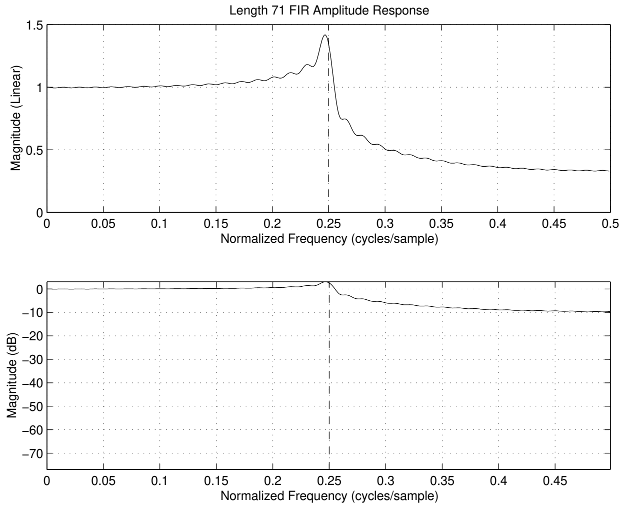
$$J_p(\hat{h}) = \sum_{n=0}^{L-1} w(n) \left|h(n) - \hat{h}(n)\right|^p + c_p \geq c_p$$

Length $L = 30$ Least-Squares LPF Design



- Desired cut-off frequency is $f_c = 1/4$
- Stopband attenuation only around 10 dB
- Passband gain has a “peak” near cut-off
- Filter is *quite poor* for audio use
- $\hat{H} = H * \text{asinc}_L$

Length $L = 71$ Least-Squares LPF Design



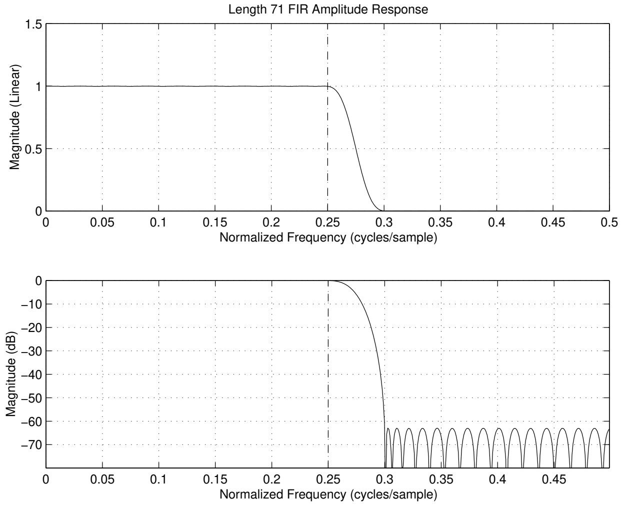
- Stopband attenuation *still* only around 10 dB
- Corner-frequency peak is *higher*, though narrower
- "Gibb's phenomenon" in the frequency domain

What we're doing wrong:

- Desired filter specification asks for too much (e.g, an infinite roll-off rate).
- Time-domain-least-squares is a poor choice of error criterion for audio work.

Length $L = 71$ Optimal Chebyshev LPF Design

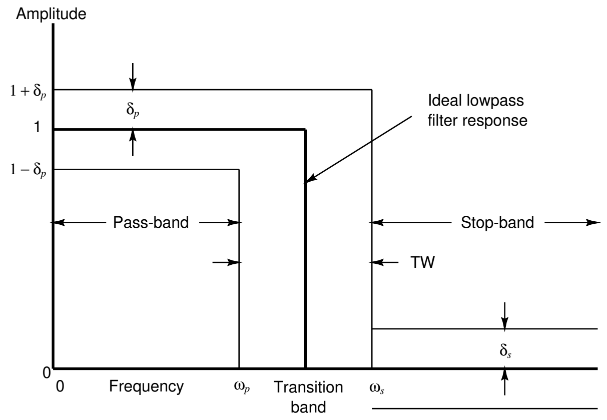
```
firpm(70,[0 0.5 0.6 1],[1 1 0 0]);
```



- Stopband attenuation better than 60 dB
- No corner-frequency peak (Gibbs overshoot)
- *Transition region* from passband to stopband *critical*
- Error is "equiripple" in both stopband (visible) and passband (not visible)
- Impulse response is slightly "impulsive" at the endpoints (not shown).

Lowpass Filter Specifications

Lowpass Filter Specifications



- δ_s : stopband ripple ($\leq 0.001 = -60$ dB typical)
- δ_p : passband ripple (≤ 0.1 dB typical)
- ω_s: stopband edge frequency
- ω_p: passband edge frequency
- TW: transition width $= \omega_s - \omega_p$
- SBA: stop-band attenuation $= -20\log(\delta_s)$

Ideal Lowpass Filter

The *ideal* lowpass filter is defined by the following specifications:

- $\omega_s = \omega_p = \omega_c \overset{\Delta}{=} 2\pi f_c \Rightarrow$ TW $= 0$
- $\delta_p = \delta_s = 0 \Rightarrow$ SBA $= \infty$

Example Ripple Calculations in Matlab

Let's first consider the passband ripple spec, ± 0.1 dB. Converting that to linear ripple amplitude gives, in Matlab,

```
format long;
dp=10^(0.1/20)-1
dp =
   0.01157945425990
```

Let's check it:

```
>> 20*log10(1+dp)
ans =
   0.10000000000000
>> 20*log10(1-dp)
ans =
  -0.10116471483635
```

Ok, close enough. Now let's set the stopband ripple to 1/10 times the passband ripple and see where we are:

```
>> ds=dp/10;
>> 20*log10(ds)
ans =
 -58.72623816882052
```

So, that's about 60 dB stop-band rejection, which is not too bad.

Setting the stopband ripple to 1/100 times the passband ripple adds another 20 dB of rejection:

```
>> ds=dp/100;
>> 20*log10(ds)
ans =
 -78.72623816882052
```

which is close to the "high fidelity" zone of 80dB SBA

- In FIR filter-design functions such as `firpm`, the *weighting* for each band is proportional to *one over the ripple amplitude* in that band. (We saw previously that the ripple amplitude is $\delta/W(\omega_k)$ where δ is minimized.)
- Thus, for a unity-gain lowpass-filter, a weighting of 1 passband and 10 in the stopband yields close to 60 dB of stopband attenuation, as derived above.
- The function `firpmord` in Matlab finds the order needed to achieve a given set of (more user friendly) specifications.

Frequency Sampling Method for FIR Filter Design

The *frequency-sampling* method for FIR filter design is perhaps the simplest and most direct technique imaginable when a desired frequency response has been specified.

- It consists simply of uniformly *sampling* the desired frequency response, and performing the inverse DFT to obtain the corresponding (finite) impulse response.
- The resulting frequency response is usually not optimal between samples
- When the desired frequency-response is *undersampled* (typical) the impulse response is *time aliased*
- It is important to compare the originally desired frequency response to the FFT of the zero-padded designed impulse response (simulated DTFT)

Window Method for FIR Filter Design

- In practical FFT processors, we are limited to a *finite duration* impulse response (FIR filters).
- An "obvious" method for FIR filter design is to simply *window* the ideal impulse response, just like we window ideal sinusoids in spectrum analysis:

  $$\boxed{\hat{h}_w(n) \stackrel{\Delta}{=} w(n) \cdot h_{\text{ideal}}(n)}$$

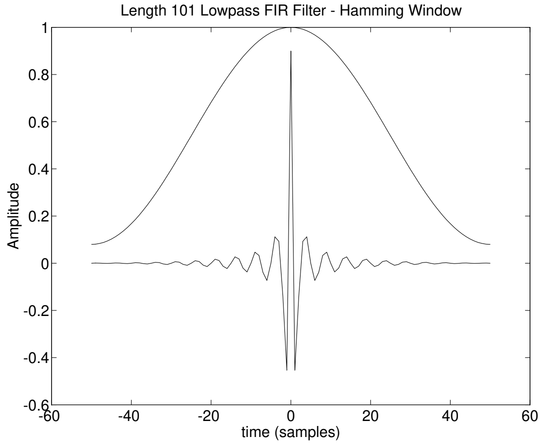
  where w is a *zero centered* window of length M (odd)
- We saw that the rectangular window is optimal in the least-squares sense, but a poor choice for typical audio filter design. What about other windows?
- Windowing is *multiplication in the time domain* $\leftrightarrow$ *convolution in the frequency domain*.
- Thus, in the window method, the *ideal frequency response is convolved with the window transform:*

  $$\boxed{\hat{H}_w(\omega) = (W * H_{\text{ideal}})(\omega)}$$

  The convolution 'smears' the response, introducing deviations in both the pass-band and stop-band.

Example

A typical windowed ideal lowpass filter response is depicted in the following diagram:



- At this point in the design, we have a non-causal (zero centered) filter.
- In order to implement this in a real time situation, we need to shift the filter by an amount equal to half its length.

