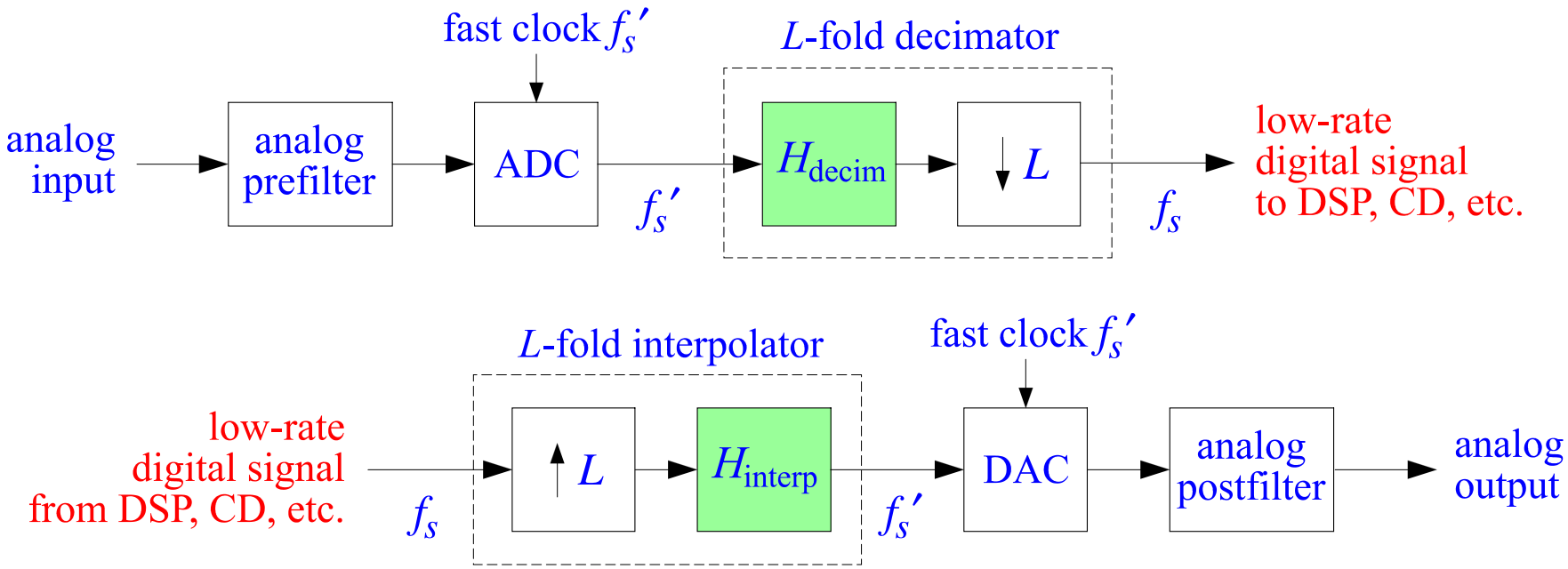
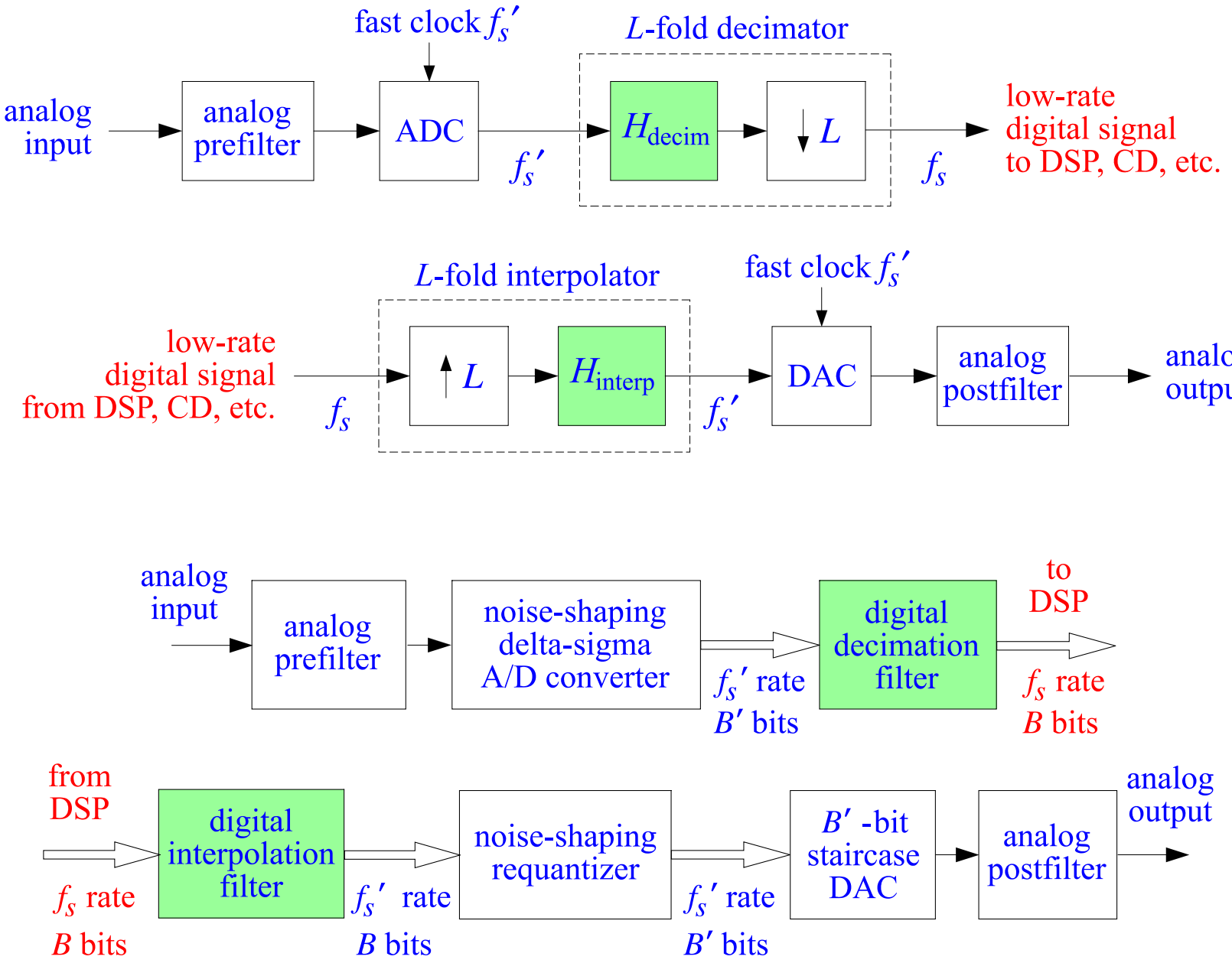
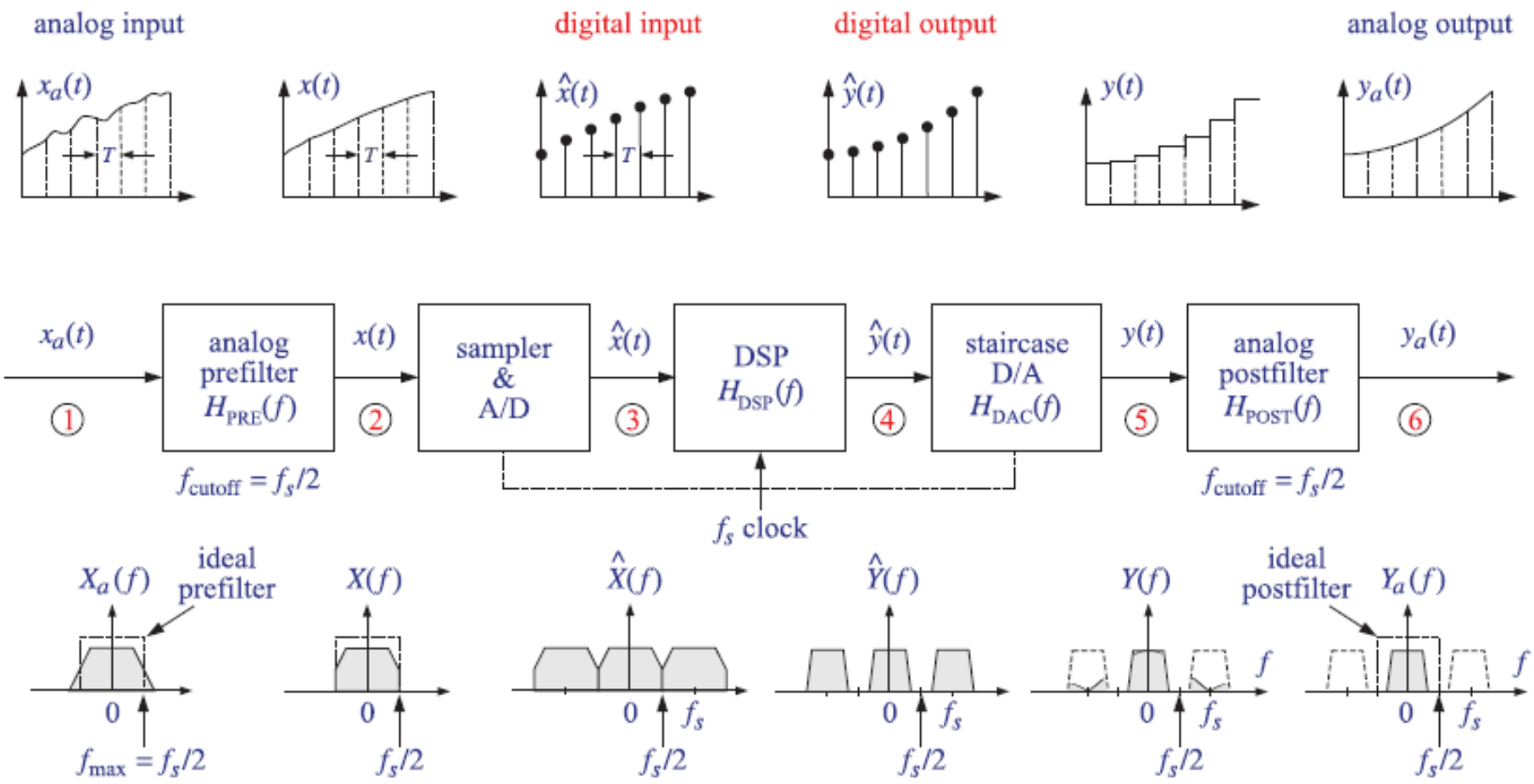
DSA – May 3, 2021

Topics: Multirate signal processing, interpolation, decimation, oversampling, interpolation filter design, direct & polyphase realizations, Kaiser designs, multistage designs, linear & hold interpolators, design examples, interpolator design with DAC & postfilter equalization, Bessel postfilters, decimation filter design, sample rate converters, noise-shaping delta-sigma quantizers, oversampled DSP systems.




fast clock f_s'
L-fold decimator
analog input
analog prefilter
ADC
f_s'
H_{decim}
$\downarrow L$
f_s
low-rate digital signal to DSP, CD, etc.
L-fold interpolator
fast clock f_s'
low-rate digital signal from DSP, CD, etc.
f_s
$\uparrow L$
H_{interp}
f_s'
DAC
analog postfilter
analog output
analog input
analog prefilter
noise-shaping delta-sigma A/D converter
f_s' rate
B' bits
digital decimation filter
to DSP
f_s rate
B bits
from DSP
f_s rate
B bits
digital interpolation filter
f_s' rate
B bits
noise-shaping requantizer
f_s' rate
B' bits
B' -bit staircase DAC
analog postfilter
analog output

conventional DSP system with ideal specifications

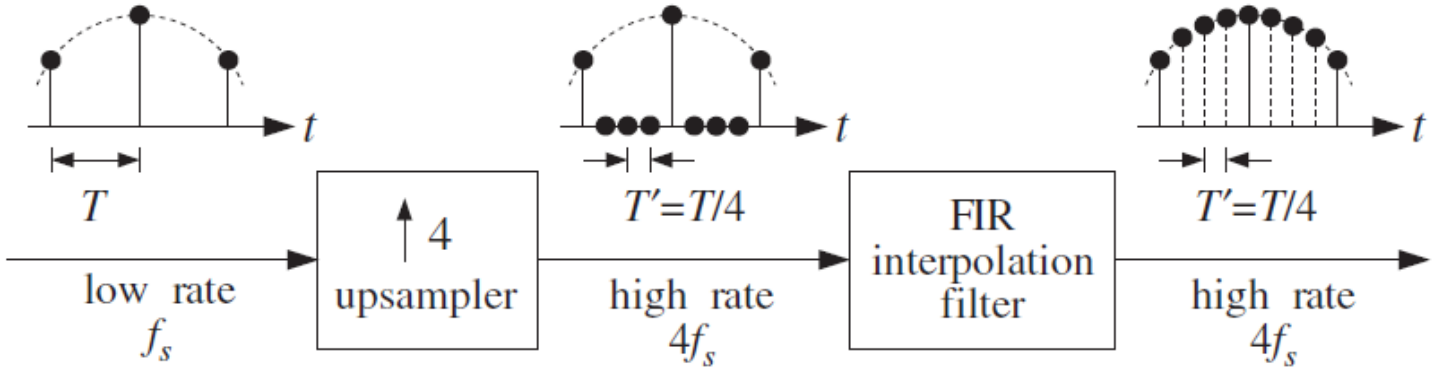


Interpolation and Oversampling

Sampling rate changes are useful in many applications, such as interconnecting digital processing systems operating at different rates. Sampling rate increase is accomplished by interpolation, that is, the process of inserting additional samples between the original low-rate samples.

The inserted, or interpolated, samples are *calculated* by an FIR digital filter (IIR filters can also be used, but are less common in practice.)

This is illustrated below for the case of a 4-fold interpolator which increases the sampling rate by a factor of four, that is, $f_s{}' = 4f_s$.



With respect to the fast time scale, the low-rate samples may be thought of as being separated by three zero samples. The 4-fold rate expander or upsampler simply inserts three zero samples for every low-rate sample. The job of the FIR filter is to replace the three zeros by the calculated interpolated values.

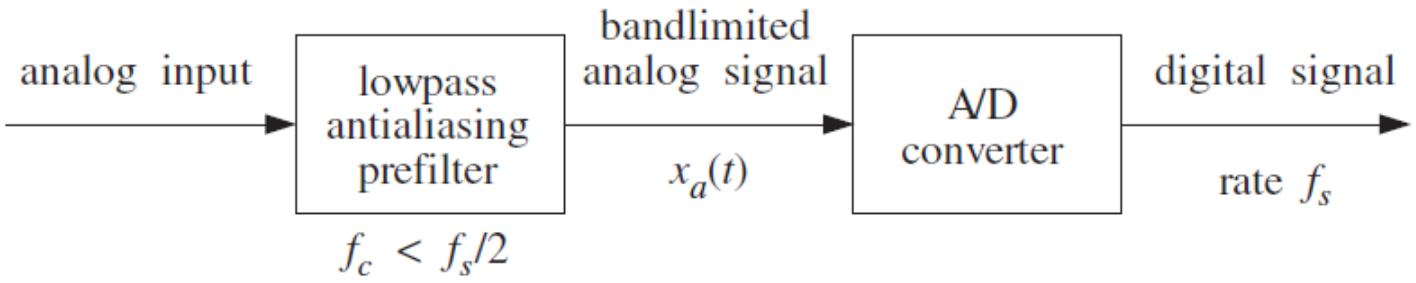
The interpolating filter is sometimes called an oversampling digital filter because it operates at the fast rate $4f_s$. However, because only one out of every four input samples is non-zero, the required filtering operations may be rearranged in such a way as to operate only on the low-rate samples, thus, effectively reducing the computational requirements of the filter—by a factor of four in this case.

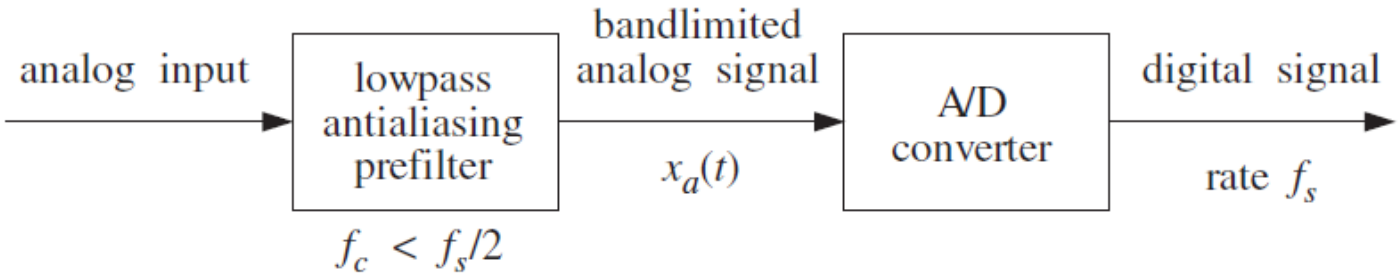
This is accomplished by replacing the high-rate interpolating FIR filter by four shorter FIR subfilters, known as polyphase filters, operating at the *low* rate f_s. The length of each subfilter is one-quarter that of the original filter. Because each low-rate input sample generates four high-rate interpolated outputs (itself and three others), each of the four low-rate subfilters is dedicated to computing only one of the four outputs.

Such realization is computationally efficient and lends itself naturally to parallel multiprocessor hardware implementations in which a different DSP chip may be used to implement each subfilter.

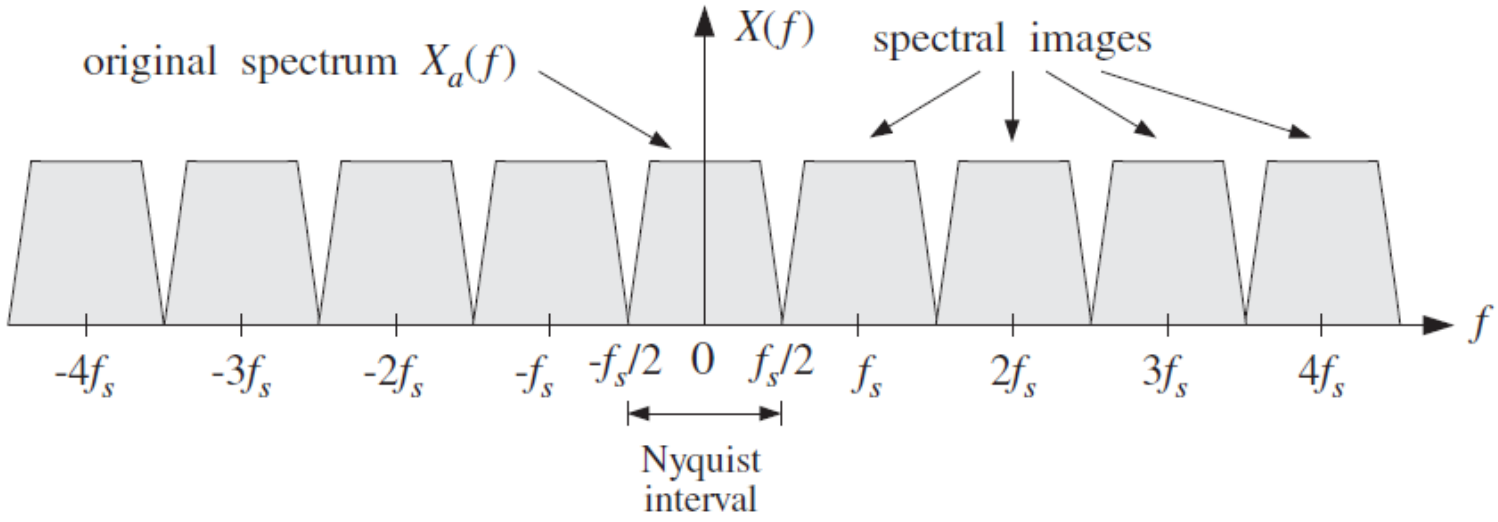
An interesting application of interpolation is the use of oversampling digital filters in audio playback systems, such as smartphones, CD, MP3, or DAT players, where they help to alleviate the need for high-quality analog anti-image postfilters in the playback system. Moreover, each high-rate sample can be requantized without loss of quality to fewer number of bits (even as low as 1 bit per sample) using appropriate noise shaping quantizers, thus, trading off bits for samples and simplifying the structure of the analog part of the playback system.

To understand the motivation behind this application, consider an analog signal sampled at a rate f_s, such as 44.1 kHz for digital audio. The analog signal is prefiltered by an analog lowpass *antialiasing prefilter* having cut-off frequency $f_c \leq f_s/2$ and then sampled at rate f_s and quantized. This operation is shown below.

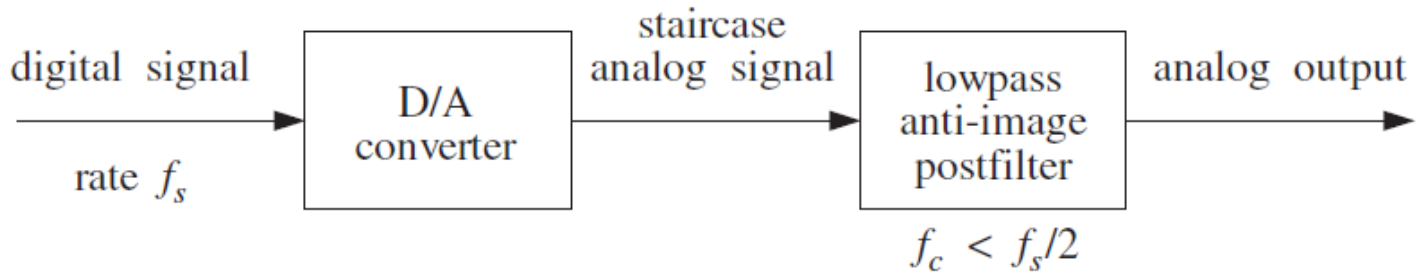




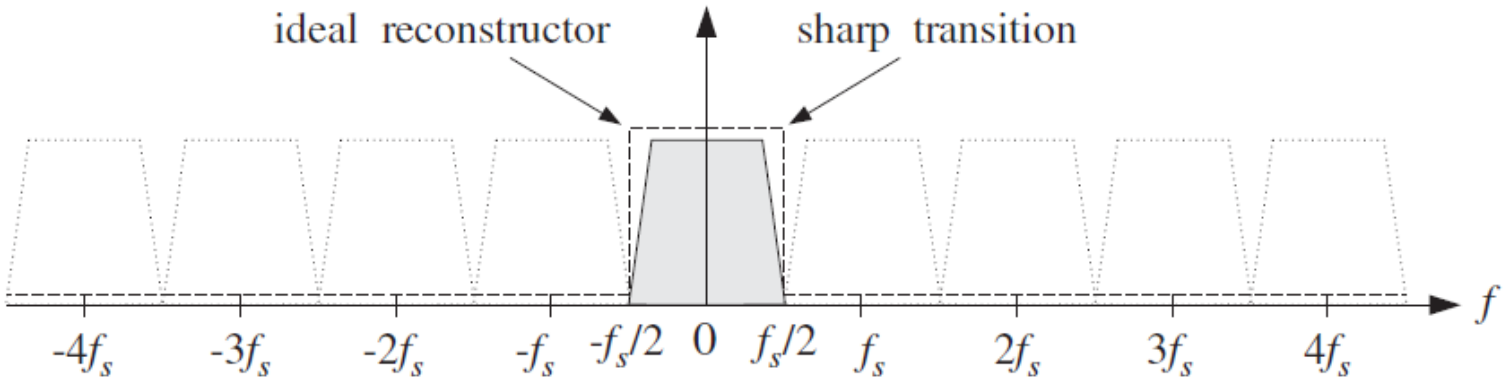
The prefilter ensures that the spectral images generated by the sampling process at integral multiples of f_s do not overlap, as required by the sampling theorem. This is shown below (we ignore here the scaling factor $1/T$).



After digital processing, the sampled signal is reconstructed back to analog form by a D/A staircase reconstructor, followed by an analog anti-image lowpass postfilter with effective cutoff $f_s/2$, as below.

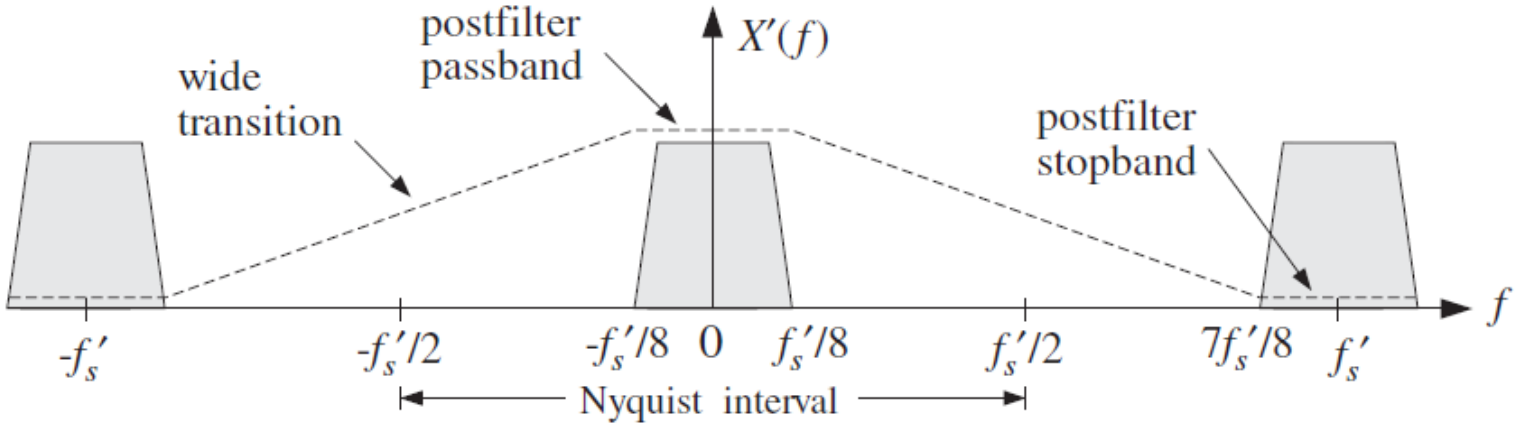


The D/A converter, with its typical $\sin x/x$ response, removes the spectral images partially; the postfilter completes their removal. The combination of the staircase DAC and the postfilter emulates the *ideal* reconstructing analog filter. The ideal reconstructor is a lowpass filter with cutoff the Nyquist frequency $f_s/2$. It has a very sharp transition between its passband, that is, the Nyquist interval, and its stopband, as shown below.



In hi-fi applications such as digital audio, to maintain high quality in the resulting reconstructed analog signal, a very high quality analog postfilter is required, which may be expensive.

