MUS421/EE367B Lecture 9D: Nonuniform FFT Filter Banks

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 9, 2020

Outline

- Overview
- Octave Filter Banks
- Summing FFT Bins
- Nonuniform FFT Filter Banks[1]

[1] **Reference:** "Audio FFT Filter Banks, J.O. Smith, DAFX-2009, http://dafx09.como.polimi.it/proceedings/papers/paper_92.pdf

Overview

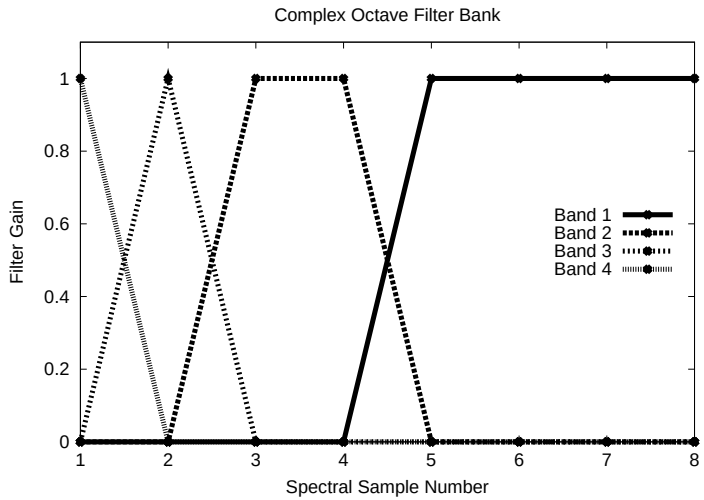
- We've seen the need for *nonuniform* spectral resolution in audio spectrograms
- The FFT gives the *same* resolution at every bin
- We can *sum* adjacent FFT bins to lower the frequency resolution
- More generally, we can *inverse FFT* adjacent FFT bins to compute a time-domain signal corresponding to a particular frequency band
- We consider the *octave filter bank* as an example

Perspective

- We are talking today about FilterBank Summation (FBS) based on the STFT
- Most STFT applications are based on the OLA interpretation (where frames overlap-add in the time domain)
- FBS application examples include the *vocoders*
- FBS-STFT mastery gives valuable insights for advanced applications of the STFT

Octave Filter Banks

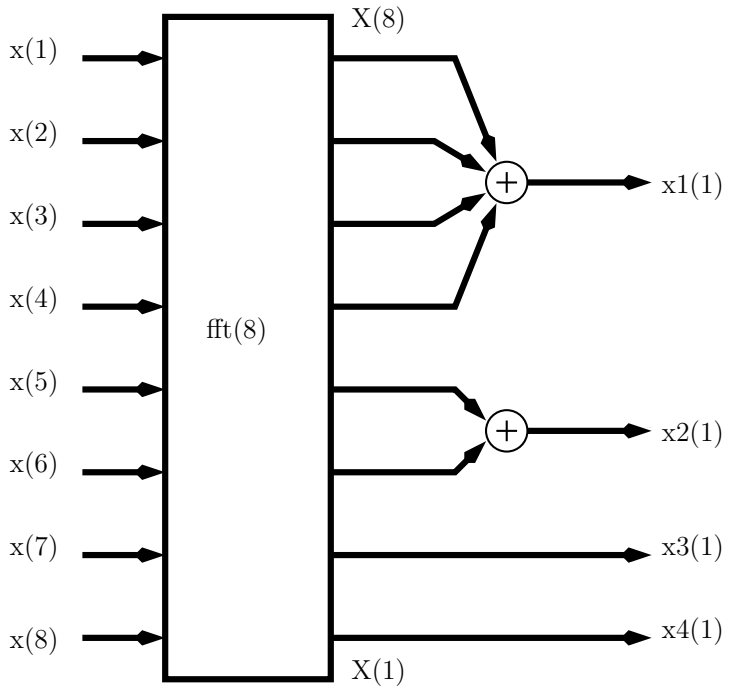
Simplified Schematic



Simple octave filter bank for *complex* signals.

