Lecture 12, Multirate Signal Processing
Polyphase Representation

Last time we saw how to obtain the polyphase representation for the filtering and downsampling operation of **1 filter**. We now extend this formulation for a **bank of N filters**. Here we not only have 1 filter, but N filters in our **analysis filter bank**, and hence can assemble the N filter outputs or N subbands into a vector of N elements,

$$\boldsymbol{Y}(z) := [Y_0(z), Y_1(z), \ldots, Y_{N-1}(z)]$$

We also now have **N filters** instead of 1 filter, and we assemble the polyphase vectors of our filters into a matrix, in which each column corresponds to an anlalysis filter,

$$\boldsymbol{H}(z) := [\boldsymbol{H}_{\boldsymbol{0}}^{\boldsymbol{T}}(z), \boldsymbol{H}_{\boldsymbol{1}}^{\boldsymbol{T}}(z), \ldots, \boldsymbol{H}_{\boldsymbol{N-1}}^{\boldsymbol{T}}(z)] =$$

$$= \begin{bmatrix} H_{N-1,0}(z) & H_{N-1,1}(z) & \ldots & H_{N-1,N-1}(z) \\ H_{N-2,0}(z) & \ldots & \ldots & \vdots \\ \vdots & \ldots & \ddots & \vdots \\ H_{0,0}(z) & \ldots & \ldots & H_{0,N-1}(z) \end{bmatrix}$$

with

$$H_{n,k}(z) = \sum_{m=0}^{\infty} h_k(mN+n) \cdot z^{-m}$$

This **matrix of polynomials** now contains all our N impulse responses of our analysis filter bank, just like in our block transform case, but with **arbitrary long filters**. Unlike our

transform case, here we now have z-transforms (polynomials in z) as our matrix entries. This now enables us to write the filtering and downsampling operations of our entire analysis filter bank as a simple multiplication using the above **polyphase matrix!** This is a very important result:

$$\boldsymbol{Y}(z)=\boldsymbol{X}(z)\cdot\boldsymbol{H}(z)$$

Mathematically this looks similar to the block transform case, but with z-transforms! Observe that this equation now contains **all the samples** of our **input signal** and also of our **subband signals** and our impulse responses, because we use the z-transformed signals. The z-transform converts a potentially **infinite sequence** into just a scalar or an element (its z-transform), which can be seen as a 1x1 matrix (the z-transform of that sequence is one big polynomial). All the input samples are in vector **X**(z). This is important because it allows us to use longer filters than just 1 block, longer than with the block transforms.

Observe: For an **implementation**, we can write the polyphase matrix as a polynomial of matrices $H(m)$,

$$\boldsymbol{H}(z)=\sum_{m=0}^{\infty} H(m)\cdot z^{-m}$$

where the elements of the n'th row and k'th column of $H(m)$ are

$$H_{n,k}(m)=h_k(mN+n)$$

In Python the exponent m would appear as the third dimension, as "slices" of matrices,

$$\texttt{Hp[n,k,m]}= h_k(mN+n)$$

In conclusion: We can write the entire analysis filter bank with its N filters and downsamplers by N with a size NxN polyphase matrix $\boldsymbol{H}(z)$, which is multiplied by the polyphase vector of the signal $\boldsymbol{X}(z)$.

Synthesis Filter Bank

Just as for the block transform case, we can also get a corresponding formulation for the synthesis filter bank. For the synthesis filter bank, we can now also write the reconstructed sequence $\hat{x}(n)$ in terms of blocks with index m and phases n, and obtain the synthesis upsampling and convolution as (see also eq. (8) of lecture (11))

$$\hat{x}(mN+n)=\sum_{k=0}^{N-1}\sum_{m'=0}^{L/N-1} y_k(m-m')\cdot g_k(m'N+n)$$

(with some still to be determined g_k)

We can now also use vectors for our sequences of blocks to simplify this equation, using a vector for our reconstructed signal and for our k'th synthesis filter (we start again with looking at just 1 filter),

$$\hat{\boldsymbol{x}}(m)=[\hat{x}(mN),\hat{x}(mN+1),...,\hat{x}(mN+N-1)]$$
$$\boldsymbol{g_k}(m)=[g_k(mN),g_k(mN+1),...,g_k(mN+N-1)]$$