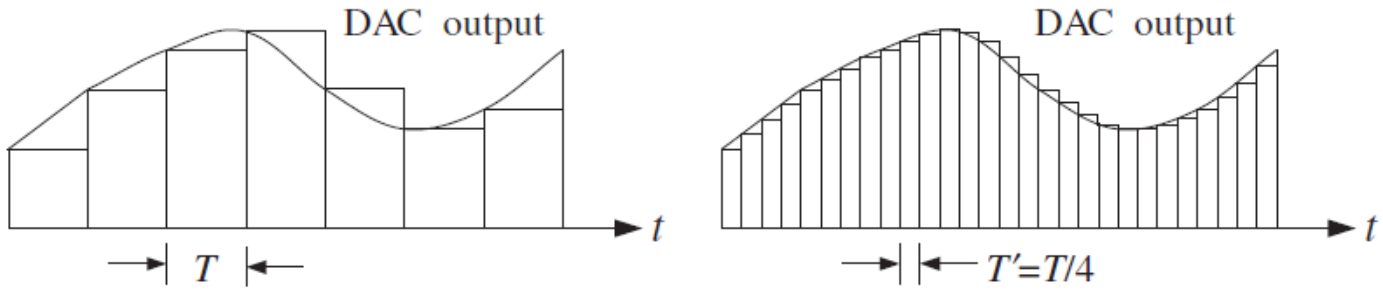
One way to alleviate the need for a high quality postfilter is to increase the sampling rate. This would cause the spectral images to be more widely separated and, therefore, require a less stringent, simpler, lowpass postfilter. This is depicted below, for a new sampling rate that is four times higher than required, $f_s' = 4f_s$.



The *passband* of the postfilter extends up to $f_{\text{pass}} = f_s'/8 = f_s/2$, but its *stopband* need only begin at $f_{\text{stop}} = f_s' - f_s'/8 = 7f_s'/8$.

It is this wide transition region between passband and stopband that allows the use of a less stringent postfilter. For example, in oversampled digital audio applications, simple third-order Butterworth or Bessel analog postfilters are used.

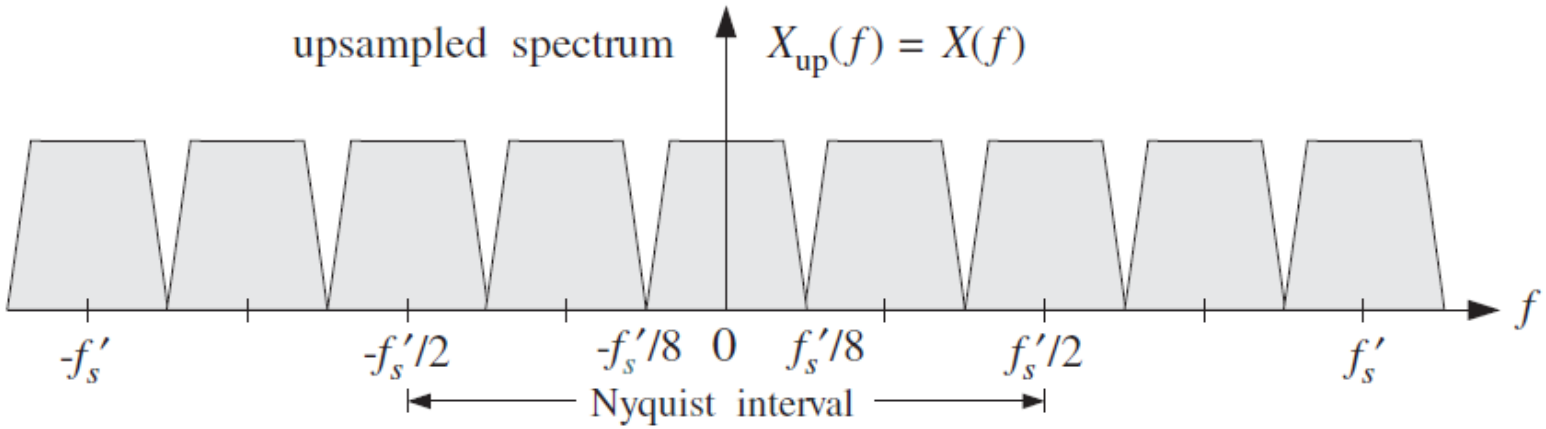
The same conclusion can also be drawn in the time domain. The figure below shows the staircase output of the D/A converter for the two sampling rates f_s and $f_s' = 4f_s$. It is evident that the higher the sampling rate, the more closely the staircase output approximates the true signal, and the easier it is for the postfilter to smooth out the staircase levels.



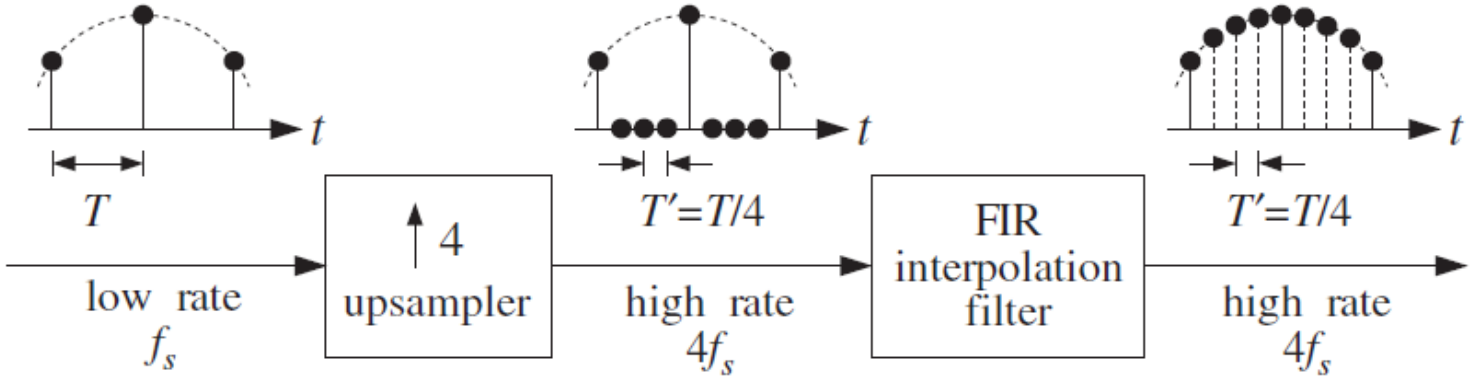
The above approach, however, is impractical because it requires the actual *resampling* of the analog signal at the higher rate f_s'. For example, in a CD player or in an MP3 file, the low rate samples are already stored at the prescribed rate of 44.1 kHz and the audio signal cannot be resampled.

The philosophy of oversampling is to increase the sampling rate *digitally* using an interpolation filter which operates only on the available *low-rate* input samples.

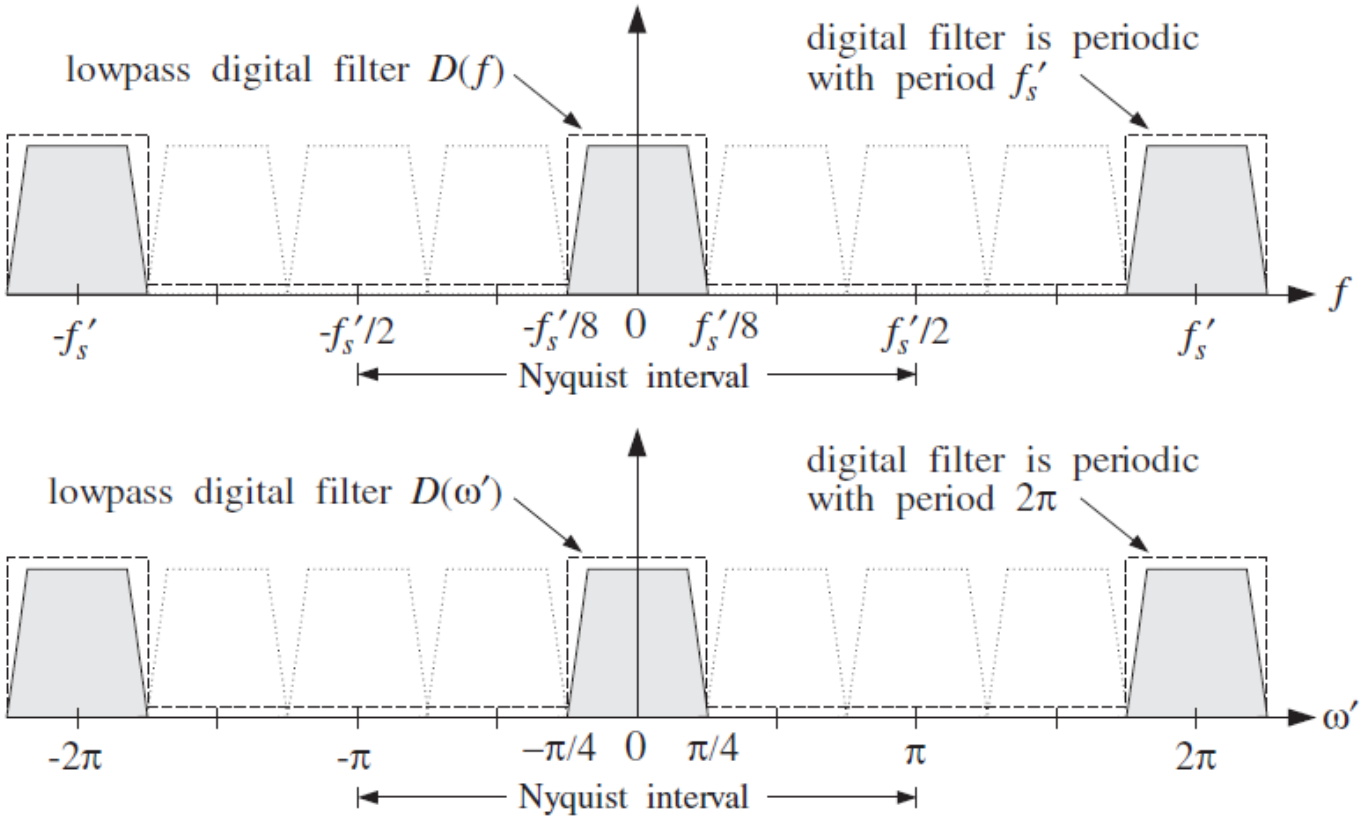
With respect to the new rate f_s' and new Nyquist interval $[-f_s'/2, f_s'/2]$, the spectrum of the low-rate samples will be as shown below, assuming that the initial sampling process was perfect and that the f_s-replicas are not overlapping.



This is also the spectrum of the high-rate upsampled signal at the output of the rate expander.



A digital lowpass FIR filter with cutoff frequency $f_s'/8$ and operating at the high rate f_s', would eliminate the three spectral replicas that lie between replicas at multiples of f_s', as shown below, and the resulting spectrum would be as if the signal that had been sampled at the high rate f_s',

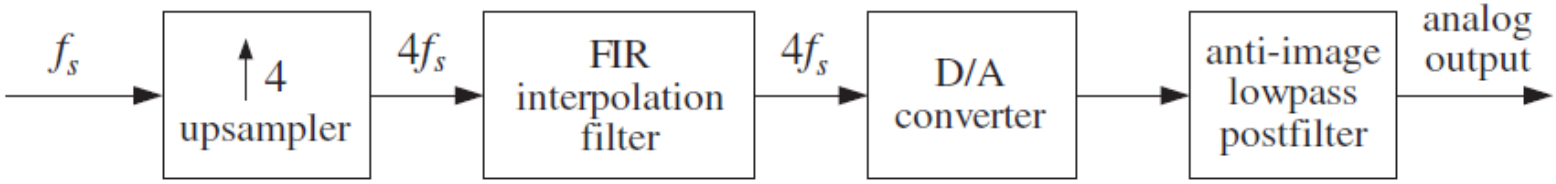


The digital filter's response is plotted here both with respect to the physical frequency f in Hz, and the corresponding digital frequency normalized to the higher rate f_s', that is, $\omega' = 2\pi f / f_s'$, in radians/sample.

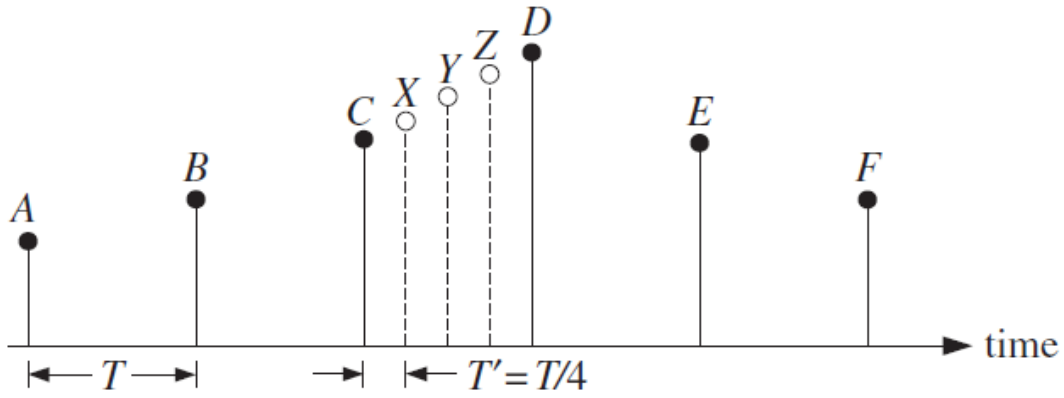
The digital filter, being periodic in f with period f_s', cannot remove the spectral replicas that are centered at integral multiples of f_s'. Those are removed later by the D/A reconstructor and the anti-image analog postfilter.

In summary, a substantial part of the analog reconstruction process is accomplished by DSP methods, that is, using a digital oversampling filter to remove several adjacent spectral replicas and thereby easing the requirements of the analog postfilter. The required sharp transition characteristics of the overall reconstructor are provided by the digital filter.

Thus, the high-quality *analog* postfilter is traded off for a high-quality *digital* filter operating at a higher sampling rate. The overall system is depicted below.



How does an interpolation filter operate in the time domain and calculate the missing signal values between low-rate samples? To illustrate the type of operations it must carry out, consider a 4-fold interpolator and a set of six successive low-rate samples $\{A, B, C, D, E, F\}$ as shown below.

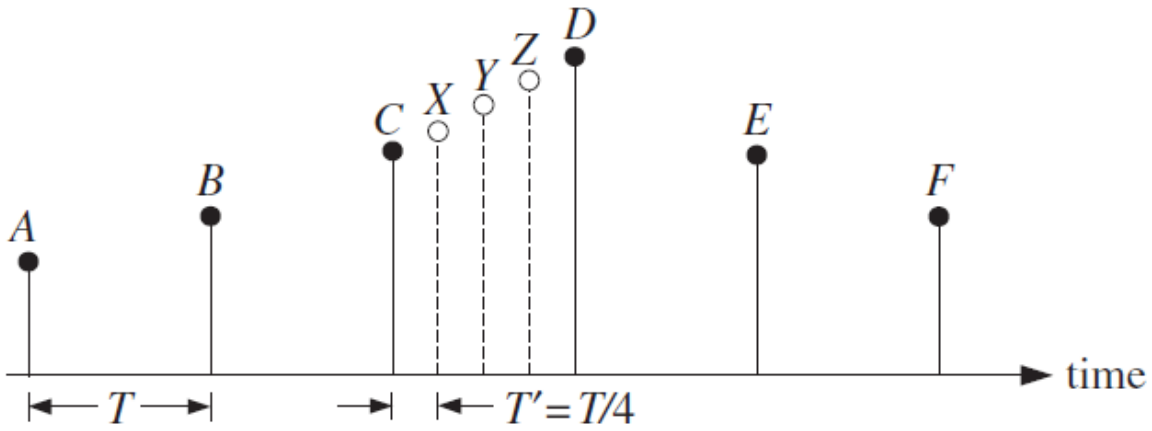


The filter calculates three intermediate samples, such as $\{X, Y, Z\}$, between any two low-rate samples by forming *linear combinations* of the surrounding low-rate samples.

Depending on the type of interpolator and desired quality of the calculated values, several different ways of calculating $\{X, Y, Z\}$ are possible. For example, the simplest one is to keep the value of the previous sample C constant throughout the sampling interval and define:

$$X = Y = Z = C$$

This choice corresponds to the so-called hold interpolator.



Another simple possibility is to interpolate linearly between samples $\{C, D\}$ calculating $\{X, Y, Z\}$ as follows:

$$X = 0.75C + 0.25D$$

$$Y = 0.50C + 0.50D$$

$$Z = 0.25C + 0.75D$$

Indeed, the straight line connecting C and D is parametrized as

$$C + \frac{D - C}{T}\, t$$

for $0 \leq t \leq T$. Setting $t = T'$, $2T'$, $3T'$, with $T' = T/4$, gives the above expressions for $\{X, Y, Z\}$.

For more accurate interpolation, more surrounding samples must be taken into account. For example, using the four samples $\{A, B, C, D\}$, we have:

$$X = -0.18B + 0.90C + 0.30D - 0.13E$$

$$Y = -0.21B + 0.64C + 0.64D - 0.21E$$

$$Z = -0.13B + 0.30C + 0.90D - 0.18E$$

corresponding to a length-17 FIR approximation to the ideal interpolation filter (this is derived later.)

Similarly, a length-25 approximation to the ideal interpolator uses six surrounding low-rate samples $\{A, B, C, D, E, F\}$, as follows:

$$X = 0.10A - 0.18B + 0.90C + 0.30D - 0.13E + 0.08F$$

$$Y = 0.13A - 0.21B + 0.64C + 0.64D - 0.21E + 0.13F$$

$$Z = 0.08A - 0.13B + 0.30C + 0.90D - 0.18E + 0.10F$$

In general, the more the surrounding samples, the more accurate the calculated values. It is not unsual to use 20–30 surrounding low-rate samples.

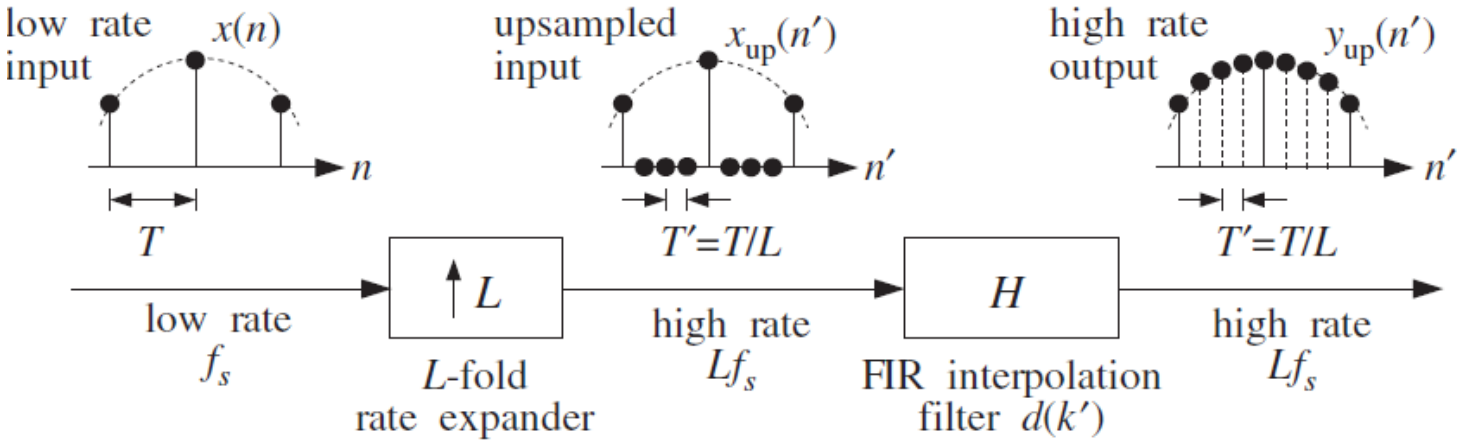
The above interpolation expressions do not quite look like the convolutional equations of linear filtering. They are special cases of the polyphase realizations of the interpolation filters and are equivalent to convolution.

They will be discussed in detail below, where starting with the frequency domain specifications of the inteprolation filter, its impulse response and corresponding direct and polyphase realization forms are derived.

Interpolation Filter Design

Direct Form

Consider the general case of an L-fold interpolator, which increases the sampling rate by a factor of L, that is, $f_s' = Lf_s$. The L-fold rate expander inserts $L-1$ zeros between adjacent low-rate samples and the corresponding $L-1$ interpolated values are calculated by an FIR digital filter operating at the high rate Lf_s.



Let $x(n)$ denote the low-rate samples that are input to the rate expander and let $x_{\text{up}}(n')$ be its high-rate output, consisting of the low-rate samples separated by $L-1$ zeros. With respect to the high-rate time index n', the low-rate samples occur every L high-rate ones, that is, at integral multiples of L, $n' = nL$,

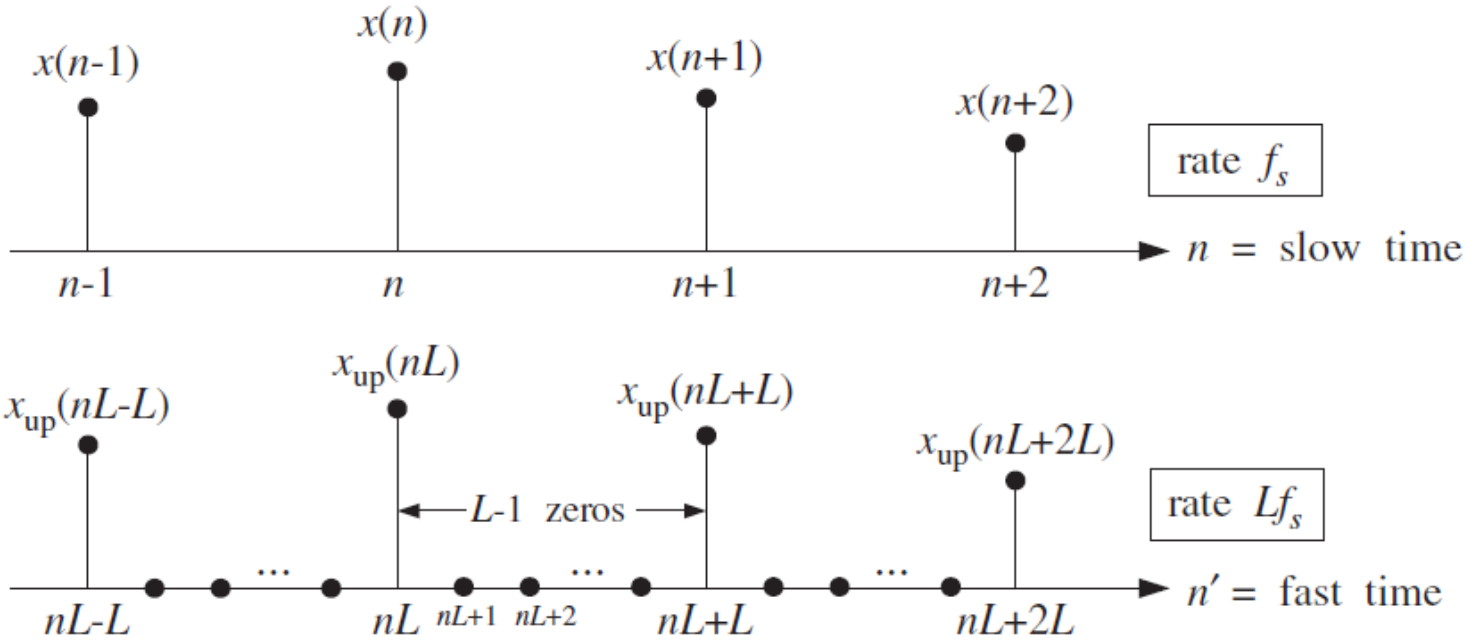
$$x_{\text{up}}(nL) = x(n)$$

The $L-1$ intermediate samples between $x_{\text{up}}(nL)$ and $x_{\text{up}}(nL+L)$ are zero:

$$x_{\text{up}}(nL+i) = 0, \qquad i = 1, 2, \ldots, L-1$$

This is shown below. More compactly, the upsampled signal $x_{\text{up}}(n')$ can be defined with respect to the high-rate time index n' by:

$$\boxed{x_{\text{up}}(n') = \begin{cases} x(n), & \text{if } n' = nL \\ 0, & \text{otherwise} \end{cases}}$$



Given an arbitrary value of the high-rate index n', we can always write it *uniquely* in the form,

$$n' = nL + i$$

where i is restricted to the range of values, $i = 0, 1, \ldots, L-1$. The integers n and i are the quotient and remainder of the division of n' by L.

Intuitively, this means that n' will either fall exactly on a low-rate sample (when $i = 0$), or will fall strictly between two of them ($i \neq 0$). Using $T = LT'$, we find the absolute time in seconds corresponding to n'

$$t = n'T' = nLT' + iT' = nT + iT'$$

that is, it will be offset from a low-rate sampling time by i high-rate sampling units T'. The interpolated values must be computed at these times.

The ideal L-fold interpolation filter is a lowpass filter, operating at the fast rate ${f_s}'$, with cutoff frequency equal to the *low-rate* Nyquist frequency $f_c = f_s/2$, or in terms of ${f_s}'$,

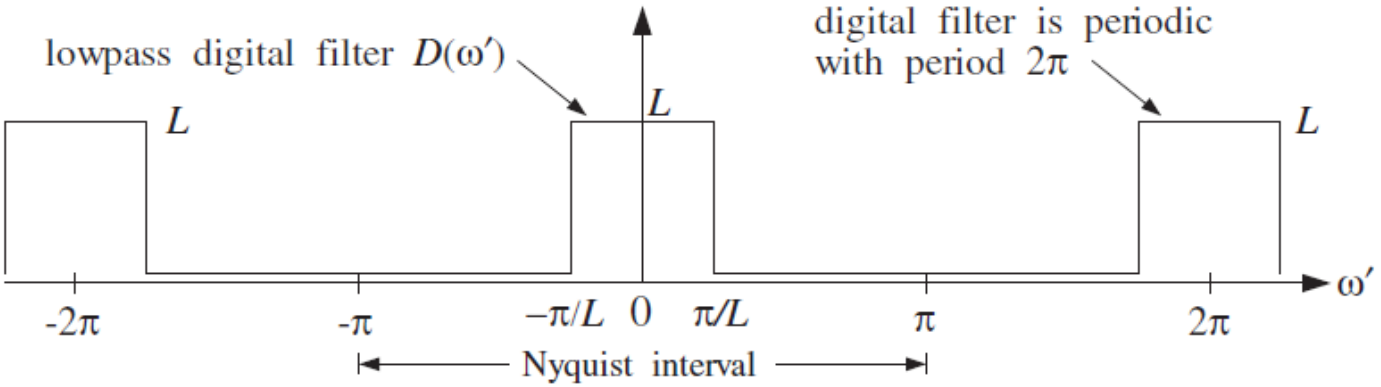
$$\boxed{f_c = \frac{f_s}{2} = \frac{{f_s}'}{2L}}$$

and expressed in units of the digital frequency, $\omega' = 2\pi f / {f_s}'$

$$\boxed{\omega_c' = \frac{2\pi f_c}{{f_s}'} = \frac{\pi}{L}}$$

The frequency response of this filter is shown below. Its passband gain is taken to be L instead of unity. This is justified below. The ideal impulse response coefficients are obtained from the inverse Fourier transform:

$$d(k') = \int_{-\pi}^{\pi} D(\omega') e^{j\omega' k'} \frac{d\omega'}{2\pi} = \int_{-\pi/L}^{\pi/L} L e^{j\omega' k'} \frac{d\omega'}{2\pi} = \frac{\sin(\pi k'/L)}{\pi k'/L}$$



An *FIR approximation* to the ideal interpolator is obtained by truncating $d(k')$ to finite length, say $N = 2LM + 1$:

$$\boxed{d(k') = \frac{\sin(\pi k'/L)}{\pi k'/L}} \qquad -LM \le k' \le LM$$

sinc-interpolator

A *causal* version of the filter is obtained by delaying it by LM samples:

$$h(n') = d(n' - LM) = \frac{\sin\big(\pi(n' - LM)/L\big)}{\pi(n' - LM)/L}, \qquad n' = 0, 1, \ldots, N-1$$

And a *windowed* version is obtained by:

$$h(n') = w(n')\, d(n' - LM)\,, \qquad n' = 0, 1, \ldots, N-1$$

where $w(n')$ is an appropriate length-N window, such as Hamming:

$$w(n') = 0.54 - 0.46 \cos\left(\frac{2\pi n'}{N-1}\right), \qquad n' = 0, 1, \ldots, N-1$$

The output of the ideal FIR interpolation filter is obtained by the convolution of the upsampled input $x_{\text{up}}(n')$ with the impulse response $d(k')$:

$$\boxed{y_{\text{up}}(n') = \sum_{k'=-LM}^{LM} d(k') x_{\text{up}}(n' - k')}$$

Polyphase Form

The interpolated values between the low-rate samples $x_{\text{up}}(nL)$ and $x_{\text{up}}(nL+L)$, that is, the values at the high-rate time instants, $n' = nL + i$, are calculated by the filter as follows:

$$y_{\text{up}}(nL+i) = \sum_{k'=-LM}^{LM} d(k')x_{\text{up}}(nL+i-k'), \qquad i = 0, 1, \ldots, L-1$$

Writing uniquely, $k' = kL+j$, with $0 \le j \le L-1$, and replacing the single summation over k' by a double summation over k and j, we find

$$y_{\text{up}}(nL+i) = \sum_{k=-M}^{M-1} \sum_{j=0}^{L-1} d(kL+j)x_{\text{up}}(nL+i-kL-j)$$

To be precise, for the case $i = 0$, the summation over k should be over the range $-M \le k \le M$. But as we will see shortly, the term $k = M$ does not contribute to the sum.

