1 Polyphase filter bank : Maximal Decimation

Consider the following operation on the sequence $x[n]$,

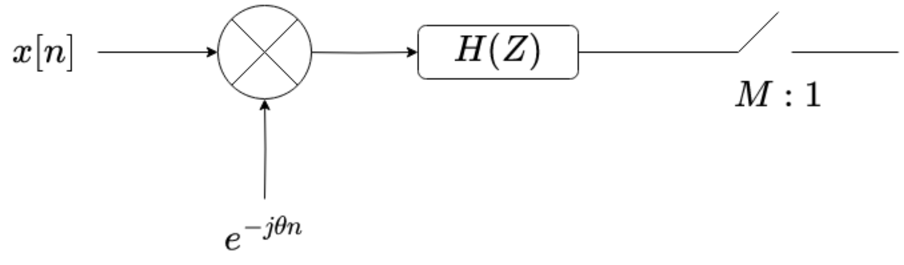


Figure 1: Downconvert, filter and then downsample.

1. The frequency band of the input centered at θ is first downconverted to baseband (this is signal speak for shifting the entire frequency spectrum of $x[n]$ to the left by θ).
2. Then, the downconverted input is passed through a filter $H(Z)$ a notation which simply represents

$$H(Z) = \sum_{n=0}^{\infty} h_n z^{-n}.$$

   In this notation, we multiply the z-transform of $x[n]$ by $H(Z)$ which in time domain simply implies convolution of $x[n]$ by the filter coefficients h_n.
3. Finally, the output of the filter is downsampled/subsampled by a factor of M. This is signalspeak for only keeping every M-th element in the input. Note that the filter is needed, otherwise, a naive subsampling causes aliasing effects in the spectral domain.

This downconvert, filter and then subsample pattern is routinely performed in the case of analyzing isolated frequency bands of the input signals. Therefore,

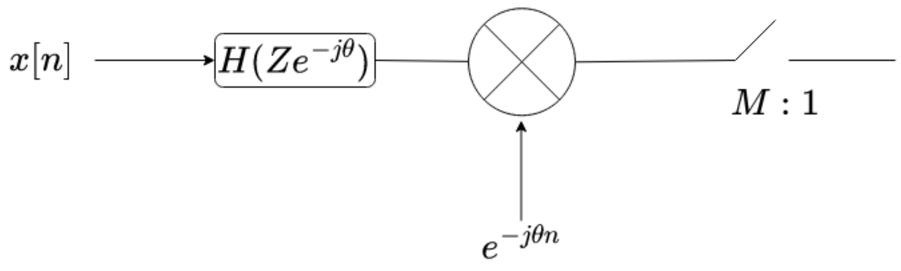


Figure 2: Equivalence theorem

it is of utmost improtance that this be done quickly. We explore the technique of polyphase decomposition of $H(Z)$ to do this. We first note the following:

$$\sum_{k=-\infty}^{\infty} x[k]e^{-j\theta k}h_{n-k} = e^{-j\theta n}\sum_{k} x[k]h_{n-k}e^{j\theta(n-k)}. \tag{1}$$

This is as if, the filter coefficients h_n have been replaced by $h_n e^{j\theta n}$. This implies the rearrangement shown in the figure in Figure 2. A result that is known as the equivalence theorem. We can then shift the multiplication by the complex sinusoid across the downsampler, and we get this: The final