- Three bands: Passbands 4,2,1 bins (samples) wide
- Leftover dc band (needed for perfect reconstruction)
- "sample number" $\stackrel{\Delta}{=}$ "bin number plus 1" (matlab)

Summing FFT Bins to get Wider Bands



- Instead of simple summing, a *weighting* can be used, corresponding to some desired *smoothing kernel*
- Summed bands can *overlap* (*e.g.*, when weighted)
- In general, we define an *overlap-add decomposition* of the *spectrum* (FBS = *dual* of OLA in time)

Summing STFT Bins

Reasoning:

- The STFT implements a *uniform* FIR filter bank
- Each FFT bin = one time-sample of filter-bank output in one filter channel
- Summing adjacent filter-bank signals sums corresponding passbands to create a wider passband
- Summing adjacent FFT bins in the STFT, therefore, synthesizes one time-sample from a *wider* passband implemented using the FFT

But note:

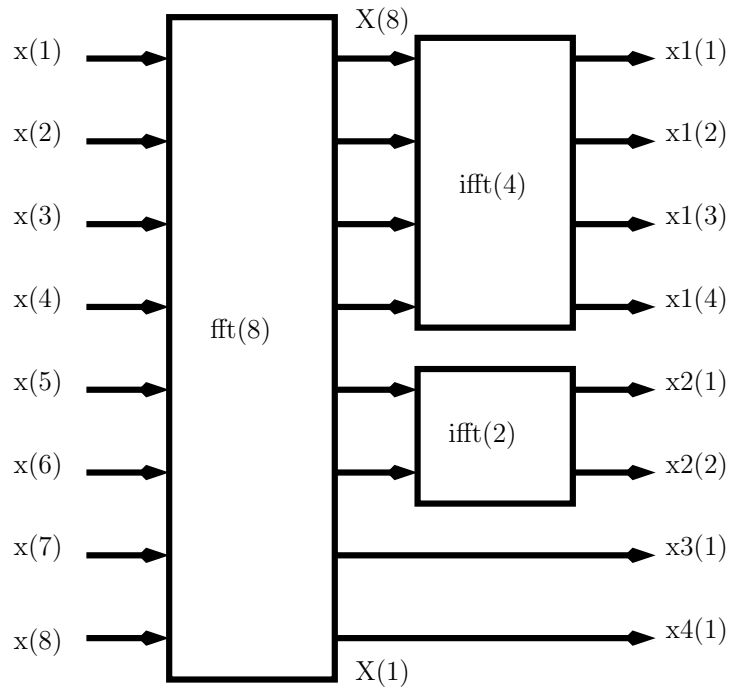
- The wider passband created by adjacent-channel summing corresponds to a *shorter impulse response* in the time domain
- The maximum STFT hop size R is therefore reduced
- This means narrower bands will be *oversampled* in the time domain, due to the smaller hop-size R

Avoiding STFT Oversampling in Time

Idea: *Don't* shorten the hop-size R, and instead compute *multiple* time-samples for the wider frequency bands:

- One FFT hop yields all needed samples in each band
- All filter-bank channels can use the same hop size R without redundancy
- If the narrowest band gets one sample per FFT hop, then a band N times as wide must produce N samples per hop

Inverse FFTs on Wider Bands



- FFT implementation of one frame of simple octave filter bank
- Successive frames *non-overlapping* (rectangular window, $R = M$)

Matlab Example

Desired frequency responses ($N = 8$):

```
H = [ ...
   0   0   0   0   1   1   1   1 ; ...
   0   0   1   1   0   0   0   0 ; ...
   0   1   0   0   0   0   0   0 ; ...
   1   0   0   0   0   0   0   0 ];
```

Let

$\texttt{X(1:N)} =$ FFT of current data frame

$\texttt{lsn(k)} =$ lowest spectral sample number for band k

$\texttt{hsn(k)} =$ highest spectral sample number for band k

(spectral sample number $\stackrel{\Delta}{=}$ bin number + 1)

For Band 1 (top row of H), `lsn(1)`=5 and `hsn(1)`=8.
The (full-rate) time-domain signal corresponding to band k is computed by

```
BandK = X(lsn(k):hsn(k));
z1 = zeros(1,lsn(k)-1);
z2 = zeros(1,N-hsn(k));
BandKzp = [z1, BandK, z2];
x(k,:) = ifft(BandKzp); % Output signal, band k
```


