AUDIO FFT FILTER BANKS

Julius O. Smith

Center for Computer Research in Music and Acoustics (CCRMA)
Music Dept., Stanford University, Stanford, CA 94305, USA
jos@ccrma.stanford.edu

ABSTRACT

FFT-based nonuniform filter banks are proposed based on channel-sized inverse FFTs applied to nonuniform frequency-partitions (or overlap-add decompositions) of the Short Time Fourier Transform (STFT). Audio filter banks (particularly octave filter banks) are considered as application examples. Trade-offs discussed include perfect reconstruction, aliasing cancellation, flexibility of filter-channel band edges, use of the FFT for speed, multirate time-domain channel signals, time-varying filtering, and associated issues.

1. INTRODUCTION

It is well known that the frequency resolution of human hearing decreases with frequency [1, 2]. As a result, any "auditory filter bank" must be a *nonuniform* filter bank in which the channel bandwidths increase with frequency over most of the spectrum. A classic approximate example is the *third-octave filter bank* [3]. A simpler (cruder) approximation is the *octave filter bank* [3], also called a *dyadic filter bank* when implemented using a binary tree structure [4]. Both are examples of *constant-Q filter banks*, in which the bandwidth of each filter-bank channel is proportional to center frequency [5]. Approximate auditory filter banks, such as constant-Q filter banks, have extensive applications in basic hearing research, audio engineering, and digital audio effects.

If the output signals from all channels of a constant-Q filter bank are *sampled* at a particular time, we obtain what may be called a constant-Q *transform* [6]. A constant-Q transform can be efficiently implemented by smoothing the output of a Fast Fourier Transform (FFT) [7]. More generally, a *multiresolution spectrogram* can be implemented by combining FFTs of different lengths and advancing the FFTs forward through time [8, 9]. Such nonuniform filter banks can also be implemented using the Goetzel algorithm [10].

While the topic of *filter banks* is well developed in the literature, including constant-Q, nonuniform FFT-based, and wavelet filter banks, it appears the elementary practical methods presented in this paper have been overlooked to date. In particular, classic nonuniform FFT filter banks as described in [11] have not offered the *perfect reconstruction* property [4] in which the filter-bank sum yields the input signal exactly (to within a delay and/or scale factor) when the filter-band signals are not modified. The voluminous literature on perfect-reconstruction filter banks [4] addresses nonuniform filter banks, such as dyadic filter banks designed based on pseudo quadrature mirror filter designs, but simpler STFT methods do not yet appear to be incorporated.

This paper can be viewed as an extension of [7] to the FFT filter-bank case. Alternatively, it may be viewed as a novel method for nonuniform FIR filter-bank design and implementation, based on STFT methodology, with arbitrarily accurate reconstruction and controlled aliasing in the downsampled case. While this paper considers only auditory (approximately constant-Q) filter banks, the method works equally well for arbitrary nonuniform spectral partitions and overlap-add decompositions.

2. BASIC IDEA

The basic idea is to partition FFT bins into the desired nonuniform bands, and perform smaller inverse FFTs on each subband to synthesize downsampled time-domain signals in each band. A simple example length 8 FFT octave filter bank is shown in Fig. 1. The remainder of this section presents the basic theory in a tutorial style that recount the reasoning leading to the idea. To maximize the audience for which this paper is readable and practically useful, mathematical notation is replaced by high-level signal operations in matlab (the most commonly used high-level language for signal processing [12]). For a tutorial development of the mathematical theory of these operations, see for example [13, 9].

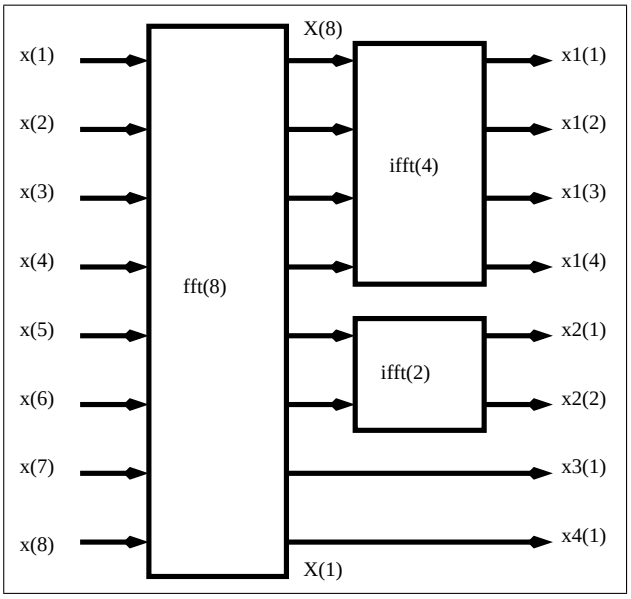

Figure 1: *FFT implementation of one frame of the simple octave filter bank of Fig. 2 on page 3. Successive frames are non-overlapping (rectangular window advances its full length each frame).*

The next section begins with further explication of the (overly) simple example of Fig. 1, followed by discussion of further refine-

ments for achieving audio filter banks of more practical complexity and quality.

2.1. Summing STFT Bins

In the Short-Time Fourier Transform, which implements a *uniform* FIR filter bank [14, 9], each FFT bin can be regarded as one sample of the filter-bank output in one channel [15]. It is elementary that summing adjacent filter-bank signals sums the corresponding passbands to create a wider passband. Summing adjacent FFT bins in the STFT, therefore, synthesizes one sample from a wider passband implemented using the FFT. This is essentially how a constant-Q transform is created from an FFT in [7] (using a different frequency-weighting, or "smoothing kernel"). However, when making a filter bank, as opposed to only a transform used for spectrographic purposes, we must be able to step the FFT through time and compute properly sampled time-domain filter-bank signals.