We define the ith polyphase subfilter, for $i = 0, 1, \ldots, L-1$, by,

$$\boxed{d_i(k) = d(kL + i)}, \qquad -M \leq k \leq M - 1$$

Then, we can rewrite the ith interpolated sample value as:

$$y_{\text{up}}(nL + i) = \sum_{k=-M}^{M-1} \sum_{j=0}^{L-1} d_j(k) x_{\text{up}}(nL - kL + i - j)$$

But the upsampled input signal is non-zero only at times that are integral multiples of L. Therefore,

$$x_{\text{up}}(nL - kL + i - j) = 0, \qquad \text{if } i \neq j$$

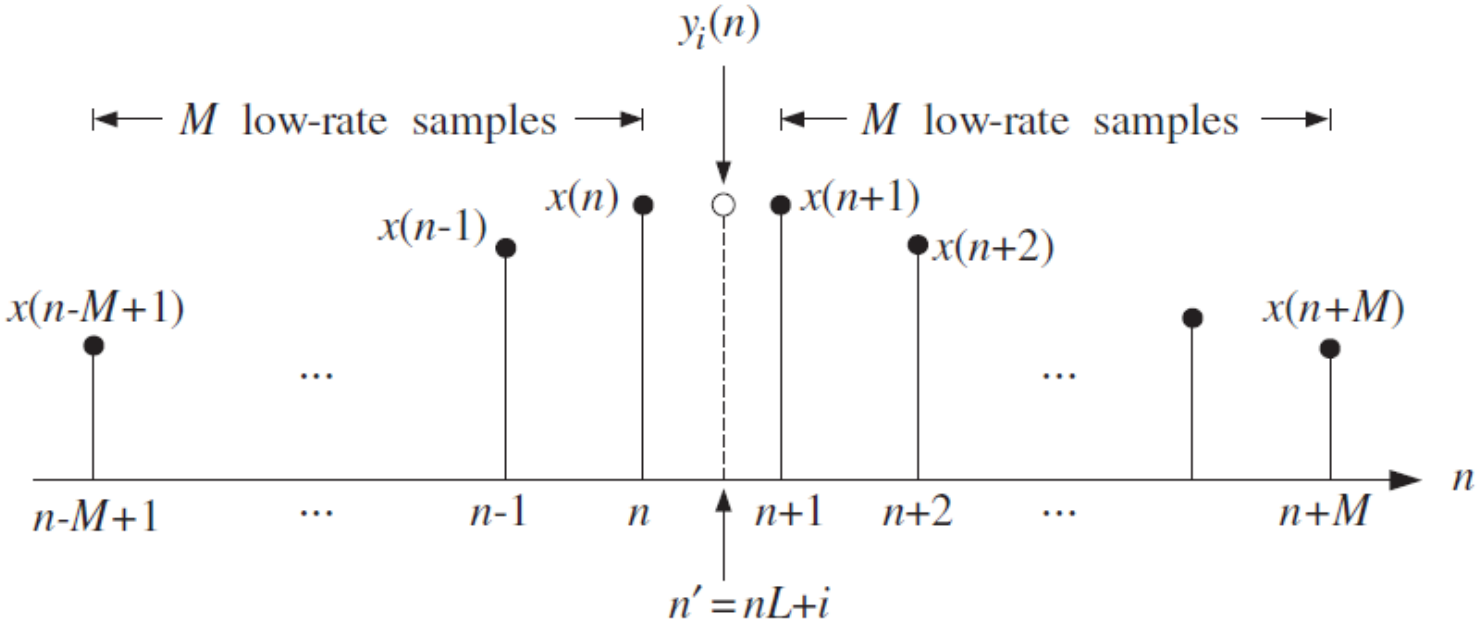
This follows from the fact that $|i - j| \leq L - 1$. Thus, keeping only the $j = i$ term in the above convolution sum, we obtain,

$$y_{\text{up}}(nL + i) = \sum_{k=-M}^{M-1} d_i(k) x_{\text{up}}(nL - kL), \qquad i = 0, 1, \ldots, L-1$$

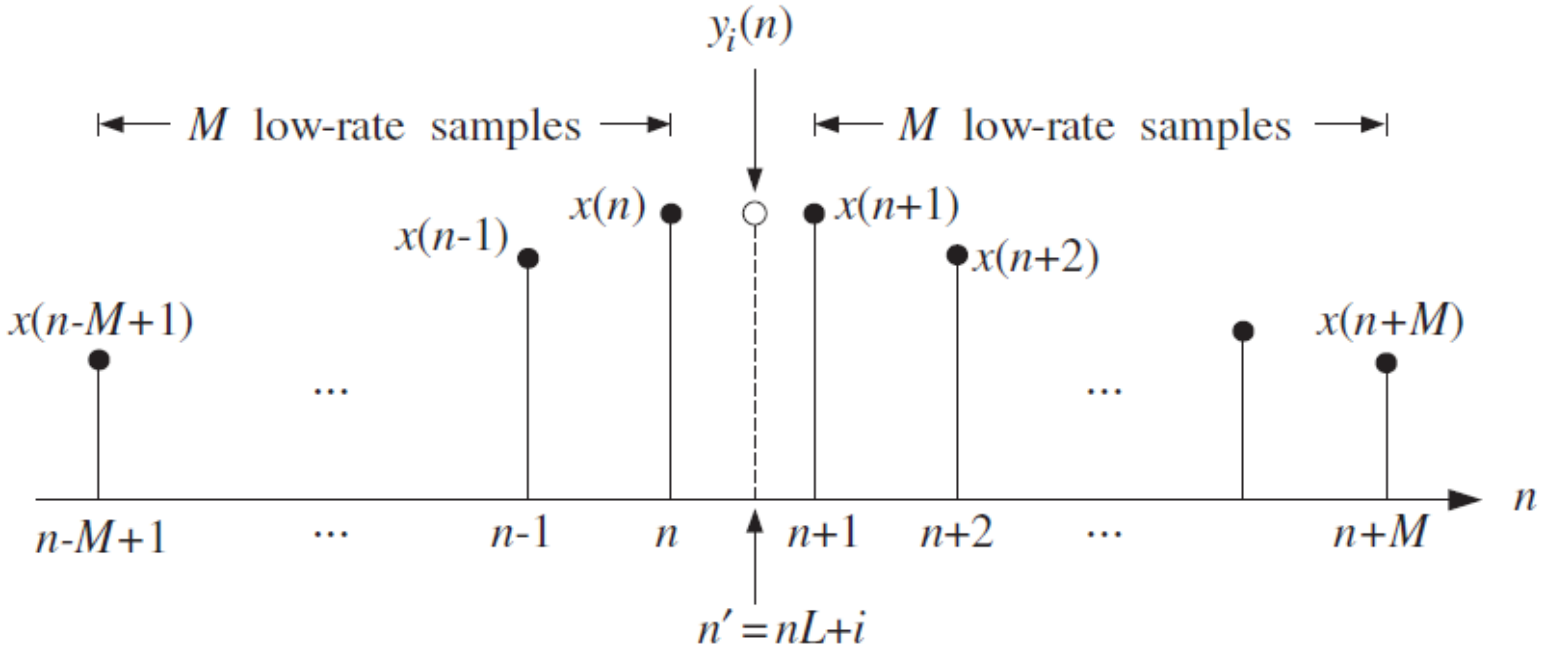
or, since $x_{\text{up}}(nL) = x(n)$, we have in terms of the low-rate samples:

$$\boxed{y_i(n) = \sum_{k=-M}^{M-1} d_i(k)x(n-k)} \qquad i = 0, 1, \ldots, L-1$$

where we set $y_i(n) = y_{\text{up}}(nL + i)$. Thus, the ith interpolated value, is computed by the ith polyphase subfilter, $d_i(k)$, which has length $2M$ and is acting only on the low-rate input samples $x(n)$. Each interpolated value is computed as a linear combination of M low-rate samples above and M below the desired interpolation time, as shown below.



$$y_i(n) = \sum_{k=-M}^{M-1} d_i(k)x(n-k) \qquad i = 0, 1, \ldots, L-1$$



Using the L subfilters, interpolation is performed at a *reduced* computational cost as compared with the cost of the full, length-N, interpolation filter $d(k')$ acting on the upsampled signal $x_{\text{up}}(n')$.

The computational cost of the full direct-form is essentially $2LM$ multiplications per interpolated value, or, $2L^2M$ multiplications for computing L interpolated values.

By contrast, the polyphase form requires $2M$ multiplications per polyphase subfilter, or, $2LM$ multiplications for L interpolated values.

Thus, the polyphase subfilter implementation achieves a factor of L in computational savings. Another way to view the computational rate is in terms of the total number of *multiplications per second* required for the filtering operations, that is,

$$\begin{aligned} R &= N(Lf_s) = NLf_s \quad \text{(direct form)} \\ R &= L(2M)f_s = Nf_s \quad \text{(polyphase form)} \end{aligned}$$

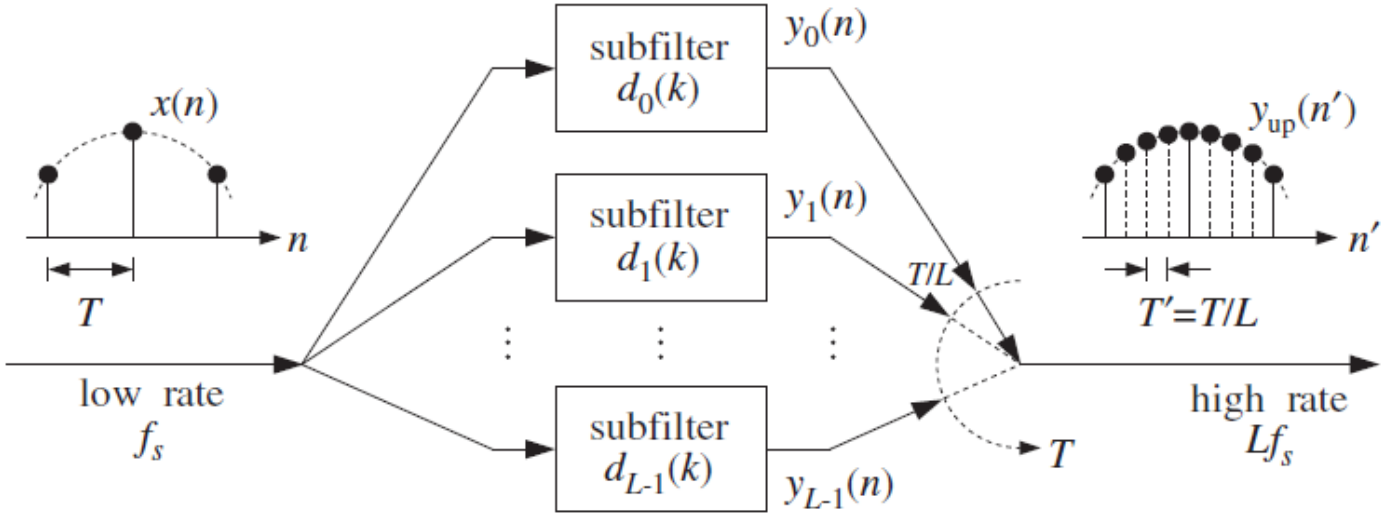
where in the *direct* form we have a single filter of length N operating at rate Lf_s and in the *polyphase* form we have L filters operating at f_s, each having computational rate $(2M)f_s$ multiplications per second.

Actually, there are $L-1$ subfilters of length $(2M)$ and one, the filter $d_0(k)$, of length $(2M+1)$, giving rise to

$$R = (L-1)(2M)f_s + (2M+1)f_s = Nf_s$$

where $N = 2LM + 1$.

The polyphase implementation is depicted below, where during each low-rate sampling period T, the commutator reads, in sequence of $T' = T/L$ seconds, the L interpolated values at the outputs of the subfilters.



This can be seen more formally, as follows. Let ζ^{-1} denote the unit delay with respect to the high rate Lf_s and let z^{-1} denote the low-rate delay. Since L high-rate delays equal one low-rate delay, that is, $LT' = T$, we will have:

$$z = \zeta^L \qquad \Leftrightarrow \qquad \zeta = z^{1/L}$$

Next, we consider the 0th polyphase subfilter, $d_0(k)$, which plays a special role. We have,

$$d_0(k) = d(kL) = \frac{\sin(\pi k)}{\pi k} = \delta(k), \qquad -M \le k \le M$$

and therefore, its output will be trivially equal to its input, that is, equal to the low-rate input sample $x(n) = x_{\text{up}}(nL)$. Indeed,

$$y_0(n) = y_{\text{up}}(nL) = \sum_{k=-M}^{M-1} d_0(k)x(n-k) = \sum_{k=-M}^{M-1} \delta(k)x(n-k) = x(n)$$

This property is preserved even for the windowed case, because all windows $w(n')$ are equal to unity at their middle. This result justifies the choice for the passband gain of the interpolator filter to be equal to L instead of 1. If the gain were 1, we would have, $y_{\text{up}}(nL) = x(n)/L$.

The causal filter implementation requires that we either delay the output or *advance the input* by M units. We choose the latter. The polyphase subfilters can be made causal by a delay of M low-rate samples:

$$h_i(n) = d_i(n-M) = d\big((n-M)L + i\big) = d(nL + i - LM)$$

for, $n = 0, 1, \ldots, 2M-1$.

For the windowed case, we have:

$$h_i(n) = d(nL + i - LM)w(nL + i), \qquad n = 0, 1, \ldots, 2M - 1$$

In terms of the causal subfilters $h_i(n)$, the filtering equation becomes,

$$y_i(n) = \sum_{k=-M}^{M-1} d_i(k)x(n-k) = \sum_{k=-M}^{M-1} h_i(k+M)x(n-k)$$

or, setting $m = k + M$ and $k = m - M$,

$$\boxed{y_i(n) = \sum_{m=0}^{P} h_i(m)x(M + n - m)}, \qquad i = 0, 1, \ldots, L - 1$$

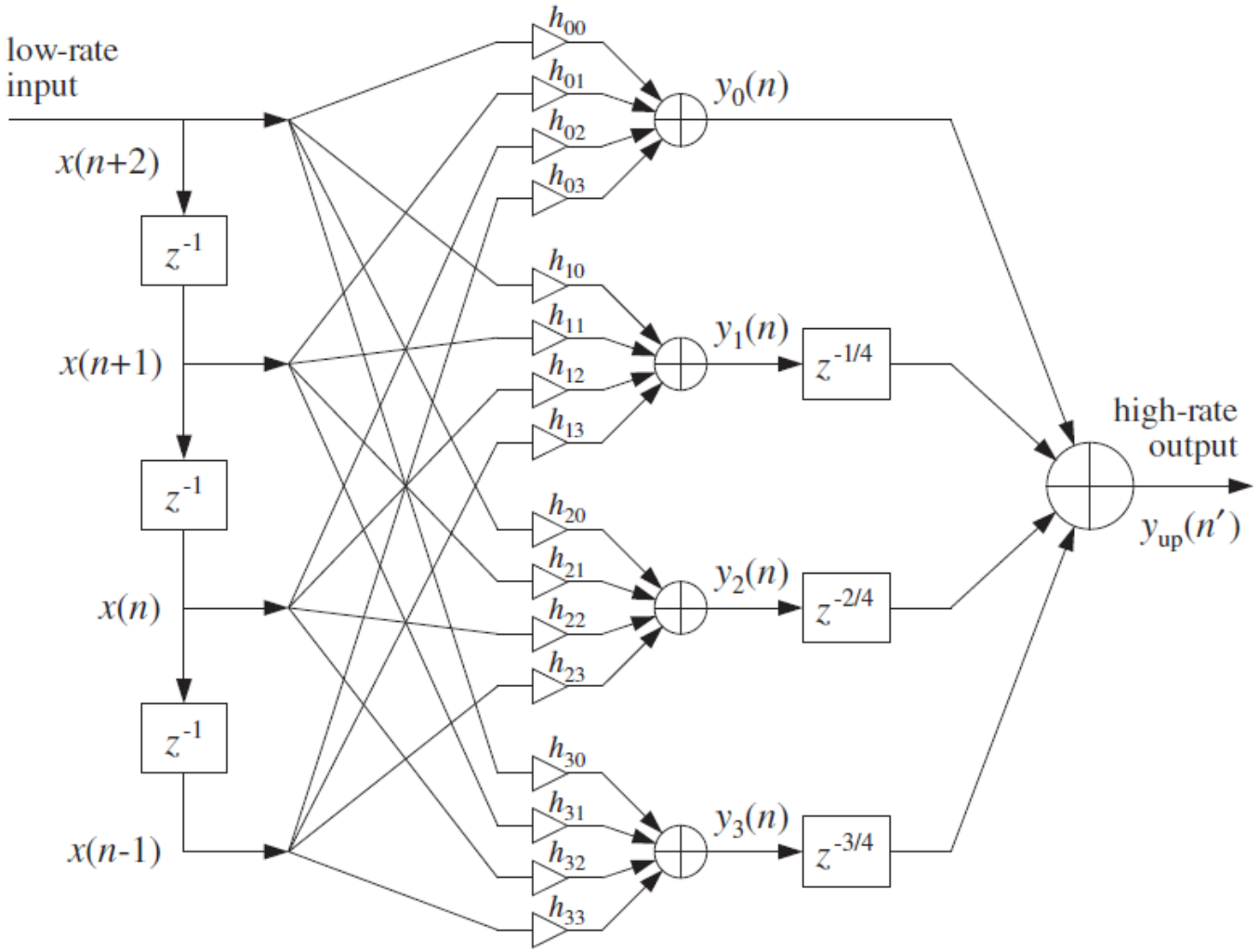
where $P = 2M - 1$ denotes the *order* of each polyphase subfilter. In other words, the interpolated samples are obtained by ordinary *causal* FIR filtering of the *time-advanced* low-rate input samples. The same result can also be obtained by z-transforms. The causal definition reads in the z-domain:

$$H_i(z) = z^{-M} D_i(z)$$

where z^{-1} represents a low-rate delay. Similarly, for the output,

$$Y_i(z) = D_i(z)X(z) = H_i(z)\left[z^M X(z)\right]$$

The sample-by-sample processing implementation requires a *common* low-rate tapped delay line which is used in sequence by *all* the subfilters $h_i(n)$ before its contents are updated. The figure below shows a concrete example when $L = 4$ and $M = 2$.



The required time-advance by M samples is implemented by initially filling the delay line with the first M low-rate samples. The internal states of the tapped delay line can be defined as

$$w_m(n) = x(M + n - m), \qquad m = 0, 1, \ldots, P$$

Then, the filtering equation can be written as the dot-product,

$$y_i(n) = \mathbf{h}_i^T \mathbf{w}(n) = \text{dot}\big(P, \mathbf{h}_i, \mathbf{w}(n)\big), \quad i = 0, 1, \ldots, L-1$$

After computing the outputs of the L subfilters, the internal state $\mathbf{w}$ may be updated to the next time instant by a call to the I2SP function, **delay**, which shifts the contents:

$$w_m(n+1) = w_{m-1}(n), \qquad m = 1, 2, \ldots, P$$

This leads to the following *sample processing algorithm* for the polyphase form: Initialize the internal state vector, $\mathbf{w}(n) = [w_0(n), w_1(n), \ldots, w_P(n)]^T$ by filling it with the first M low-rate input samples, $x_0, x_1, \ldots, x_{M-1}$, that is, at time $n = 0$ start with,

$$\mathbf{w}(0) = [0,\ x_{M-1},\ x_{M-2},\ \ldots,\ x_0,\ \underbrace{0,\ 0,\ \ldots,\ 0}_{M-1 \text{ zeros}}]^T$$

The value $w_0(0)$ need not be initialized—it is read as the current input sample. If the low-rate samples are being read sequentially from a file or an input port, then this initialization can be implemented by the following algorithm:

for $m = M$ down to $m = 1$ do:
    read low-rate input sample x
    $w_m = x$

Then, proceed by reading each successive low-rate sample, $x(M + n)$, $n = 0, 1, \ldots$, and processing it by the algorithm:

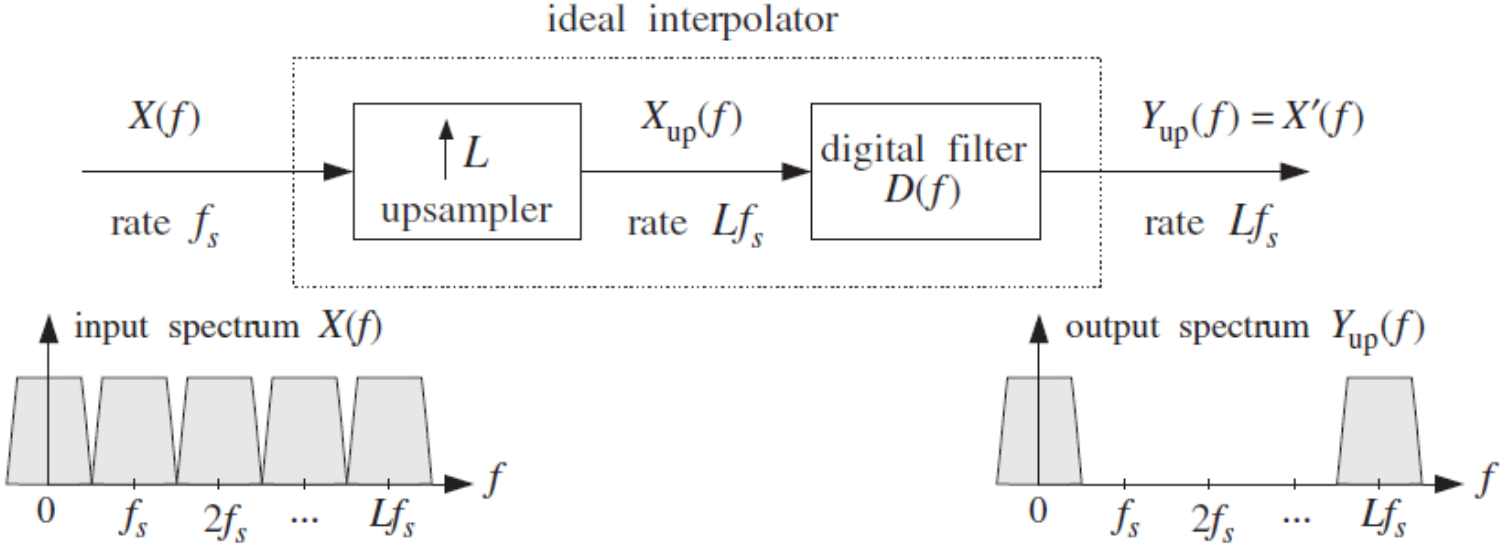
for each low-rate input sample x do:
    $w_0 = x$
    for $i = 0, 1, \ldots, L-1$ compute:
        $y_i = \mathbf{h}_i^T \mathbf{w}$
    delay$(P, \mathbf{w})$

where the function, delay$(P, \mathbf{w})$, represents the shifting of the delay line, that is, replacing the current $\mathbf{w}$, by the next one, with the first P elements shifted to become the last P elements (with w_0 overwritten in the next call),

$$\underbrace{[w_0,\ w_1,\ \ldots,\ w_{P-1}}_{\text{first } P},\ w_P]^T \quad \Rightarrow \quad [w_0,\ \underbrace{w_0,\ w_1,\ \ldots,\ w_{P-1}}_{\text{shifted}}]^T$$

The effect of the ideal interpolator in the frequency domain is shown below. The input spectrum consists of replicas at multiples of the *input* sampling rate f_s.

The filter removes all of these replicas, *except* those that are multiples of the *output* rate Lf_s. The output spectrum consists only of replicas at multiples of Lf_s. (The scaling by the gain L is not shown.)



Kaiser Window Designs

Digital interpolation filters can be designed by a variety of filter design methods, such as the Fourier series method with windowing, Parks-McClellan, or even IIR designs. Here we summarize FIR designs based on the Kaiser window method.