Now we can re-write our synthesis equation as (with L the length of the synthesis filters)

$$\hat{\boldsymbol{x}}(m)=\sum_{k=0}^{N-1}\sum_{m'=0}^{L/N-1} y_k(m-m')\,\boldsymbol{g_k}(m')$$

The inner sum is a convolution,

$$\hat{\boldsymbol{x}}(m)=\sum_{k=0}^{N-1} y_k(m)*\boldsymbol{g_k}(m)$$

where we now no longer have our phase index n, because we now have output blocks instead of samples.
The inner sum is again a convolution, which turns into a multiplication using the z-transform,

$$\hat{\boldsymbol{X}}(z)=\sum_{k=0}^{N-1} Y_k(z)\boldsymbol{G_k}(z)$$

Now we can extend this notation to our bank of N synthesis filters using our subband vector

$\boldsymbol{Y}(z)$, and the synthesis polyphase matrix. Since the output is the sum of all subbands, we obtain our polyphase matrix by collecting all our polyphase (row) vectors of our synthesis filters $\boldsymbol{G_k}(z)$ into a matrix, such that the **outer sum** of the above equation **turns into a matrix multiplication,**

$$\hat{\boldsymbol{X}}(z)=[Y_0(z), Y_1(z), \ldots, Y_{N-1}(z)]\cdot \boldsymbol{G}(z)$$

where each row of $\boldsymbol{G}(z)$ now contains one synthesis filter, or

$$\hat{\boldsymbol{X}}(z)=\boldsymbol{Y}(z)\cdot \boldsymbol{G}(z) \text{ with}$$

$$\boldsymbol{G}(z):=\begin{bmatrix} \boldsymbol{G_0}(z) \\ \boldsymbol{G_1}(z) \\ \vdots \\ \boldsymbol{G_{N-1}}(z) \end{bmatrix}=\begin{bmatrix} G_{0,0}(z) & G_{0,1}(z) & \ldots & G_{0,N-1}(z) \\ G_{1,0}(z) & \ldots & \ldots & \vdots \\ \vdots & \ldots & \ddots & \vdots \\ G_{N-1,0}(z) & \ldots & \ldots & G_{N-1,N-1}(z) \end{bmatrix}$$

with

$$G_{k,n}(z)=\sum_{m=0}^{\infty} g_k(mN+n)\, z^{-m}$$

This is the **synthesis polyphase matrix**. Observe that for this polyphase matrix, the indices for the subbands k and for the phase n are in reversed order compared to the analysis polyphase matrix.

In Conclusion: Again we turned the mathematically very complex operation of upsampling and synthesis filtering into a

mathematically very simple operation with the **multiplication** of the **subband vector** with the **polyphase matrix**!

$$\hat{\boldsymbol{X}}(z)=\boldsymbol{Y}(z)\cdot\boldsymbol{G}(z)$$

Perfect Reconstruction

Perfect reconstruction (PR) is defined as a reconstructed signal which is identical to the original signal except for a delay,

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

with some delay n_d at the original sampling rate. This delay usually results from our filtering and the downsampling and upsampling operations. To obtain PR we can simply take a look at the output of our synthesis filter bank,

$$\hat{\boldsymbol{X}}(z)=\boldsymbol{Y}(z)\cdot\boldsymbol{G}(z)=\boldsymbol{X}(z)\cdot\boldsymbol{H}(z)\cdot\boldsymbol{G}(z)$$

The structure of the polyphase analysis and synthesis filter bank can be seen also in the following structure,