The wider passband created by adjacent-channel summing requires a higher sampling rate in the time domain to avoid aliasing. As a result, the maximum STFT "hop size" is limited by the widest passband in the filter bank. For audio filter banks, low-frequency channels have narrow bandwidths, while high-frequency channels are wider, thereby forcing a smaller hop size for the STFT. This means that the low-frequency channels are heavily oversampled when the high-frequency channels are merely adequately sampled (in time) [7, 8]. In an octave filter-bank, for example, the top octave, occupying the entire upper half of the spectrum, requires a time-domain step-size of no more than two samples, if aliasing of the band is to be avoided. Each octave down is then oversampled (in time) by an additional factor of 2.

2.2. Inverse Transforming STFT Bin Groups

The solution proposed here is to compute *multiple* time samples for each high-frequency channel, so that one hop of the FFT produces all needed samples in each band in order that all channels can use the same hop size without redundancy. If the narrowest band produces one sample per hop, then a band N times as wide must produce at least N samples per hop.

To efficiently compute multiple time samples from the frequency-samples (FFT bins) of a given band, an inverse FFT can be used, as shown in Fig. 1. In matlab notation, let `X(1:N)` denote the FFT (length `N`) of the current frame of data in the STFT, and denote the lower and upper spectral samples for band `k` by `lsn(k)` and `hsn(k)`, respectively. Then we may compute (in matlab) the full-rate time-domain signal corresponding to band `k` as follows:

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

where `x(k,:)` denotes the output signal vector (length `N`) for the `k`th filter-bank channel for the current time-domain STFT window.

Let $\texttt{Nk} = \texttt{hsn(k)} - \texttt{lsn(k)} + 1$ denote the number of FFT bins in band `k`. When `Nk` is a power of 2, we can apply an inverse FFT only to the nonzero samples in the band:

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

where `xd{k}` now denotes a *cell array* for holding the downsampled signal vectors (since the downsampling factor is typically different for different bands).

We may relate `x(k,:)` and `xd{k}` by noting that, when `Lk = N/Nk` is an integer, we have that the relation

```
BandK == alias(BandKzp,Lk)        % (2)
```

is true, for each element, where `alias(x,K)` denotes aliasing of `K` equal partitions of the vector `x`:[1]

```
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
```

By the *aliasing theorem* (*downsampling theorem*) for Discrete Fourier Transforms (DFTs) [13], the relation `(2)` in the frequency domain corresponds to

```
xd{k} == Lk * x(k,1:Lk:end)
```

in the time domain, *i.e.*, `xd{k}` is obtained from `x(k,:)` by means of *downsampling* by the factor `Lk`. This produces `N/Lk == Nk` samples. That is, for a band that is `Nk` bins wide, we obtain `Nk` time-domain samples for each STFT frame, when critically sampled. (At the full rate, we obtain `N` samples from each channel for each frame.)

We thus see that taking an inverse FFT of only the bins in a given channel (`BandK` above) computes the critically downsampled signal `xdk` for that channel. Downsampling factors between 1 and `Lk` can be implemented by choosing a zero-padding factor for the band somewhere between 1 and `Lk`. (In (1), the zero-padding factor is `N/Nk`.)

Note that this filter bank has the *perfect reconstruction* (PR) property [4]. That is, the original input signal can be exactly reconstructed (to within a delay and possible scale factor) from the channel signals. The PR property follows immediately from the exact invertibility of the FFT. Specifically, we can recover the original signal frame by taking an FFT of each channel-signal frame and abutting the spectral bins so computed to form the original frame spectrum. Of course, the underlying STFT (uniform filter bank) must be PR as well, as it routinely is in practice [9].

2.3. Improving the Channel Filters

Recall that each FFT bin can be viewed as a sample from a bandpass filter whose frequency response is a frequency-shift of the FFT-window Fourier transform [9].[2] Therefore, the frequency response of a channel filter obtained by summing `Nk` adjacent FFT bins is given by the sum of `Nk` window transforms, one for each FFT bin in the sum. As a result, the stopband of the channel filter frequency response is a sum of `Nk` successive window side lobes, and by controlling window side-lobe level, we may control the stopband gain of the channel filters.

The *transition width* from passband to stopband, on the other hand, is given by the main-lobe width of the window transform [9]. In the previous section, by zero-padding the band (line `(1)` above), we implicitly assumed a transition width of one bin. Only the length `N` rectangular window can be reasonably said to have a one-bin transition from passband to stopband. Since the first side lobe of a rectangular window transform is only about 13 dB below

[1] A more efficient implementation can use `reshape` and `sum`.

[2] `http://ccrma.stanford.edu/~jos/sasp/dftfb.html`

the main lobe, the rectangular window gives poor stopband performance, as illustrated in Fig. 4 on page 4. Moreover, we often need FFT data windows to be shorter than the FFT size `N` (*i.e.*, we often need zero-padding in the time domain) so that the frame spectrum will be oversampled, enabling various spectral processing such as linear filtering [9].

One might wonder how the length `N` rectangular window can be all that bad when it gives the perfect reconstruction property, as demonstrated in the previous section. The answer is that there is a lot of aliasing in the channel signals, when downsampled, but this aliasing is exactly canceled in the reconstruction, provided the channel signals were not modified in any way [9].

