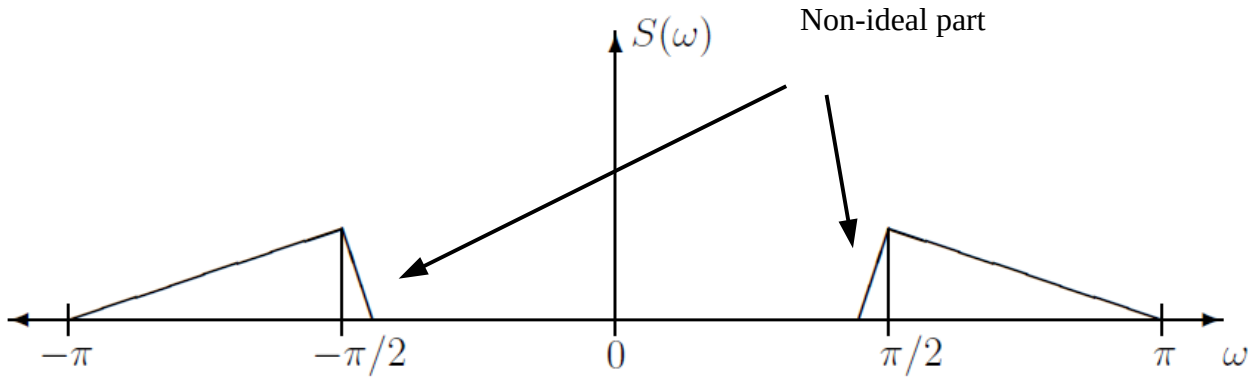
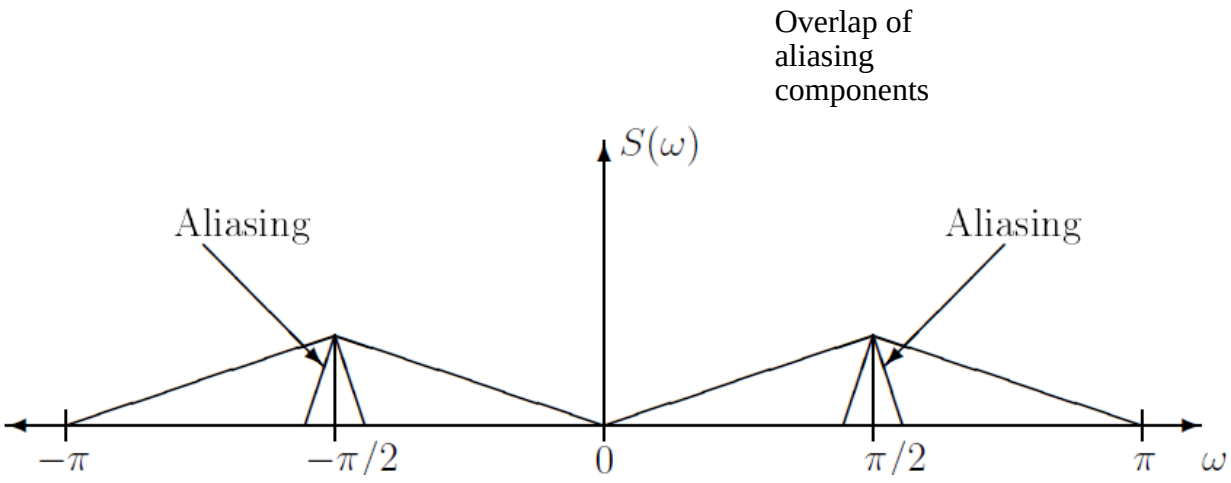
Lecture 9, Multirate Signal Processing
Non-ideal Filters

We saw that we can get perfect reconstruction in a filter bank, using ideal filters (ideally we remove all aliasing components). But in practice, we don't have ideal filters, they are not realizable, because their impulse response stretches from minus infinity to infinity.
How can we obtain **perfect reconstruction** using realizable **non-ideal filters**, and still "**critical sampling**" (meaning the down sampling rate N is equal to the number of sub bands, as in the case of ideal filters)? Let's first take a look at the resulting spectrum when we use non-ideal filters. We start with the spectrum at the output of a non-ideal high-pass,



Observe that the slope at pi/2 and -pi/2 is not infinite, but we get a more or less slow transition into higher attenuation towards lower frequencies. Hence we also get frequencies