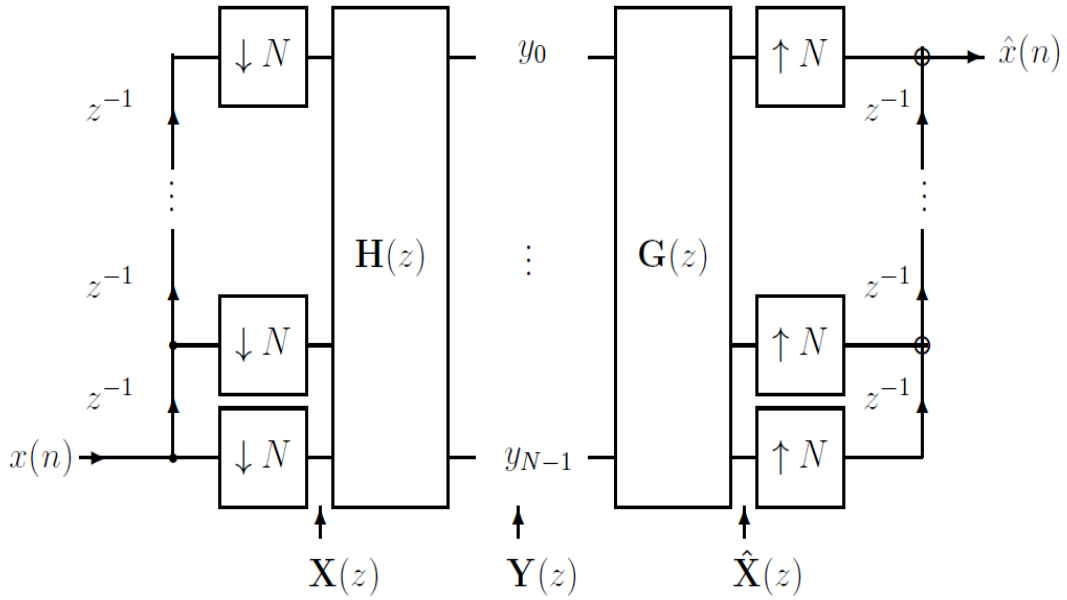

Figure 2.9: Polyphase representation of an N band filter bank with critical sampling. Observe that z^{-1} always means a delay by 1 sample, independent of the sampling rate.

The structure with the delays and downsamplers on the left of the analysis polyphase matrix converts a sequence of samples at the high sampling rate into a sequence of blocks at the low sampling rate ("blocking"), which is what out Python function "x2polyphase.py" is doing. Conversely, the structure to the right of the synthesis polyphase matrix converts a sequence of blocks at the low sampling rate into a sequence of samples at the high sampling rate ("de-blocking").

Here we can see that we obtain perfect reconstruction if we have the synthesis polyphase matrix as the inverse of the analysis polyphase times a delay by d blocks,

$$\boldsymbol{G}(z)=z^{-d}\boldsymbol{H}^{-1}(z)$$

(where d here is the delay at the downsampled

rate). This is basically again like in our block transform case. This is now the constraint for obtaining PR. The question is, how do we obtain filters for PR? How do we invert a polyphase matrix, containing the polynomials? How do we get "good" synthesis filters?
A simple approach is analog to **orthogonal** block transform matrices, where the inverse is simply the transpose matrix. There the analysis and synthesis filters are identical, except for the time reversal for the analysis filters.
The corresponding property of polyphase matrices is called "**para-unitary**" (see also: Vaidyanathan: Multirate Systems and Filter Banks). It is defined as,

$$\boldsymbol{H}^{-1}(z)=\boldsymbol{H}^{T}(z^{-1}) \qquad (1)$$

(The transposition is the case for real valued coefficients in the polyphase matrix. Otherwise we need conjugate transposition for the coeffients).
This definition is very similar to the definition for orthonormal matrices, except that we have the “ z^{-1} “ on the right hand side (which corresponds to a time-reversal).
The advantage (or rather: one of its advantages) of a polyphase matrix with this property is, that we don't need to explicitly compute its inverse, we just need to transpose

it and replace all z's by z^{-1} .

Observe: if our polynomials in the polyphase matrix **only have zero'th order** (only a constant, no z, only $z^0=1$), then the polyphase matrix is identical to a **transform matrix** (for instance our DCT4). We obtain this simple case if our filters are no longer than N, they fit into one block.