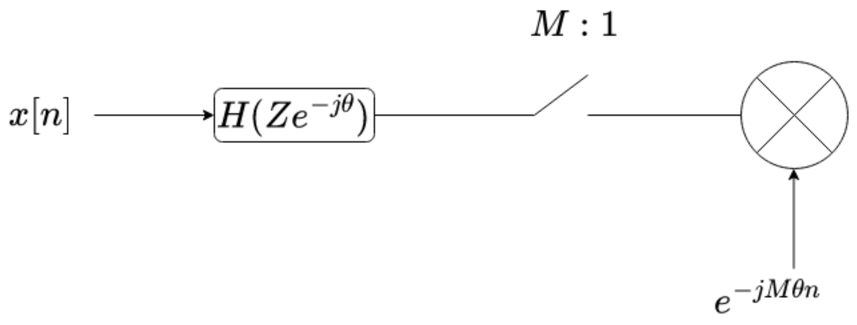


Figure 3: Equivalence theorem with downsample removed

multiplication can be removed with the assumption that

$$M\theta = 2\pi l \tag{2}$$

so that we are only interested in center frequencies that are integer multiples of the downsampling rate. Thus, we need to analyze the situation shown in Figure 4.

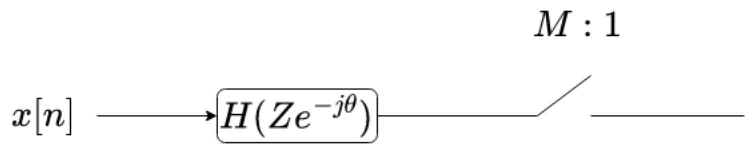


Figure 4: Equivalence theorem with downsample removed

We next note that as written here, we are computing the convolution of the filter with the input at the full rate, i.e., we are computing all the values of the output, but discarding every M-th value. This is wasteful, and the Polyphase Channelizer algorithm is designed to remedy this.

Instead of thinking about delays and polyphase components etc., let us directly compute every M-th output of the filtering operation, and see if we observe any patterns. The Z-transforms of the input $x[n]$ (which we assume starts at time instant $n = 1$) is simply

$$X(z) = x_1 z^{-1} + x_2 z^{-2} + \cdots \tag{3}$$

while the filter Z-transform is (note that we are taking downconversion into account from the get-go, in the following, note $\xi = e^{j\theta}$)

$$H(Ze^{-j\theta}) = h_0 + h_1 \xi z^{-1} + h_2 \xi^2 z^{-2} + \cdots$$

Now, simply multiply $H(Z\xi^{-1})$ with $X(Z)$ to obtain

$$F(Z,\xi) = \Bigg(x_1 Z^{-1} + x_2 Z^{-2} + \cdots \Bigg) \cdot \Bigg(h_0 + h_1 \xi Z^{-1} + h_2 \xi^2 Z^{-2} + \cdots \Bigg). \tag{4}$$

Since we are interested in every M-th output of this product, it makes sense to group together all those filter coefficients whose indices leave the same remainder when divided by M. Thus, we decompose the product $F(Z,\xi)$ as

$$F(Z,\xi) = F_0(Z,\xi) + F_1(Z,\xi) + \cdots + F_{M-1}(Z,\xi) \tag{5}$$

Here, the notation $F_j(Z,\xi)$ indicates the portion of F that depends on filter coefficients h_k such that $k = pM + j$ for some p. Note futher that, at the output due to the downsamplig, we are only interested in terms in the above that involve Z^{-lM} for l an integer. We also note in passing $\xi^M = 1$. Here is the structure of the first few $F_j(Z,\xi)[\xi^{Mk}]$ (a notation that means we only keep terms involving ξ^{Mk}),

$$\begin{aligned}
F_0(Z,\xi)[\xi^{Mk}] &= (h_0x_M)Z^{-M} + (h_Mx_M + h_0x_{2M})Z^{-2M} \\
&\quad + (h_0x_{3M} + h_Mx_{2M} + h_{2M}x_M)z^{-3M} + \cdots \\
&= G_0(Z) \\
F_1(Z,\xi)[\xi^{Mk}] &= (\xi h_1x_{M-1})Z^{-M} + \xi(h_{M+1}x_{M-1} + h_1x_{2M-1})Z^{-2M} \\
&\quad + \xi(h_1x_{3M-1} + h_{M+1}x_{2M-1} + h_{2M+1}x_{M-1})z^{-3M} + \cdots \\
&= \xi G_1(Z) \\
F_2(Z,\xi)[\xi^{Mk}] &= \xi^2(h_2x_{M-2})Z^{-M} + \xi^2(h_{M+2}x_{M-2} + h_2x_{2M-2})Z^{-2M} \\
&\quad + \xi^2(h_2x_{3M-2} + h_{M+2}x_{2M-2} + h_{2M+2}x_{M-2})z^{-3M}) + \cdots \\
&= \xi^2G_2(Z)
\end{aligned}$$

We observe,

1. The j-th component F_j depends on ξ^j solely as the factor ξ^j multiplied by a quantity $G_j(Z)$ that does not depend on ξ.