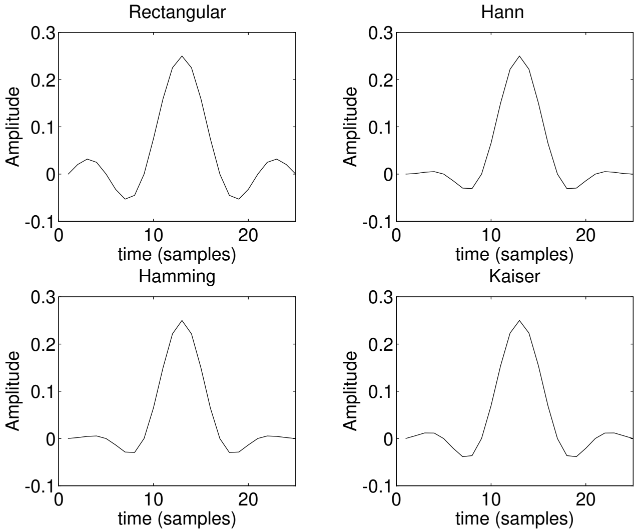
$$\hat{h}_{\textsf{causal}}(n) = \hat{h}\left(n - \frac{M-1}{2}\right)$$

- This shift in the time domain results in a *linear phase term* in the frequency domain:

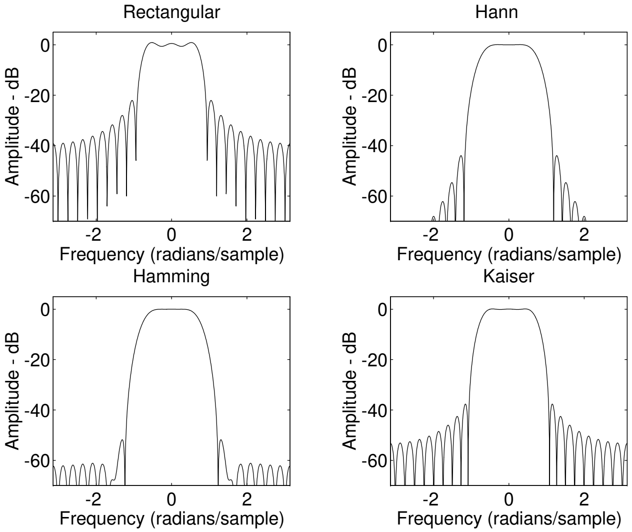
$$\hat{H}_{\textsf{causal}}(\omega) = e^{-j\omega\frac{M-1}{2}}\hat{H}(\omega)$$

Examples

- length 25 FIR filter
- $\omega_c = \pi/4$
- $h_{\text{ideal}}(n) = \frac{\sin(\omega_c n)}{\pi n} \qquad n \in \mathbb{Z}$
- We will examine several windows:

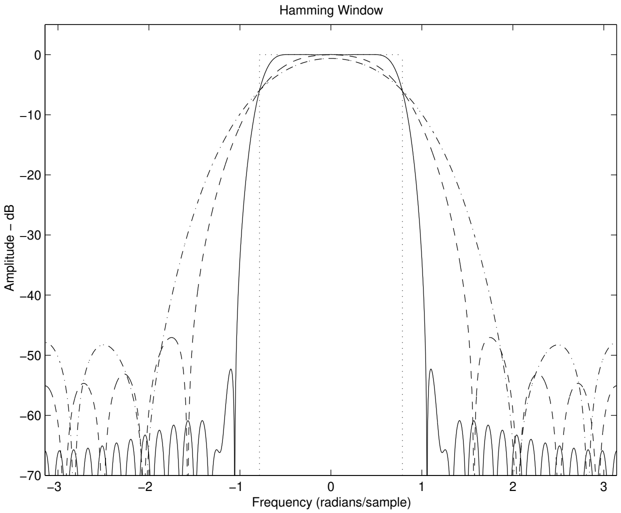


Amplitude Response of Examples, $M = 25$



- If the pass band width is set to zero, we obtain the respective *window transforms* (asinc, 3 asincs, asinch→asinc)
- Otherwise we see the convolution of the window transform with the ideal rectangular lowpass frequency response $\Rightarrow$
  - more stopband attenuation
  - wider passband (desired passband extended by transition-band on each side that is one main-lobe wide)

Effect of Changing Filter Length, $M = 11, 15, 41$



Window Method in the Frequency Domain

$$(1)\quad \hat{H}(\omega) = (W * H)(\omega) = \int_{-\pi}^{\pi} W(\nu)H(\omega-\nu)\,d\nu$$

$$= \boxed{\langle W, \textsc{Shift}_\omega\{\textsc{Flip}(\overline{H})\}\rangle}$$

For the *ideal lowpass filter:*

- Main-lobe is widened by convolution with rectangle to become the *pass band*
- Side-lobes are convolved with the rectangle to form the *stop band* (adjacent-sidelobe partial cancellation occurs, resulting SBA<sidel-lobe-level, except for dc-pass where SBA=SLL)

$$(2)\quad \hat{H}(\omega) = (H * W)(\omega) = \int_{-\pi}^{\pi} H(\nu)W(\omega-\nu)\,d\nu$$

$$= \boxed{\langle H, \textsc{Shift}_\omega\{\textsc{Flip}(\overline{W})\}\rangle}$$

For the *ideal lowpass filter:*

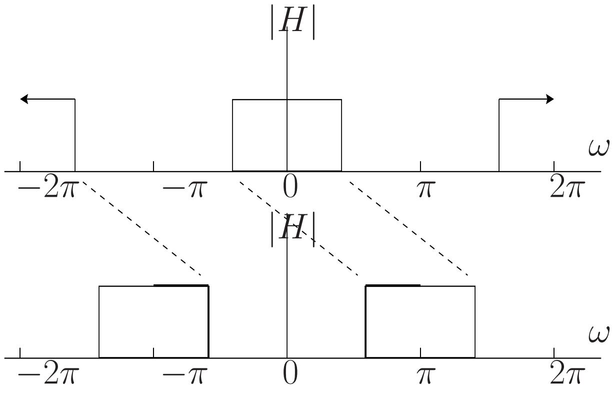
- Passband is widened by convolution with main lobe to give a *transition band*
- Transition band is clearly one main-lobe-width wide
- Stop-band made from sliding width-$2\omega_c$ integral over the window side-lobes

Other Types of Filters

Ideal highpass, bandpass, and bandstop filters can all be considered as modifications of an ideal lowpass filter.

Highpass Filter from Lowpass

A **highpass filter** can be constructed by shifting a lowpass filter by an amount equal to half the sampling rate.



This is equivalent to a modulation by $e^{-j\pi n}$ in the time domain. Hence,

$$\begin{aligned} H_{\textsf{hp}}(\omega) &= \textsc{Shift}_{\pi}\{H_{\textsf{lp}}(\omega)\} \;=\; H_{\textsf{lp}}(\omega-\pi) \quad (T=1) \\ \leftrightarrow \quad h_{\textsf{hp}}(n) &= h_{\textsf{lp}}(n)e^{-j\pi n} \\ &= h_{\textsf{lp}}(n)(-1)^n \end{aligned}$$

Summarizing:

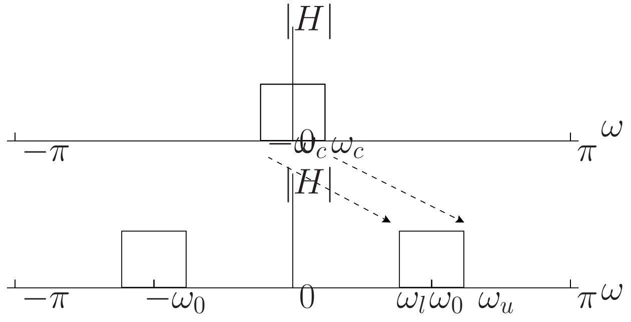
- To design a high-pass filter, first design a finite length, causal lowpass filter
  - Use a lowpass cut-off frequency equal to $\pi-$(highpass cut-off)
  - Design the lowpass filter as in the previous section
- Modulate the impulse reponse by $(-1)^n$ to exchange dc and $f_s/2$

Alternatively, if the passband of the prototype lowpass filter $\hat{H}(\omega)$ is *very flat* or *equal-ripple* about unity gain, a highpass filter can be designed by simply subtracting from 1:

$$\begin{aligned} H_{\textsf{hp}}(\omega) &\overset{\Delta}{=} 1 - H_{\textsf{lp}}(\omega) \\ &\leftrightarrow \delta(n) - h_{lp}(n) \end{aligned}$$

Bandpass Filter from Lowpass

A bandpass can be derived by designing the appropriate lowpass filter (with cut-off frequency equal to half the width of the bandpass), and then *modulating* it to the desired center frequency ω_c. In the frequency domain, this means shifting two copies of the lowpass prototype, one to the left by $-\omega_c$, and the other to the right by ω_c. The two shifted copies are added together to give a bandpass filter having a real impulse response.



To design a bandpass filter:

- Design a lowpass prototype filter of the desired width (cut-off $\omega_c = \omega_u - \omega_0 = (\omega_u - \omega_l)/2$)
- Shift to the left and right by ω_0 and sum: $h_{\textsf{bp}}(n) = 2\cos(n\omega_0)h_{\textsf{lp}}(n)$

Band-Stop Filters

A bandstop filter can be constructed by first designing a bandpass with similar specifications. As for highpass design, we have two cases (π-rotation and subtraction from 1 in the frequency domain).

"FIR Paint"
(FIR Filter-Design App)

- Start with basic FOSS "paint program"
- Drawing canvas includes axes for drawing a desired amplitude response
- Provide menu/buttons for choosing FIR length M and window type (Hamming, Blackman, etc.)
- Mouse cursor becomes the chosen *window transform*
- When "drawing", drop a mouse-cursor image each "FFT bin"
- Instead of overwriting with each mouse-cursor-image, *sum* it into the canvas (replacing any previous version for that one bin)
- "What you see is what you get" - really!
- Should of course display on dB scale or better
- Set as graphical EQ for the computer's sound output?