Going back to §2.1, we need to replace the zero-padded band `(1)` by a proper filtering operation in the frequency domain (a "spectral window"):

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

where the channel filter frequency response `Hk` may be prepared in advance as the appropriate weighted sum of FFT-window transforms:

```
Hideal = [z1,ones(1,Nk),z2];
Hk = cconvr(W,Hideal); % circ. conv.
```

where `z1` and `z2` are the same zero vectors defined in §2.1, and `cconvr(W,H)` denotes the *circular convolution* of two real vectors having the same length [13]:

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

Note that in this more practical case, the perfect reconstruction property no longer holds, since the operation

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

is not exactly invertible in general.[3] However, 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.

The band filters `Hk` can be said to have been designed by the *window method* for FIR filter design [16]. (See functions `fir1` and `fir2` in Octave and/or the Matlab Signal Processing Toolbox.)

3. FAST OCTAVE FILTER BANKS

Let's now apply the above technique to the design of an octave filter bank.[4] At first sight, this appears to be a natural fit, because it is immediately easy to partition a power of 2 (typical for the FFT size `N`) into octaves, each having a width in bins that is a power of 2. For example, when `N == 8`, we have the following stack of frequency-response vectors for the rectangular-window, no-zero-padding, complex-signal case:

```
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 ];
```

Figure 2 depicts the resulting filter bank schematically in the frequency domain.[5] Thus, `H(1,:)` is the frequency-response for the top (first) octave, `H(2,:)` is the frequency-response for the next-to-top (second) octave, `H(3,:)` is the next octave down, and `H(4,:)` is the "remainder" of the spectrum. (In every octave filter bank, there is a final low-frequency band.

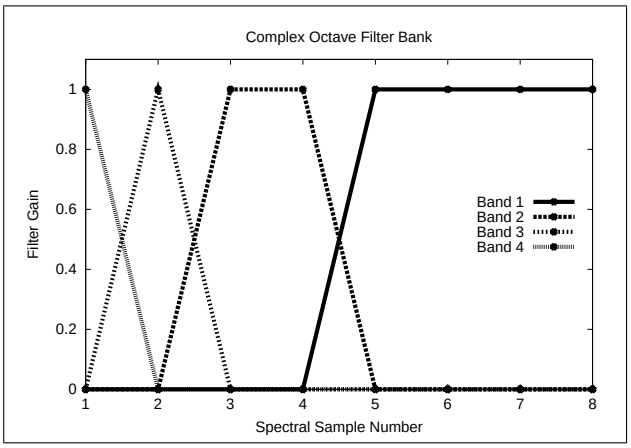

Figure 2: *Simple octave filter bank for complex signals.*

A schematic of the filter-bank implementation using FFTs is shown in Fig. 1 on page 1. The division by `N` called for by `ifft(N)` is often spread out among the `ifft` butterfly stages as divisions by 2 (one-bit right-shifts in fixed-point arithmetic) after each of them. For conservation of dynamic range, half of the right-shifts can be spread out among every other forward-FFT butterfly stage [17] as well, thereby implementing the *normalized* DFT (NDFT) [13].

The simple spectral partitioning of Fig. 2 is appropriate for *complex* signals—those for which the spectrum is regarded as spanning 0 Hz to the sampling rate. For real signals, we need a spectral partition more like that in Fig. 3.

Unfortunately, the number of spectral samples in Fig. 3 is 14—not a power of 2. (The previous length 8 complex case maps to length 14 real case because the dc and Nyquist-limit samples do not have complex-conjugate counterparts.) Discarding the sample at the Nyquist limit—number 8 in Fig. 3—does not help, since that gives 13 samples—still not a power of 2. In summary, there is no obvious way to octave-partition the spectral samples of a real

[3] 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) [9]. These zeros depend on the window-length being `N` (no zero-padding), and on the window-type being rectangular ("no window").

[4] Do not confuse the Octave program—a free, open-source implementation of the matlab language—with the *musical* octave: a frequency interval spanning a factor of two. Here, "Octave" (capitalized) refers to the program, while "octave" refers to the frequency interval.

[5] The samples are connected by straight lines to make them visible. The true responses for the left two bands are aliased sinc functions (asinc). The next octave up is a sum of two asincs, and the rightmost band (top octave) is a sum of four asincs. A properly interpolated frequency response for this filter bank is shown in Fig. 4.

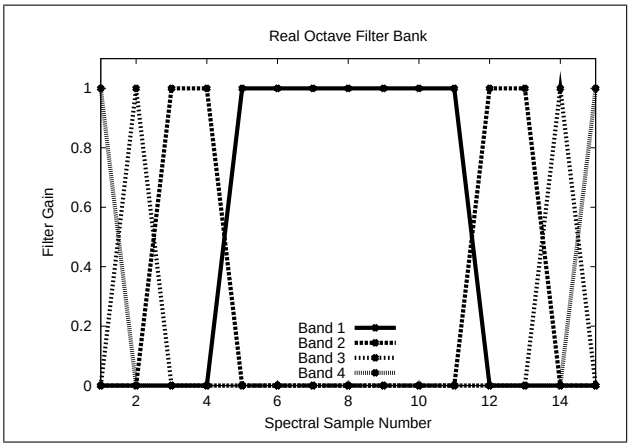

Figure 3: *Simple octave filter bank for real signals.*

signal while maintaining the power-of-2 condition for each band and symmetrically partitioning positive and negative frequencies.

A real signal can of course be converted to its corresponding "analytic signal" by filtering out the negative-frequency components. This is normally done by designing a *Hilbert transform filter* [18, 16, 9]. However, 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 zero the negative-frequency components.