2. To compute $G_j(Z)$, simply take the input samples $x_{M-j}, x_{2M-j}, x_{3M-j}, \cdots$, and perform the convolution of these samples with the filter coefficients $h_j, h_{M+j}, h_{2M+j} \cdots$.

3. Note that before these observations, we were computing convolutions between filter and input at full rate. But now, we only need to perform one convolution computation every M samples of the input. This represents the gains we have made with this observation.

4. Thus, the Z-transform of the output at a given value of the center frequency, set by $\xi = e^{j\theta}$, is simply

$$\sum_{j=0}^{M-1} \xi^j G_j(Z)$$

5. This represents the output at a given value of the center frequency. What if we want all of them? This structure suggests that the answer is to compute $G_j(Z)$ for each j, and expand

$$G_j(Z) = \sum_k f_{jk} Z^{-kM}$$

The actual output as we saw above has the Z-transform $\sum_j \xi^j G_j(Z)$. This suggests that the output time series in each channel is simply given by

$$T_k = \sum_j \xi^j f_{jk}$$

This is nothing but the coloumnwise IFFT of the matrix f_{jk}.

Offline Channelizer : This is the computation one needs to perform to get disjoint channels from the input array $x[n]$ given filter coefficients h_n. One first arranges the input array $x[n]$ in the following layout in a buffer array

$$X = \begin{pmatrix} x_M & x_{2M} & x_{3M} & \cdots \\ x_{M-1} & x_{2M-1} & x_{3M-1} & \cdots \\ x_{M-2} & x_{2M-2} & x_{3M-2} & \cdots \\ & \vdots & & \\ x_1 & x_{M+1} & x_{2M+1} & \cdots \end{pmatrix}$$

This step will involve copy of the input into the layout given above. Successive input elements are mapped onto non-contiguous portions of the buffer array in the above layout, and therefore, this step is a time consuming one. Filter coefficients will be stored in the polyphase layout

$$H = \begin{pmatrix} h_0 & h_M & h_{2M} & \cdots \\ h_1 & h_{M+1} & h_{2M+1} & \cdots \\ h_2 & h_{M+2} & h_{2M+2} & \cdots \\ & \vdots & & \\ h_{M-1} & h_{2M-1} & h_{3M-1} & \cdots \end{pmatrix}$$

The output is computed simply as

$$\text{Column-wise IFFT}\Bigg(\text{Row-wise Convolve}\Bigg(X, H\Bigg)\Bigg)$$

For the off-line version, we use FFT based convolution algorithms as they would be faster than multiplcation based convolution.

Online Channelizer: One can also construct a streaming channelizer by a simple modification of the algorithm above. By a streaming channelizer, we mean that the output in each channel is computed one sample at a time.

Since there are M channels, the channelizer has to wait for M input samples to compute one output sample in each channel. This is simply a consequence of the downsampling operation. (One can infact start computing new outputs given $M/2$ samples actually, but we don't do that micro-optimization here).

Call the maximum number of (non-zero) filter coefficients per channel as the number of taps per channel (denoted as T). Then, one can construct a streaming channelizer by maintaining M circular buffers of size T each, one for each polyphase component of the filter.