Critically Downsampled Band Signals

For critical sampling in band k, simply omit the spectral zero padding:

```
xd{k} = ifft(BandK);
```

where

$\texttt{xd}\{\texttt{k}\} =$ *cell array* for "downsampled" signal vectors

The length of $\texttt{xd}\{\texttt{k}\}$ is

$$\begin{aligned} N_k &= \texttt{hsn(k)} - \texttt{lsn(k)} + 1 \\ &= \texttt{number of FFT bins in band } k \\ &= \texttt{power of 2 (which we will ensure)} \end{aligned}$$

Perfect Reconstruction Property

Note that when $L_k \stackrel{\Delta}{=} N/N_k$ is an integer, we have that

$$\texttt{BandK} = \textsc{Alias}_{L_k}(\texttt{BandKzp})$$

where $\textsc{Alias}_L(X)$ denotes the aliasing of vector X by the factor L (the number of aliasing spectral partitions). In matlab we could define the alias operator by

```
function y = alias(x,L)
  Nx = length(x);
  Np = Nx/L; % aliasing-partition length
  y = zeros(Np,1);
  for i=1:L
    y = y + x(1+(i-1)*Np : i*Np)(:);
  end
```

and then have that

```
BandK == alias(BandKzp,Lk)
```

is true for each element

(Note: `reshape` and `sum` give a faster `alias` function)

Perfect Reconstruction, Time-Domain View

By the *aliasing theorem* (*downsampling theorem*), the relation

$$\texttt{BandK} = \frac{1}{L_k}\textsc{Alias}_{L_k}(\texttt{BandKzp})$$

in the frequency domain corresponds to

$$\texttt{xdk} == \texttt{x(k,1:Lk:end)}$$

in the time domain (in matlab notation)

- That is, in the time domain, `xd{k}` is obtained from `x(k,:)` by *downsampling* by the factor `Lk`
- This produces `N/Lk == Nk` samples in `xd{k}`
- Thus, for a band that is `Nk` bins wide, we obtain `Nk` time-domain samples for each STFT frame [when critically sampled, again using a rectangular window with no overlap ($R = M$)]
- (At the full rate—no downsampling—we obtain `N` samples from each channel for each frame)

Summary of Nonuniform FFT Filter Banks

We see that

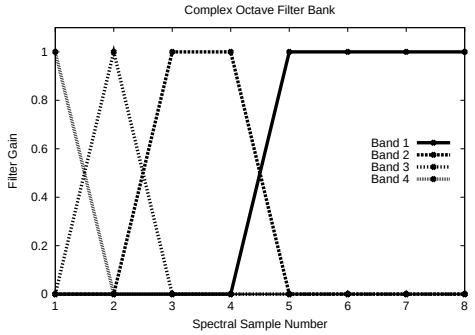
- Taking the inverse FFT of only the bins in a given filter-bank channel computes the critically downsampled signal for that channel
- Zero-padding each band out to N samples yields full-rate time-domain signals (no downsampling)
- Downsampling factors between 1 and `Lk` can be implemented by choosing a zero-padding factor for the band somewhere between 1 and `Lk` samples.
- The filter bank has the *perfect reconstruction* (PR) property, *i.e.*,

  > *The original input signal is exactly reconstructed (to within a delay and possible scale factor) from the channel signals by performing FFTs on each and reassembling the original signal spectrum for the frame*

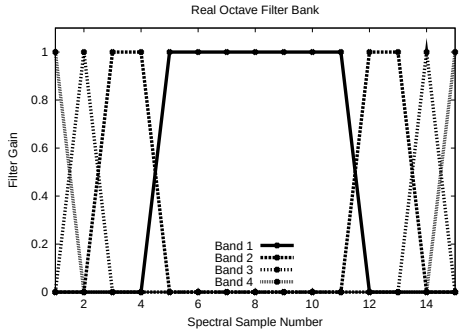
  In other words, the PR property follows immediately from the invertibility of the FFT

Real Octave Filter Banks

Recall that our simple octave filter bank is for *complex signals:*

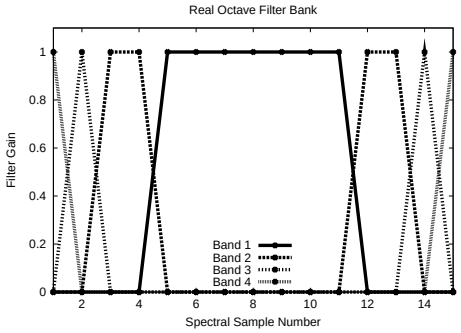


A corresponding real filter bank looks like this:



Unfortunately, the number of spectral samples is now 15—not a power of 2

Problem with Real Octave Filter Banks



- Number of spectral samples is not a power of 2
- Discarding sample at half the sampling rate (number 8 above) does not help, since that gives 14 samples
- There is no obvious way to octave-partition the spectral samples of a *real* signal while maintaining the power-of-2 condition for each band (for the Cooley-Tukey FFT).

Converting Real Signals to Complex

- Any real signal can be converted to its corresponding "analytic signal" by filtering out the negative-frequency components
- This is normally done by designing a *Hilbert transform filter*
- Such filters are large-order FIR filters, exactly like we are trying to design!
- If we design our filter bank properly, we can use *it* to eliminate the negative-frequency components!

Spectral Rotation of Real Signals

- Note that if we *rotate* the spectrum of a real signal by *half* a bin, we obtain $N/2$ positive-frequency samples and $N/2$ negative-frequency samples, with no sample at dc or at the Nyquist limit
- This is nice for audio signals because dc is inaudible and the Nyquist limit is a degenerate point of the spectrum that, for example, cannot have a phase other than 0 or π
- If N is a power of 2, then so is $N/2$, and the octave-band partitioning of the previous subsection can be applied separately to each half of the spectrum:

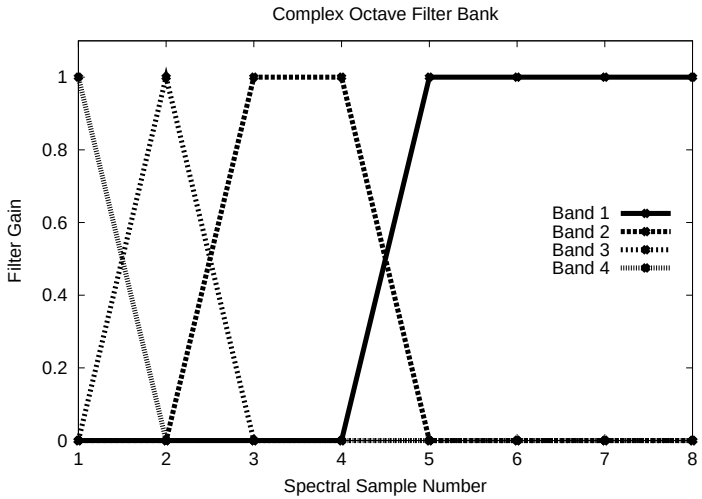
Matlab for Spectral Half-Bin Rotation

```
LN = round(log2(N)); % number of octave bands
shifter = exp(-j*pi*[0:N-1]/N); % half-bin
xs = x .* shifter; % apply spectral shift
X = fft(xs,N); % project xs onto rotated basis
XP = X(1:N/2); % positive-frequency components
XN = X(N:-1:N/2+1); % neg.-frequency components
YP = dcells(XP); % partition to octave bands
YN = dcells(XN); % ditto for neg. frequencies
YPe = dcells2spec(YP); % unpack "dyadic cells"
YNe = dcells2spec(YN); % unpack neg. freqs
YNeflr = fliplr(YNe);  % undo former flip
ys = ifft([YPe,YNeflr],N,2);
y = real(ones(LN,1)*conj(shifter) .* ys);
% = octave filter-bank signals (real)
yr = sum(y); % filter-bank sum (should equal x)

% Alternate reconstruction:
yP = ifft([YPe,zeros(size(YPe))],N,2);
yN = ifft([zeros(size(YNeflr)),YNeflr],N,2);
ys = yP + yN;
```

Improving the Channel Filters

Consider again the simple octave filter bank:



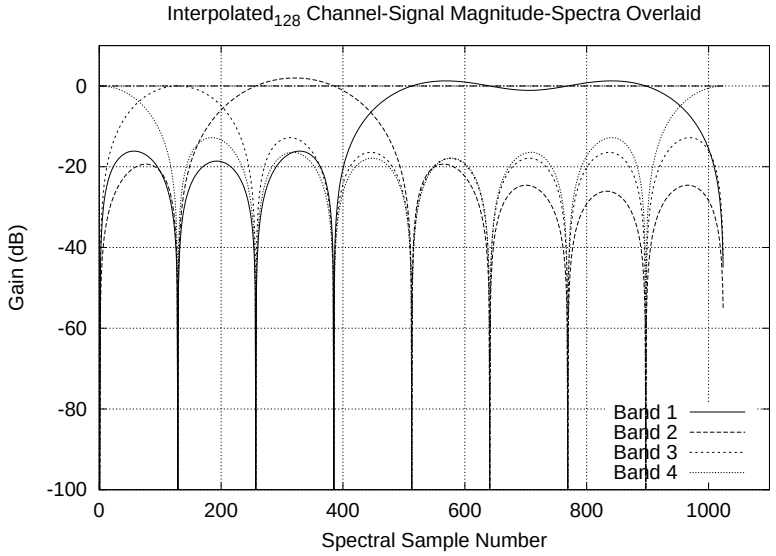
Simple octave filter bank for *complex* signals.

To test our FFT implementation, let's

1. feed it an impulse,
2. FFT each output signal (with zero padding), and
3. overlay the spectral magnitudes

This shows the true amplitude response of the filter bank.

FFT Octave Filter Bank Frequency Response



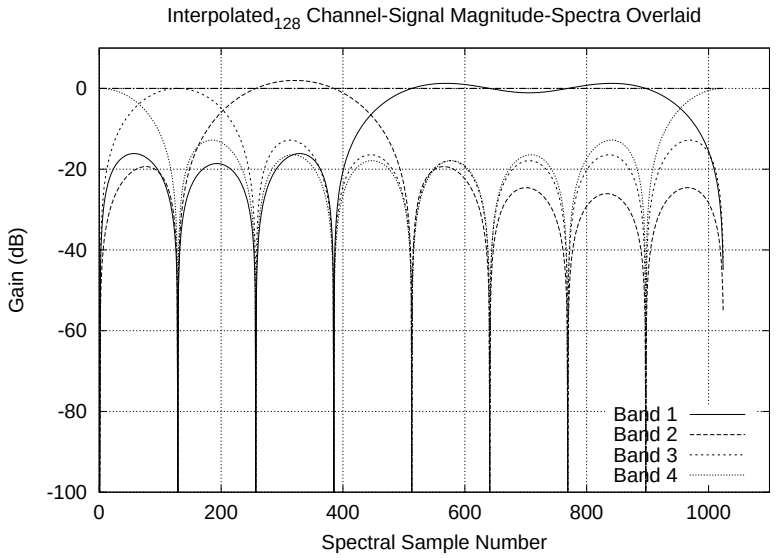
We see that our filter-bank has poor channel isolation:

- By using a non-overlapping spectral partition (recall that we simply zero-padded each spectral band out to the full IFFT size), we implicitly assumed a transition width of *one bin*
- Only the length `N` rectangular window has a one-bin transition from "pass-bin" to stopband (side lobes)

- However, the rectangular window gives poor stopband performance (see figure)

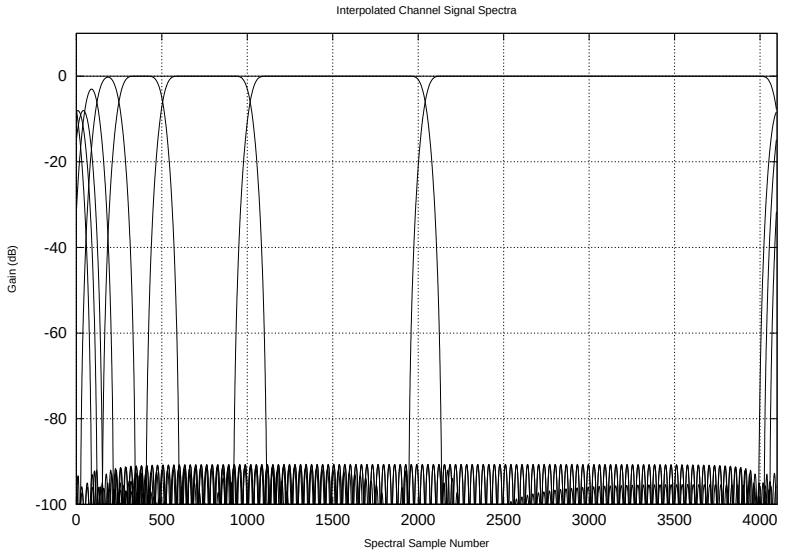
Furthermore, if we *downsample* these filter bands, then