3.1. 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 typically desirable 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 section can be applied separately to each half of the spectrum:

```
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)
```

In the above listing, the function `dcells` simply forms a cell array in matlab containing the spectral partition ("dyadic cells"). The function `dcells2spec` is the inverse of `dcells`, taking a spectral partition and unpacking it to produce a usual spectrum vector.

3.2. Improving the Octave Band Filters

To see the true filter-bank frequency response corresponding to Fig. 2, we may feed an impulse to the filter bank and take a long FFT (for zero padding) of the channel signals to produce the interpolated response shown in Fig. 4.

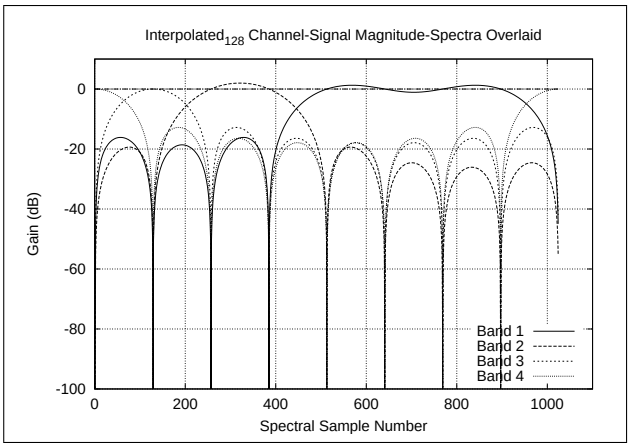

Figure 4: *Channel-frequency-response overlay for the octave filter bank shown in Fig. 2.*

The horizontal line along 0 dB in Fig. 4 was obtained by summing the channel responses, indicating that it is a perfect-reconstruction filter-bank, as expected. However, the stopband performance of the channels is quite poor, being comparable to the sidelobes of a rectangular window transform (an aliased sinc function). In fact, the stopband is identical to the rectangular-window sidelobes for the two lowest bands. Notice that the original eight samples of Fig. 2 still lie along the 0-dB line, and there are still zeros in each channel response beneath the unity-gain point of every other channel's response, so Fig. 2 can be obtained by sampling Fig. 4 at those eight points. However, the interpolated response shows that the filter bank is quite poor by audio standards.

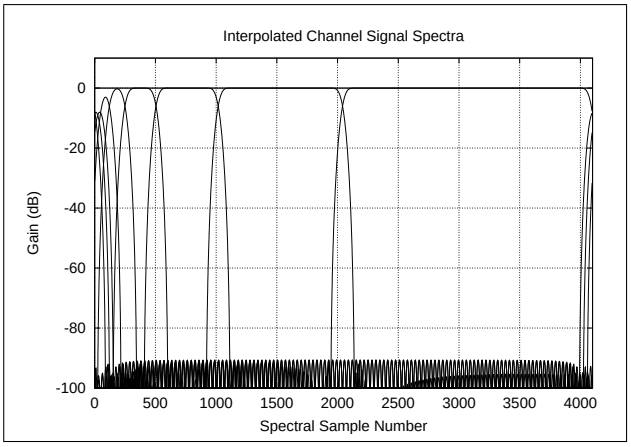

Figure 5: *Channel-frequency-response overlay for a complex-signal octave filter bank, designed using a length 127 Dolph-Chebyshev window (80 dB stopband attenuation) and length 256 FFT.*

Figure 5 shows an octave filter bank (again for complex signals) that is better designed for audio usage. Instead of basing the channel filter prototype on the rectangular window, a *Dolph-Chebyshev window* was used [matlab: `chebwin(127,80)`]. The

FFT size is about twice the filter length, thereby allowing for a data frame of equal length (to the filter) without incurring time aliasing, in the usual way for STFTs [9]. The data window is chosen to overlap-add to a constant, as typical in STFTs, so its choice does not affect the quality of the filter-bank output channel signals. We therefore may continue to use a rectangular data window that advances by its full length each frame. Choosing an odd filter length facilitates zero-phase offline STFT processing, in which the middle FIR sample is treated as occurring at time zero, so that there is no delay in any filter-bank channel [13].

The filter bank is PR in the full-rate case because the underlying STFT is PR, in the absense of modifications, and because the channel filter-bank is constant-overlap-add in the frequency domain (again in the absence of modifications) according to STFT theory [9].

3.3. Aliasing on Downsampling

While the filter bank of Fig. 5 gives good stopband rejection, there is still a lot of *aliasing* when the bands are critically sampled. This happens because the *transition bands* are aliased about their midpoints. This can be seen in Fig. 5 by noting that aliasing "folding frequencies" lie at the crossover point between each pair of bands. An overlay of the spectra of the downsampled filter-bank outputs, for an impulse input, is shown in Fig. 6.