Summary of the Window Method

Basic Idea:

$$\hat{h} \stackrel{\Delta}{=} w \cdot h \quad \longleftrightarrow \quad \hat{H} \; = \; W * H \quad \approx \quad \delta * H \; = \; H$$

For the Ideal Lowpass Filter:

- Start with the ideal impulse response (*e.g.*, sinc function):
  - $h(t) = \sin(\pi t/T)/(\pi t/T)$
  - Infinite duration
  - Non-causal
- *Window* the infinite duration sinc to get a finite-duration (FIR) filter:
  - Window *length* determines transition width (≈ main lobe width)
  - Window *type* determines the sidelobe level (SLL)
  - Rectangular window yields the optimal filter in the unweighted least squares sense (not usually great)
- Shift the noncausal (zero centered) filter to get a causal (linear-phase) filter
  - The linear-phase slope equals half the filter length

- Other types of filters (highpass, bandpass, bandstop) are designed by first designing a low pass, and applying transformations

Important Points (Window Method)

- Easy to design
- Can design extremely long FIR filters without numerical problems
- Not usually optimum in any sense
- See `fir1` for FIR lowpass/highpass/bandpass/bandstop by the window method in Octave and/or the Matlab Signal Processing Tool Box
- See `fir2` for more general FIR filter design by the window method in Octave and/or the MSPTB

Remarks (Window-Method)

- Transition width TW ($\approx$ main lobe width) determined by window length M and window type:
  - As window gets longer, its main lobe narrows
  - Narrower main lobe $\Rightarrow$ narrower transition band
  - Tighter specs demand longer filters
  - Length $M \propto K/\text{TW}$ in general, where $K = 1$ for rectangular, 2 for generalized Hamming, 3 for Blackman, etc.
- Pass-band and stop-band ripple are determined by window side lobes
  - We do not have individual control over passband and stopband ripple (limitation of window method)
  - Ripple determined by window used
  - Ripple normally largest around transition band

Hilbert Transform Filter Design Example: A Case Study in FIR Filter Design

Design Example:

- Ideal Single-Sideband Filter (Hilbert transformer)
- Window Method using Kaiser Window
- Optimal Chebyshev using Remez Exchange

Problem Statement

- Design a "negative-frequency filter" which converts a real signal into its complex "analytic signal" counterpart by filtering out negative frequencies
- Equivalently, design a "single sideband" (SSB) filter which outputs a single-sideband signal in response to a complex-conjugate pair of sidebands (a Hermitian spectrum)
- We could also call it a "positive-frequency-pass" filter
- The *imaginary part* of such a filter is called a *Hilbert transform filter*
- The ideal Hilbert transform is an *allpass filter* that *delays positive-frequency components* by 90 degrees, and *advances negative-frequency components* by 90 degrees

Hilbert Transform

$$
\begin{aligned}
y(t) &\stackrel{\Delta}{=} (h_i * x)(t) && \text{(Hilbert transform of } x\text{)} \\
h_i(t) &\stackrel{\Delta}{=} \frac{1}{\pi t} && \text{(Hilbert transform “kernel”)} \\
H_i(\omega) &\stackrel{\Delta}{=} \begin{cases} -j, & \omega > 0 \\ j, & \omega < 0 \\ 0, & \omega = 0 \end{cases} && \text{(Hilbert frequency response)} \\
x_a(t) &\stackrel{\Delta}{=} x(t) + jy(t) && \text{(Analytic signal from } x\text{)} \\
&= \frac{1}{\pi}\int_0^\infty X(\omega)e^{j\omega t}d\omega && \text{(Note: Lower limit usually } -\infty\text{)}
\end{aligned}
$$

Proof:

$$
\begin{aligned}
X_a(\omega) &= X(\omega) + jY(\omega) && \text{(By linearity of Fourier transform)} \\
&= X_+ + X_- && \text{(Apply frequency response)} \\
&\quad +j[-jX_+ + jX_-] \\
&= 2X_+(\omega)
\end{aligned}
$$

Ideal Bandlimited Impulse Response

Since we are working with sampled data, the ideal single-side-band filter impulse response is

$$h(n) = \text{IDTFT}_n(u) \triangleq \frac{1}{2\pi}\int_{-\pi}^{\pi} u(\omega)e^{j\omega n}d\omega$$

where $u(\omega)$ is a unit step function in the frequency domain:

$$u(\omega) \triangleq \begin{cases} 1, & \omega \geq 0 \\ 0, & \omega < 0 \end{cases}$$

We can evaluate the IDTFT directly as follows:

$$\begin{aligned} h(n) &= \frac{1}{2\pi}\int_{0}^{\pi} e^{j\omega n}d\omega = \frac{1}{2\pi jn}e^{j\omega n}\Big|_{0}^{\pi} = \frac{e^{j\pi n}-1}{2\pi jn} \\ &= \frac{(-1)^n - 1}{2\pi jn} = \begin{cases} 0, & n \text{ even}, n \neq 0 \\ j\dfrac{1}{\pi n}, & n \text{ odd} \end{cases} \end{aligned}$$

For $n = 0$, going back to the original integral gives

$$h(0) = \frac{1}{2\pi}\int_{0}^{\pi} e^{j\omega\cdot 0}d\omega = \frac{1}{2}.$$

Thus, the real part of $h(n)$ is $\delta(n)/2$, and the imaginary part is $1/(\pi n)$ for *odd* n and *zero otherwise* (as a result of bandlimiting). There is no aliasing.

Ideal Bandlimited Impulse Response, Cont.

We defined the *bandlimited* single-side-band filter by direct calculation:

$$\begin{aligned} h(n) &= \text{IDTFT}_n(u) \triangleq \frac{1}{2\pi}\int_{0}^{\pi} e^{j\omega n}d\omega \\ &= \begin{cases} 1/2, & n = 0 \\ 0, & n \text{ even} \\ \dfrac{j}{\pi n}, & n \text{ odd} \end{cases} \end{aligned}$$

- An impulsive real-part is expected (passing the input real-part unchanged, only scaled)
- An imaginary part $j/(\pi n)$ would be the result of sampling $j/(\pi t)$ with $f_s = 1$, but ...
- The zeros for even n are surprising, but that's the spectrum of a square pulse train (both bandlimiting and *periodic extension*)

We also need a *transition band:*

- We'll "draw" an estimate in the frequency domain, take a (large) inverse dft, and window that
- The *transition width* will be at least the *main-lobe width* of the window transform

Kaiser Window

Kaiser window in causal, linear-phase form:

```
w = kaiser(M,beta); % M=length=257, beta=8
```

Zero-pad by a factor of 8 and convert to zero-centered form (N = FFT size = 2048):

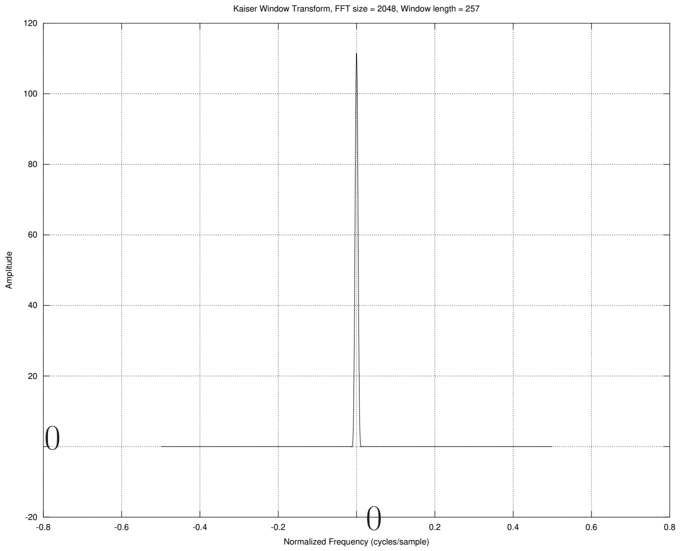
```
wzp = [w((M+1)/2:M),zeros(1,N-M),w(1:(M-1)/2)];
```

Kaiser window transform:

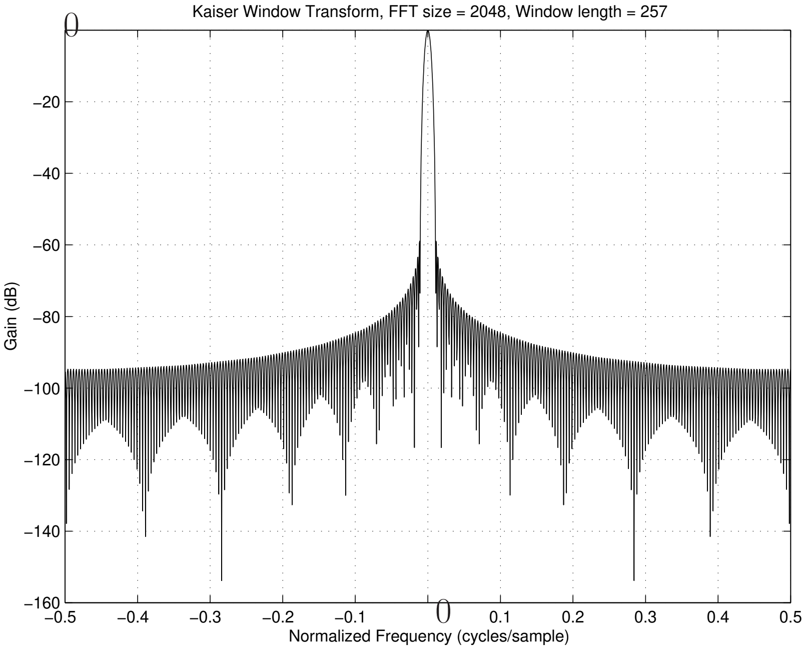
```
W = fft(wzp);
```

Note: `W = fft(w,N)` would set up `w` in *causal* form in `W` instead of *zero-phase* form

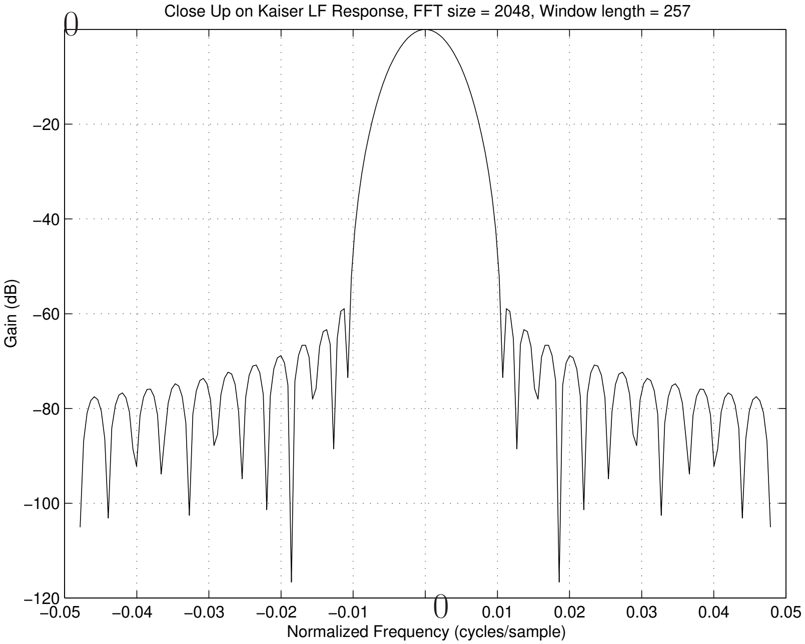
Kaiser Window Transform (Real, Linear Scale)