We follow the design steps of I2SP–Sect. 10.2, but use f_s' in place of f_s, because the interpolation filter is operating at the fast rate f_s'. For any length-N window $w(n')$, the interpolator's impulse response is computed by

$$h(n') = w(n')d(n' - LM), \qquad n' = 0, 1, \ldots, N-1 = 2LM$$

The L length-$(2M)$ *polyphase subfilters* are defined in terms of $h(n')$ as follows. For $i = 0, 1, \ldots, L-1$,

$$h_i(n) = h(nL + i), \qquad n = 0, 1, \ldots, 2M-1$$

For a Kaiser window design, we start by specifying the desired stopband attenuation A in dB, and desired transition width Δf about the ideal cutoff frequency:

$$f_c = \frac{f_s'}{2L} = \frac{f_s}{2}$$

so that the passband and stopband frequencies are:

$$f_{\text{pass}} = f_c - \frac{1}{2}\Delta f, \quad f_{\text{stop}} = f_c + \frac{1}{2}\Delta f$$

The Kaiser window parameters are calculated by:

$$\delta = 10^{-A/20}$$

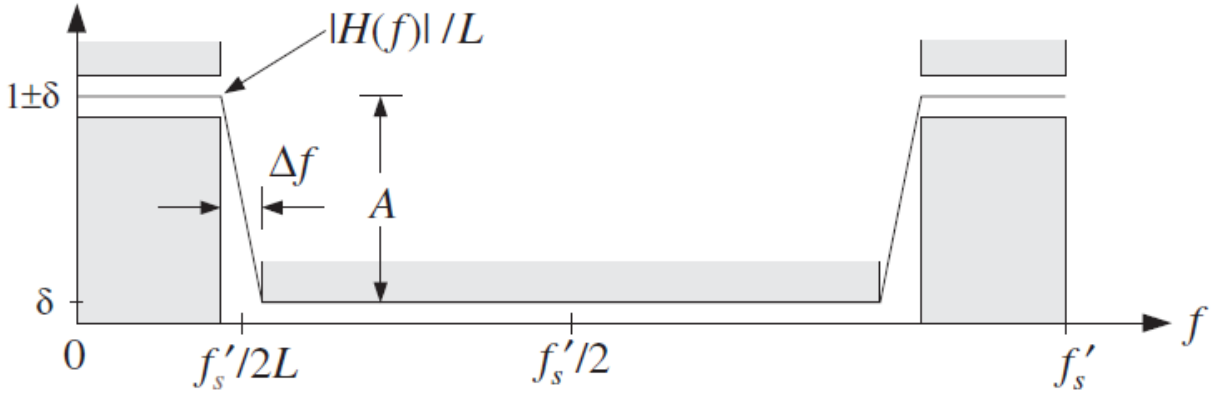
$$D = \frac{A - 7.95}{14.36}$$

$$\alpha = 0.1102(A - 8.7) \qquad \text{(because, typically, } A > 50 \text{ dB)}$$

$$N - 1 \geq \frac{Df_s'}{\Delta f} = \frac{DLf_s}{\Delta f} = \frac{DL}{\Delta F}$$

where we used f_s' in the formula for N and set $\Delta F = \Delta f / f_s$. Then, N must be rounded up to the smallest odd integer of the form, $N = 2LM + 1$ satisfying the above inequality.

The design specifications are shown in the figure below.



The designed length-N impulse response is,

$$h(n') = w(n')d(n' - LM), \qquad n' = 0, 1, \ldots, N-1 = 2LM$$

with $w(n')$ given for $n' = 0, 1, \ldots, N-1$:

$$w(n') = \frac{I_0\left(\alpha\sqrt{1-(n'-LM)^2/(LM)^2}\right)}{I_0(\alpha)}$$

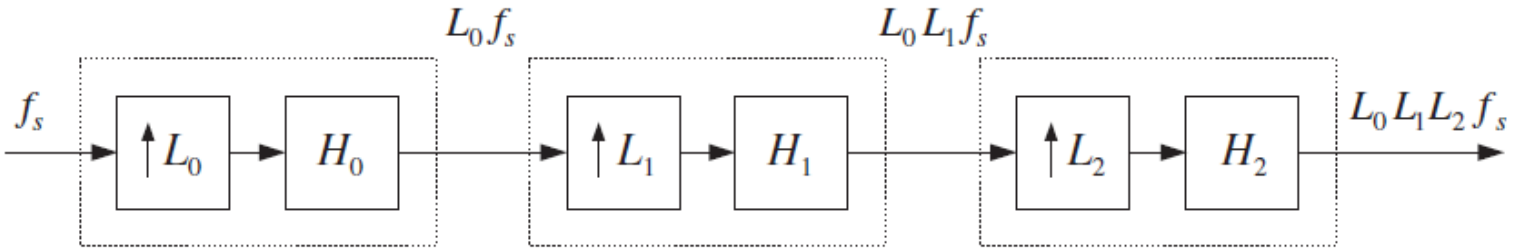
The frequency response of the designed filter may be computed by:

$$H(f) = \sum_{n'=0}^{N-1} h(n')e^{-2\pi jfn'/f_s'} = \sum_{n'=0}^{N-1} h(n')e^{-2\pi jfn'/(Lf_s)}$$

The designed filter $h(n')$ can be implemented in its direct or polyphase forms.

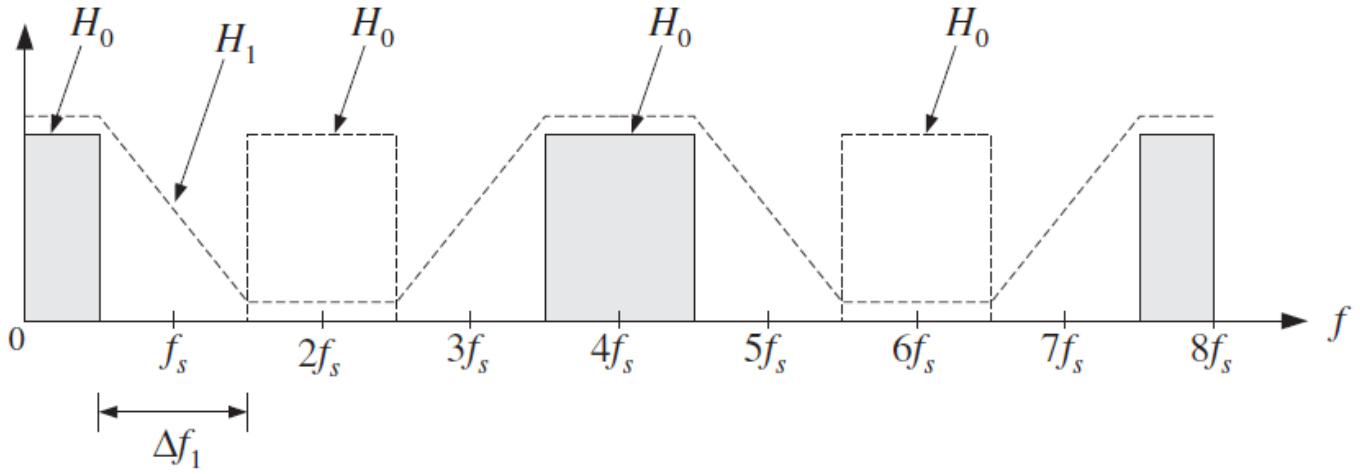
Multistage Designs

Interpolation filters can also be implemented in a multistage form, whereby the sampling rate is gradually increased in stages until the final rate is reached. An example is shown below. The first filter increases the sampling rate by a factor of L_0, the second by a factor of L_1, and the third by L_2, so that the overall interpolation factor is $L = L_0 L_1 L_2$. Such multistage realizations allow additional savings in the overall computational rate of the interpolator.



The first filter $H_0(f)$ must have the most *stringent* specifications in the sense that it has the desired transition width Δf, which is typically very narrow. The remaining stages have much *wider* transition widths and therefore smaller filter lengths.

To see this, consider the design of a 4-fold interpolator realized as the cascade of two 2-fold interpolators, $L = L_0 L_1$, with $L_0 = L_1 = 2$. The desired ideal frequency characteristics of the two interpolation filters are depicted below.



The first filter H_0 is operating at the intermediate rate $f_s' = L_0 f_s = 2f_s$ and is designed to act as an ideal lowpass filter with cutoff $f_c = f_s/2 = f_s'/4$. It removes all the replicas at multiples of its input rate f_s, except those that are multiples of its output rate $2f_s$. It can be designed using a Kaiser window. For example, assuming a narrow transition width Δf about f_c, and a stopband attenuation A, we have,

$$N_0 - 1 = \frac{D f_s'}{\Delta f} = \frac{D(2f_s)}{\Delta f} = \frac{2D}{\Delta F}, \qquad \Delta F = \frac{\Delta f}{f_s}$$

where D is the Kaiser D-factor that depends on the attenuation A.

The second interpolator H_1 is operating at the rate of $2f_s{}' = 4f_s$, and must remove all replicas at multiples of its input rate $2f_s$, except those that are multiples of its output rate $4f_s$. Therefore, it will have a wider transition width given by

$$\Delta f_1 = f_s{}' - f_s = 2f_s - f_s = f_s$$

Its Kaiser length will be:

$$N_1 - 1 = \frac{D(4f_s)}{\Delta f_1} = \frac{D(4f_s)}{f_s} = 4D$$

The combined effect of the two interpolation filters is to remove every three intervening replicas leaving only the replicas at multiples of $4f_s$. Because H_0 is operating at rate $2f_s$ and H_1 at rate $4f_s$, the corresponding frequency responses will be:

$$H_0(f) = \sum_{n'=0}^{N_0-1} h_0(n')e^{-2\pi jfn'/(2f_s)}, \qquad H_1(f) = \sum_{n'=0}^{N_1-1} h_1(n')e^{-2\pi jfn'/(4f_s)}$$

Assuming that both filters $h_0(n')$ and $h_1(n')$ are realized in their polyphase forms, then the total computational rate of the multistage case will be, in MACs per second:

$$R_{\text{multi}} = N_0 f_s + N_1(2f_s) \simeq \left(\frac{2D}{\Delta F} + 8D\right) f_s = \frac{2D}{\Delta F}(1 + 4\Delta F) f_s$$

By contrast, a single stage design would have filter length:

$$N - 1 = \frac{D(Lf_s)}{\Delta f} = \frac{4D}{\Delta F}$$

and polyphase computational rate:

$$R_{\text{single}} = N f_s \simeq \frac{4D}{\Delta F} f_s$$

The relative performance of the two implementations will be

$$\frac{R_{\text{multi}}}{R_{\text{single}}} = \frac{1 + 4\Delta F}{2} = \frac{1}{2} + 2\Delta F$$

Computational savings will take place whenever:

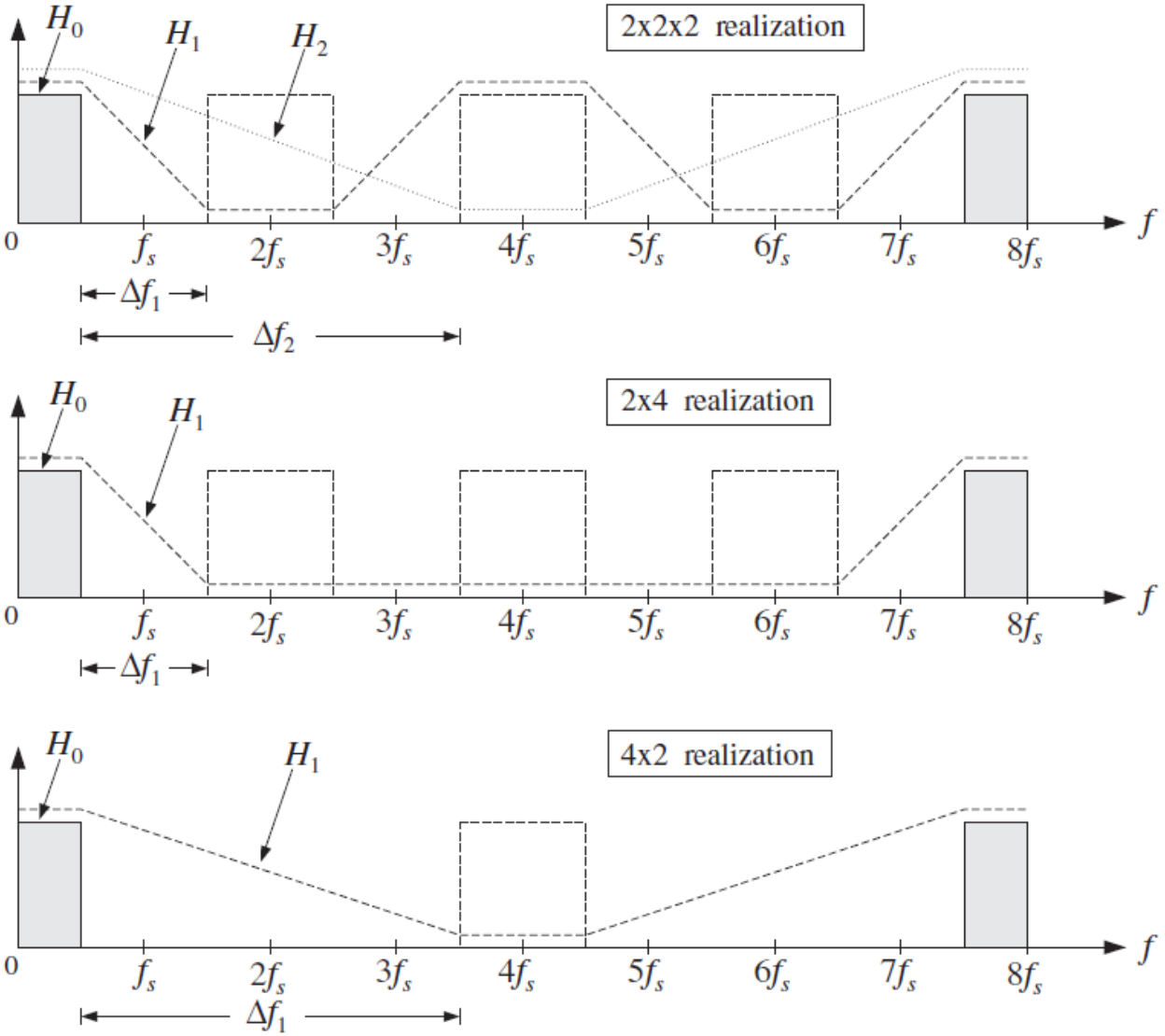
$$\frac{1}{2} + 2\Delta F < 1 \qquad \Leftrightarrow \qquad \Delta F < \frac{1}{4}$$

which is easily satisfied because typically ΔF is of the order of $1/10$.

As another example, consider an 8-fold interpolator which can be realized in three different multistage ways:

$$8 = 2 \times 2 \times 2 = 2 \times 4 = 4 \times 2$$

The frequency characteristics of the different stages are shown below.



The interpolator at each stage removes all replicas at multiples of *its input* rate, except those that are multiples of *its output* rate. In all three cases, the combined effect is to remove every seven intervening replicas leaving only the replicas at the multiples of $8f_s$.

The computational rates of the three multistage cases versus a single stage can be derived as above (see I2SP–Sect. 12.2.5),

$$\frac{R_{\text{multi}}}{R_{\text{single}}} = \frac{1}{4} + 2\Delta F \qquad (2\times 4 \text{ case})$$

$$\frac{R_{\text{multi}}}{R_{\text{single}}} = \frac{1}{4} + \frac{7}{3}\Delta F \qquad (2\times 2\times 2 \text{ case})$$

$$\frac{R_{\text{multi}}}{R_{\text{single}}} = \frac{1}{2} + \frac{4}{3}\Delta F \qquad (4\times 2 \text{ case})$$

Comparing the three multistage cases, it appears that the 2×4 case is more efficient than the $2\times 2\times 2$ case, which is more efficient than the 4×2 case. Indeed,

$$\frac{1}{4} + 2\Delta F < \frac{1}{4} + \frac{7}{3}\Delta F < \frac{1}{2} + \frac{4}{3}\Delta F$$

the second inequality being valid for $\Delta F < 1/4$.

Linear and Hold Interpolators

An ideal interpolator may be thought of as the *sampled* version of the ideal *analog* reconstructor, sampled at the high rate $f_s{}'$, that is, with $T' = T/L$,

$$d(k') = h(k'T') = \left.\frac{\sin(\pi t/T)}{\pi t/T}\right|_{t=k'T'} = \frac{\sin(\pi k'/L)}{\pi k'/L}$$

This relationship can be applied to other analog reconstructors, resulting in simpler interpolators. For any analog reconstructor $h(t)$ that reconstructs the low-rate samples by

$$y_a(t) = \sum_m h(t - mT)x(m)$$

the interpolated samples are obtained by resampling $y_a(t)$ at the rate $f_s{}'$:

$$y_a(n'T') = \sum_m h(n'T' - mT)x(m)$$

which can be written in the form

$$y_{\text{up}}(n') = \sum_m d(n' - mL)x(m)$$

where $d(k')$ is obtained from $h(t)$ via the above mapping. The interpolation equation can be written also in terms of the *upsampled* version of $x(n)$

$$y_{\text{up}}(n') = \sum_{m'} d(n' - m')x_{\text{up}}(m') = \sum_{k'} d(k')x_{\text{up}}(n' - k')$$

Two of the most common interpolators are the hold and linear interpolators resulting from the sample/hold and linear analog reconstructors having impulse responses:

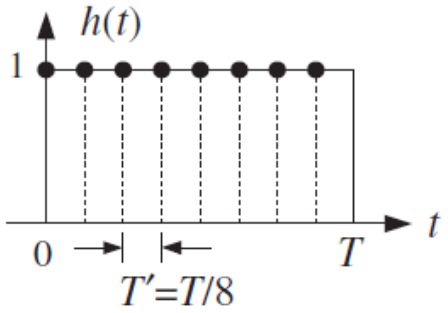
$$h(t) = \begin{cases} 1, & \text{if } 0 \le t < T \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad h(t) = \begin{cases} 1 - \dfrac{|t|}{T}, & \text{if } |t| \le T \\ 0, & \text{otherwise} \end{cases}$$

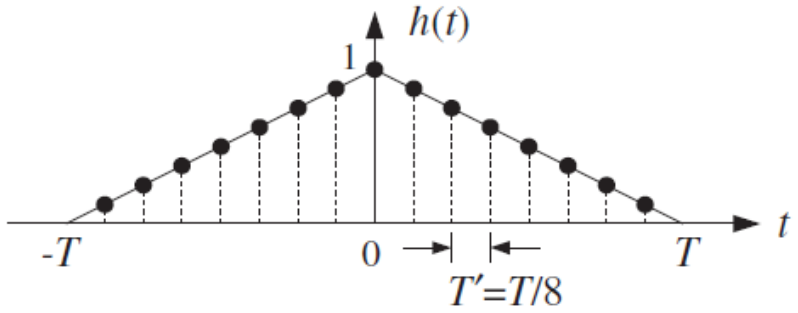
Setting $t = k'T'$ in these definitions, and using the relationship $T = LT'$, we find the following discrete-time versions:

$$\text{(hold)} \qquad d(k') = \begin{cases} 1, & \text{if } 0 \le k' \le L-1 \\ 0, & \text{otherwise} \end{cases} = u(k') - u(k'-L)$$

$$\text{(linear)} \qquad d(k') = \begin{cases} 1 - \dfrac{|k'|}{L}, & \text{if } |k'| \le L-1 \\ 0, & \text{otherwise} \end{cases}$$

Note that in the linear case, the endpoints $k' = \pm L$ are not considered because $d(k')$ vanishes there. The figure below shows the sampled impulse responses for $L = 8$.





The filtering operations of these interpolators are very simple. The hold interpolator holds each low-rate sample constant for L high-rate sampling times. In other words, each low-rate sample is *repeated* L times at the high rate. The linear interpolator interpolates *linearly* between a given low-rate sample and the next one.

To see this, we rewrite the filtering equation in its polyphase form. Setting, $n' = nL + i$ and $k' = kL + j$, we have,

$$y_{\text{up}}(nL + i) = \sum_k d_i(k)x(n - k)$$

where $d_i(k)$ are the corresponding polyphase subfilters:

$$d_i(k) = d(kL + i), \qquad i = 0, 1, \ldots, L - 1$$

and they are found to be (see I2SP–Sect. 12.3 for details),

$$\text{(hold)} \qquad d_i(k) = \delta(k)$$

$$\text{(linear)} \qquad d_i(k) = \big(1 - \frac{i}{L}\big)\delta(k) + \frac{i}{L}\delta(k + 1)$$

for $i = 0, 1, \ldots, L - 1$.

Inserting these impulse responses into the filtering equations, we obtain the explicit forms of the interpolation equations in the two cases. For the hold case,

$$\boxed{y_{\text{up}}(nL+i) = x(n)}, \qquad i = 0, 1, \ldots, L-1$$

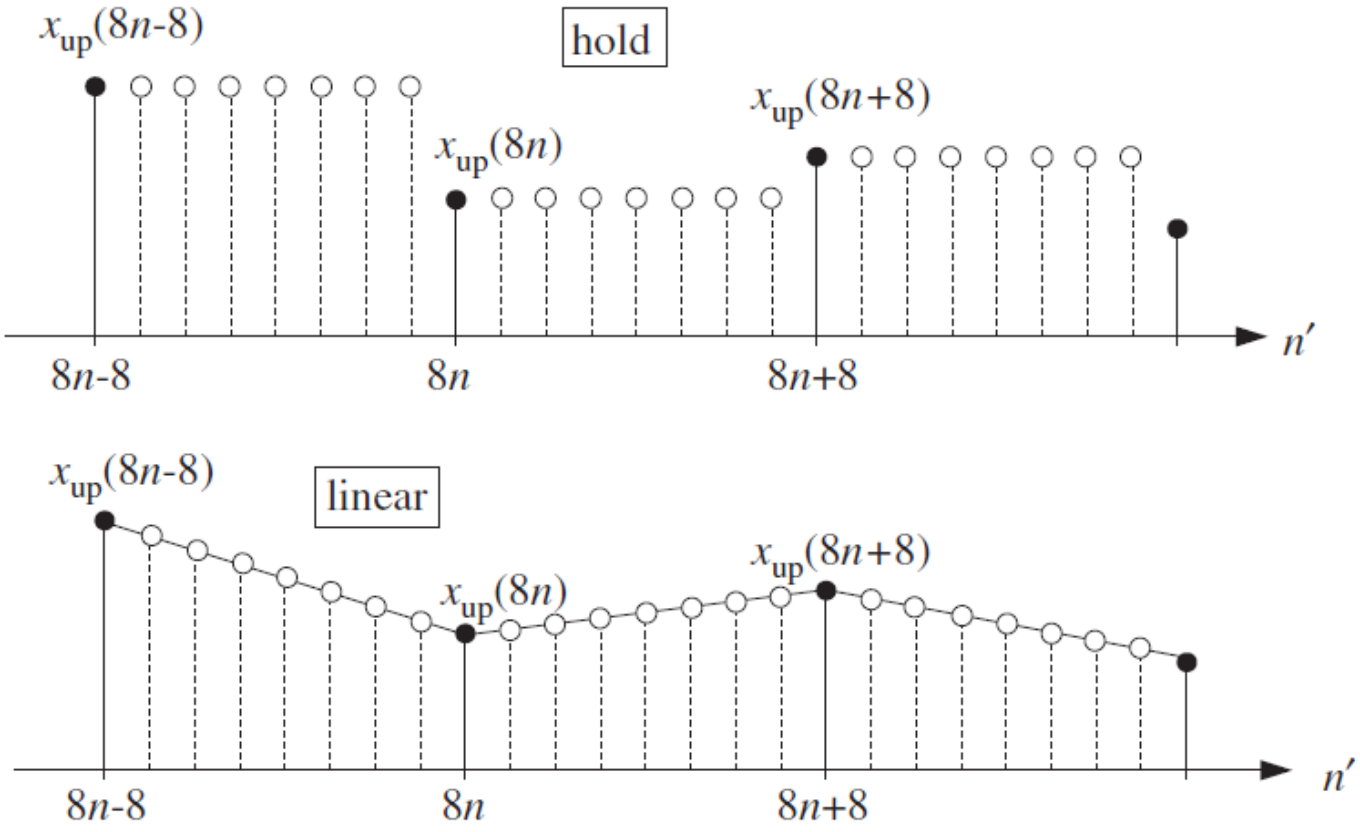
so that each low-rate sample is repeated L times. And, for the linear case,

$$\boxed{y_{\text{up}}(nL+i) = \left(1 - \frac{i}{L}\right)x(n) + \frac{i}{L}x(n+1)}, \qquad i = 0, 1, \ldots, L-1$$

corresponding to linearly weighting the two successive low-rate samples $x(n)$ and $x(n+1)$. For example, when $L = 8$ the eight interpolated samples between $x(n)$ and $x(n+1)$ are calculated by:

$$\begin{aligned}
y_{\text{up}}(8n) &= x(n) \\
y_{\text{up}}(8n+1) &= 0.875\,x(n) + 0.125\,x(n+1) \\
y_{\text{up}}(8n+2) &= 0.750\,x(n) + 0.250\,x(n+1) \\
y_{\text{up}}(8n+3) &= 0.625\,x(n) + 0.375\,x(n+1) \\
y_{\text{up}}(8n+4) &= 0.500\,x(n) + 0.500\,x(n+1) \\
y_{\text{up}}(8n+5) &= 0.375\,x(n) + 0.625\,x(n+1) \\
y_{\text{up}}(8n+6) &= 0.250\,x(n) + 0.750\,x(n+1) \\
y_{\text{up}}(8n+7) &= 0.125\,x(n) + 0.875\,x(n+1)
\end{aligned}$$

The figure below shows the interpolated signal using 8-fold hold and linear interpolators.



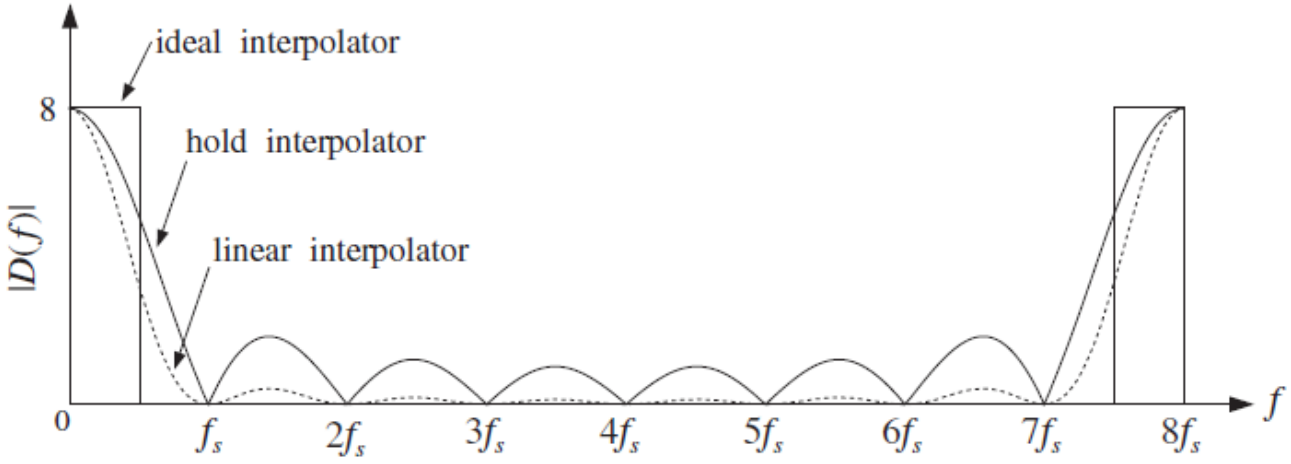
To understand the extent to which the hold and linear interpolators differ from the ideal interpolator, we determine their frequency responses (see I2SP–Sect. 12.3for details). For the hold case,

$$D(f) = \frac{1 - e^{-jL\omega'}}{1 - e^{-j\omega'}} = \frac{\sin(L\omega'/2)}{\sin(\omega'/2)} e^{-j(L-1)\omega'/2} = \frac{\sin(\pi f/f_s)}{\sin(\pi f/Lf_s)} e^{-j\pi(L-1)f/Lf_s}$$

And, for the linear case,

$$D(f) = \frac{1}{L}\left|\frac{\sin(L\omega'/2)}{\sin(\omega'/2)}\right|^2 = \frac{1}{L}\left|\frac{\sin(\pi f/f_s)}{\sin(\pi f/Lf_s)}\right|^2$$

Both frequency responses are periodic in f with period $f_s' = Lf_s$ and vanish at all multiples of f_s which are *not* multiples of Lf_s. Therefore, they partially remove the spectral replicas that are between multiples of f_s'. They are shown below for the case $L = 8$, together with the ideal response.