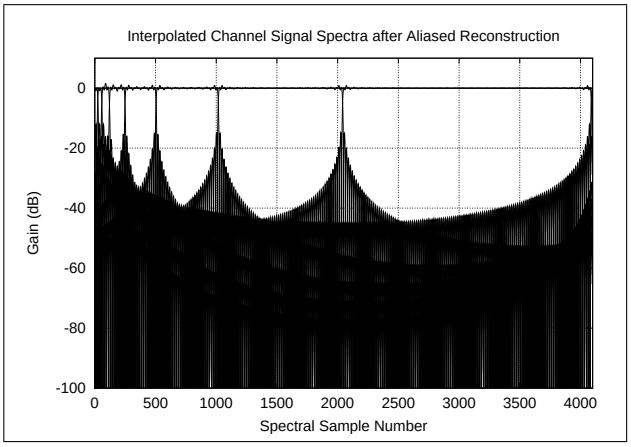

Figure 6: *Same as Fig. 5 obtained by critically downsampling each channel signal, zero-padding, and performing an FFT. All the observable stopband error happens to cancel out in the filter-bank sum because the input signal is an impulse, in which case the reconstruction remains exact.*

Figure 7 shows the aliased spectral signal bands (prior to inverse STFT) for a *step* 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. The signal and error spectra are shown overlaid in Fig. 8. In this case, the aliasing causes significant error in the reconstruction.

3.4. Restricting Aliasing to Stopbands

To eliminate the relatively heavy transition-band aliasing (when critically sampling the channel signals), we can define *overlapping bands* such that each band encompasses its transition bands on either side. However, unless a full $2\times$ oversampling is provided for each band (which is one easy solution), the bandwidth (in bins) is no longer a power of two, thereby thwarting use of radix-2 inverse-FFTs to compute the time-domain band signals.

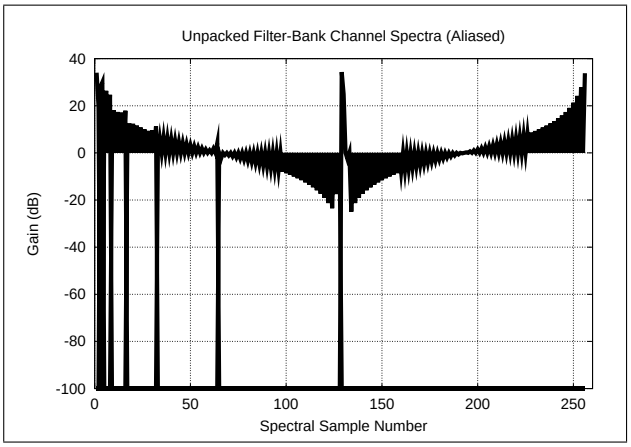

Figure 7: *Same filter bank as in Fig. 6 but driven by a step input.*

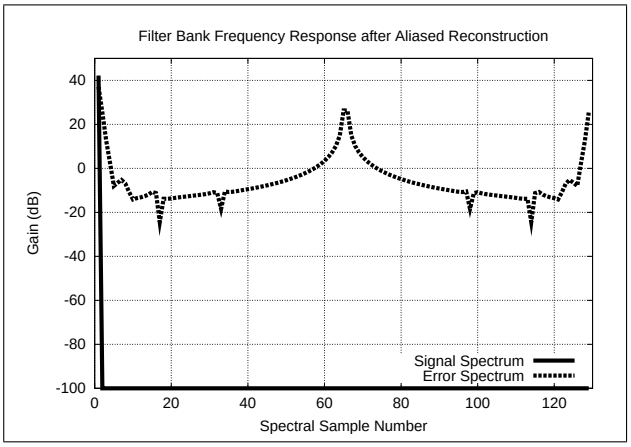

Figure 8: *Signal spectrum (an impulse, since the time signal is a step) and error spectrum for the case of Fig. 7. Note the large error near half the sampling rate.*

To keep the channel bandwidths at powers of two while restricting aliasing to stopband energy only, each IFFT input band can be widened beyond the passband plus two transition bands so that it includes just enough stopband to reach the next power of two.[6] If the stopband energy is negligible, the band can be simply zero-padded outside of the transition bands instead of including additional stopband; in other words, each frequency band may be individually zero-padded to the next higher power of two. Since the passband widths and IFFT sizes are now decoupled, the channel IFFTs will overlap by varying amounts, in general.

The bands also may have any shape that sums to a constant frequency response; in other words, while we have been considering only spectral *partitions* in this paper, any *overlap-add decomposi-*

[6] We do not really have to restrict consideration to powers of two, as there are many fast Fourier transform algorithms for various composite and prime lengths [19, 20].

tion of the spectrum can be used. (See the gammatone and gammachirp filter banks for examples of heavily overlapping bands in an audio filter bank [21].) Both zero-padding and overlap-add-decomposition are terms normally applied in the time domain [9]. Thus, nonuniform filter banks may be formulated as the Fourier dual of nonuniform time windows. Figure 9 shows how the example of Fig. 5 is modified by this strategy.

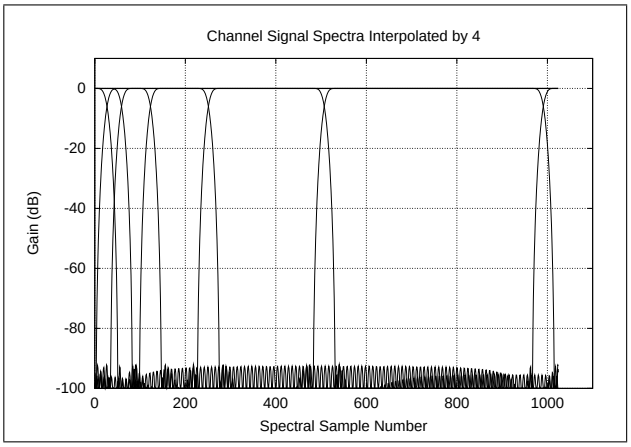

Figure 9: *Channel-frequency-response overlay for an octave filter bank designed using a length 127 Dolph-Chebyshev window (80 dB SBA) and length 256 FFT size.*

In summary, passbands of arbitrary width and shape are embedded in overlapping IFFT bands that are a power-of-2 wide. As a result of this flexibility, the frequency-rotation trick of §3.1 is no longer needed for real filter banks. Instead, we simply allocate any desired bands between dc and half the sampling rate, and conjugate-symmetry dictates the rest. In addition to a left-over "dc-Nyquist" band, there is a similar residual "Nyquist-limit" band (a typically negligible band about half the sampling rate). In other words, 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.