- Each channel signal will contain much *aliasing*.
- However, we know this FFT filter bank is PR, so all such aliasing must *cancel out* in the reconstruction (filter-bank sum)



Improving the FFT Octave Filter Bank

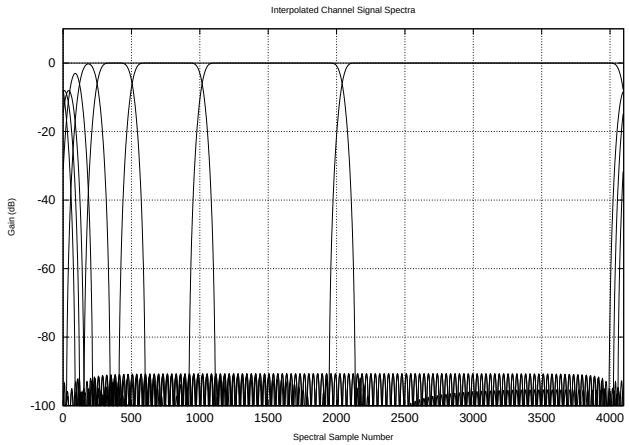
Improve filter-bank via *spectral* overlap-add:



Channel-magnitude-response overlay, octave filter-bank

- Each zero-padded "rectangularly windowed" spectral band is replaced by a proper "spectral windowing operation" in the frequency domain (Chebyshev)
- Disjoint spectral partition replaced by an *overlap-add decomposition* (OLAD) in the frequency domain

FFT Octave Filter-Bank Band Filters



Spectral window = frequency response of FIR filter designed by the *window method*:

- Dolph-Chebyshev window
- Length $N_h = 127$
- Side-lobe level $= -80$ dB
- Zero phase
- Length 256 FFT (analysis window can still be rectangular with $R = M$)
- matlab: `chebwin(127,80)`

Matlab for Improved Octave FFT Filter Bank

Implementation of band `k`:

```
BandK2 = Hk .* X;
x(k,:) = ifft(BandK2); % full rate
BandK2a = alias(BandK2,Nk);
xd{k} = ifft(BandK2a); % critically sampled
```

Channel frequency response `Hk`:

```
W = fft(w,N); % where w = chebwin(127,80)
Hideal = [z1,ones(1,Nk),z2];
Hk = cconvr(W,Hideal); % circ. conv.
```

where `z1` and `z2` are zero padding vectors.

The band filters `Hk` can be said to have been designed by the *window method* for FIR filter design.

`cconvr(W,H)` = *circular convolution* of two real vectors having the same length:

```
function [Y] = cconvr(W,X)
  wc=fft(W);  xc=fft(X);
  yc = wc .* xc;
  Y = real(ifft(yc));
```

Properties of Improved FFT Filter Bank

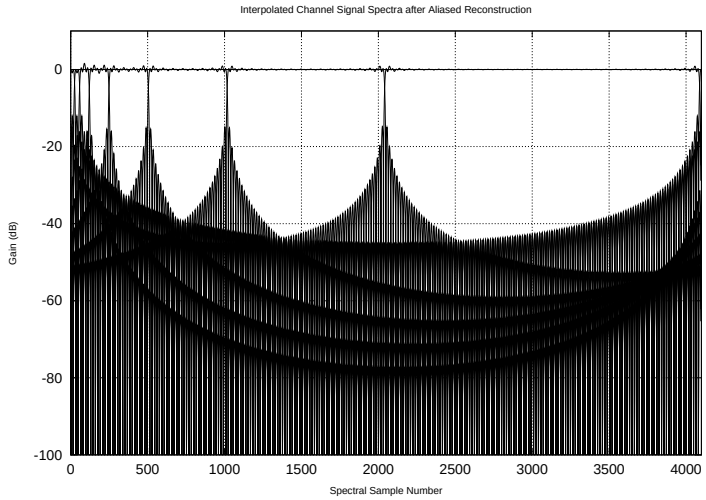
Perfect reconstruction *no longer holds*, since

```
BandK2a = alias(Hk .* X, Nk);
```