Because of their simple structure, linear and hold interpolators are used in *multistage* implementations of interpolators, especially in the latter stages that have higher sampling rates.

Example – 4-fold Interpolators

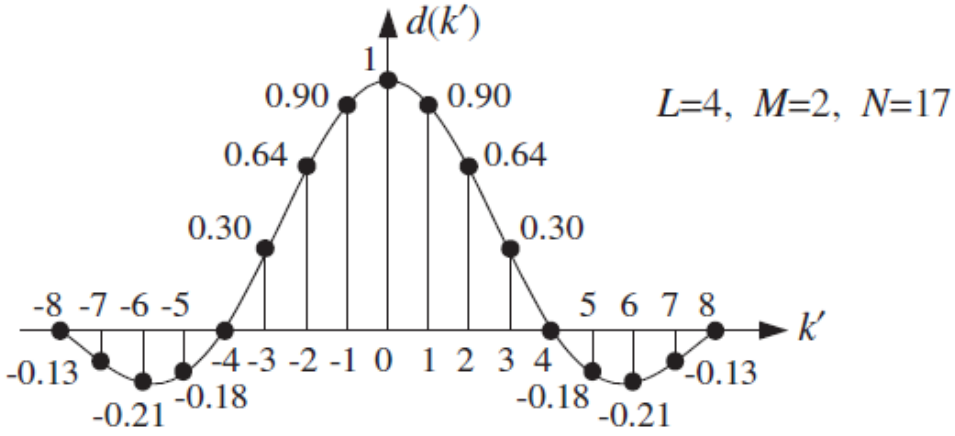
Consider the case of a 4-fold interpolator having $L = 4$ and polyphase filter length $2M = 4$ or $M = 2$. This corresponds to a filter length $N = 2LM + 1 = 17$. The ideal impulse response will be:

$$d(k') = \frac{\sin(\pi k'/4)}{\pi k'/4}, \qquad -8 \le k' \le 8$$

or, numerically,

$$\mathbf{h} = \mathbf{d} = [0,\ -0.13,\ -0.21,\ -0.18,\ 0,\ 0.30,\ 0.64,\ 0.90,\ 1,\ 0.90,\ 0.64,\ 0.30,\ 0,\ -0.18,\ -0.21,\ -0.13,\ 0]$$

where **h** is the causal version, with time origin shifted to the *left* of the vector, and **d** is the symmetric one with time origin at the *middle* of the vector. This truncated ideal impulse response is shown below.



The four polyphase subfilters are defined as follows, for $i = 0, 1, 2, 3$,

$$d_i(k) = d(4k + i), \qquad -2 \le k \le 1$$

They are extracted from $\mathbf{h}$ by taking every fourth entry, starting with the ith entry:

$$\begin{aligned}
\mathbf{h}_0 &= \mathbf{d}_0 = [0,\ 0,\ 1,\ 0] \\
\mathbf{h}_1 &= \mathbf{d}_1 = [-0.13,\ 0.30,\ 0.90,\ -0.18] \\
\mathbf{h}_2 &= \mathbf{d}_2 = [-0.21,\ 0.64,\ 0.64,\ -0.21] \\
\mathbf{h}_3 &= \mathbf{d}_3 = [-0.18,\ 0.90,\ 0.30,\ -0.13]
\end{aligned}$$

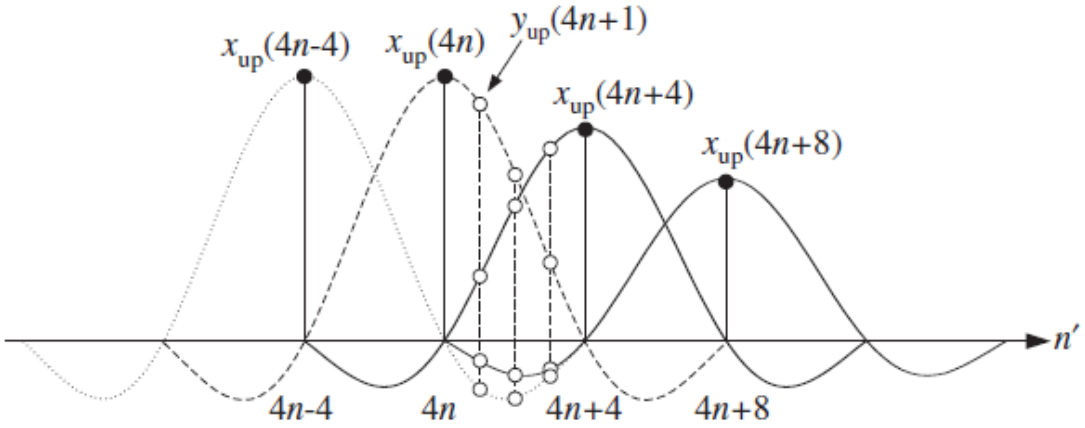
The interpolated samples between $x(n) = x_{\text{up}}(4n)$ and $x(n+1) = x_{\text{up}}(4n+4)$ are calculated from the polyphase filtering equation, applied here with $M = 2$ and $P = 2M - 1 = 3$,

$$y_i(n) = \sum_{m=0}^{P} h_i(m) x(M + n - m)\,, \qquad i = 0, 1, \ldots, L-1$$

All four subfilters act on the (time-advanced) low-rate input samples $\{x(n+2), x(n+1), x(n), x(n-1)\}$, or, $\{x_{\text{up}}(4n+8), x_{\text{up}}(4n+4), x_{\text{up}}(4n), x_{\text{up}}(4n-4)\}$. The polyphase equations can be cast in a compact matrix form:

$$\begin{bmatrix} y_{\text{up}}(4n) \\ y_{\text{up}}(4n+1) \\ y_{\text{up}}(4n+2) \\ y_{\text{up}}(4n+3) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -0.13 & 0.30 & 0.90 & -0.18 \\ -0.21 & 0.64 & 0.64 & -0.21 \\ -0.18 & 0.90 & 0.30 & -0.13 \end{bmatrix} \begin{bmatrix} x_{\text{up}}(4n+8) \\ x_{\text{up}}(4n+4) \\ x_{\text{up}}(4n) \\ x_{\text{up}}(4n-4) \end{bmatrix}$$

These results can be understood more intuitively using the LTI form of convolution, that is, superimposing the full length-17 symmetric impulse response **d** at the four contributing low-rate samples and summing up their contributions at the four desired time instants, that is, at $n' = 4n + i$, $i = 0, 1, 2, 3$, as shown below.



For example, referring to the sampled impulse response values, we find that at time instant $4n + 1$, the input sample $x_{up}(4n + 8)$ will contribute an amount $-0.13x_{up}(4n + 8)$, the sample $x_{up}(4n + 4)$ will contribute an amount $0.30x_{up}(4n + 4)$, the sample $x_{up}(4n)$ will contribute $0.90x_{up}(4n)$, and the sample $x_{up}(4n - 4)$ an amount $-0.18x_{up}(4n - 4)$. The interpolated value is built up from these four contributions:

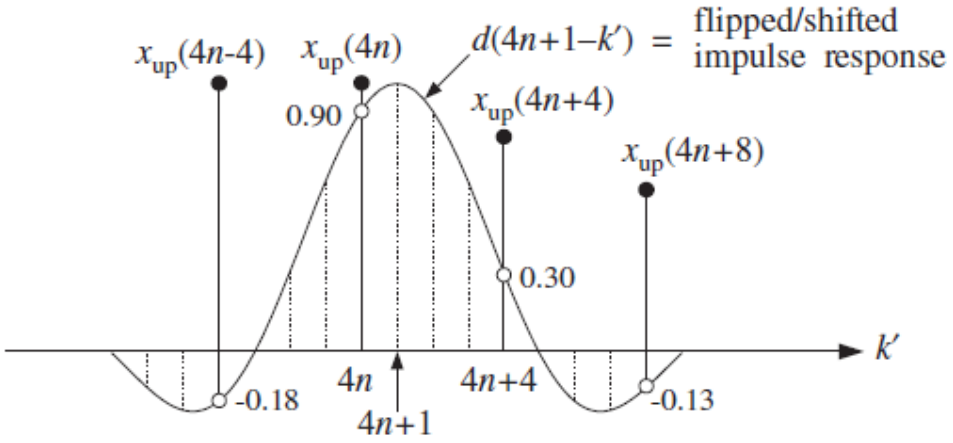
$$y_{up}(4n+1) = -0.13x_{up}(4n+8)+0.30x_{up}(4n+4)+0.90x_{up}(4n)-0.18x_{up}(4n-4)$$

Similarly, it should be evident that, $y_{up}(4n) = x_{up}(4n)$, with the contributions of the other low-rate inputs vanishing at time instant $4n$.

We may also use the flip-and-slide form of convolution, in which the impulse response $d(k')$ is flipped, delayed, and positioned at the sampling instant n' to be computed. For example, at $n' = 4n + 1$, we have:

$$y_{\text{up}}(4n+1) = \sum_{k'} d(4n+1-k')x_{\text{up}}(k')$$

The figure below shows this operation. Because of symmetry, the flipped impulse response is the same as in the LTI form. It is then translated to $n' = 4n + 1$ and the above linear combination is performed.



The only contributions come from the low-rate samples that fall within the finite extent of the impulse response. Thus, only the terms $k' = 4n-4, 4n, 4n+4, 4n+8$ contribute, and each is weighted by the appropriate impulse response values that are read off from the figure, that is, $\{-0.18, 0.90, 0.30, -0.13\}$, so that again:

$$y_{\text{up}}(4n+1) = -0.18x_{\text{up}}(4n-4)+0.90x_{\text{up}}(4n)+0.30x_{\text{up}}(4n+4)-0.13x_{\text{up}}(4n+8)$$

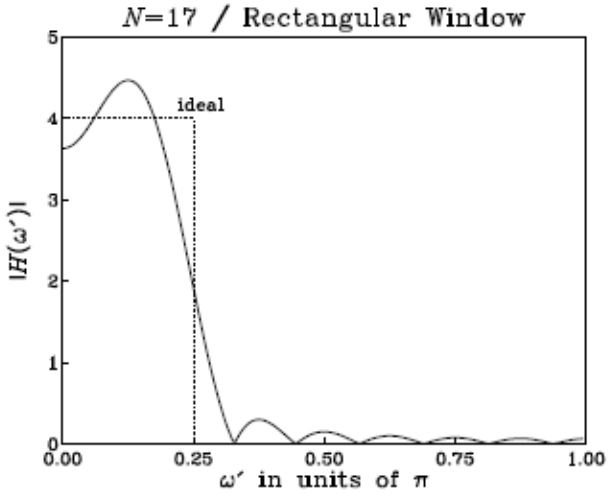
The Hamming windowed version of the filter is obtained by multiplying the full length-17 filter response **h** by a length-17 Hamming window. The resulting impulse response becomes:

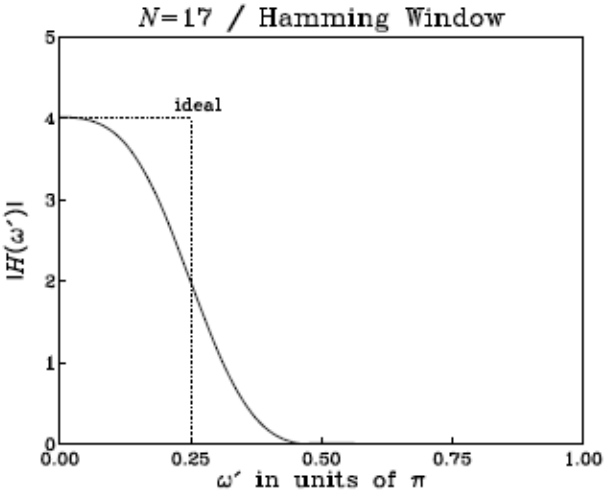
$$\mathbf{h} = [0,\ -0.02,\ -0.05,\ -0.07,\ 0,\ 0.22,\ 0.55,\ 0.87,\ 1,\ 0.87,\ 0.55,\ 0.22,\ 0,\ -0.07,\ -0.05,\ -0.02,\ 0]$$

The polyphase interpolation equations become in this case:

$$\begin{bmatrix} y_{\text{up}}(4n) \\ y_{\text{up}}(4n+1) \\ y_{\text{up}}(4n+2) \\ y_{\text{up}}(4n+3) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -0.02 & 0.22 & 0.87 & -0.07 \\ -0.05 & 0.55 & 0.55 & -0.05 \\ -0.07 & 0.87 & 0.22 & -0.02 \end{bmatrix} \begin{bmatrix} x_{\text{up}}(4n+8) \\ x_{\text{up}}(4n+4) \\ x_{\text{up}}(4n) \\ x_{\text{up}}(4n-4) \end{bmatrix}$$

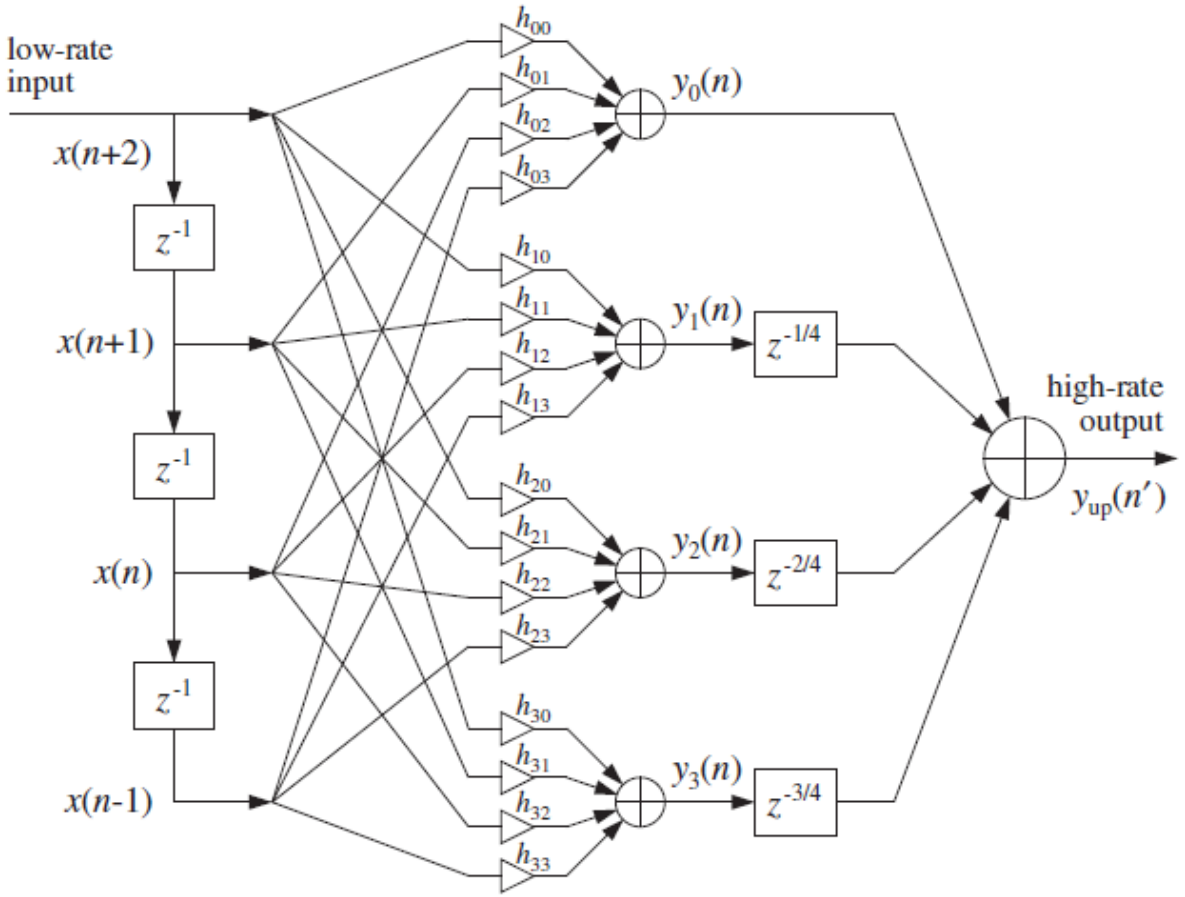
The graphs below compare the *magnitude responses* of the rectangularly and Hamming windowed interpolation filters.





A *block diagram* realization of the polyphase form for this example is shown below, with all the subfilters using the *same tapped delay line* holding the incoming low-rate samples, where the indicated polyphase subfilters are defined by, for $i = 0, 1, 2, 3$.

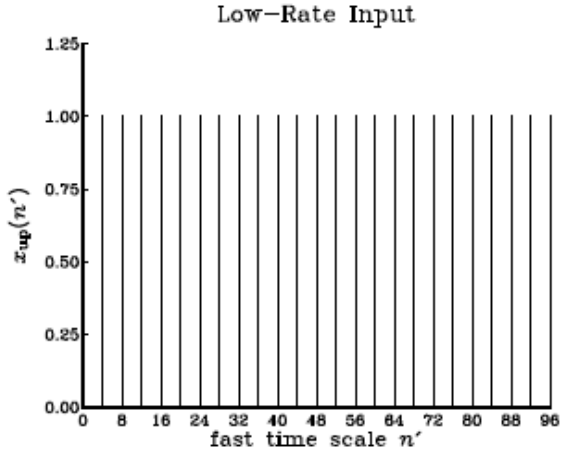
$$\mathbf{h}_i = [h_{i0},\ h_{i1},\ h_{i2},\ h_{i3}], \qquad H_i(z) = h_{i0} + h_{i1}z^{-1} + h_{i2}z^{-2} + h_{i3}z^{-3}$$

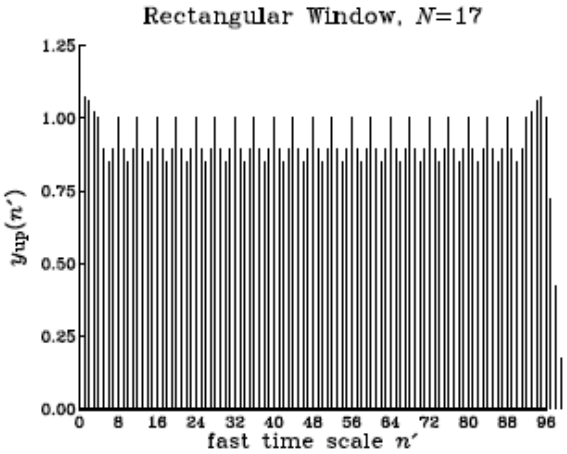


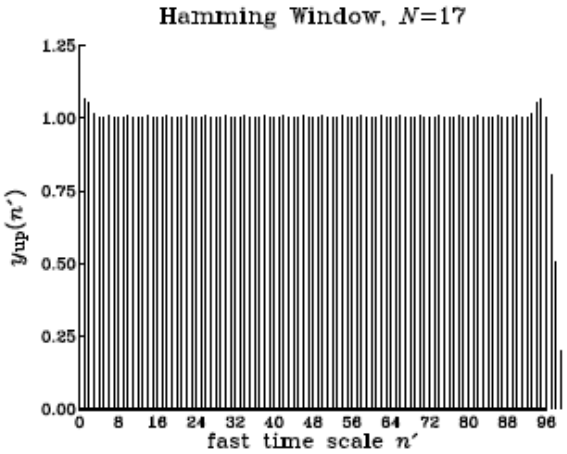
As a concrete filtering example, consider the following low-rate input signal $x(n)$ consisting of 25 DC samples, and depicted below with respect to the fast time scale:

$$x(n) = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$$

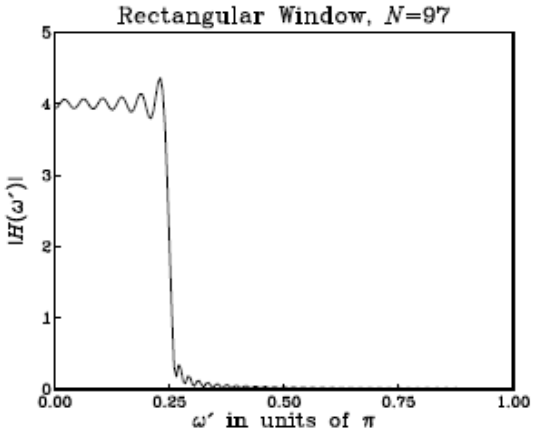
The interpolated values between these low-rate samples are shown below for the cases of the rectangularly and Hamming windowed interpolating filters. The input-on and input-off transients are evident.

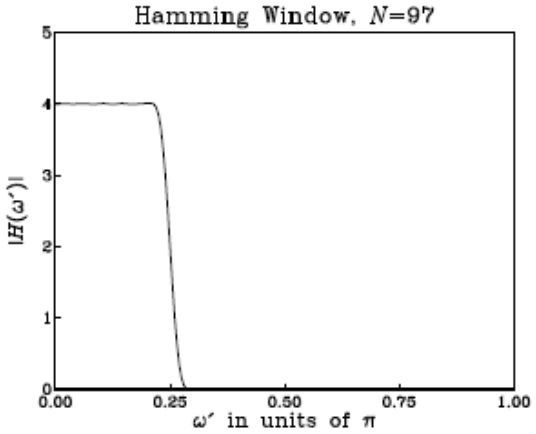


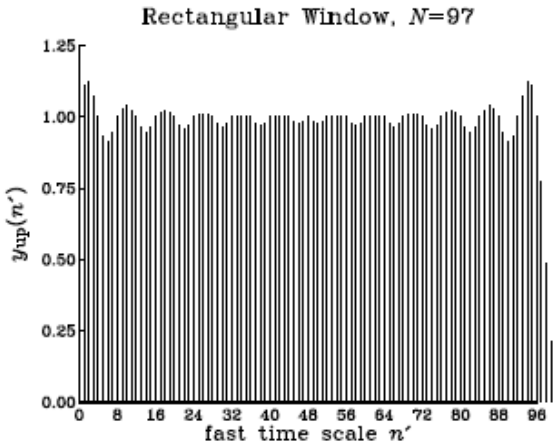


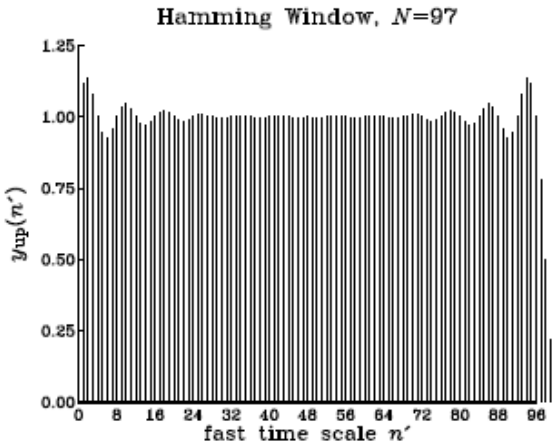


As another example, consider the case of $L = 4$ and $M = 12$, that is, interpolation filter length $N = 2LM + 1 = 97$. This is a more realistic length for typical 4-fold oversampling digital filters used in audio playback systems. The corresponding rectangularly and Hamming windowed magnitude responses are shown below, as are the interpolated output signals from these two filters, for the same low-rate input signal $x(n)$. Note the longer input-on and input-off transients.

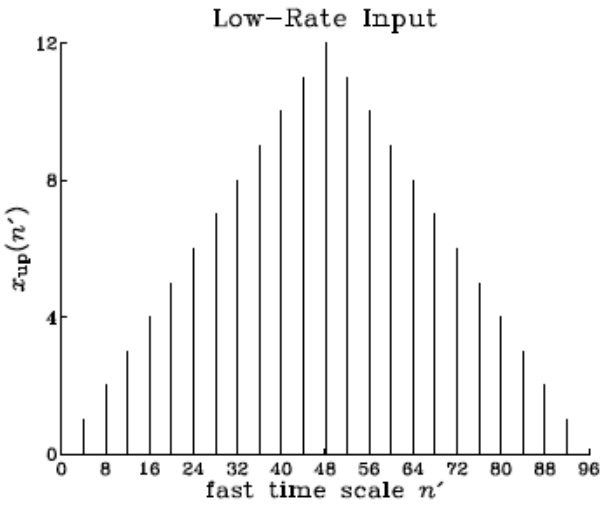


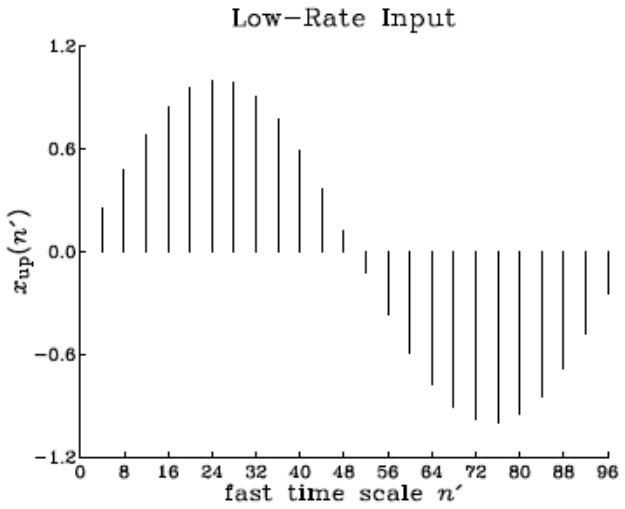


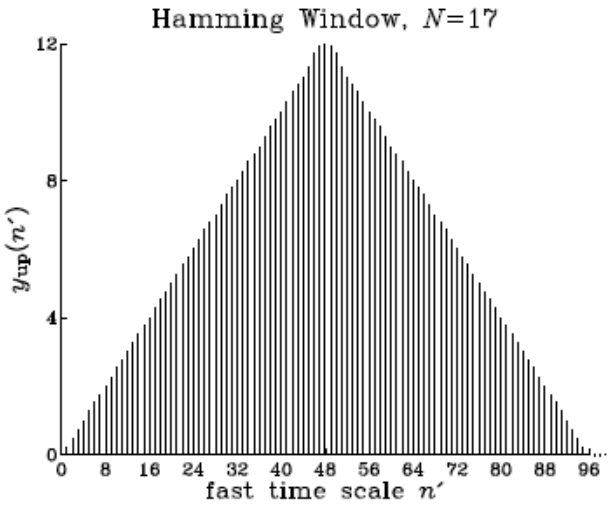


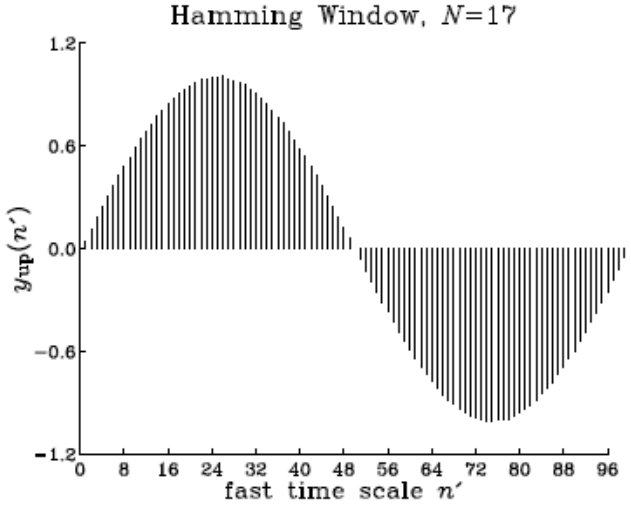


Here is another 4-fold interplation example for triangular and sinusoidal input signals using a length-17 Hamming design.