The minimum channel bandwidth is defined here as two transition bands plus one bin (*i.e.*, the minimum passband width is zero, corresponding to one bin, or one spectral sample). For the Dolph-Chebyshev window, the transition bandwidth is known in closed form [22, 9]. In our examples, we have a length 127 window with 80 dB stopband attenuation in the lowpass prototype [matlab command `chebwin(127,80)`], corresponding to a transition width of 6.35 bins in a length 256 FFT, which was rounded up to 7 bins in software for simplicity of band allocation. Therefore, our minimum channel bandwidth is 15 bins (two transition bands plus one sample for the band center). The next highest power of two is 16, so that is our minimum encompassing IFFT length for any band.

The dc and Nyquist channels are combined into a single channel containing the left-over residual filter-bank response consisting of a low transition down from dc and a high-frequency transition up to the sampling rate (in the complex-signal case). When `N` is sufficiently large so that these bands contain no audible energy, they may be discarded. We include them in all examples here so as to preserve the (near) perfect reconstruction property of the filter bank. Thus, the 7-bin dc channel is combined with the 7-bin Nyquist channel to form a single 16-bin encompassing residual band that may be discarded in many audio applications (when the initial FFT size is sufficiently large for the sampling rate used).

In the example of Fig. 9, the initial FFT size is 256, and the channel bandwidths (passbands only, excluding transitions), from top to bottom, are 121, 64, 32, 16, and 8 bins. The top band is reduced by 7 bins to leave a transition band to the sampling rate. Similarly, the lowest band lies above a transition band consisting of bins 0-6. The encompassing IFFTs (containing transitions) are lengths 256, 128, 64, 32, 32, for the interior bands, and a length 32 IFFT handles the dc and Nyquist bands (which are combined into a single 14-bin band about dc, which occupies 28 bins when the transition bands are appended). Letting [lo,hi] denote a band by its lower and upper bin limit, the nonoverlapping adjacent passband edges in "spectral samples"[7] of the interior bands are [8, 15], [16, 31], [32, 63], [64, 127], and [128, 248]; the overlapping encompassing IFFT band edges are then [1, 32], [9, 40], [25, 88], [57, 184], [1, 256], *i.e.*, they each contain a passband and two transition bands, and have a power-of-2 length. The downsampling factor for each channel can be computed as the initial FFT size (256) divided by the IFFT size (32, 32, 64, 128, or 256) for the channel.

Figure 10 shows the counterpart of Fig. 6 for this example. In this case, the aliased signal energy comes only from channel-filter stopbands. For narrow bands, the aliasing is suppressed by at least 80 dB (the side-lobe level of the chosen Dolph-Chebyshev window transform). For bands wider than one bin (the minimum bandwidth in this example is the dc-Nyquist band at 14 bins), the stopband consists of a sum of shifts of the window-transform sidelobes, and these are found to be more than 80 dB down due to cancellation (more than 90 dB down in most bands of this example).

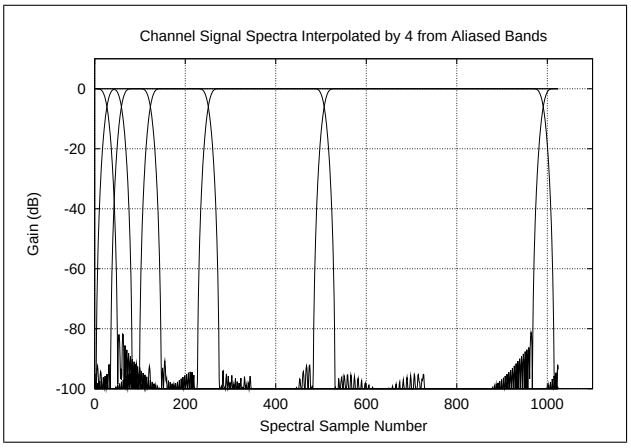

Figure 10: *Same example as in Fig. 9 but looking at the effects of aliasing to to channel-signal downsampling. Compare to Fig. 6.*

3.4.1. Tightening the IFFTs

In this example the top band is not downsampled at all, and the interior bands are oversampled by approximately 2. This is because the desired passband widths started out at a power of 2, so that the addition of transition bands forced the next higher power of 2

[7] Spectral samples are defined here as "bin numbers plus 1", that is, spectral samples are numbered from 1 as in matlab, rather than from 0, as in the signal processing literature.

for the IFFT size. Narrowing the width of the top band from 121 bins to $128 - 2 \cdot 7 = 114$ bins would enable use of a length 128 IFFT for the top band, and similarly for the lower bands. In other words, when the desired spectral partition is that of an ideal octave filter bank, as sketched in Fig. 2, narrowing each octave-band by twice the transition width of the lowpass prototype filter (and "covering down" to keep them adjacent) will produce a relatively "tight" FFT filterbank design in which the IFFT sizes remain the same length as in the heavily aliased case discussed above (Fig. 6). When applied to the octave filter bank, the passbands become a little narrower than one octave. We may call this a *quasi octave filter bank*.