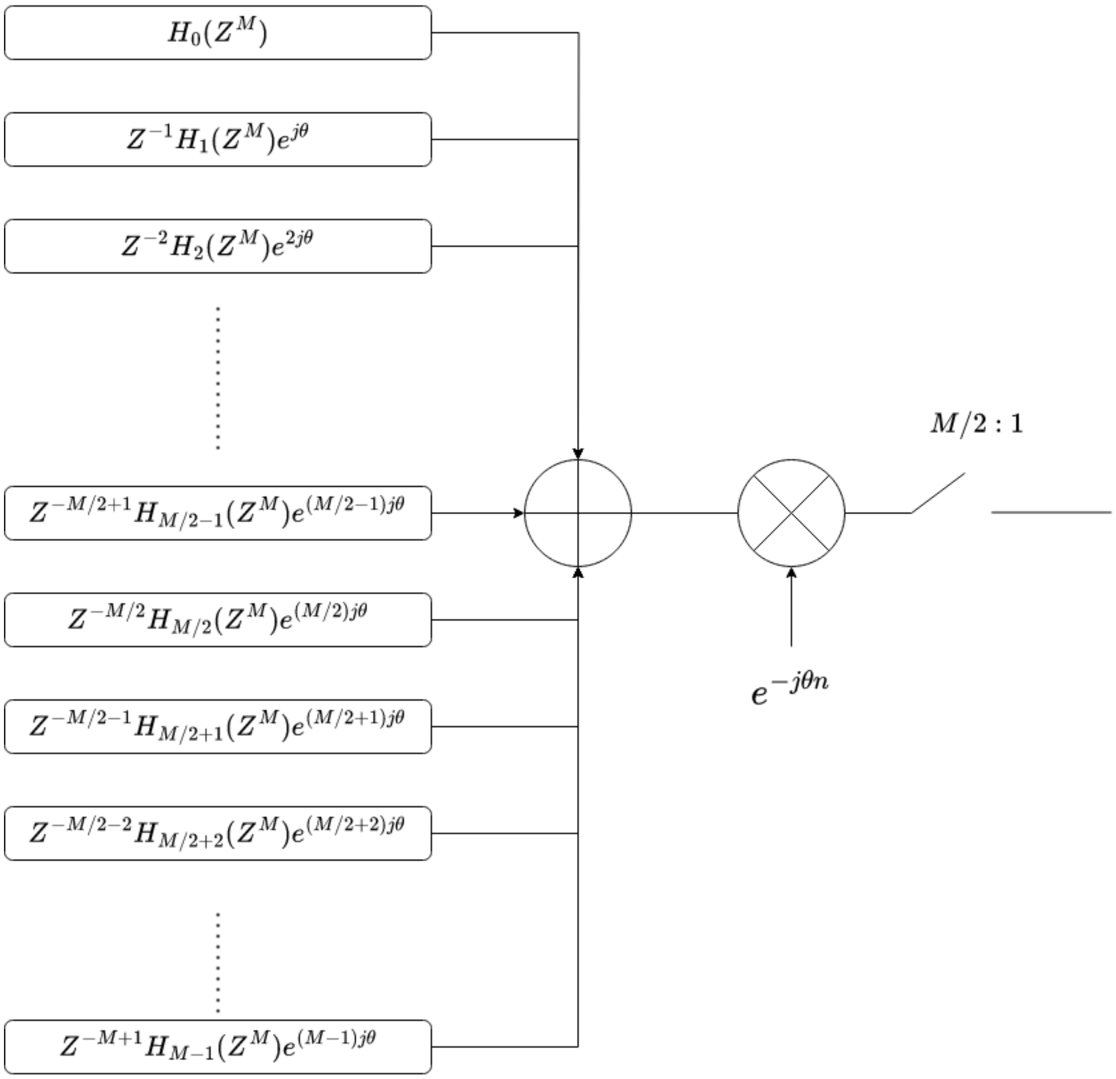
The first sample in each batch of M new samples is sent to the last polyphase component, the second to the second-last polyphase component, and so on. The insertion of these new samples updates the buffers. Then, compute the convolution of each buffer with the corresponding polyphase filter component by simple addition and multiplication. This would be a number for each channel. Collect them in a vector and then take the IFFT of the resulting vector. That would be the updated output from the channelizer given the M inputs.