Next, consider a 4-fold interpolation filter of length $N = 25$. Here, we have

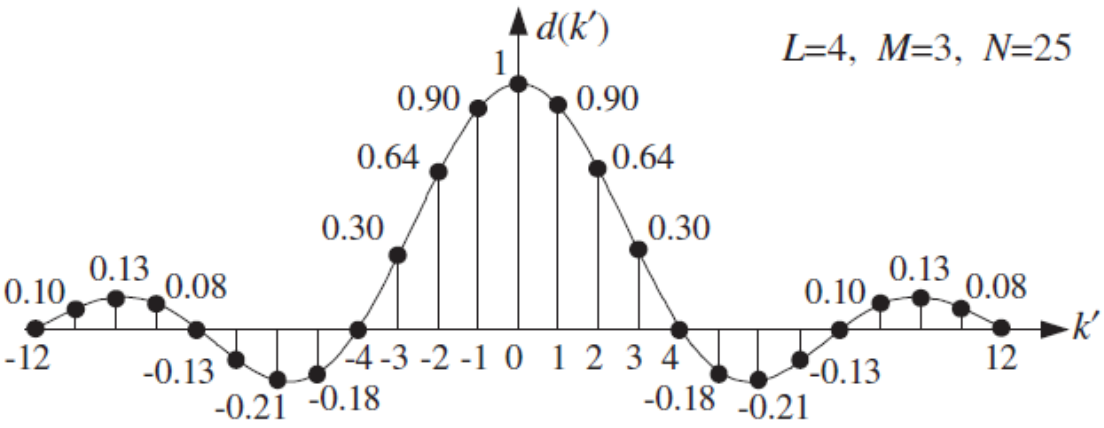
$$N = 2LM + 1 = 25 \quad \Rightarrow \quad M = (N-1)/2L = (25-1)/8 = 3$$

Thus, we use 3 low-rate samples above and three below every interpolated value to be computed. The length-25 ideal interpolation impulse response is calculated from, and shown below,

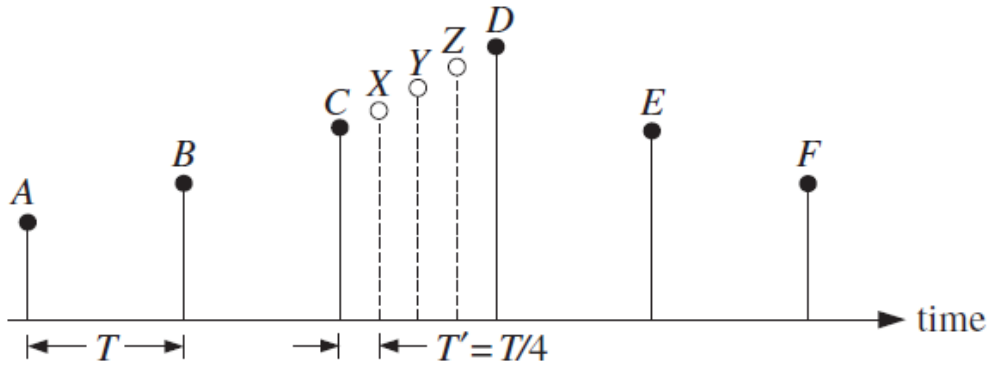
$$d(k') = \frac{\sin(\pi k'/4)}{\pi k'/4}, \quad \text{for} \quad -12 \le k' \le 12$$

This generates the numerical values (listed with 2-digit accuracy),

$$\begin{aligned} \mathbf{d} = \; [&0, \quad 0.08, \quad 0.13, \quad 0.10, \quad 0.00, \quad -0.13, \quad -0.21, \quad -0.18, \\ &0, \quad 0.30, \quad 0.64, \quad 0.90, \quad 1.00, \quad 0.90, \quad 0.64, \quad 0.30, \\ &0, \quad -0.18, \quad -0.21, \quad -0.13, \quad 0.00, \quad 0.10, \quad 0.13, \quad 0.08, \quad 0] \end{aligned}$$



Given $2M = 6$ low-rate samples $\{A, B, C, D, E, F\}$ as shown below,

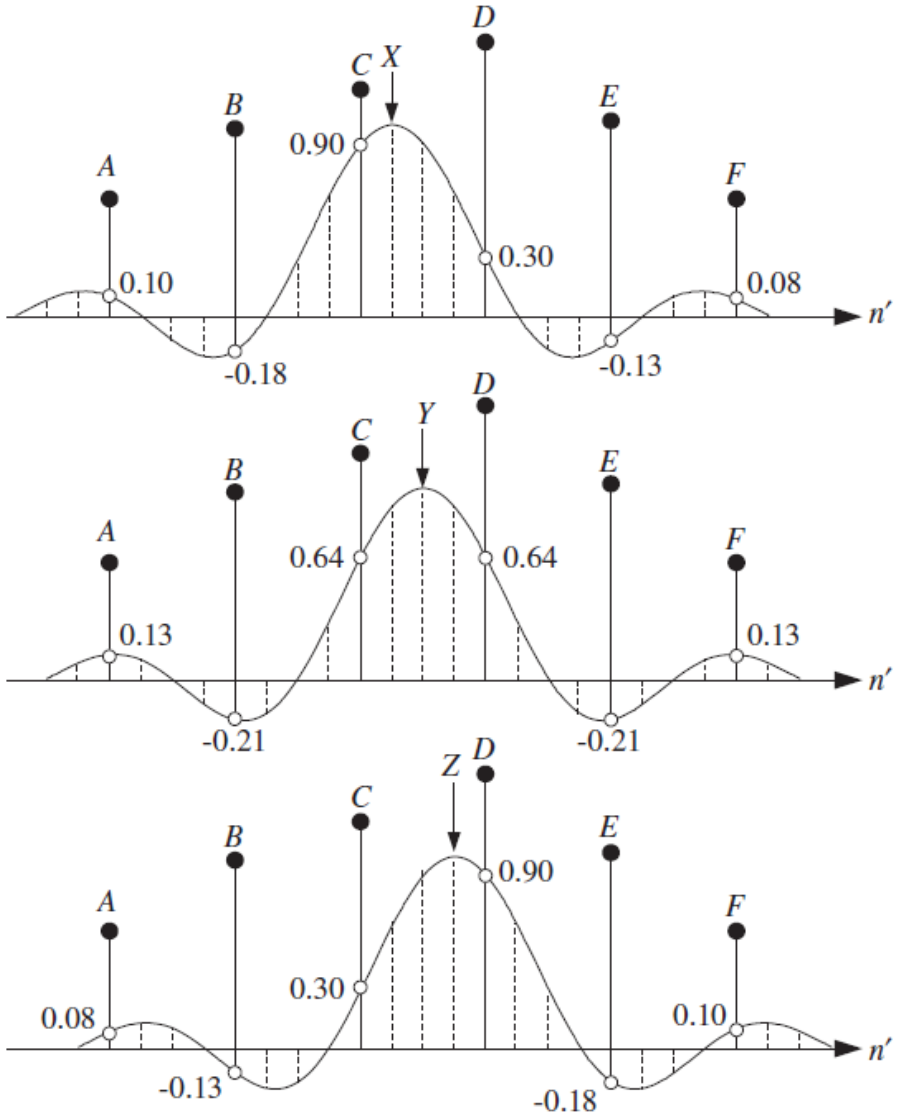


The three values $\{X, Y, Z\}$ interpolated between C and D are calculated by convolving **d** with the upsampled low-rate samples. Using the flip-and-slide form of convolution, we position the impulse response **d** at the three successive positions, and read off the values of $d(k')$ where it intersects with the low-rate samples, and add up the results. This is epicted below.

The corresponding linear combinations of low-rate samples may be rewritten in their polyphase matrix form (with the low-rate samples listed from the latest down to the earliest).

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.08 & -0.13 & 0.30 & 0.90 & -0.18 & 0.10 \\ 0.13 & -0.21 & 0.64 & 0.64 & -0.21 & 0.13 \\ 0.10 & -0.18 & 0.90 & 0.30 & -0.13 & 0.08 \end{bmatrix} \begin{bmatrix} F \\ E \\ D \\ C \\ B \\ A \end{bmatrix}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.08 & -0.13 & 0.30 & 0.90 & -0.18 & 0.10 \\ 0.13 & -0.21 & 0.64 & 0.64 & -0.21 & 0.13 \\ 0.10 & -0.18 & 0.90 & 0.30 & -0.13 & 0.08 \end{bmatrix} \begin{bmatrix} F \\ E \\ D \\ C \\ B \\ A \end{bmatrix}$$



DAC Equalization

In an oversampling DSP system, the interpolator output samples are reconstructed by a staircase D/A converter operating at the high rate $f_s' = Lf_s$. Its frequency response (normalized to unity gain at DC) is

$$H_{\text{dac}}(f) = \frac{\sin(\pi f / f_s')}{\pi f / f_s'} e^{-j\pi f / f_s'}$$

It causes some attenuation within the Nyquist interval, with maximum of about 4 dB at the Nyquist frequency $f_s'/2$. For an L-fold interpolation filter which has cutoff at $f_c = f_s/2 = f_s'/2L$, the maximum attenuation within the filter's passband will be:

$$|H_{\text{dac}}(f_c)| = \left| \frac{\sin(\pi f_c / f_s')}{\pi f_c / f_s'} \right| = \frac{\sin(\pi/2L)}{\pi/2L}$$

For large values of the oversampling ratio L, this attenuation is insignificant, approaching 0 dB. Thus, one of the benefits of oversampling is that the aperture effect of the DAC can be neglected.

However, for smaller values of L (for example, $L \leq 8$) it may be desirable to compensate this attenuation by designing the interpolation filter to have an *inverse* shape to the $\sin x/x$ DAC response over the relevant passband range.

The desired *equalized* ideal interpolation filter can then be defined by the following equation:

$$D(f) = \begin{cases} LD_{\text{eq}}(f), & \text{if } |f| \leq \frac{f_s}{2} \\ 0, & \text{if } \frac{f_s}{2} < |f| \leq \frac{f_s'}{2} \end{cases}$$

where $D_{\text{eq}}(f)$ is essentially the inverse response $1/H_{\text{dac}}(f)$ with the phase removed in order to keep $D(f)$ real and even in f:

$$D_{\text{eq}}(f) = \frac{\pi f/f_s'}{\sin(\pi f/f_s')}, \qquad |f| \leq \frac{f_s}{2}$$

In units of the high-rate digital frequency $\omega' = 2\pi f/f_s'$, we have,

$$D(\omega') = \begin{cases} L\,\dfrac{\omega'/2}{\sin(\omega'/2)}, & \text{if } |\omega'| \leq \dfrac{\pi}{L} \\ 0, & \text{if } \dfrac{\pi}{L} < |\omega'| \leq \pi \end{cases}$$

Such a filter can be designed by the frequency sampling design method. If the filter order is known, say $N = 2LM + 1$, then we can compute the desired filter weights by the inverse N-point DFT:

$$\widetilde{d}(k') = \frac{1}{N} \sum_{i=-LM}^{LM} D(\omega_i') e^{j\omega_i' k'}, \qquad -LM \le k' \le LM$$

where ω_i' are the N DFT frequencies spanning the symmetric Nyquist interval $[-\pi, \pi]$:

$$\omega_i' = \frac{2\pi i}{N}, \qquad -LM \le i \le LM$$

The designed causal windowed filter will be,

$$h(n') = \widetilde{d}(n' - LM)\, w(n'), \qquad 0 \le n' \le N - 1$$

In the Hamming window case, we must assume a desired value for N. In the Kaiser case, we may start with a desired stopband attenuation A and transition width Δf, and then determine the filter length N and the window parameter α. Because the filter is sloping upwards in the passband, to achieve a true attenuation A in the stopband, we may have to carry out the design with a slightly larger value of A.

Note also that because $\widetilde{d}(k')$ and $D(\omega')$ are real-valued, we may replace the right-hand side of the IDFT by its real part and write it in the cosine form:

$$\widetilde{d}(k') = \frac{1}{N} \sum_{i=-LM}^{LM} D(\omega_i') \cos(\omega_i' k'), \qquad -LM \leq k' \leq LM$$

and because $D(\omega')$ is even in ω'

$$\widetilde{d}(k') = \frac{1}{N} \left[D(\omega_0') + 2 \sum_{i=1}^{LM} D(\omega_i') \cos(\omega_i' k') \right], \qquad -LM \leq k' \leq LM$$

where $D(\omega_0') = D(0) = L$. This expression can be simplified even further by noting that $D(\omega_i')$ is non-zero only for

$$< \omega_i' < \frac{\pi}{L} \quad \Rightarrow \quad 0 < \frac{2\pi i}{N} < \frac{\pi}{L} \quad \Rightarrow \quad 0 < i < \frac{N}{2L} = \frac{2LM+1}{2L} = M + \frac{1}{2L}$$

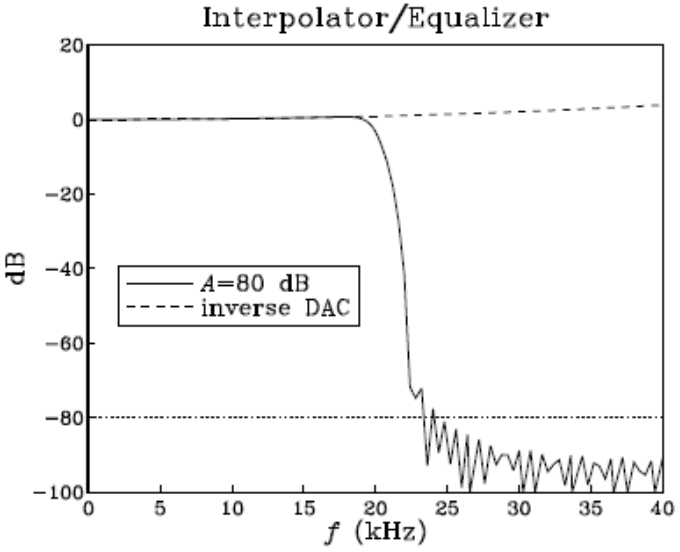
Thus, the summation can be restricted over $1 \leq i \leq M$, giving

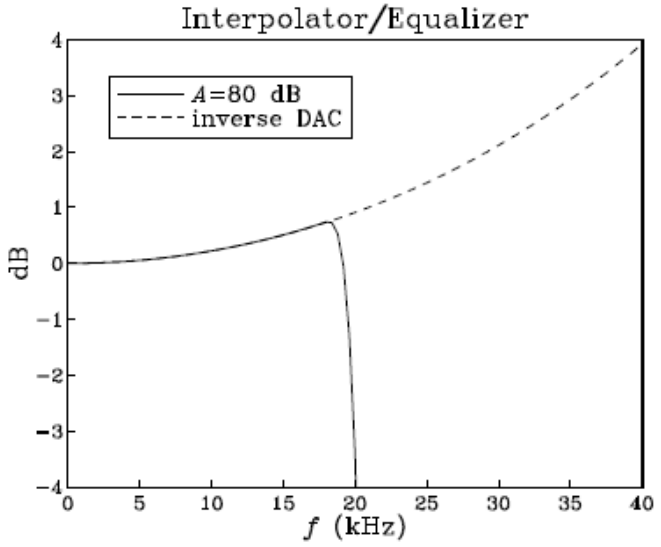
$$\widetilde{d}(k') = \frac{L}{N} \left[1 + 2 \sum_{i=1}^{M} \frac{\omega_i'/2}{\sin(\omega_i'/2)} \cos(\omega_i' k') \right], \qquad -LM \leq k' \leq LM$$

As a first example, consider a 2-times oversampling filter for digital audio. Assume a nominal audio rate of $f_s = 40$ kHz, transition width $\Delta f = 5$ kHz, and stopband attenuation $A = 80$ dB. The normalized width is $\Delta F = \Delta f / f_s = 0.125$. The Kaiser D parameter and filter length N will be

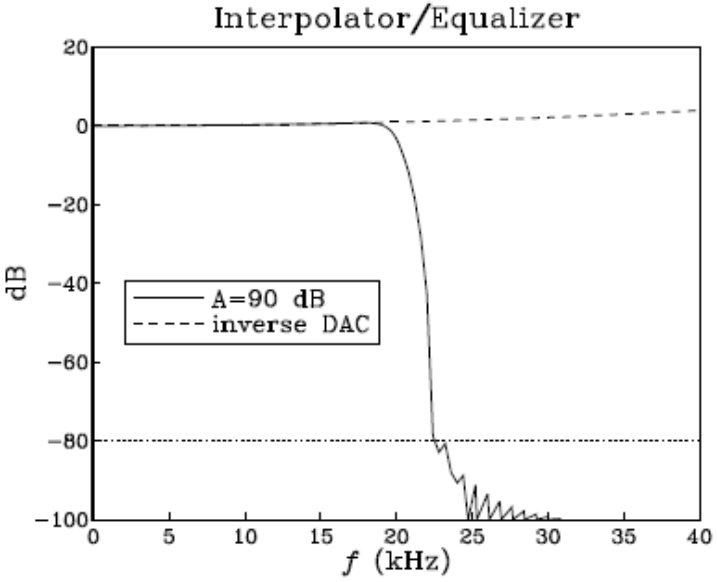
$$D = \frac{A - 7.95}{14.36} = 5.017, \qquad N - 1 \geq = \frac{DL}{\Delta F} = 80.3$$

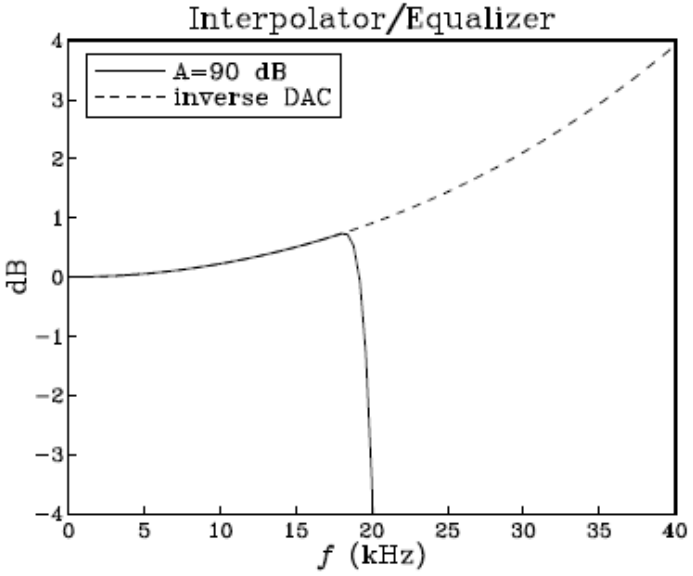
which rounds up to $N = 85$. The designed filter is shown below together with the inverse DAC response $1/|H_{\text{dac}}(f)|$, plotted over the high-rate Nyquist interval. On the right, the passband is shown in a magnified scale. Notice how the inverse DAC response reaches 4 dB at the Nyquist frequency of $f_s'/2 = 40$ kHz.





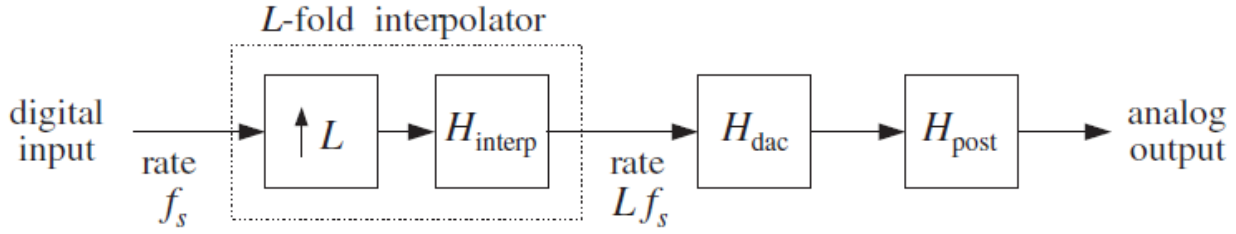
The actual stopband attenuation is somewhat less than the prescribed 80 dB, namely, about 72 dB at $f = f_s/2 + \Delta f/2 = 22.5$ kHz. Thus, we may wish to redesign the filter starting out with a somewhat larger attenuation. For example, assuming $A = 90$ dB, we obtain filter length $N = 93$. The redesigned filter is shown below, achieving 80 dB attenuation at 22.5 kHz.

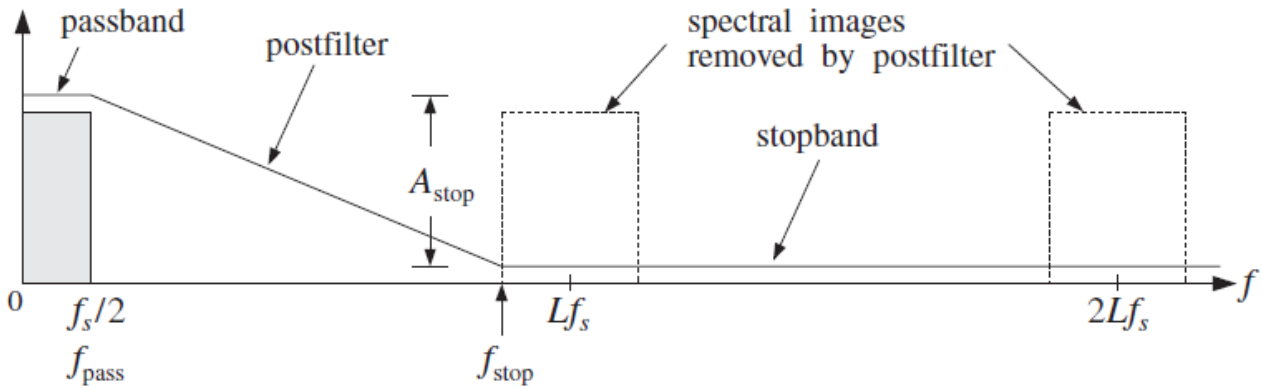




Postfilter Design and Equalization

In addition to compensating for the attenuation of the DAC, one may wish to compensate for other effects. For example, the staircase output of the DAC will be fed into an analog anti-image postfilter which introduces its own slight attenuation within the desired passband. This attenuation can be equalized digitally by the interpolation filter.





Because of oversampling, the postfilter will have a wide transition region resulting in low filter order, such as 2 or 3. The postfilter must provide enough attenuation in its stopband to remove the spectral images of the interpolated signal at multiples of f_s', which cannot be removed by the interplator.

Its passband must extend up to the low-rate Nyquist frequency, and its stopband begins at the left edge of the first spectral image, that is,

$$\boxed{f_{\text{pass}} = \frac{f_s}{2}, \qquad f_{\text{stop}} = f_s{}' - f_{\text{pass}} = Lf_s - \frac{f_s}{2}}$$

At f_{stop}, the DAC already provides a certain amount of attenuation given by:

$$|H_{\text{dac}}(f_{\text{stop}})| = \left|\frac{\sin(\pi f_{\text{stop}}/f_s{}')}{\pi f_{\text{stop}}/f_s{}'}\right| = \frac{\sin(\pi - \pi/2L)}{\pi - \pi/2L} = \frac{\sin(\pi/2L)}{\pi - \pi/2L}$$

which, for large L, becomes approximately:

$$|H_{\text{dac}}(f_{\text{stop}})| = \frac{\sin(\pi/2L)}{\pi - \pi/2L} \simeq \frac{1}{2L}$$

or, in dB

$$A_{\text{dac}} \simeq 20\log_{10}(2L)$$

The analog postfilter must supply an additional amount of attenuation A_{stop}, raising the total attenuation at f_{stop} to a desired level, say A_{tot} dB:

$$A_{\text{tot}} = A_{\text{dac}} + A_{\text{stop}}$$

For example, suppose $f_s = 40$ kHz and $L = 4$, and we require the total suppression of the replicas to be more than $A_{\text{tot}} = 60$ dB. The stopband frequency will be $f_{\text{stop}} = Lf_s - f_s/2 = 160 - 20 = 140$ kHz. At that frequency the DAC will provide an attenuation $A_{\text{dac}} = 20\log_{10}(8) = 18$ dB, and therefore the postfilter must provide the rest:

$$A_{\text{stop}} = A_{\text{tot}} - A_{\text{dac}} = 60 - 18 = 42 \text{ dB}$$

Suppose we use a third-order Butterworth filter with magnitude response,

$$|H_{\text{post}}(f)|^2 = \frac{1}{1 + \left(\frac{f}{f_0}\right)^6}$$

and attenuation in dB:

$$A_{\text{post}}(f) = -10\log_{10}|H_{\text{post}}(f)|^2 = 10\log_{10}\left[1 + \left(\frac{f}{f_0}\right)^6\right]$$

where f_0 is the 3-dB normalization frequency to be determined.

The requirement that at f_{stop} the attenuation be equal to A_{stop} gives:

$$A_{\text{stop}} = 10\log_{10}\left[1 + \left(\frac{f_{\text{stop}}}{f_0}\right)^6\right]$$

which can be solved for f_0:

$$f_0 = f_{\text{stop}}\left[10^{A_{\text{stop}}/10} - 1\right]^{-1/6} = 140 \cdot \left[10^{42/10} - 1\right]^{-1/6} = 28 \text{ kHz}$$

The third-order Butterworth analog transfer function of the postfilter is:

$$H_{\text{post}}(s) = \frac{1}{1 + 2\left(\frac{s}{\Omega_0}\right) + 2\left(\frac{s}{\Omega_0}\right)^2 + \left(\frac{s}{\Omega_0}\right)^3}$$

where $\Omega_0 = 2\pi f_0$. This postfilter will adequately remove the spectral images at multiples of $f_s{}'$, but it will also cause a small amount of attenuation within the desired passband. The maximum passband attenuations caused by the postfilter and the DAC at $f_{\text{pass}} = f_s/2$ can be computed as follows,

$$A_{\text{post}}(f_{\text{pass}}) = 10\log_{10}\left[1 + \left(\frac{f_{\text{pass}}}{f_0}\right)^6\right] = 10\log_{10}\left[1 + \left(\frac{20}{28}\right)^6\right] = 0.54 \text{ dB}$$

$$A_{\text{dac}}(f_{\text{pass}}) = -20\log_{10}\left[\frac{\sin(\pi/2L)}{\pi/2L}\right] = 0.22 \text{ dB}$$

resulting in a total passband attenuation of $0.54 + 0.22 = 0.76$ dB.