3.4.2. Real Filter Bank Example

Finally, Fig. 11 shows the appearance of the octave filter bank for real signals. In this case, bands are constructed between dc and *half* the sampling rate, and conjugate symmetry is used to automatically construct the desired bands between half the sampling rate and the sampling rate. The band allocation algorithm therefore needs no modification for the real-signal case.

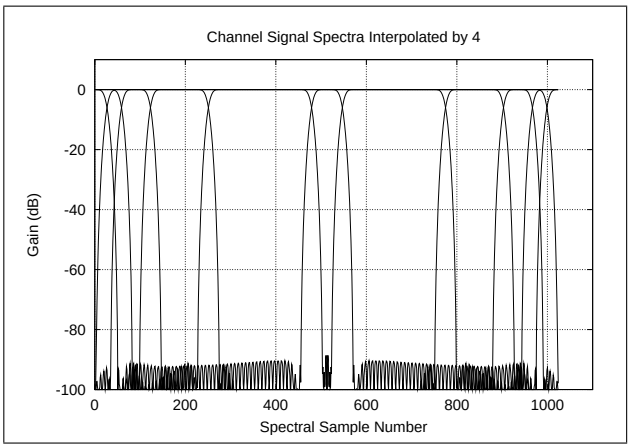

Figure 11: *Channel-frequency-response overlay for a* real-signal *octave filter bank, designed using a length 127 Dolph-Chebyshev window and length 256 FFT size (no downsampling). Compare the complex-signal filter bank in Fig. 9.*

3.5. Optimal Band Filters

In the filter-bank literature, one class of filter banks is called "cosine modulated" filter banks. DFT filter banks are similar. The lowpass-filter prototype in such filter banks can be used in place of the Dolph-Chebyshev window used here. Therefore, any result on optimal design of cosine-modulated filter banks can be adapted to this context. See, for example, [23, 24]. Note, however, that in principle a separate optimization is needed for each different channel bandwidth. An optimal lowpass prototype only optimizes channels having a one-bin passband, since the protype frequency-response is merely shifted in frequency (cosine-modulated in time) to create the channel frequency response. Wider channels are made by summing such channel responses, which alters the stopbands.

In practice, the Dolph-Chebyshev window, used in the examples of this paper, typically yields a filter bank magnitude frequency response that is optimal in the Chebyshev sense because

1. there can be only one lowpass prototype filter in any modulated filter bank (such as the DFT filter bank),
2. the prototype itself is the optimal Chebyshev lowpass filter of minimum bandwidth, and
3. summing shifted copies of the prototype frequency response generally *improves* the stopband rejection over that of the prototype, thus meeting the Chebyshev optimality requirement for the filter-bank as a whole.

All channel bands, whatever their width, are constructed by some linear combination of shifted copies of the lowpass protoype frequency response. The Dolph-Chebyshev window is precisely optimal (in the Chebyshev sense) for any passband that is one bin wide. Summing shifts of the window transform to synthesize wider bands has been observed to invariably improve the stopband rejection significantly. The examples shown above illustrate the margin beyond 80 dB stopband rejection achieved for the octave filter bank.

The Dolph-Chebyshev window has faint impulsive endpoints on the order of the sidelobe level (about 50 dB down in the 80-dB-SBA examples above), and in some applications, this could be considered an undesirable time-domain characteristic. To eliminate them, an optimal Chebyshev window may be designed by means of linear programming with a time-domain monotonicity constraint [9].[8] Alternatively, of course, other windows can be used, such as the Kaiser, or three-term Blackman-Harris window, to name just two [9]. In this paper we use only the Chebyshev window in order to facilitate comparisons. Effects changing the window type have exactly the same effect as one would expect from experience with the window method for FIR filter design [16, 9].

4. TIME VARYING NONUNIFORM FFT FILTER BANKS

As is well known, the STFT can be modified independently from one frame to the next over time [14, 9]. In the present context, it is natural to consider the case of changing the spectral partition (or spectral overlap-add decomposition more generally) from one frame to the next.

In the time-invariant case discussed above, there was no reason not to use a length M rectangular window and hop size M for the underlying STFT. That is, there was no reason to use overlapping (in time) or tapered blocks of the input signal. However, when the filter-bank (spectral partition/OLA-decomposition) changes from frame to frame, it may be preferable to use some tapered window, such as the triangular (Bartlett) or raised-cosine (Hann) window along with a compatible overlap (such as 50% for triangular or Hann). This gives the effect of cross-fading from one spectral partition to the next, instead of suddenly introducing the new filtering at a frame boundary.

An effect similar to cross-fading is obtained by using zero-phase (instead of causal) channel filters. When the channel filters are zero phase, the current output frame consists of the superposition of itself filtered by the current spectral partition, plus the causal "ringing" from the previous frame, plus the anticausal "prering" from the next frame. Zero-phase channel filters can of course be used in conjunction with a time-domain overlap-add decomposition of the input signal [9].

If a time-varying filter bank is used with downsampling, the algorithm presented above may cause the downsampling factor may

[8] `https://ccrma.stanford.edu/~jos/sasp/-Window_Design_Linear_Programming.html`

vary over time in a given band. Using the rule of "zero-padding" (widening) each band-plus-its-transitions to the next higher power of two can cause the sampling rate in a given channel to change by factors of two over time. Allowing this would yield successive frames at different sampling rates in the time domain. When it is desired to have only one sampling rate for each filterbank channel (especially needed when time-frames overlap), one simple solution is to zero-pad to the same power of two for each time-frame of a given band, given simply as its maximum over time. That is, a fixed sampling rate is chosen for the time-varying band that accommodates its maximum bandwidth over time. Thus, uniformly sampled time-varying bands must have a maximum bandwidth imposed (or measured in the offline case).