Kaiser Window Transform (Decibel Scale)



Kaiser window transform, close-up on main lobe



Oversimplified Window Method

Sample ideal Hilbert-transform kernel $h(t) = 1/\pi t$ to get

$$\hat{h}_i(n) \triangleq \frac{1}{\pi n T} \qquad \text{(\textit{Sampled} Hilbert transform kernel)}$$

$$\hat{h}_w(n) \triangleq w(n)\hat{h}_i(n) \qquad \text{(\textit{Windowed} ideal impulse response)}$$

$$\hat{H}_w(\omega) = (W * \hat{H})(n) \qquad \text{(\textit{Smoothed} ideal frequency response)}$$

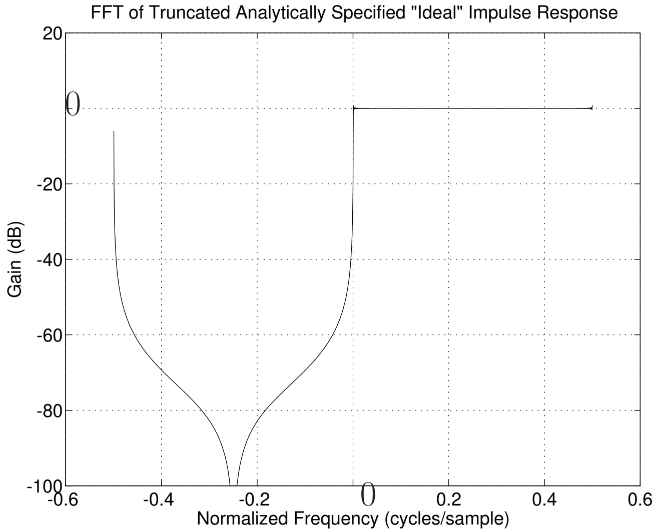
Design Parameters:

```
fs = 22050;     % sampling rate (Hz)
T = 1/fs;       % sampling period (sec)
M = 257;        % FIR filter length = window length
N = 8*(M-1);    % for interpolated spectral displays
beta = 8;       % beta for Kaiser window
```

Filter Design:

```
% Choose our sampled time axis to avoid time zero
% to avoid a division by zero in hr below:
n = [-N/2+0.5:N/2-0.5];    % Time axis (avoid t=0)
hi = T ./ (pi*n*T);        % Sampled Hilbert kernel
hr = sin(pi*n) ./ (pi*n);  % 1/2 sample delay filter
h = (hr + j*hi)/2;         % Sampled ideal final filter
plot(f,fftshift(max(-100,20*log10(abs(fft(h)))))); grid;
...
```

FFT of Truncated(2048) Ideal Impulse Response

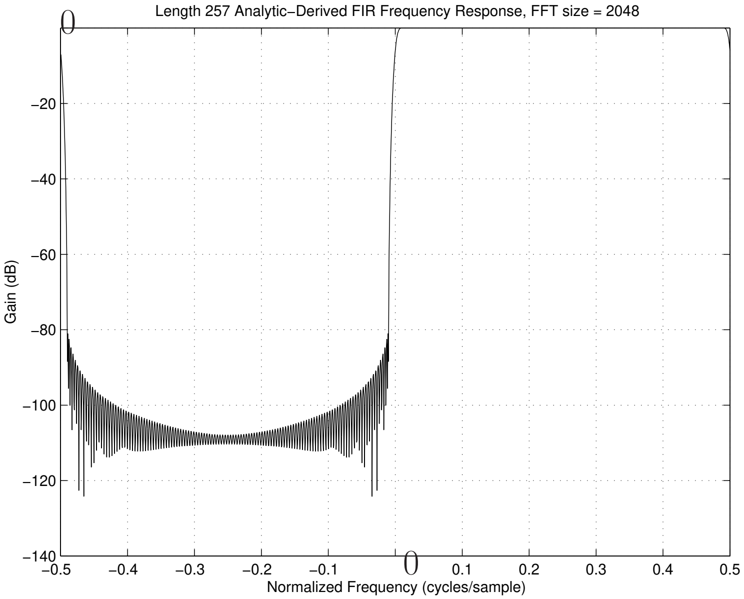


```
h = [h(N/2+1:N),h(1:N/2)]; % zero-centered form (almost)
hw = wzp .* h; % Apply window to ideal impulse response
Hw = fft(hw);  % Frequency response we really get

% Compute total stopband attenuation:
ierr = norm(Hw(N/2+2:N))/norm(Hw)
     = 0.0378 = 3.8 percent
```

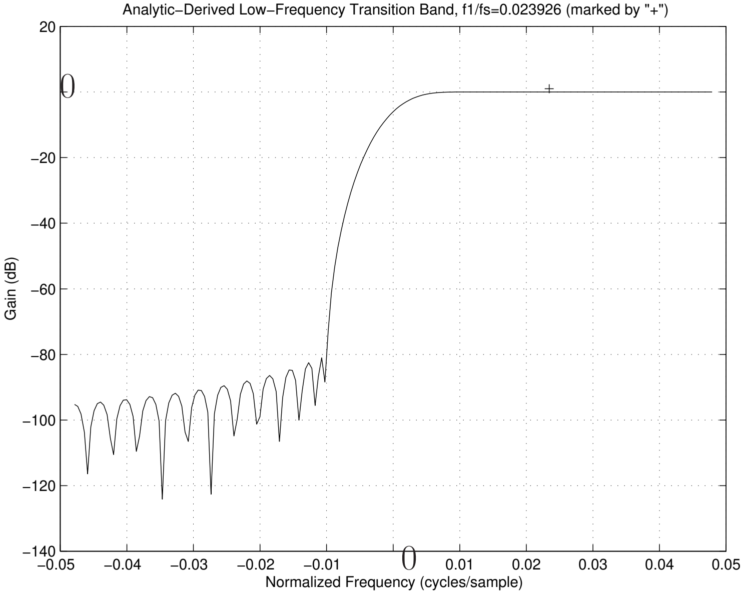
For spectral displays, rotate buffer so negative frequencies are on the left and dc is at $k = N/2$:

```
Hwp = [Hw(N/2+2:N), Hw(1:N/2+1)]; % Neg. freqs on left
Hwpn = abs(Hwp); Hwpn = Hwpn/max(Hwpn);
plot(f,20*log10(Hwpn)); grid;
...
```

FIR Frequency Response by the Window Method



Zoom in on low-frequency transition band



- Poorly positioned transition region from positive to negative frequencies.
- Need to first *bandpass-filter* the ideal impulse response.
- High-frequency transition identical, by symmetry

Window Method applied to Bandlimited Ideal Impulse Response

Further Design Parameters:

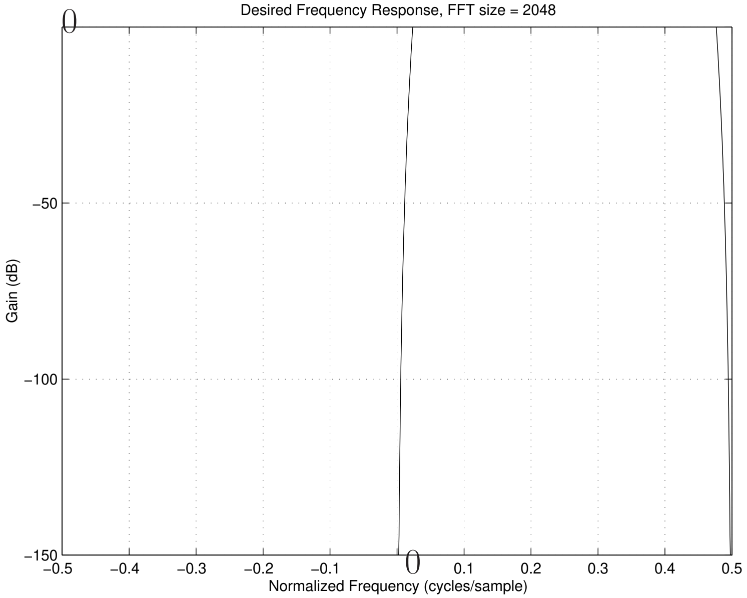
```
f1 = 530;       % lower passband limit (Hz)
N = 2^(nextpow2(8*fs/f1)) % FFT size from xition width
if N<8*M, N = 8*(M-1); end; % spectral interpolation
```

Bandlimited "Ideal" Hilbert transformer frequency response (FR), including *transition bands* Δ matched to FFT size N such that $N \gg f_s/\Delta$:

```
H = [ ([0:k1-2]/(k1-1)).^8,ones(1,k2-k1+1), ...
   ([k1-2:-1:0]/(k1-1)).^8,zeros(1,N/2-1)];
Hp = [H(N/2+2:N), H(1:N/2+1)]; // pos. freqs only
f = [-0.5 + 1/N : 1/N : 0.5]; normalized freq axis
plot(f,Hp); grid; ...
```

- Linear transition to 8th power ⇒ $6 \times 8 = 48$ dB/octave roll-off to dc instead of only 6 dB/octave (transition looks like 8 poles at dc instead of 1)
- Our time-domain window will smooth the abrupt corner at the passband edge