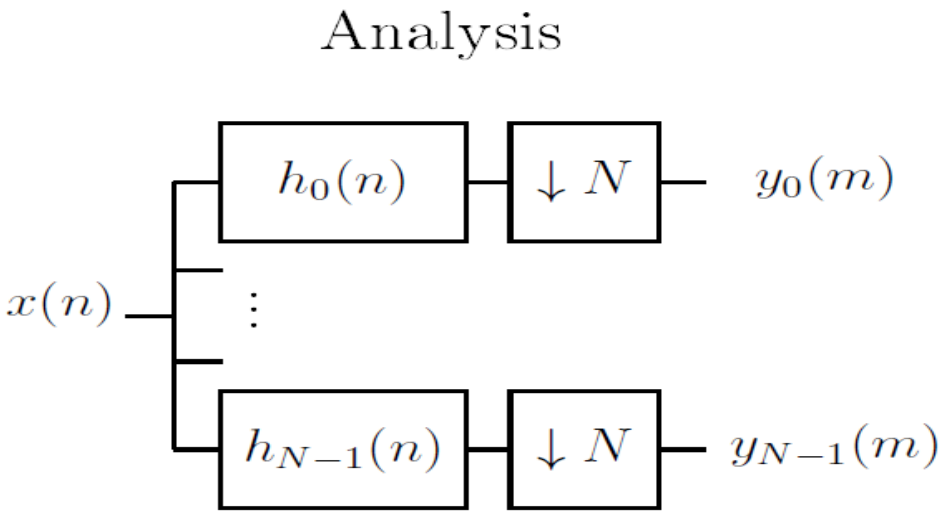
below pi/2 (or above -pi/2) in our signal. This will become a problem after downsampling by a factor of 2 (for now just the multiplication with the delta impulse train), as can be seen in the next pictures. Remember that for the multiplication with the delta train we get aliasing components at frequency shifts of 2pi/N, hence in this case pi,

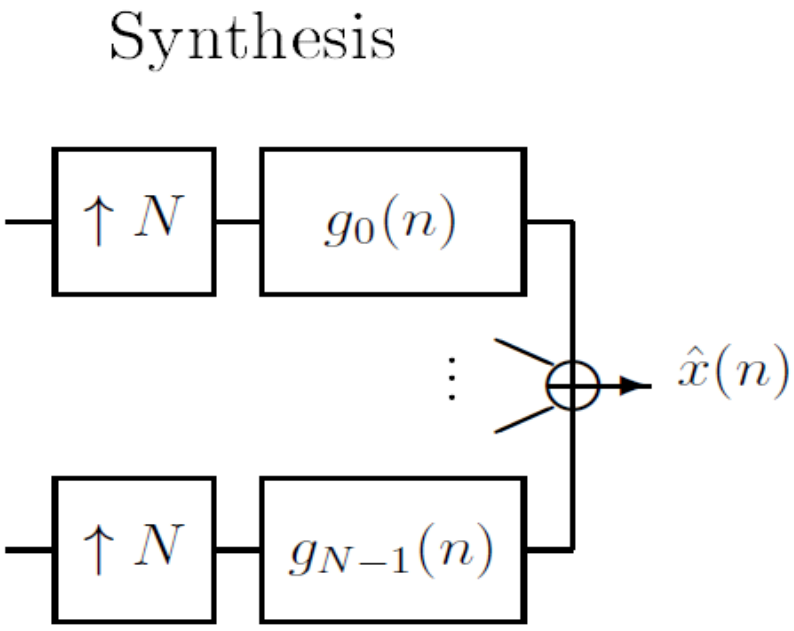


Observe, even if we now used ideal filters for the synthesis for the high-pass part, we would still have the **overlapped aliasing component**, and hence no perfect reconstruction.

Filter Banks

Filter banks consist of a bank of subbband filters which cover the entire frequency band of our signal, followed by downsampling in the analysis filter bank, and preceded by upsampling in the synthesis filter bank. This can be seen in the following pictures,





In the picture, $x(n)$ is the input signal (for instance our audio signal), $y_k(m)$ are the downsampled subband signals (with k the subband index and m the index at the lower sampling rate), and $\hat{x}(n)$ is the reconstructed signal.
As described in lecture 2, $\downarrow N$ Symbolizes downsampling by a factor of N (including removal of the zeros). If $x(n)$ is the signal we downsample, then we also write the downsampled signal as

$$x_{n_0}^{\downarrow N}(m) := x(mN + n_0)$$

where n_0 is the index of the first sample we keep in the downsampling, or the "**phase**", with $0 \leqslant n_0 \leqslant N-1$.
$\uparrow N$ symbolizes upsampling by a factor of N, including insertion of the zeros. If $y(m)$ is the signal we upsample, then we also write the upsampled signal as

$$y_{n_0}^{\uparrow N}(n) = \begin{cases} y(m), n = mN + n_0 \\ 0, else \end{cases}$$

where n_0 is the index of the first non-zero sample in the upsampled signal, or the phase, with
$0 \leqslant n_0 \leqslant N-1$.