2 Polyphase Filterbank: Non-maximal decimation

In the previous version of the channelizer, the bandwidth of each channel was the same as the spacing between channels. This is only good for analyzing static bands where we know the activity in advance.

For bands with dynamic activity, we would like to combine multiple channels together. For this to happen, the filter bank needs to be a near perfect reconstruction filter bank, and this is only possible is channel spacing is smaller than the bandwidth of each channel.

An easy way to get this, is to keep M channels at the output, but increasing the sample rate of each channel by a factor of two, as in the diagram below,



$H_0(Z^M)$
$Z^{-1}H_1(Z^M)e^{j\theta}$
$Z^{-2}H_2(Z^M)e^{2j\theta}$
$Z^{-M/2+1}H_{M/2-1}(Z^M)e^{(M/2-1)j\theta}$
$Z^{-M/2}H_{M/2}(Z^M)e^{(M/2)j\theta}$
$Z^{-M/2-1}H_{M/2+1}(Z^M)e^{(M/2+1)j\theta}$
$Z^{-M/2-2}H_{M/2+2}(Z^M)e^{(M/2+2)j\theta}$
$Z^{-M+1}H_{M-1}(Z^M)e^{(M-1)j\theta}$
$M/2:1$
$e^{-j\theta n}$

Figure 5: Non-maximal decimated channelizer construction

Note that here we have written down the structure we found in the previous section for the maximally decimated channelizer. We push the downsampler to the left, and use the Noble identity (the previous section can be thought of as roughly a derivation of the Noble identity for downsampling), and get the following arrangement,