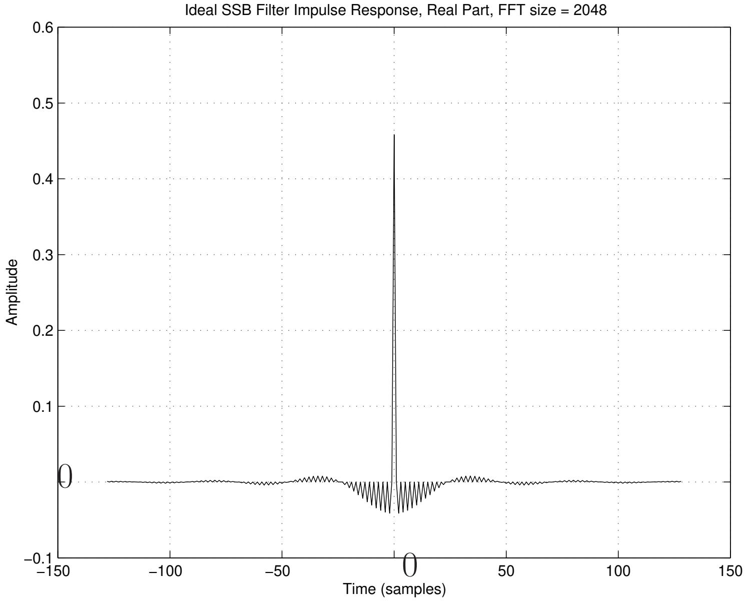
Desired Frequency Response



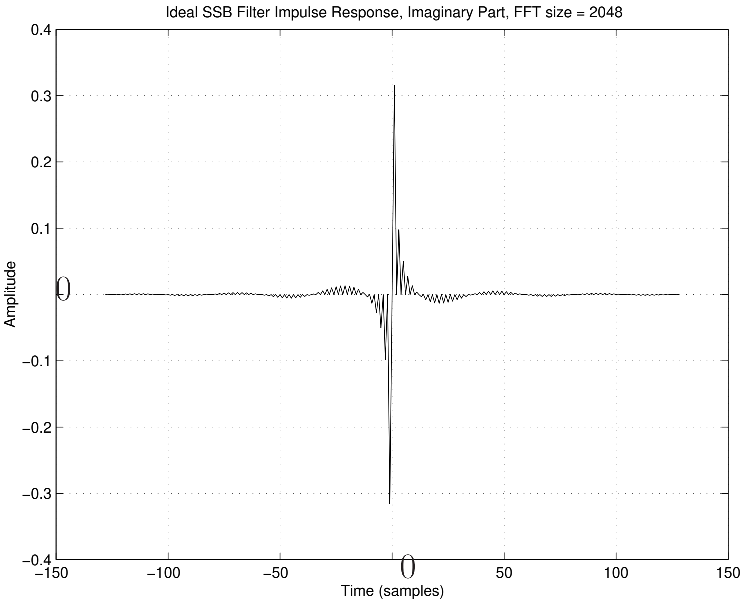
Desired Impulse Response

```
h = ifft(H); % zero-centered form

% rough estimate of time-aliasing error:
aerr = norm(h(N/2-N/32:N/2+N/32))/norm(h)

aerr =
   4.8300e-04

% for plots, prefer negative times on the left:
hp = [h(N/2+2:N), h(1:N/2+1)];
```

Real Part of Desired Impulse Response



Imaginary Part of Desired Impulse Response



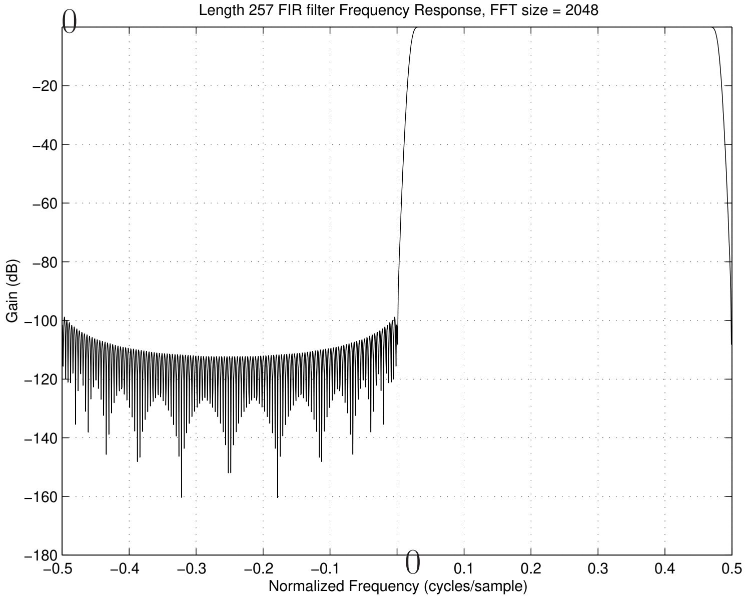
Apply window to ideal impulse response:

```
hw = wzp .* h; % Final FIR coefficients
Hw = fft(hw);  % Frequency response we'll see
```

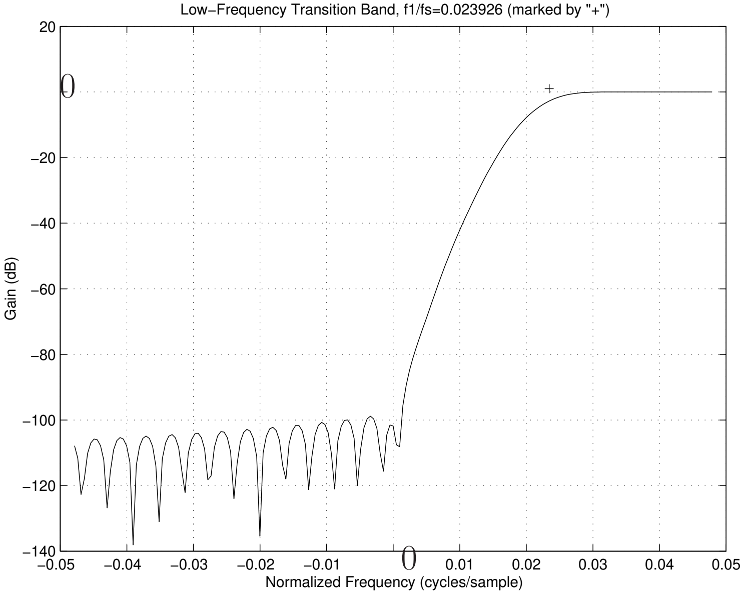
Total stopband energy:

```
ierr = norm(Hw(N/2+2:N))/norm(Hw)
disp(sprintf(['Stop-band energy is %g percent of',...
             total spectral energy'], 100*ierr));
```

Final FIR Filter Frequency Response



Zoom-in on Transition Region of Final Frequency Response



- Transition band better positioned
- ≈ 100 dB stob-band attenuation

Remez Multiple Exchange Method

Let’s now try the Remez multiple exchange algorithm (“Parks-McClellan algorithm”) for optimal Chebyshev (equiripple) FIR filter design, and compare it to our window-method design.

Step 1: Design an optimal *lowpass* filter of the desired width (takes quite a while):

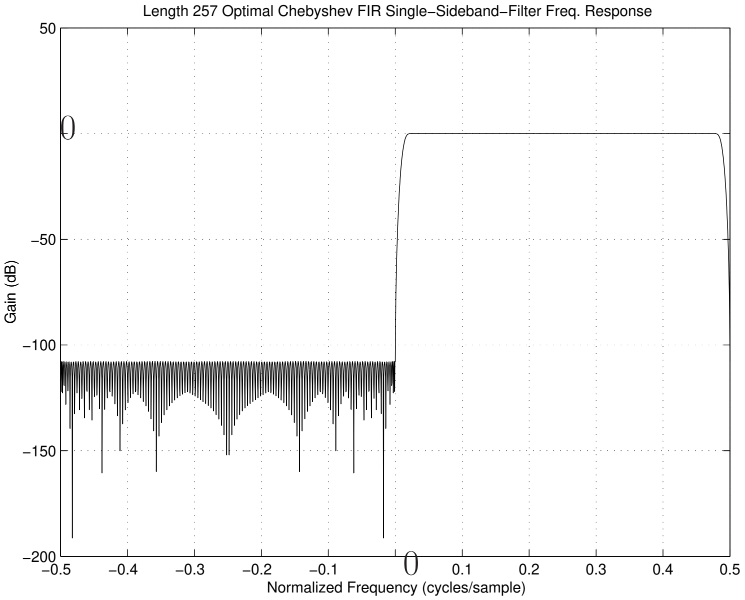
```
M = 257;      % FIR filter length
fs = 22050;   % sampling rate (Hz)
fn = fs/2;    % Nyquist limit (Hz)
f1 = 530;     % lower passband limit (Hz)
f2 = fn - f1; % upper passpand limit (see text, p. 136)
hrm = firpm(M-1, [0,(f2-fs/4)/fn,0.5,1], ...
                        [1,1,0,0], [1,10]);
```

- The weighting [1,P] says make the passband ripple P times that of stopband
- For steady-state audio spectra, passband ripple ≈ 0.1 dB is good
    - However, consider an FM signal in the passband $\Rightarrow$ passband ripple becomes AM sidebands
    - Here, we allow the passband ripple to be 10 times the stopband ripple, as a compromise

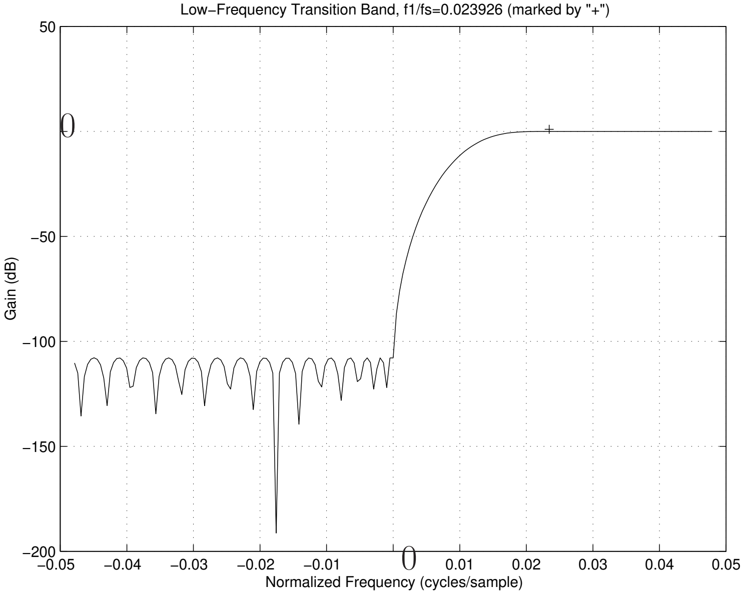
Step 2: Modulate the impulse response to make it a *single-sideband filter*. I.e., right-shift by $\pi/2$ in the frequency domain (rotate the frequency response counterclockwise along the unit circle by 90 degrees):

```
hr = hrm .* j .^ [0:M-1];
```