is no longer exactly invertible

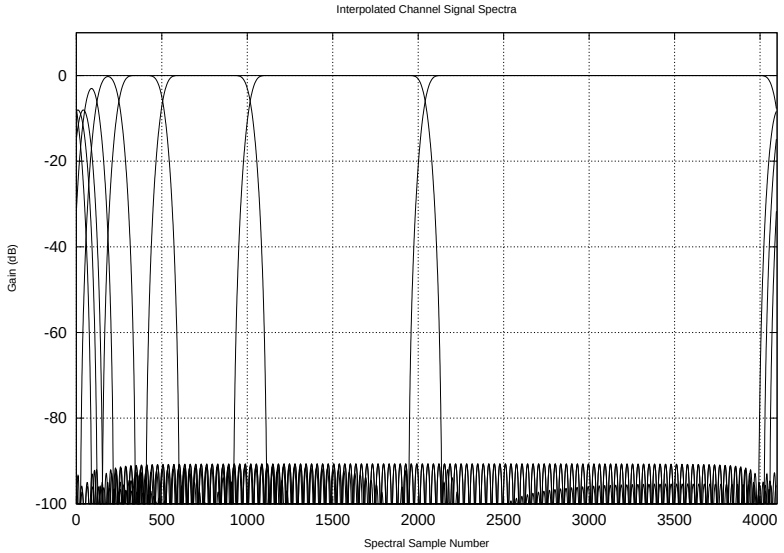
- When the FFT window is a length `N` rectangular window, then `alias(Hk .* X, Nk) == BandK`, as defined above, and there is no aliasing after all
- More precisely, the aliased spectral samples all happen to be zeros of the window transform (which is an aliased sinc function)
- These zeros depend on the window-length being `N` (no zero-padding), and on the window-type being rectangular ("no window")
- We may approach perfect reconstruction *arbitrarily closely* by only aliasing *stopband intervals* onto the passband, and by increasing the stopband attenuation of `Hk` as desired
- In contrast to the PR case, we do not rely on aliasing cancellation, which is valuable when the channel signals are to be modified

Octave Filter Bank Using Chebyshev Window



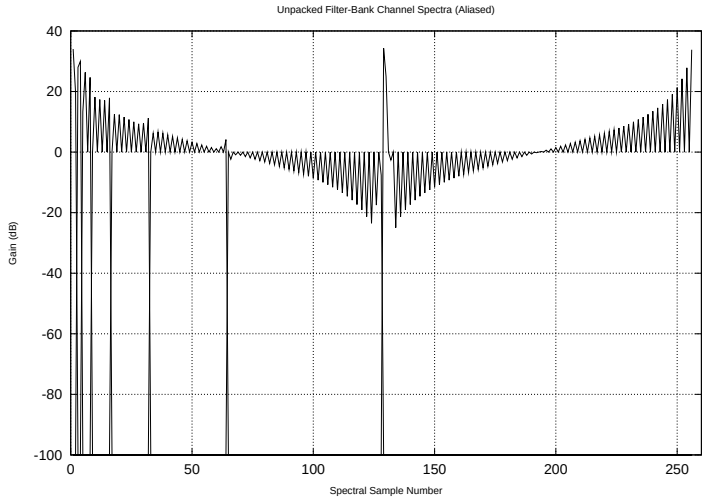
- Same as before but with *critical downsampling* of each channel signal
- All this stopband error *cancels out* in the filter-bank sum because the input signal is an *impulse* (reconstruction remains exact)