Such combined attenuation of the DAC and postfilter can be equalized by the interpolator filter. Using the frequency sampling design, we replace the interpolator's defining equation by the doubly equalized version:

$$D(\omega') = \begin{cases} L\left[\dfrac{\omega'/2}{\sin(\omega'/2)}\right]\left[1+\left(\dfrac{\omega'}{\omega_0}\right)^6\right]^{1/2}, & \text{if } |\omega'| \le \dfrac{\pi}{L} \\ 0, & \text{if } \dfrac{\pi}{L} < |\omega'| \le \pi \end{cases}$$

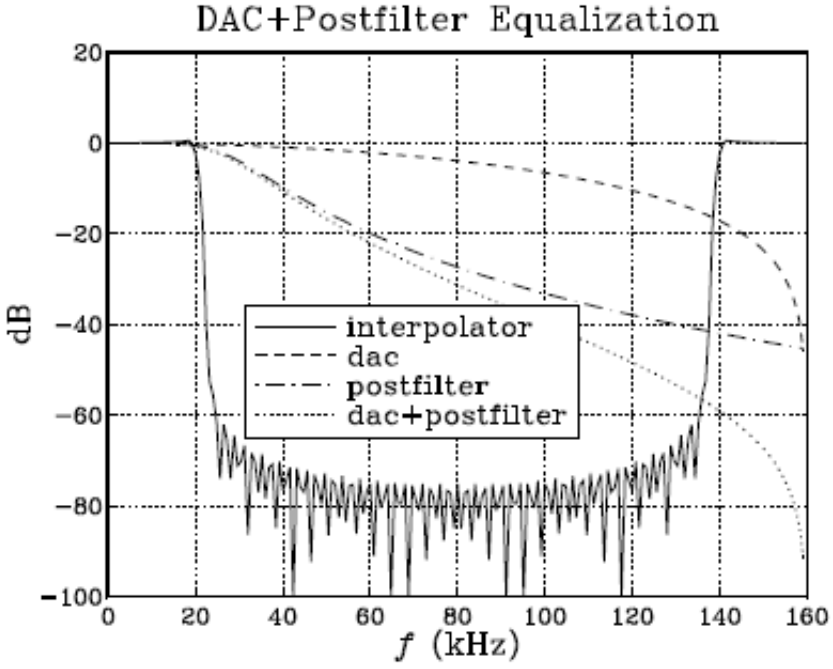
where $\omega_0 = 2\pi f_0 / f_s{}'$, with impulse response coefficients calculated by:

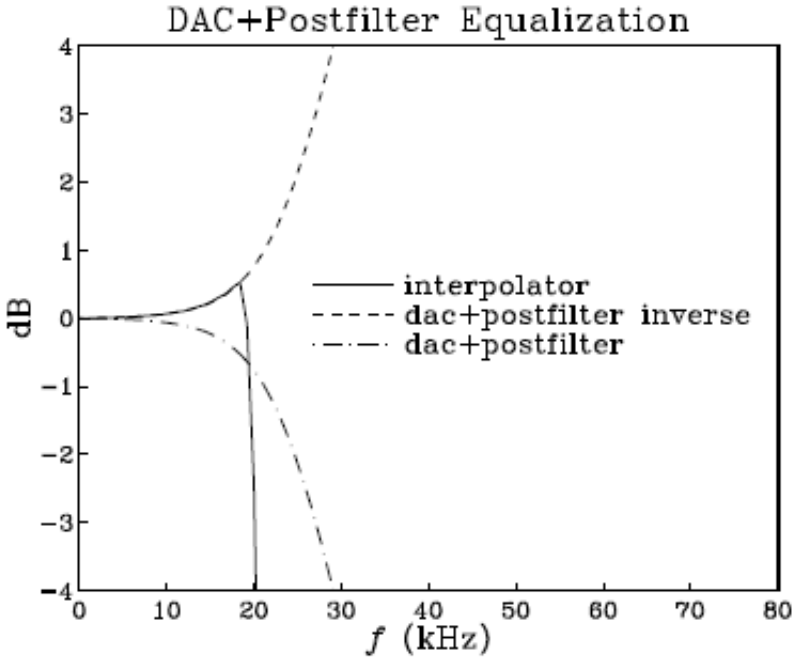
$$\widetilde{d}(k') = \frac{L}{N}\left[1 + 2\sum_{i=1}^{M}\left[\frac{\omega_i'/2}{\sin(\omega_i'/2)}\right]\left[1+\left(\frac{\omega_i'}{\omega_0}\right)^6\right]^{1/2}\cos(\omega_i' k')\right]$$

for $-LM \le k' \le LM$. These coefficients must be weighted by an appropriate window. The figure below shows shows a Kaiser design corresponding to interpolator stopband attenuation of $A = 60$ dB and a transition width of $\Delta f = 5$ kHz. As before, the resulting filter length is $N = 121$.

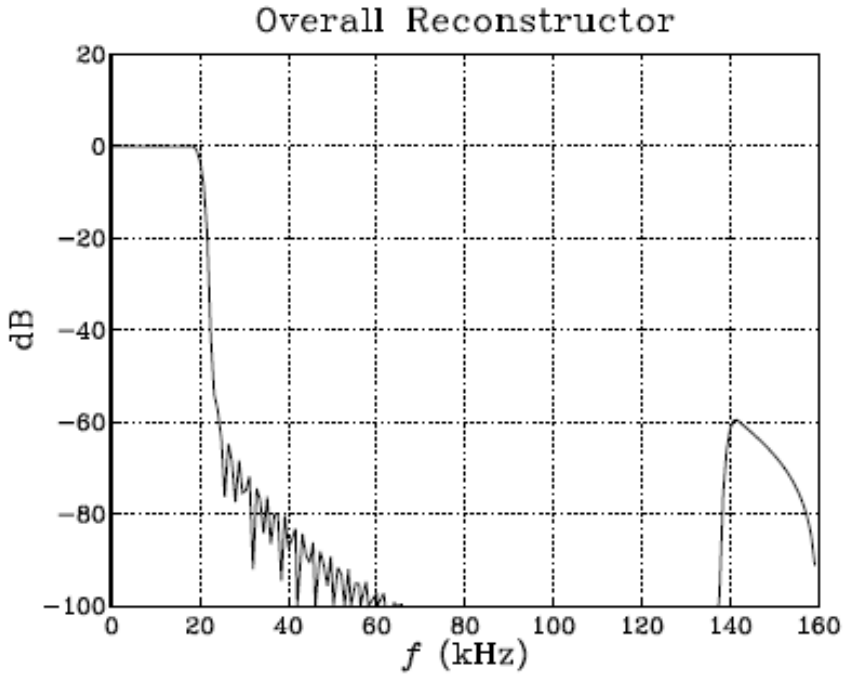
For reference, the DAC response $H_{\text{dac}}(f)$, postfilter response $H_{\text{post}}(f)$, and total response $H_{\text{dac}}(f)H_{\text{post}}(f)$ are superimposed on the figure. Notice how they meet their respective specifications at $f_{\text{stop}} = 140$ kHz. The DAC response vanishes (i.e., it has infinite attenuation) at $f_s{}' = 160$ kHz and all its multiples.

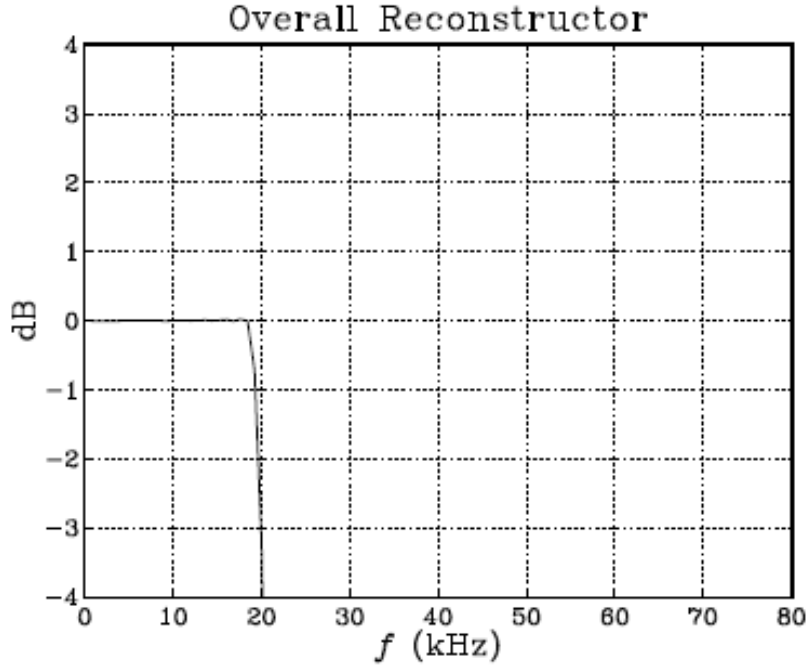
The figure also shows the filter's passband in a magnified scale, together with the plots of the total filter $H_{\text{dac}}(f)H_{\text{post}}(f)$ and total inverse filter $1/\big(H_{\text{dac}}(f)H_{\text{post}}(f)\big)$.





The effective overall analog reconstructor $H_{\text{interp}}(f)H_{\text{dac}}(f)H_{\text{post}}(f)$, consisting of the equalized interpolator, DAC, and postfilter, is shown below. The spectral images at multiples of $f_s' = 160$ kHz are suppressed by more than 60 dB and the 20 kHz passband is essentially flat. The figure also shows the passband in a magnified scale.





Bessel Postfilters

In digital audio applications, Bessel postfilters may also be used instead of Butterworth filters. Bessel filters provide the additional benefit that they have approximately *linear phase* over their passband.

The transfer function and magnitude response of a third-order Bessel filter are given by,

$$H_{\text{post}}(s) = \frac{15}{15 + 15\left(\frac{s}{\Omega_0}\right) + 6\left(\frac{s}{\Omega_0}\right)^2 + \left(\frac{s}{\Omega_0}\right)^3}$$

$$|H_{\text{post}}(f)|^2 = \frac{225}{225 + 45\left(\frac{f}{f_0}\right)^2 + 6\left(\frac{f}{f_0}\right)^4 + \left(\frac{f}{f_0}\right)^6}$$

where $\Omega_0 = 2\pi f_0$ and f_0 is related to the 3-dB frequency of the filter by

$$f_{3\text{dB}} = 1.75 f_0$$

The passband attenuation of this filter can be equalized digitally in a similar fashion. For equal 3-dB frequencies, Bessel filters fall off somewhat less sharply than Butterworth ones, thus, suppressing the spectral images by a lesser amount.

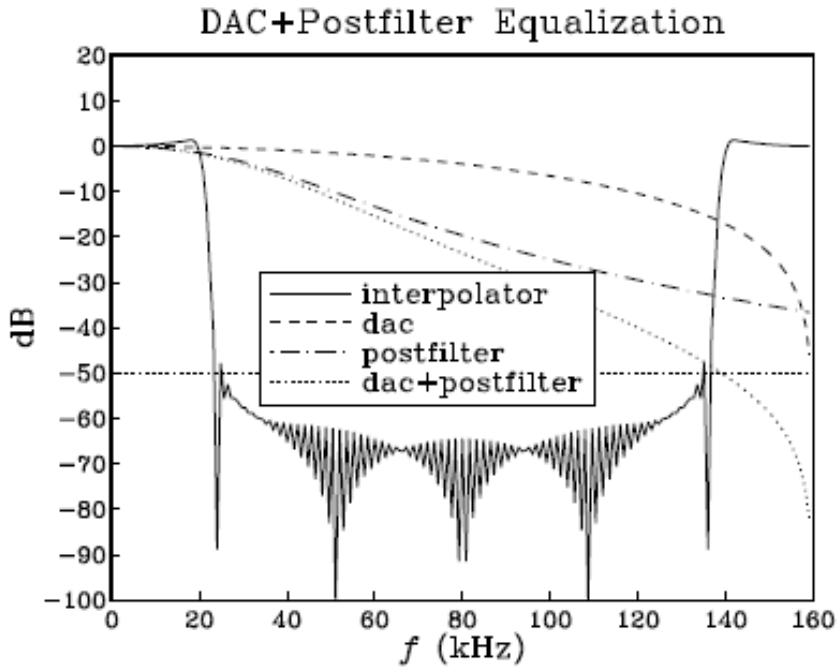
For example, for the previous 3-dB frequency $f_{3\text{dB}} = 28$ kHz, we find the normalization frequency $f_0 = f_{3\text{dB}}/1.75 = 16$ kHz. The corresponding postfilter attenuations at the passband and stopband frequencies, $f_{\text{pass}} = 20$ kHz and $f_{\text{stop}} = 140$ kHz, calculated to be:

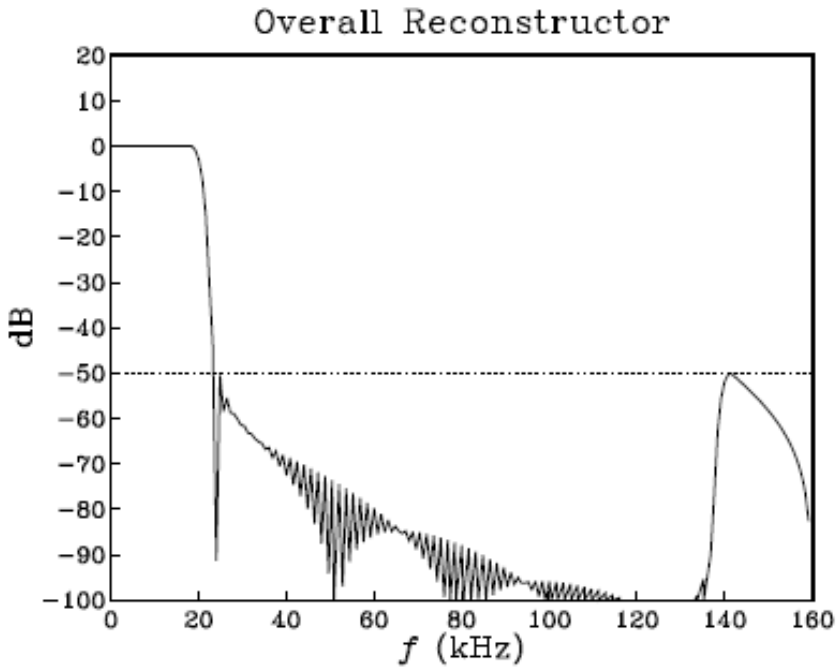
$$A_{\text{post}}(f_{\text{pass}}) = 1.44 \text{ dB}, \qquad A_{\text{post}}(f_{\text{stop}}) = 33 \text{ dB}$$

Thus, the DAC/postfilter combination will only achieve a total stopband attenuation of $33 + 18 = 51$ dB for the removal of the spectral images. Similarly, the total passband attenuation to be compensated by the interpolator will be $0.22 + 1.44 = 1.66$ dB. If 51 dB suppression of the spectral images is acceptable, then we may redesign the interpolator so that it suppresses its stopband also by 51 dB. With $A = 51$ and $L = 4$, the redesigned interpolator will have Kaiser parameters:

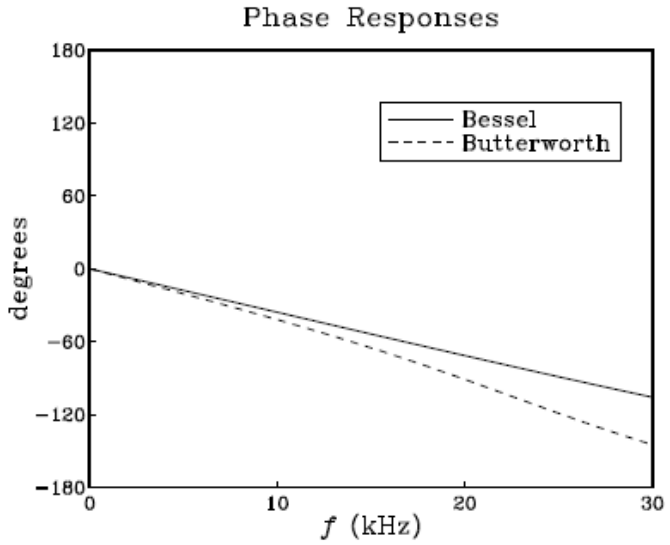
$$D = \frac{A - 7.95}{14.36} = 2.998\,, \quad N - 1 \geq = \frac{DL}{\Delta F} = 95.93 \quad \Rightarrow \quad N = 97$$

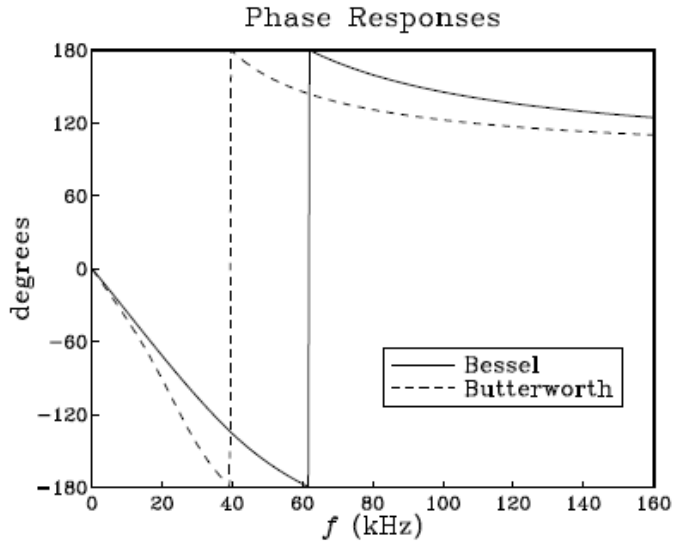
The redesigned equalized interpolation filter and the effective overall reconstruction filter are shown below. The overall reconstructor has a flat passband and suppresses all spectral images by at least 51 dB.

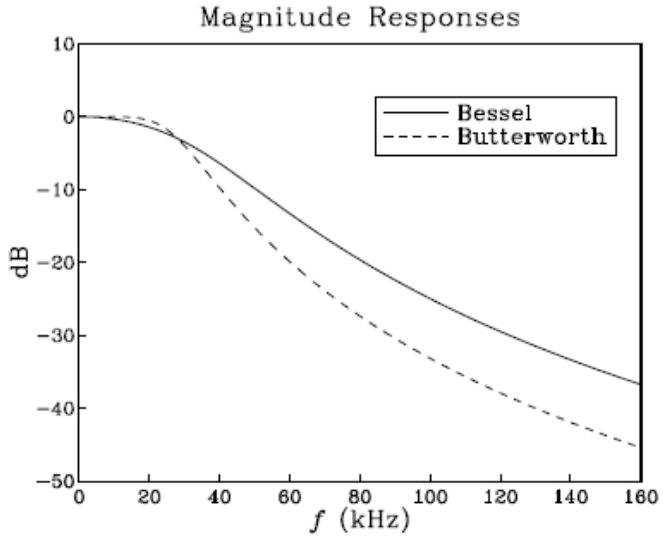




The differences between the phase responses and between the magnitude responses of the above Butterworth and Bessel postiflters are shown in the graphs below. As before, the 3-dB frequencies were $f_0 = 28$ kHz for the Butterworth filter, and $f_0 = 16$ kHz for Bessel.





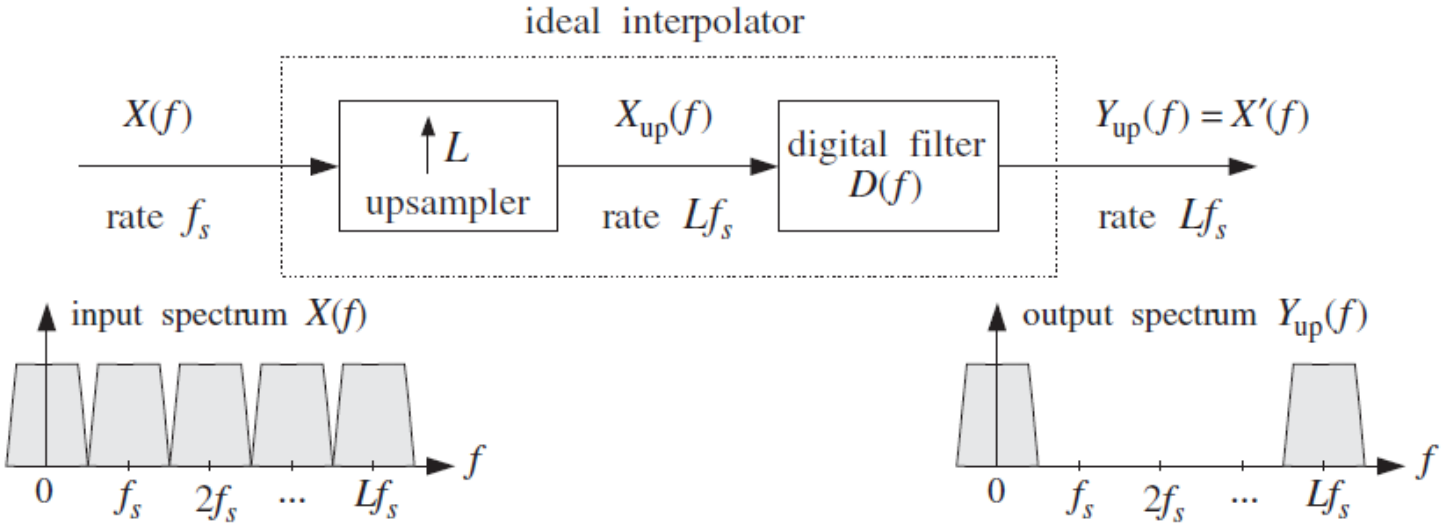


Decimation and Oversampling

Decimation by an integer factor L is the reverse of interpolation, that is, decreasing the sampling rate from the high rate $f_s{}'$ to the lower rate f_s,

$$f_s{}' \Rightarrow f_s = \frac{f_s{}'}{L}$$

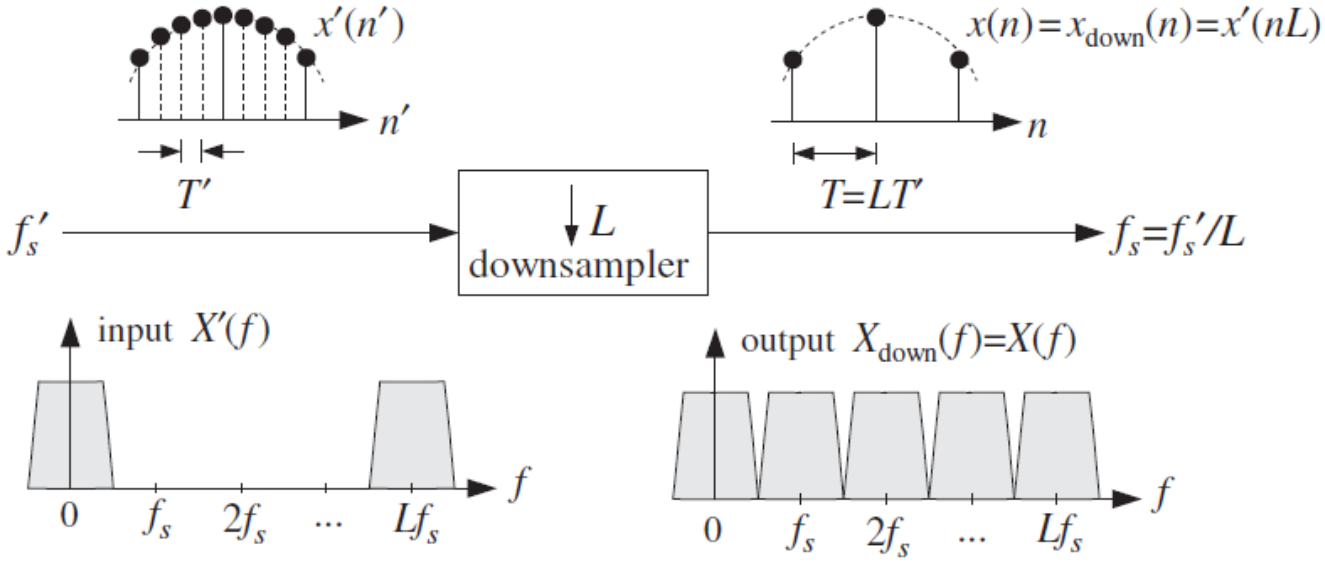
An ideal interpolator replaces a low-rate signal $x(n)$ by the high-rate interpolated signal $x'(n')$, which would ideally correspond to the resampling of the analog signal at the higher rate. As shown below,



where, the spectrum $X'(f)$ of $x'(n')$ is the spectrum of $x(n)$ with $L-1$ spectral images removed between multiples of $f_s{}'$.

This ideal interpolation process can be reversed by keeping from $x'(n')$ every Lth sample and discarding the $L-1$ samples that were interpolated between the low-rate ones.

This process of downsampling and its effect in the time and frequency domains is depicted below.



Formally, the downsampled signal is defined in terms of the slow time scale as follows:

$$x_{\text{down}}(n) = x'(n')\Big|_{n'=nL} = x'(nL)$$

For the ideal situation depicted above, the downsampled signal $x_{\text{down}}(n)$ *coincides* with the low-rate signal $x(n)$ that would have been obtained had the analog signal been resampled at the lower rate f_s, that is,

$$x(n) = x_{\text{down}}(n) = x'(nL)$$

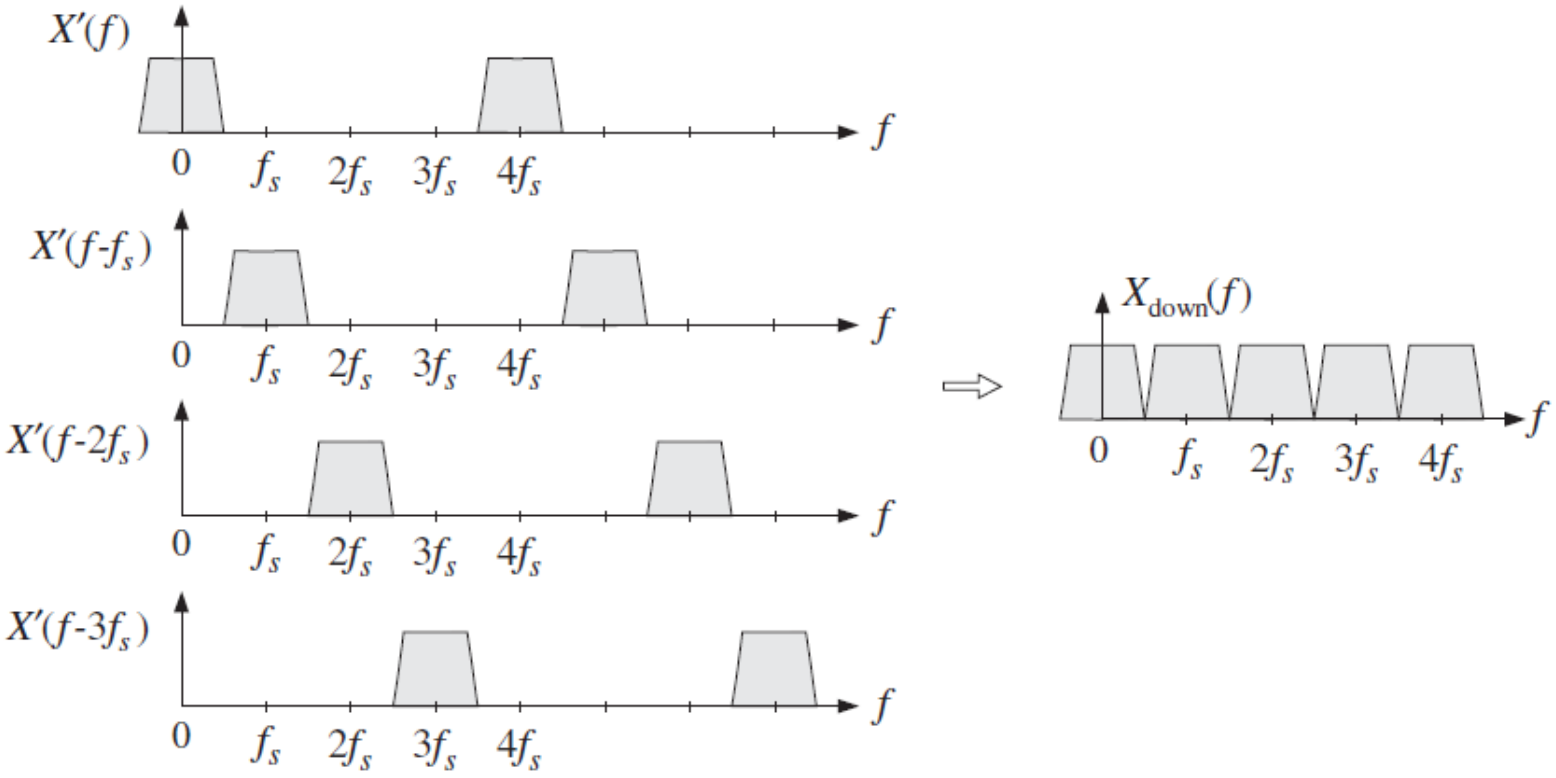
The gaps in the input spectrum $X'(f)$ are necessary to guarantee this equality. Dropping the sampling rate by a factor of L, shrinks the Nyquist interval $[-f_s'/2, f_s'/2]$ by a factor of L to the new interval $[-f_s/2, f_s/2]$. Thus, if the signal had frequency components outside the new Nyquist interval, aliasing would occur and $x_{\text{down}}(n) \neq x(n)$.