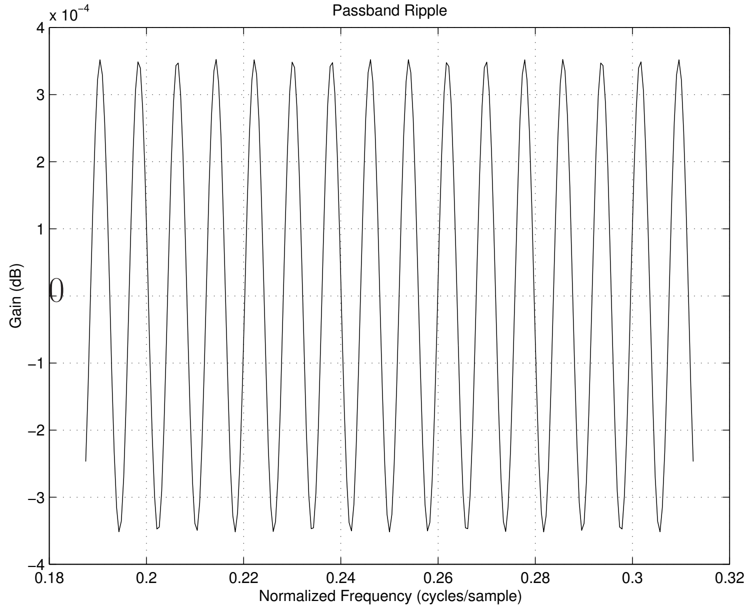
Optimal Chebyshev FIR(257) Frequency Response



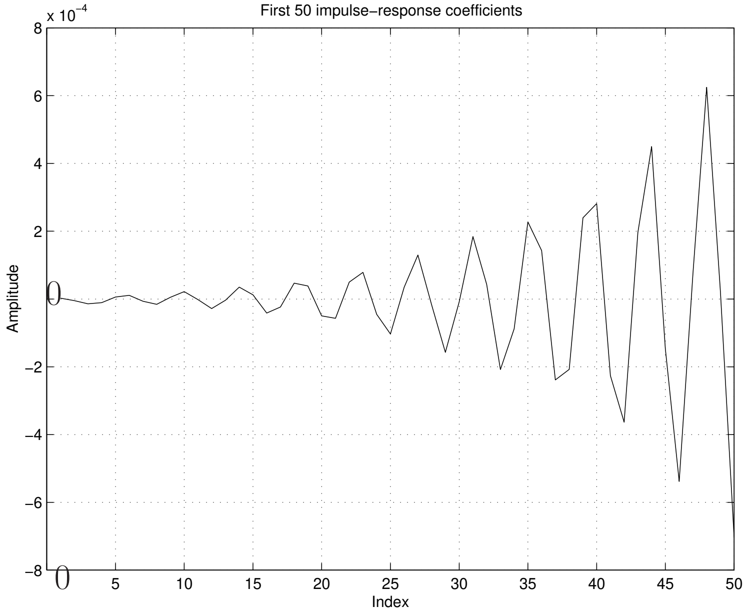
Zoom-In on Transition Band



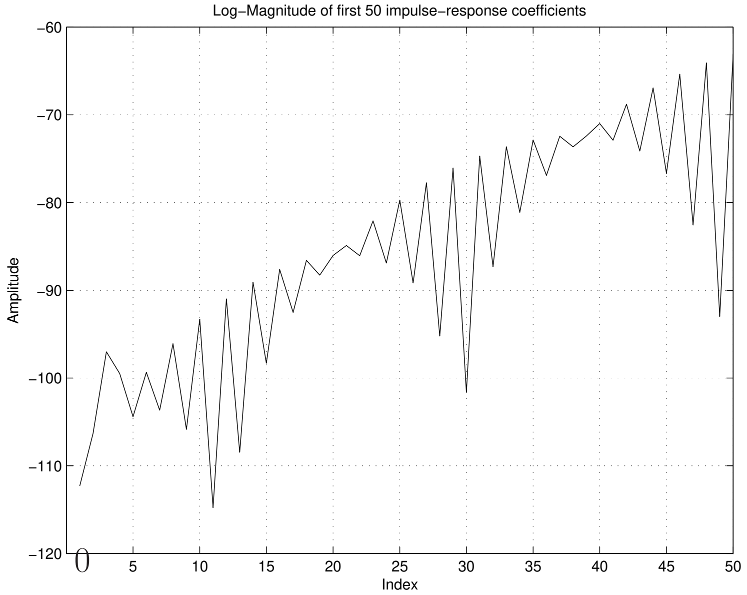
Passband Ripple



Initial Impulse Response



Log-Magnitude of Initial Impulse Response



Summary of Trade-Offs

Window-method filter:

$(+)$ Filter length can be thousands of taps

$(-)$ Stopband "droops" (rolls off) — not L^∞ optimal

$(-)$ Passband edge tuned slightly too low

Optimal Remez-Exchange filter:

$(+)$ Stopband is ideal, equiripple

$(+)$ Transition region close to *half* that of the window method

$(+)$ Pass-band is ideal, equiripple

$(+)$ Time-domain impulses from *passband* ripple are small

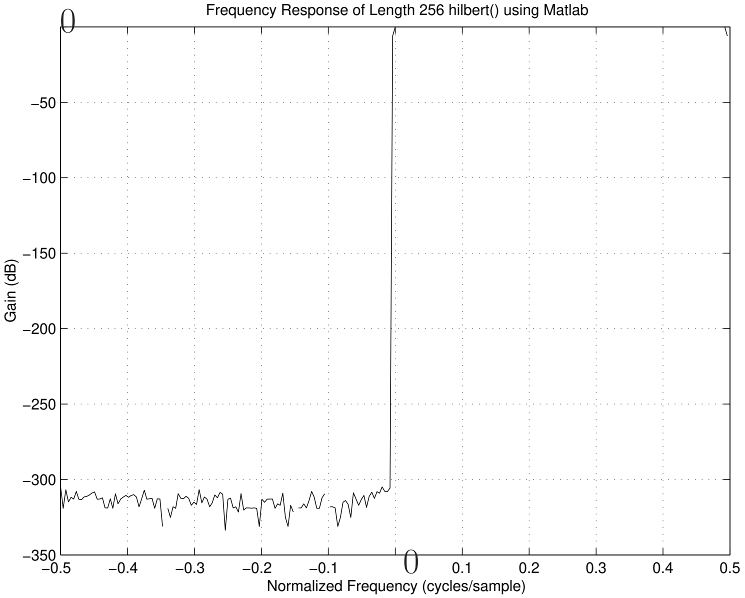
$(-)$ Design time orders of magnitude larger than that for window method

$(-)$ Fails to converge for filters much longer than 256 taps (Need to increase working precision to get longer filters)

Matlab hilbert() function

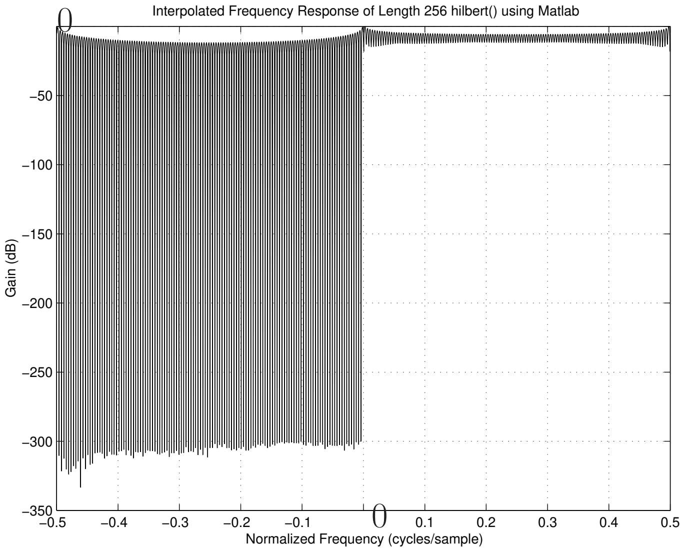
The Matlab Hilbert function returns an analytic signal given its real part.

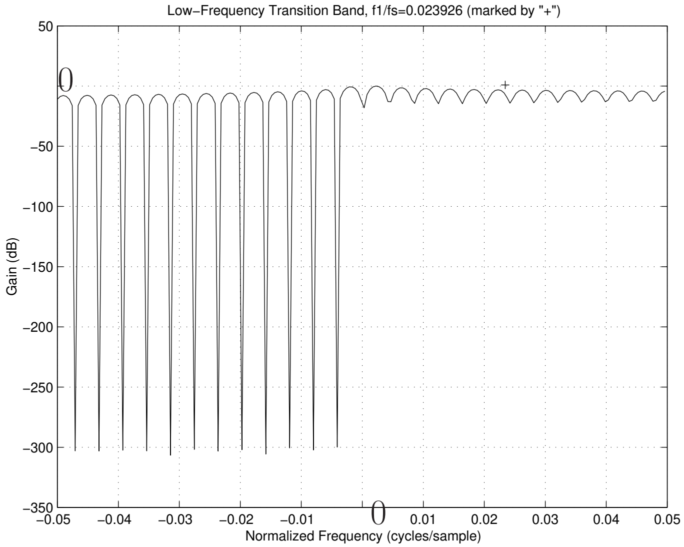
```
Nh = M-2; % This looks best
delta = [1,zeros(1,Nh)]; % zero-centered impulse
hh = hilbert(delta); % zeros negative-freq FFT bins
Hh = fft(hh); % FIR frequency response
```



Looks pretty good, but let's zero-pad the impulse response to interpolate the frequency response:

```
hhzp = [hh(Lh/2+1:Lh), zeros(1,N-Lh), hh(1:Lh/2)];
Hhzp = fft(hhzp); % Frequency response
```





Problems:

- Transition bandwidth only 2 bins wide
- Massive time aliasing
- Only correct for *periodic* signals with period exactly equal to filter length Nh. In this case, time aliasing is equivalent to overlap-add from adjacent periods.