The analysis can be found for instance in audio

encoders, and the synthesis in audio decoders. We would like to have this subband decomposition of our signal, because in this way we can process each subband signal differently, for instance according to the different sensitivities of our ear at different frequencies.
We can obtain the different subband filters of our filter bank for instance by using our window method and modulation.

Our **goal** is **perfect reconstruction**, which means the output signal is identical to the input signal except for some delay d:

$$\hat{x}(n) = x(n-d)$$

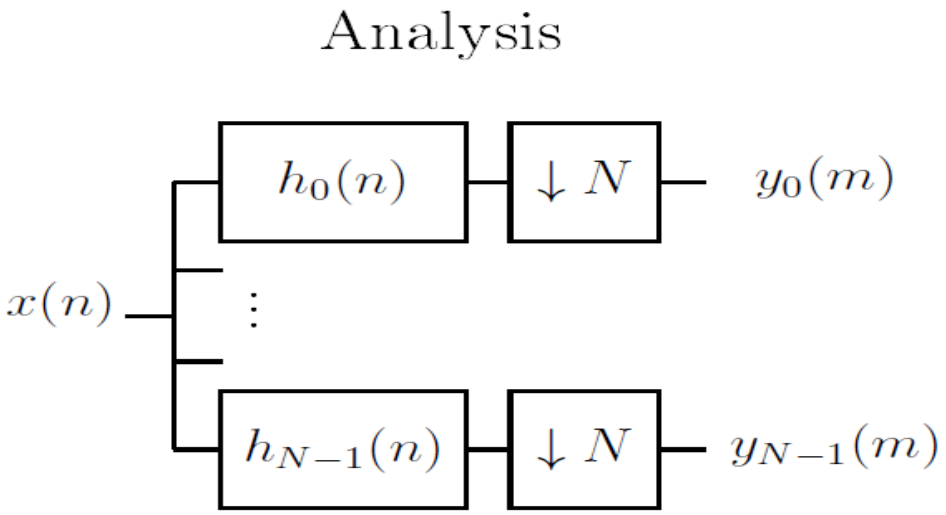
and **good frequency responses**.

So the only hope we have to **obtain perfect reconstruction** with non-ideal filters (and still critical sampling) in filter banks, is that somehow the aliasing from different subbands cancels out during the synthesis process, when we add up all the subbands (if we add e.g. the low pass signal and the high pass signal for the reconstruction); or in other words, we would like the sum of all aliasing components at the output of the synthesis filters to become zero.
The question now is: how do we obtain this goal?
We could now analyse all the aliasing components, to find out when they cancel each

other. This is how researchers actually approached this problem first, for instance in the first derivation of the so-called Time-Domain Aliasing Cancellation (TDAC) filter banks by Princen and Bradley (*J. P. Princen, A. W. Johnson und A. B. Bradley: Subband/transform coding using filter bank designs based on time domain aliasing cancellation, IEEE Proc. Intl. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*), or using the aliasing component matrices by Vetterli et al. But it turned out that a simpler mathematical approach is possible by just using the goal of overall perfect reconstruction of our filter bank, regardless of all aliasing components. This is now the goal of the following derivation. Here we need to analyse in more detail what happens in the filtering and downsampling process in our filter bank.

To come to a solution, we need to apply the z-transform to our filter bank with non-ideal filters.

Analysis Filter Bank



An analysis filter bank.