5. CONCLUSIONS

An approach to the design and efficient implementation of nonuniform, perfect- and near-perfect-reconstruction, FFT filter banks, with controllable aliasing, was presented. These filter banks can be viewed as an extension and generalization of the FFT-based constant-Q transform to the FFT filter-bank case. Alternatively, they may be seen as a new class of nonuniform FIR filter banks based on FFT block processing, with arbitrarily accurate reconstruction and controlled aliasing in the downsampled case. While only auditory (approximately constant-Q) filter banks were considered here as examples, arbitrary nonuniform spectral partitions and/or spectral overlap-add decompositions are just as easily implemented. Moreover, time-varying nonuniform filter banks may be implemented using no more than the usual additional considerations that arise in time-varying STFT processing of any kind.

6. REFERENCES

[1] H. Fastl and E. Zwicker, *Psychoacoustics: Facts and Models*, Springer Verlag, Berlin, 2006, third edition, 462pp., CD-ROM.

[2] Stanley Smith Stevens and Hallowell Davis, *Hearing: Its Psychology and Physiology*, Amer.n Inst. Physics, for the Acoust. Soc. of Amer., 1983, copy of original 1938 edition, `http://asa.aip.org/publications.html`.

[3] L. L. Beranek, *Acoustics*, American Institute of Physics, for the Acoustical Soc. of Amer., `http://asa.aip.org/publications.html`, 1986, 1st ed. 1954.

[4] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice-Hall, 1993.

[5] Julius O. Smith, *Introduction to Digital Filters with Audio Applications*, `https://ccrma.stanford.edu/~jos/filters/`, Sept. 2007, online book.

[6] Judith C. Brown, "Calculation of a constant Q spectral transform," *J. Acoust. Soc. of Amer.*, vol. 89, no. 1, pp. 425–434, 1991.

[7] Judith C. Brown and Miller S. Puckette, "An efficient algorithm for the calculation of a constant Q transform," *J. Acoust. Soc. of Amer.*, vol. 92, no. 5, pp. 2698–2701, 1992.

[8] Brian R. Glasberg and Brian C. J. Moore, "A model of loudness applicable to time-varying sounds," *J. Audio Eng. Soc.*, vol. 50, no. 5, pp. 331–342, May 2002.

[9] Julius O. Smith, *Spectral Audio Signal Processing*, `https://ccrma.stanford.edu/~jos/sasp/`, Dec. 2011, online book.

[10] Ryan J. Cassidy and Julius O. Smith III, "Efficient time-varying loudness estimation via the hopping Goertzel DFT," in *Proceedings of the IEEE International Midwest Symposium on Circuits and Systems (MWSCAS-2007)*, Aug. 2007.

[11] Larry R. Rabiner and Ron W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, 1978.

[12] Julius O. Smith, *Introduction to Matlab and Octave*, `https://ccrma.stanford.edu/~jos/matlab/`, 2003.

[13] Julius O. Smith, *Mathematics of the Discrete Fourier Transform (DFT), with Audio Applications, Second Edition*, `https://ccrma.stanford.edu/~jos/mdft/`, Apr. 2007, online book.

[14] Jont B. Allen and Larry R. Rabiner, "A unified approach to short-time Fourier analysis and synthesis," *Proc. IEEE*, vol. 65, no. 11, pp. 1558–1564, Nov. 1977.

[15] M. R. Portnoff, "Implementation of the digital phase vocoder using the fast Fourier transform," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-24, no. 3, pp. 243–248, June 1976.

[16] Larry R. Rabiner and Bernard Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, 1975.

[17] Robert Bristow-Johnson, "Tutorial on floating-point versus fixed-point," *Audio Eng. Soc. Convention*, Oct. 2008.

[18] Michael Z. Komodromos, Steve F. Russel, and Ping Tak Peter Tang, "Design of FIR Hilbert transformers and differentiators in the complex domain," *IEEE Trans. Circuits and Systems—II: Analog and Digital Signal Processing*, vol. 45, no. 1, pp. 64–67, Jan 1998.

[19] M. Frigo and S. G. Johnson, "FFTW: An adaptive software architecture for the FFT," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Seattle*, New York, 1998, vol. 3, pp. 1381–1384, IEEE Press, `http://www.fftw.org/`.

[20] (Various), *Prime-Factor FFT Algorithm*, `http://www.wikipedia.org/wiki/-Prime-factor_FFT_algorithm`, 2003.

[21] Toshio Irino and Roy D. Patterson, "A time-domain, level-dependent auditory filter: The gammachirp," *J. Acoust. Soc. of Amer.*, vol. 101, pp. 412–419, 1997.

[22] Peter Lynch, "The Dolph-Chebyshev window: A simple optimal filter," *Amer. Meteorological Soc. J. Online*, vol. 125, no. 4, pp. 655–660, Apr. 1997.

[23] Guangming Shi, Xuemei Xie, Xuyang Chen, and Wei Zhong, "Recent advances and new design method in nonuniform filter banks," in *Proc. IEEE Int. Conf. Comm., Circ. & Sys.*, June 2006, vol. 1, pp. 211–215.

[24] Zhang Zijing and Yang Yun, "A simple design method for nonuniform cosine modulated filter banks," in *IEEE Int. Symp. on Microwave, Antenna, Prop., & EMC Tech. for Wireless Comm.*, 2007, pp. 1052–1055.