In general, any real signal given to hilbert() must be interpreted as precisely one period of a periodic signal.

Optimal FIR Digital Filter Design

- Optimal Chebyshev FIR Design
- L_p Norm Minimization
  - Least squares problems
  - Min-Max problems
- Least Squares Optimization
  - PseudoInverse
- FIR Filter Design
  - Linear Phase
  - Complex Filter Design
- Other Applications

Books pertinent to Digital Filter Design

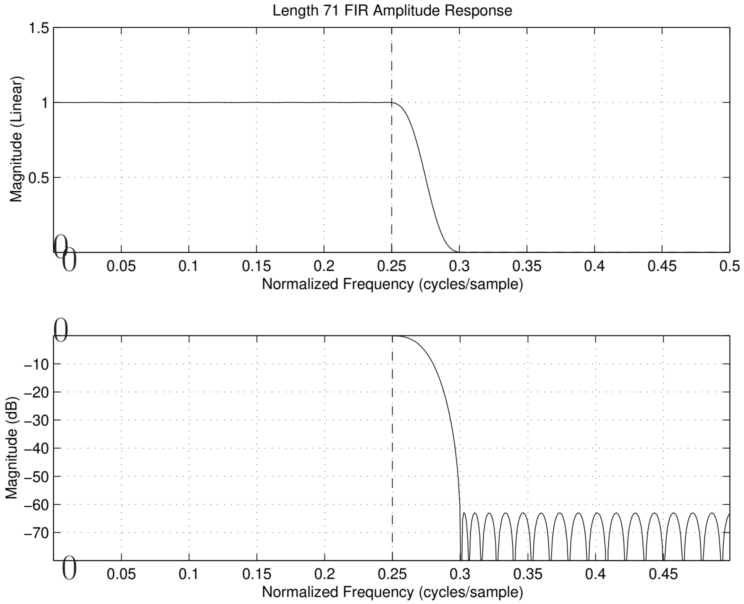
- Parks and Burrus (1987)
- Rabiner and Gold (1975)
- Oppenheim and Schafer (1975, 1999)
- Strum and Kirk (1988)
- Steiglitz (1996)
- Boyd and Vandenberghe (2004)

See the text bibliography for full citations.

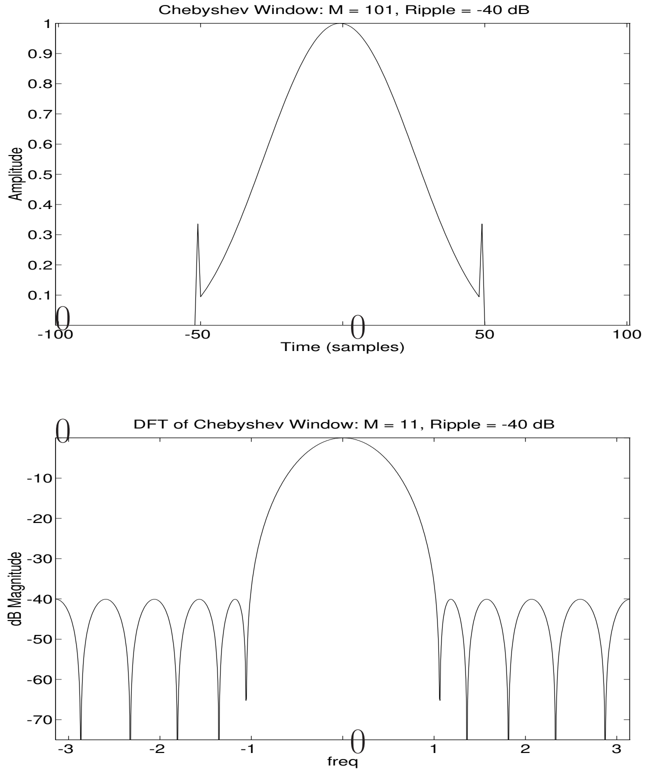
Optimal Chebyshev Filters

- Minimize the *maximum* of the weighted error
- Stopband and passband errors are *equiripple*
- Remez Multiple Exchange
  (Parks-McClellan algorithm)
- Available in Octave as `remez` and the Matlab SP Tool Box as `firpm()`, `cfirpm()`
  [formerly `remez()`, `cremez()`]

Recall Example Chebyshev FIR Lowpass



Recall Chebyshev Window and Transform



Problems with Remez Exchange

- Convergence *unlikely* for FIR filters longer than a few hundred taps or so
- Fundamentally *real* polynomial approximation on the unit circle

  $\Rightarrow$ Applies only to *linear phase* filters
- Sometimes need optimal weighted *complex* approximation
  e.g., minimum phase FIR filter design

Filter Design using Lp Norms

In terms of L_p norm minimization, the filter design problem becomes:

$$\min_h \left\| W(\omega_k) \left[H(\omega_k) - D(\omega_k) \right] \right\|_p$$

Where

- $\|\cdot\|_p$ denotes the Lp norm (defined below)
- ω_k = discrete frequency (possibly non-uniform)
- $H(\omega_k)$ = frequency response of our filter
- $D(\omega_k)$ = desired (complex) frequency response
- $W(\omega_k)$ = (optional) weighting function

This formulation applies to *both* FIR and IIR filters replacing h by the direct-form coefficients $[B, A]$. (With $A = 1$, $B = h$).

We'll look at some specific cases below

Lp Norms

The L_p norm of an N-dimensional vector x is defined as

$$\| x \|_p \stackrel{\Delta}{=} \left(\sum_{i=0}^{N-1} |x_i|^p \right)^{\frac{1}{p}}$$

Special cases

- L_1 norm

$$\| x \|_1 \stackrel{\Delta}{=} \sum_{i=0}^{N-1} |x_i|$$

  - Sum of the absolute values of the elements
  - "City block" distance

- L_2 norm

$$\| x \|_2 \stackrel{\Delta}{=} \sqrt{\sum_{i=0}^{N-1} |x_i|^2}$$

  - "Euclidean" distance
  - Minimized by "Least Squares" techniques

- L_∞-norm

$$\| x \|_\infty \stackrel{\Delta}{=} \lim_{p \to \infty} \left(\sum_{i=0}^{N-1} |x_i|^p \right)^{\frac{1}{p}}$$

In the limit as $p \to \infty$, the L_p norm of x is dominated by the maximum element of x.

Least-Squares Linear-Phase FIR Filter Design

Let the FIR filter length be $L+1$ samples, with L even, and suppose we'll initially design it to be centered about the time origin ("zero-phase"). Then the frequency response is given on our frequency grid ω_k by

$$H(\omega_k) \;=\; \sum_{n=-L/2}^{L/2} h_n e^{-j\omega_k n},\; k=0,1,2,\dots,N{-}1,\; N \gg L.$$

Enforcing *even symmetry* in the impulse response, i.e., $h_n = h_{-n}$, gives a *zero-phase* FIR filter which we can later right-shift $L/2$ samples to make a causal, *linear phase* filter. In this case, the frequency response reduces to a *sum of cosines*:

$$H(\omega_k) \;=\; h_0 + 2\sum_{n=1}^{L/2} h_n \cos(\omega_k n), \quad k=0,1,2,\dots,N-1,$$

or in matrix form:

$$\begin{bmatrix} H(\omega_0) \\ H(\omega_1) \\ \vdots \\ H(\omega_{N-1}) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 2\cos(\omega_0) & \dots & 2\cos[\omega_0(L/2)] \\ 1 & 2\cos(\omega_1) & \dots & 2\cos[\omega_1(L/2)] \\ \vdots & & & \\ 1 & 2\cos(\omega_{N-1}) & \dots & 2\cos[\omega_{N-1}(L/2)] \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{L/2} \end{bmatrix}}_{x}$$

Note that Remez exchange algorithms are also based on this formulation internally, but now $L \ll N$.

Matrix Formulation: Optimal L_2 Design, Cont'd

In matrix notation, our filter design problem can be stated

$$\min_x \|Ax-b\|_2^2$$

where, for zero-phase filters,

$$A \triangleq \begin{bmatrix} 1 & 2\cos(\omega_0) & \dots & 2\cos[\omega_0(L/2)] \\ 1 & 2\cos(\omega_1) & \dots & 2\cos[\omega_1(L/2)] \\ \vdots & & & \\ 1 & 2\cos(\omega_{N-1}) & \dots & 2\cos[\omega_{N-1}(L/2)] \end{bmatrix}$$

$$x \triangleq h$$

and $b = [D(\omega_k)]$ is the desired frequency response at the specified frequencies.