Just the filtering can be written with the convolution equation,

$$y(n') = \sum_{n=0}^{L-1} x(n'-n)\,h_k(n)$$

The filtering and subsequent **downsampling** by N of the k'th analysis filter $h_k(n)$ of impulse response length L can be written as a downsampled convolution sum with phase offset n_0 , simply with the substitution $mN+n_0 \rightarrow n'$ (and have the new argument m of y)

$$y_{n_0}^{\downarrow N}(m) = \sum_{n=0}^{L-1} x(mN+n_0-n)\,h_k(n) \qquad (2)$$

(see also lecture 1) where m is again the downsampled index (at the lower sampling

rate). n_0 Is our **phase index** for the downsampling. Often we assume $n_0=0$, which means we start downsampling at the first sample (keeping the first sample).
In this way the sum only computes the values which the donwsampler keeps. The downsampling in this equation is now the hurdle to simply applying the z-transform to turn the convolution into a multiplication, for easy invertibility.

Transforms

To look for a solution for invertibility, perfect reconstruction and alias cancellation, we now take a detour to fast implementations, using transforms, like the **DFT** and its fast implementation, the FFT. This is also the approached used by the Short Time Fourier Transform (**STFT**). If we **divide our signal** into **blocks** of length N,

$$\begin{aligned}\boldsymbol{x}(m)&=[x(mN),x(mN+1),\dots,x(mN+N-1)]\\&=[x_0^{\downarrow N}(m),\dots,x_{N-1}^{\downarrow N}(m)]\end{aligned}$$

where m is the block index or index at the lower sampling rate. You can imagine the signal as a long horizontal sequence of samples, which we subdivide into small blocks of length N. We then stack those blocks on top of each other. Then going vertically down, we obtain downsampled versions of our signal at different

phase lags,

$$\begin{array}{lcc} & x_0^{\downarrow 2}(m) & x_1^{\downarrow 2}(m) \\ & phase\,0 & phase\,1 \\ & \downarrow & \downarrow \\ \boldsymbol{x}(0), block\,0 \rightarrow & [x(0) & x(1)] \\ \boldsymbol{x}(1), block\,1 \rightarrow & [x(2) & x(3)] \end{array}$$

The DFT computes a block with N frequency or subband values for each **signal block** $\boldsymbol{x}(m)$. If $\boldsymbol{T}$ is the DFT transform matrix, then the DFT of each signal block is

$$\boldsymbol{y}(m)=\boldsymbol{x}(m)\cdot\boldsymbol{T}.$$

where $\boldsymbol{y}(m)$ contains the DFT coefficients,

$$\boldsymbol{y}(m):=[y_0(m), y_1(m), \ldots, y_{N-1}(m)]$$

Here, $y_k(m)$ is the k 'th DFT coefficient for block m . Hence index k denotes the subband or frequency, and index m is the block index or time at the lower sampling rate. Hence, if we keep k fixed we obtain a time signal, for the $k'th$ subband. If we keep m fixed, we obtain a spectrum for block m . That is why this is also called a “time/frequency” representation, since we have both, time (m) and frequency (k) indices in it. This shows that the DFT is simply a special case of an analsys filter bank with critical sampling (and perfect reconstruction through

the inverse DFT).

Python Example:

`python pyrecspecwaterfall.py`

Observe: The horizontal axis is frequency k, the vertical axis is time m, and color is the magntitude of $y_k(m)$.

Fast Implementation

As the name says, the FFT has a fast implementation. Compare it to our filter bank: per subband we need to calculate the convolution sum. If we have filters of length L, that means we have L multiplications per output sample (we neglect the sum operations) for each sub band. Without down samplers, we get LN multiplications for each block of N input samples for each subband, leading to $L N^2$ multiplications for all subbands for the block of N input samples. Now we can reduce this amount by only computing the output samples which the downsampler keeps. This reduces the computational complexity by a factor of N, resulting in LN multiplications for each input block of length N, to obtain N output values (1 value in each subband). In other words: each sample in a subband is computed by a filter of length L which needs L multiplications in the convolution sum. We need L multiplications per

sample.

If we compare it with an FFT, it only needs on the order of $N\log_2(N)$ multiplications ("order" meaning: up to a fixed factor) for each block. Hence for each subband sample we need log(N) multiplications. Compare this with the L multiplications per sample for the critically sampled filter bank.
Since usually we have $L>\log_2(N)$, this is **more efficient**.
Example: If N=1024 subbands, then we need filters which are longer than L=1024, and $L\gg\log_2(1024)=10$. Hence in this case, we save more than an order of a factor of 100 in complexity using the FFT!

Another interesting effect is, that we obtain **perfect reconstruction** using the **inverse FFT** for the synthesis process, even though the equivalent FFT filter bank has no perfect filters. So this might give us a hint on how to obtain perfect reconstruction without ideal filters. The question is, what is the **equivalent FFT filter bank**?