Aliasing on Downsampling



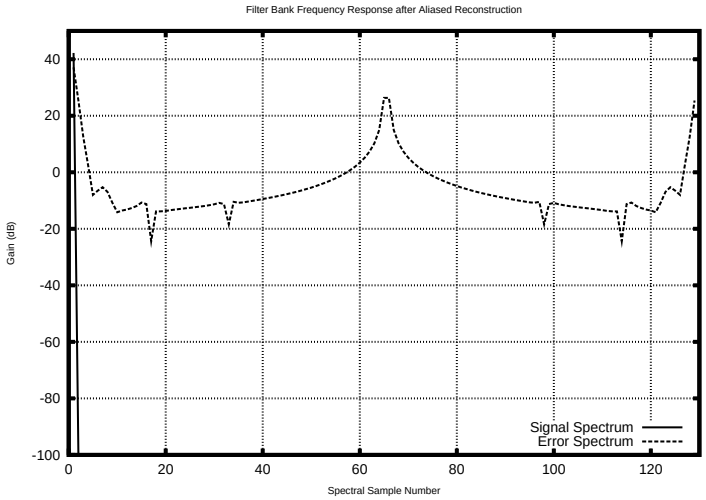
- While this filter bank gives good stopband rejection, there is still a significant amount of *aliasing* when the bands are critically sampled
- This happens because the *transition bands* are aliased about their midpoints
- This can be seen in the above amplitude-response overlay by noting that aliasing "folding frequencies" lie at the crossover point between each pair of bands

Step Response



- Aliased spectral signal bands (prior to inverse STFT) for a *step* signal input (same filter bank)
- This type of plot looks ideal for an impulse input signal because the spectrum is constant (so the aliased bands are also constant)
- Note the large slice of dc energy that has aliased from near the sampling rate to near half the sampling rate in the top octave band

Step Response Signal and Error Spectra

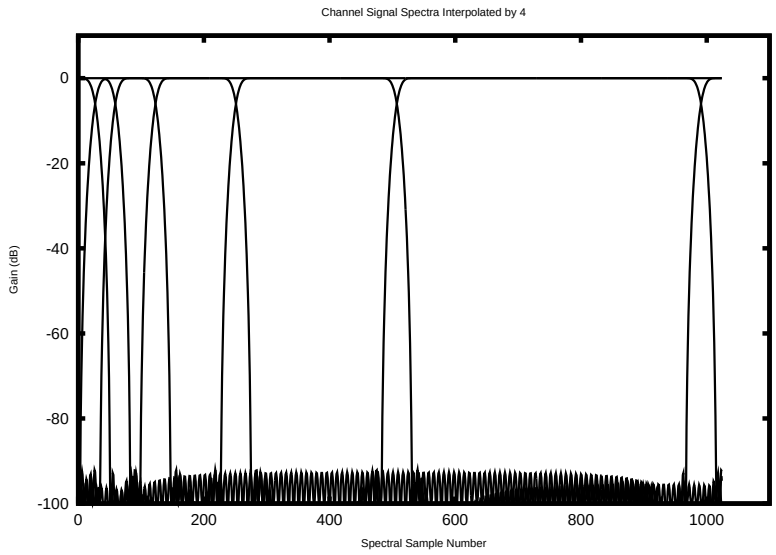


- Signal spectrum (an impulse, since the time signal is a step) and error spectrum
- Note the large error near half the sampling rate, as expected.
- In this case (step input), the aliasing causes significant error in the reconstruction

Restricting Aliasing to Stopbands

- Define *overlapping bands* such that each band encompasses the transition bands on either side
- Channel Bandwidth = filter pass-band plus both transition bands
- Include more stop-band to get to the next power of 2:
  - May fill with a continuation of the enclosed band's stopband response (or some tapering of it, or zero)
  - Since we assume stopband energy is negligible, the difference should be inconsequential
- The desired bands may overlap each other by any amount, and may have any desired shape
- The minimum channel bandwidth is two transition bands plus one bin (one FFT bin)
- For the Dolph-Chebyshev window, the transition bandwidth is known in closed form
- In summary, passbands of arbitrary width are embedded in overlapping IFFT bands that are a power-of-2 wide

Example



Channel-frequency-response overlay for an octave filter bank designed using a length 127 Dolph-Chebyshev window (80 dB SBA) and length 256 FFT size

Real Filter Banks

Since the passbands may be any width and the encompassing IFFT bands may overlap by any amount, they do not have to "pack" conveniently as power-of-two blocks

- As a result of this flexibility, the frequency-rotation trick is no longer needed for real filter banks
- Instead, allocate any desired bands between dc and half the sampling rate, and let conjugate-symmetry dictate the rest
- In addition to a left-over "dc-Nyquist" band, there is a "Nyquist-limit" band (a typically negligible band at half the sampling rate)