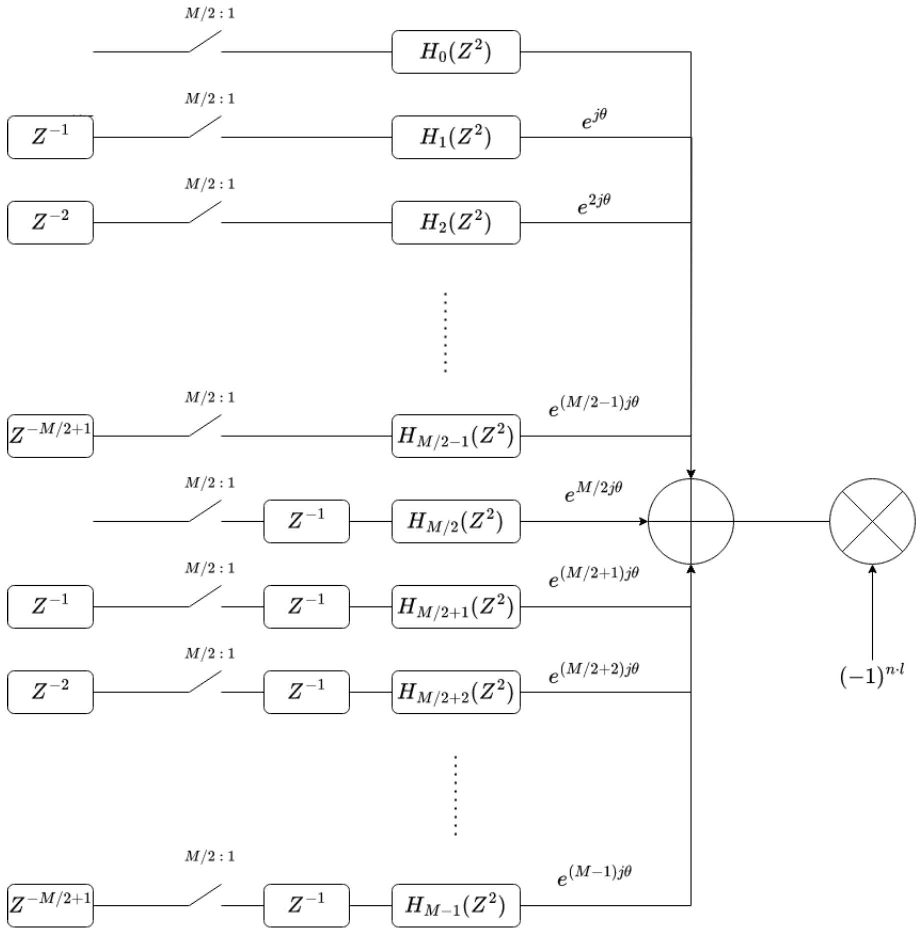


Figure 6: Non-maximal decimated channelizer construction. Note that $\theta = \frac{2\pi l}{M}$, i.e., the center frequency of the output channel is an integer multiple of the downsampled rate of the maximally decimated channelizer. Since the downsampling in this new case is half of the maximal possible, an overall multiplication by $(-1)^n$ remains for odd channels.

The delay and downsample on the left side of the figure mean that the input $X = x_0, x_1, \cdots$, needs to be arranged in the layout

$$X = \begin{pmatrix} x_{M/2} & x_M & x_{3M/2} & \cdots \\ x_{M/2-1} & x_{M-1} & x_{3M/2-1} & \cdots \\ x_{M/2-2} & x_{M-2} & x_{3M/2-2} & \cdots \\ & \vdots & & \\ x_2 & x_{M/2+2} & x_{M+2} & \cdots \\ x_1 & x_{M/2+1} & x_{M+1} & \cdots \\ x_{M/2} & x_M & x_{3M/2} & \cdots \\ x_{M/2-1} & x_{M-1} & x_{3M/2-1} & \cdots \\ x_{M/2-2} & x_{M-2} & x_{3M/2-2} & \cdots \\ & \vdots & & \\ x_2 & x_{M/2+2} & x_{M+2} & \cdots \\ x_1 & x_{M/2+1} & x_{M+1} & \cdots \end{pmatrix}$$

The filter coefficients are in the layout (the zeros in the first $M/2$ rows correspond to taking the filters $H_k(z^2)$), whereas, the delays in the last $M/2$ rows are due to the additional Z^{-1} delay in the last $M/2$ rows with the filters in the diagram above:

$$H = \begin{pmatrix} h_0 & 0 & h_M & 0 & h_{2M} & \cdots & \\ h_1 & 0 & h_{M+1} & 0 & h_{2M+1} & \cdots & \\ h_2 & 0 & h_{M+2} & 0 & h_{2M+2} & \cdots & \\ & \vdots & & & & & \\ h_{M/2-1} & 0 & h_{3M/2-1} & 0 & h_{5M/2-1} & \cdots & \\ 0 & h_{M/2} & 0 & h_{3M/2} & 0 & h_{5M/2} & \cdots \\ 0 & h_{M/2+1} & 0 & h_{3M/2+1} & 0 & h_{5M/2+1} & \cdots \\ 0 & h_{M/2+2} & 0 & h_{3M/2+2} & 0 & h_{5M/2+2} & \cdots \\ & \vdots & & & & & \\ 0 & h_{M-1} & 0 & h_{2M-1} & 0 & h_{3M-1} & \cdots \end{pmatrix}$$

Finally, define the matrix

$$T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & \cdots \\ 1 & -1 & 1 & -1 & 1 & \cdots \\ 1 & 1 & 1 & 1 & 1 & \cdots \\ 1 & -1 & 1 & -1 & 1 & \cdots \\ & \vdots & & & & \end{pmatrix}$$

Then, the output is computed as

$$Y = T \odot \text{ColumnWise IFFT}(\text{Row-wise Convolve}(X, H)) \tag{6}$$

where $\odot$ represents Hadamard elementwise multiplication. The output Y contains M channels, and each channel is sampled at $\frac{2f_s}{M}$ sample rate, with the spacing between the samples $\frac{f_s}{M}$ if the input sample rate is f_s.