- The chosen grid frequencies ω_k provide
  - effective weighting (denser points $\Rightarrow$ more weight)
  - transition bands (gaps in the grid)
- Usually there is an explicit weighting allowed, so that $\|W(\omega_k)\,[H(\omega_k)-D(\omega_k)]\|_2^2$ is minimized
- The function `firls` in Octave and the Matlab Signal Processing Tool Box implements frequency-domain weighted-least-squares FIR filter design

Least Squares Optimization

$$\hat{x} \stackrel{\Delta}{=} \arg\min_x \| Ax - b \|_2 = \arg\min_x \| Ax - b \|_2^2$$

Hence we can minimize

$$\| Ax - b \|_2^2 = (Ax - b)^T (Ax - b)$$

Expanding this, we have:

$$(Ax - b)^T (Ax - b) = (b^T - x^T A^T)(Ax - b)$$

This is quadratic in x, hence it has a *global minimum* which we can find by taking the derivative, setting it to zero, and solving for x. Doing this yields:

$$A^T Ax - A^T b = 0$$

These are the famous *normal equations* whose solution is given by:

$$\boxed{\hat{x} = \left[(A^T A)^{-1} A^T\right] b}$$

The matrix

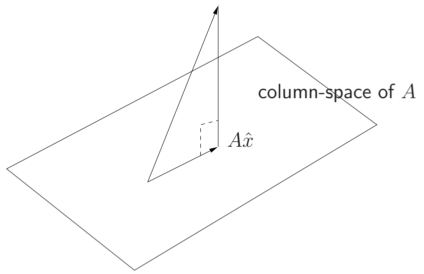
$$A^\dagger \stackrel{\Delta}{=} (A^T A)^{-1} A^T$$

is known as the *pseudo-inverse* of the matrix A.

Geometrical Interpretation of Least Squares

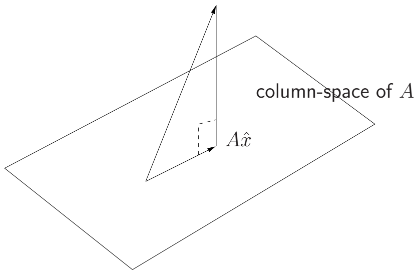
Typically, $L \ll N$, *i.e.*, the number of frequency constraints is much greater than the number of design variables (filter taps). In these cases, we have an *overdetermined* system of equations (more equations than unknowns). Therefore, we cannot generally satisfy all the equations, and we are left with minimizing some error criterion to find the "optimal compromise" solution.

In the case of least-squares approximation, we are minimizing the *Euclidean distance*, which suggests the following geometrical interpretation:



$$b = A\hat{x} + e$$

Minimize$_x \|e\|_2 = \|b - Ax\|_2$



This diagram suggests that the error vector $b - A\hat{x}$ is *orthogonal* to the column space of the matrix A, hence it must be orthogonal to each column in A.

$$A^T(b - A\hat{x}) = 0 \Rightarrow A^T A\hat{x} = A^T b$$

This is how the *orthogonality principle* can be used to derive the fact that the best least squares solution is given by

$$\hat{x} = (A^T A)^{-1} A^T b = A^\dagger b$$

Note that the pseudo-inverse $A^\dagger$ *projects* the vector b onto the column space of A.

In Matlab:

- `x = A \ b`
- `x = pinv(A) * b`

Complex FIR Filter Design

In linear-phase filter design, we assumed symmetry of our filter coefficients $[h(n) = h(-n)] \Rightarrow$

- The filter frequency response became a *sum of cosines* ("zero phase")
- The matrix A was real
- The desired magnitude response b was real
- The final zero-phase filter $\hat{x}$ could be right-shifted $L/2$ samples to get a corresponding causal linear-phase FIR filter

Now we would like to specify a *complex* frequency response. This means that:

- b is complex
- A is complex
- We still want x (our filter coefficients) to be real

If we try to use '\' or `pinv` in Matlab, we will generally get a complex result for $\hat{x}$

Summarizing our problem:

$$\min_x \| Ax - b \|_2$$

where, $A \in \mathbb{C}^{NxM}$, $b \in \mathbb{C}^{Nx1}$, and $x \in \mathbb{R}^{Mx1}$

Hence we have,

$$\min_x \| [\mathcal{R}(A) + j\mathcal{I}(A)]\, x - [\mathcal{R}(b) + j\mathcal{I}(b)] \|_2^2$$

which can be written as:

$$\min_x \| \mathcal{R}(A)x - \mathcal{R}(b) + j\, [\mathcal{I}(A)x + \mathcal{I}(b)] \, \|_2^2$$

or

$$\min_x \left\| \begin{bmatrix} \mathcal{R}(A) \\ \mathcal{I}(A) \end{bmatrix} x - \begin{bmatrix} \mathcal{R}(b) \\ \mathcal{I}(b) \end{bmatrix} \right\|_2^2$$

which is written in terms of only *real* variables.

Hence, we can use the standard least squares solvers in Matlab and end up with a *real* solution.

Related paper

"Design of Fractional Delay Filters Using Convex Optimization" (Mohonk-97, Music 421 handout):

`http://ccrma.stanford.edu/~jos/eps/mohonk97.ps.gz`

Optimal FIR Filters — Arbitrary Magnitude and Phase Specificiations

`cfirpm` (Matlab Signal Processing Toolbox) performs *complex* L_∞ FIR filter design:

- Convergence theoretically guaranteed for *arbitrary* magnitude and phase specifications versus frequency.
- Reduces to Parks-McClellan algorithm (Remez second algorithm) as a special case.
- Written by Karam and McClellan. See "Design of Optimal Digital FIR Filters with Arbitrary Magnitude and Phase Responses," by Lina J. Karam and James H. McClellan, ISCAS-96. The paper may be downloaded at

`http://www.eas.asu.edu/~karam/papers/iscas96_km.html`

MATLAB FIR Filter Design Functions

Reference:

`https://www.mathworks.com/help/signal/ug/fir-filter-design.html`

FIR Filter Design

Method	Description	Functions
Window	Apply chosen window to inverse Fourier transform of heavily interpolated desired frequency response	`fir1`, `fir2`, `kaiserord`
Least-Squares or Chebyshev Bandpass Design	Chebyshev (equal-ripple) or least-squares design on frequency bands separated by transition bands	`firls`, `firpm`, `firpmord`
Constrained Least Squares	Minimize sum of squared magnitude errors over entire frequency range subject to maximum error constraints	`fircls`, `fircls1`
Complex Chebyshev	Approximate any complex frequency response, including nonlinear phase and complex filters	`cfirpm`
Raised Cosine	Lowpass response with smooth, sinusoidal transition	`rcosdesign`

MATLAB FIR Filter Design Functions, Continued

`fir1`: Window-based finite impulse response filter design.

`b = fir1(n,Wn)` returns row vector b containing the n+1 coefficients of an order n lowpass FIR filter.

Algorithms: The window method of FIR filter design [1].

`fir2`: Frequency sampling-based finite impulse response filter design.

`b = fir2(n,f,m)` returns row vector b containing the n+1 coefficients of an order n FIR filter. The frequency-magnitude characteristics of this filter match those given by vectors f(frequency) and m(corresponding magnitude). References:[2, 3]

`firls`: Least square linear-phase FIR filter design.

`b = firls(n,f,a)` returns row vector b containing the n+1 coefficients of the order n FIR filter whose frequency-amplitude characteristics approximately match those given by vectors f and a. The output filter coefficients, or “taps,” in b obey the symmetry relation. References: [4, 5]

fircls: Constrained least square, FIR multiband filter design.

b = fircls(n,f,amp,up,lo) generates a length n+1 linear phase FIR filter b. The frequency-magnitude characteristics of this filter match those given by vectors f(frequency) and amp(amplitude).

Algorithms: uses an iterative least-squares algorithm to obtain an equiripple response. The algorithm is a multiple exchange algorithm that uses Lagrange multipliers and Kuhn-Tucker conditions on each iteration[6].

fircls1: Constrained least square, lowpass and highpass, linear phase, FIR filter design.

b = fircls1(n,wo,dp,ds) generates a lowpass FIR filter b, where n+1 is the filter length, wo is the normalized cutoff frequency in the range between 0 and 1 (where 1 corresponds to the Nyquist frequency), dp is the maximum passband deviation from 1 (passband ripple), and ds is the maximum stopband deviation from 0 (stopband ripple).

Algorithms: uses an iterative least-squares algorithm to

obtain an equiripple response. The algorithm is a multiple exchange algorithm that uses Lagrange multipliers and Kuhn-Tucker conditions on each iteration[6].

firpm: Parks-McClellan optimal FIR filter design.

b = firpm(n,f,a) returns row vector b containing the n+1 coefficients of the order n FIR filter whose frequency-amplitude characteristics match those given by vectors f and a. The filters are optimal in the sense that the maximum error between the desired frequency response and the actual frequency response is minimized. Filters designed this way exhibit an equiripple behavior in their frequency responses. Reference: [7]

cfirpm: Complex and nonlinear-phase equiripple FIR filter design.

b = cfirpm(n,f,@fresp) returns a length n+1 FIR filter with the best approximation to the desired frequency response as returned by function fresp, f is a vector of frequency band edge pairs.

Algorithms: An extended version of the Remez exchange method is implemented for the complex case[8].

References

[1] IEEE Acoustics, Speech, and Signal Processing Society. Digital Signal Processing Committee, Programs for digital signal processing, IEEE, 1979.

[2] S.K. Mitra and Y. Kuo, Digital signal processing: a computer-based approach, vol. 2, McGraw-Hill New York, 2006.

[3] L.B. Jackson, Digital filters and signal processing: with MATLAB exercises, Kluwer Academic Pub, 1996.

[4] T.W. Parks and C.S. Burrus, Digital filter design, Wiley-Interscience, 1987.

[5] A.V. Oppenheim, R.W. Schafer, J.R. Buck, et al., Discrete-time signal processing, vol. 2, Prentice hall Englewood Cliffs, NJ:, 1989.

[6] I.W. Selesnick, M. Lang, and C.S. Burrus, “Constrained least square design of fir filters without specified transition bands,” Signal Processing, IEEE Transactions on, vol. 44, no. 8, pp. 1879–1892, 1996.

[7] L.R. Rabiner, J.H. McClellan, and T.W. Parks, “Fir digital filter design techniques using weighted chebyshev approximation,” Proceedings of the IEEE, vol. 63, no. 4, pp. 595–610, 1975.

[8] L.J. Karam and J.H. McClellan, “Complex chebyshev approximation for fir filter design,” Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, vol. 42, no. 3, pp. 207–216, 1995.