In the above figure, the input spectrum was already restricted to the f_s Nyquist interval, and therefore, aliasing did not occur. The rate decrease causes the spectral images of $X'(f)$ at multiples of f_s' to be *down shifted* and become images of $X(f)$ at multiples of f_s without overlapping.

The mathematical justification of this down-shifting property is derived by expressing the equation, $x_{\text{down}}(n) = x'(nL)$, in the frequency domain. It can be shown (see I2SP–Problem 12.12) that:

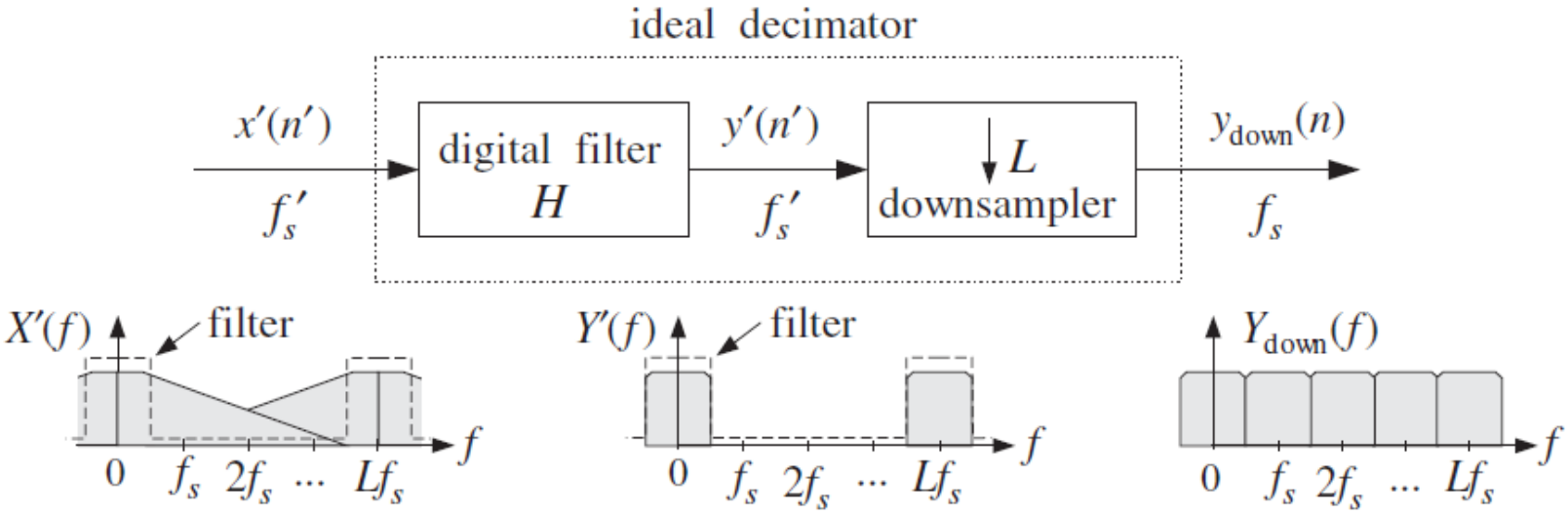
$$X_{\text{down}}(f) = \frac{1}{L} \sum_{m=0}^{L-1} X'(f - mf_s)$$

Therefore, the downsampling process causes the periodic replication of the original spectrum $X'(f)$ at multiples of the low rate f_s. This operation is depicted below for $L = 4$.



In general, if the high-rate signal $x'(n')$ has frequency components outside the low-rate Nyquist interval $[-f_s/2, f_s/2]$, then downsampling alone is not sufficient to perform decimation. For example, noise in the signal, such as quantization noise arising from the A/D conversion process, will have such frequency components.

To avoid the aliasing that will arise by the downsampling spectrum replication property, the high-rate input $x'(n')$ must be prefiltered by a digital lowpass filter, called the decimation filter. The combined filter/downsampler system is called a decimator and is depicted below.



The filter operates at the high rate f_s' and has cutoff frequency $f_c = f_s/2 = f_s'/2L$. It is similar to the ideal interpolation filter, except its DC gain is unity instead of L. The high-rate output of the filter is downsampled to obtain the desired low-rate decimated signal, with non-overlapping down-shifted replicas:

$$y_{\text{down}}(n) = y'(nL), \qquad Y_{\text{down}}(f) = \frac{1}{L}\sum_{m=0}^{L-1} Y'(f - mf_s)$$

The design of the decimation filter is identical to that of the interpolation filter. For example, a length-N FIR decimator can be obtained by windowing the (causal) ideal impulse response:

$$h(n') = w(n')d(n' - LM), \qquad \text{where} \quad d(k') = \frac{\sin(\pi k'/L)}{\pi k'}$$

where $n' = 0, 1, \ldots, N-1$, and $N = 2LM + 1$. A Kaiser window $w(n')$ may be used. The downsampled output is obtained by:

$$\boxed{y_{\text{down}}(n) = y'(nL) = \sum_{m'=0}^{N-1} h(m')x'(nL - m')}$$

Because only every Lth output of the filter is needed, the overall computational rate is reduced by a factor of L, that is,

$$R = \frac{1}{L} N f_s{}' = N f_s$$

This is similar to the savings of the polyphase form of interpolation.

A simple implementation uses a length-N tapped delay line into which the high-rate input samples are shifted at the high rate $f_s{}'$. Every L inputs, its contents are used to perform the filter's dot product output computation.

Denoting the N-dimensional column-vectors of the impulse response and of the internal states of the filter by,

$$\mathbf{h} = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-1} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{bmatrix}$$

then, we may state the filtering/downsampling algorithm as follows:

for each high-rate input sample x' do:
$w_0 = x'$
for every Lth input, compute output:
$y_{\text{down}} = \mathbf{h}^T \mathbf{w}$
delay$(N-1, \mathbf{w})$]

where the function, delay$(N-1, \mathbf{w})$, represents the shifting of the delay line, that is, replacing the current $\mathbf{w}$, by the next one,

$$\mathbf{w} = \big[\, w_0, \underbrace{w_0,\ w_1,\ \ldots,\ w_{N-2}}_{\text{shifted}} \,\big]^T$$

Multistage implementations of decimators are also possible. The proper ordering of the decimation stages is the *reverse* of the interpolation case, that is, the decimator with the most *stringent* specifications is placed *last*.

Often, the earlier decimators, which also have the highest rates, are chosen to have *simplified* structures, such as simple averaging filters. For example, the decimation version of the hold interpolator is obtained by dividing by L to restore its DC gain to unity:

$$H(\zeta) = \frac{1}{L}\,\frac{1-\zeta^{-L}}{1-\zeta^{-1}} = \frac{1}{L}\left[1+\zeta^{-1}+\zeta^{-2}+\cdots+\zeta^{-(L-1)}\right]$$

where ζ^{-1} is one high-rate delay. Thus, the decimator becomes a simple FIR *averaging* filter that averages L successive high-rate samples:

$$y_{\text{down}}(n) = \frac{x'(nL)+x'(nL-1)+x'(nL-2)+\cdots+x'(nL-L+1)}{L}$$

If so desired, the cruder passbands of the earlier decimators can be equalized by the last decimator, which can also equalize any imperfect passband of the analog antialiasing prefilter used prior to sampling.

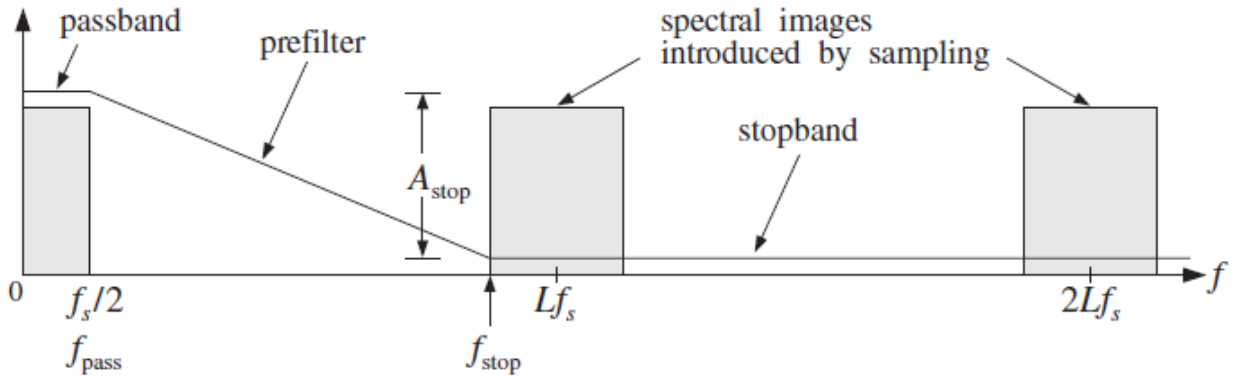
One of the main uses of decimators is to alleviate the need for high-quality analog prefilters, much as the interpolators ease the specifications of the anti-image postfilters.

This idea is used in many current applications, such as the sampling systems of DAT machines, PC sound cards, speech CODECs, and various types of delta-sigma A/D converter chips.

Sampling an analog signal, such as audio, at its nominal Nyquist rate f_s would require a high-quality analog prefilter to bandlimit the input to the Nyquist frequency $f_{\text{max}} = f_s/2$.

In a sampling system that uses oversampling and decimation, the analog input is first prefiltered by a simple prefilter and then sampled at the higher rate $f_s' = Lf_s$.

The decimation filter then reduces the bandwidth of the sampled signal to $f_s/2$. The sharp cutoffs at the Nyquist frequency $f_s/2$ are provided by the digital decimation filter instead of the prefilter. The specifications of the analog prefilter and decimator are shown below.



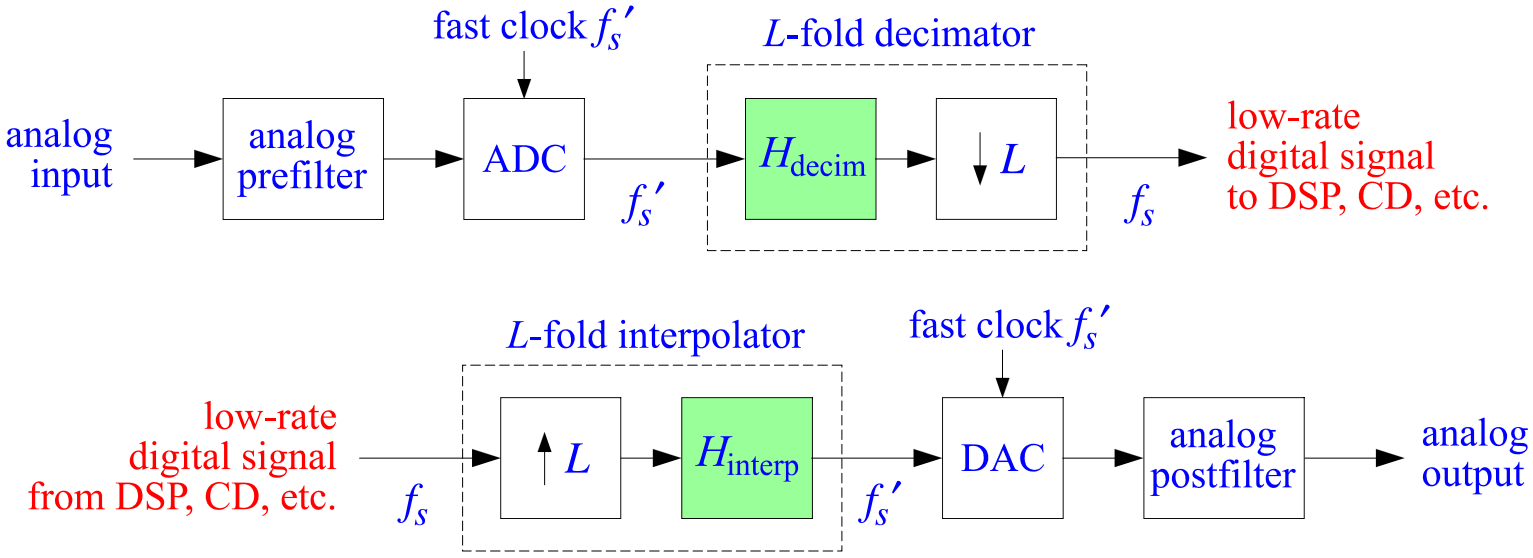
The decimator removes all frequencies from the range $[f_s/2, Lf_s - f_s/2]$. But because of periodicity, it cannot remove any frequencies in the range $Lf_s \pm f_s/2$. Such frequencies, if present in the analog input, must be removed by the prefilter prior to sampling; otherwise they will be aliased back into the desired Nyquist interval $[-f_s/2, f_s/2]$. Therefore, the prefilter's passband and stopband frequencies are:

$$\boxed{f_{\text{pass}} = \frac{f_s}{2}, \qquad f_{\text{stop}} = Lf_s - \frac{f_s}{2}}$$

The transition width of the prefilter is $\Delta f = f_{\text{stop}} - f_{\text{pass}} = (L-1)f_s$ and gets wider with the oversampling ratio L. Hence, the filter's complexity reduces with increasing L.

In summary, oversampling in conjunction with decimation and interpolation alleviates the need for high-quality analog prefilters and postfilters by assigning the burden of achieving sharp transition characteristics to the *digital* filters.

The figure below shows an oversampling DSP system in which sampling and reconstruction are carried out at the fast rate f_s', and any intermediate digital processing at the low rate f_s.



A second major benefit of oversampling is that it also simplifies the structure of the A/D and D/A converters shown in the figure, so that they require fewer bits without sacrificing quality.

This is accomplished by the principle of feedback quantization, which we discuss later on. The changes in the above figure are to replace the conventional ADC block by a delta-sigma ADC operating at fewer bits (even 1 bit), and insert between the output interpolator and the DAC a noise shaping quantizer that requantizes the output to fewer bits.

Sampling Rate Converters

Interpolators and decimators are examples of sampling rate converters that change the rate by *integer* factors. A more general sampling rate converter can change the rate by an arbitrary rational factor, say L/M, so that the output rate will be related to the input rate by:

$$\boxed{f_s{}' = \frac{L}{M} f_s}$$

Such rate changes are necessary in practice for interfacing DSP systems that might be operating at different rates. For example, to convert digital audio for broadcasting, sampled at 32 kHz, to digital audio for a DAT machine, sampled at 48 kHz, one must use a conversion factor of $48/32 = 3/2$. Similarly, to convert DAT audio to CD audio at 44.1 kHz, one must use the factor $44.1/48 = 147/160$.

The rate conversion can be accomplished by first increasing the rate by a factor of L to the high rate $f_s{}'' = Lf_s$ using an L-fold interpolator, and then decreasing the rate by a factor of M down to $f_s{}' = f_s{}''/M = Lf_s/M$ using an M-fold decimator.

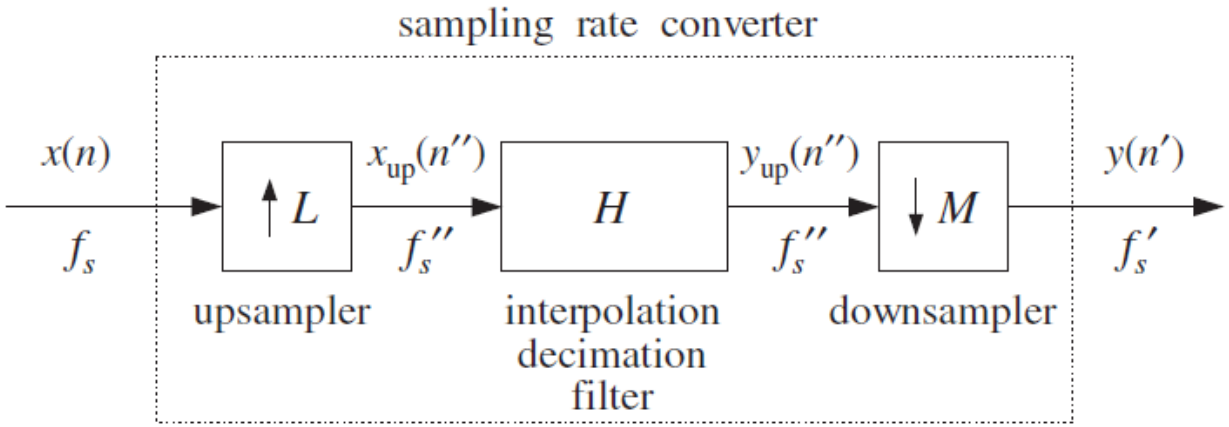
Note that f_s'' is an integer multiple of both the input and output rates, and the corresponding sampling time interval, $T'' = 1/f_s''$, is an integer fraction of both the input and output sampling times T and T':

$$f_s'' = Lf_s = Mf_s', \qquad T'' = \frac{T}{L} = \frac{T'}{M}$$

or,

$$\underbrace{f_s'' = Lf_s}_{\text{upsample}}, \quad \underbrace{f_s' = \frac{f_s''}{M}}_{\text{downsample}}, \quad \underbrace{T'' = \frac{T}{L}}_{\text{upsample}}, \quad \underbrace{T' = MT''}_{\text{downsample}}$$

Because both the interpolation and the decimation filters are operating at the same high rate of f_s'' and both are lowpass filters, they may be combined into a single lowpass filter preceded by an upsampler and followed by a downsampler, as shown below



The interpolation filter must have cutoff frequency $f_c = f_s''/2L = f_s/2$ and the decimation filter $f_c = f_s''/2M = f_s'/2$. Thus, the cutoff frequency of the common filter must be chosen to be the *minimum* of the two:

$$\boxed{f_c = \frac{1}{2}\min(f_s, f_s')}$$

which can be written also in the alternative forms:

$$f_c = \min\left(1, \frac{L}{M}\right)\frac{f_s}{2} = \min\left(\frac{M}{L}, 1\right)\frac{f_s'}{2} = \min\left(\frac{1}{L}, \frac{1}{M}\right)\frac{f_s''}{2}$$

In units of the high-rate digital frequency $\omega'' = 2\pi f/f_s''$, we have:

$$\boxed{\omega_c'' = \frac{2\pi f_c}{f_s''} = \min\left(\frac{\pi}{L}, \frac{\pi}{M}\right) = \frac{\pi}{\max(L, M)}}$$

When $f_s' > f_s$, the common filter acts as an anti-image postfilter for the upsampler, removing the spectral replicas at multiples of f_s but not at multiples of Lf_s. When $f_s' < f_s$, it acts as an antialiasing prefilter for the downsampler, making sure that the down-shifted replicas at multiples of f_s' do not overlap.

The design of the filter is straightforward. Assuming a filter length N of the form $N = 2LK + 1$ (we use K instead of M to avoid confusion with the downsampling factor M), and passband gain of L, we define the windowed impulse response, with respect to the high-rate time index $n'' = 0, 1, \ldots, N-1$:

$$h(n'') = w(n'')d(n'' - LK), \quad \text{where} \quad d(k'') = L\,\frac{\sin(\omega_c'' k'')}{\pi k''}$$

where $w(n'')$ is any desired length-N window. Its L polyphase subfilters, each of length $2K$, are defined for, $i = 0, 1, \ldots, L-1$:

$$h_i(n) = h(Ln + i), \qquad n = 0, 1, \ldots, 2K-1$$

Next, we discuss the time-domain operation and implementation of the converter. The input signal $x(n)$ is upsampled to the high rate ${f_s}''$. Then, the upsampled input $x_{\text{up}}(n'')$ is filtered, generating the interpolated output $y_{\text{up}}(n'')$, which is then downsampled by keeping one out of every M samples, that is, setting $n'' = Mn'$ to obtain the desired signal $y(n')$ resampled at rate ${f_s}'$. Thus, we have:

$$y_{\text{up}}(n'') = \sum_{m''=0}^{N-1} h(m'')x_{\text{up}}(n'' - m'')$$

$$y(n') = y_{\text{up}}(n'')\Big|_{n''=Mn'} = y_{\text{up}}(Mn')$$

The interpolation operation can be implemented efficiently in its polyphase realization. Setting $n'' = Ln + i$, with $i = 0, 1, \ldots, L-1$, we obtain the ith sample interpolated between the input samples $x(n)$ and $x(n+1)$, by,

$$y_i(n) = y_{\text{up}}(Ln + i) = \sum_{m=0}^{P} h_i(m)x(n-m) = \mathbf{h}_i^T \mathbf{w}(n)$$

where $P = 2K-1$ is the *order* of the polyphase subfilters (the time-advance required for causal operation is not shown here), and the low-rate tapped delay line, $\mathbf{w} = [w_0, w_1, \ldots, w_P]^T$, is used by all polyphase subfilters before it is updated.

Because the downsampler keeps only every Mth filter output, it is not necessary to compute all L interpolated outputs between input samples. Only those interpolated values that correspond to the output time grid need be computed.

Given an output sample time, $n'' = Mn'$, we can write it uniquely in the form, $Mn' = Ln+i$, where $0 \le i \le L-1$. It follows that the downsampled output will be the ith interpolated value arising from the current input $x(n)$ and computed as the output of the ith polyphase subfilter $\mathbf{h}_i$:

$$y(n') = y_{\text{up}}(Mn') = y_{\text{up}}(Ln + i) = y_i(n)$$

The pattern of polyphase indices i that correspond to successive output times n' repeats with period L, and depends only on the relative values of L and M. Therefore, for the purpose of deriving a sample processing implementation of the converter, it proves convenient to think in terms of blocks of output samples of length L. The total time duration of such an output block is $LT' = MT$,

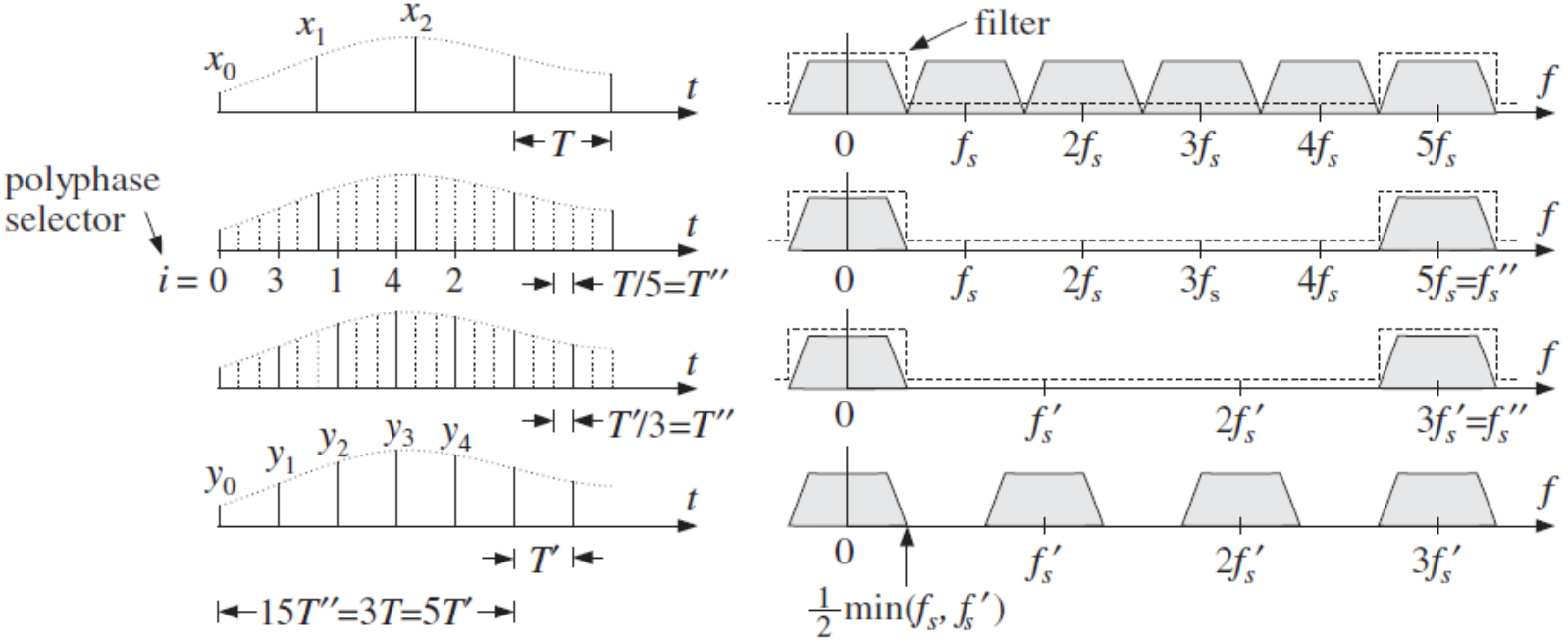
$$\boxed{T_{\text{block}} = LT' = LMT'' = MT}$$

Thus, within each output time block there are M input samples, LM high-rate interpolated samples, and L output samples. The M input samples get interpolated into the LM high-rate ones, from which the L output samples are selected.

The computational rate is M times smaller than the polyphase rate Nf_s required for full interpolation. Indeed, we have $2K$ MACs per polyphase filter output and L polyphase outputs in each period T_{block}, that is, $R = 2KL/T_{\text{block}} = N/T_{\text{block}} = N/MT = Nf_s/M$. Equivalently, we have *one* polyphase output in each output period T', $R = 2K/T' = 2Kf_s{}'$. Thus,

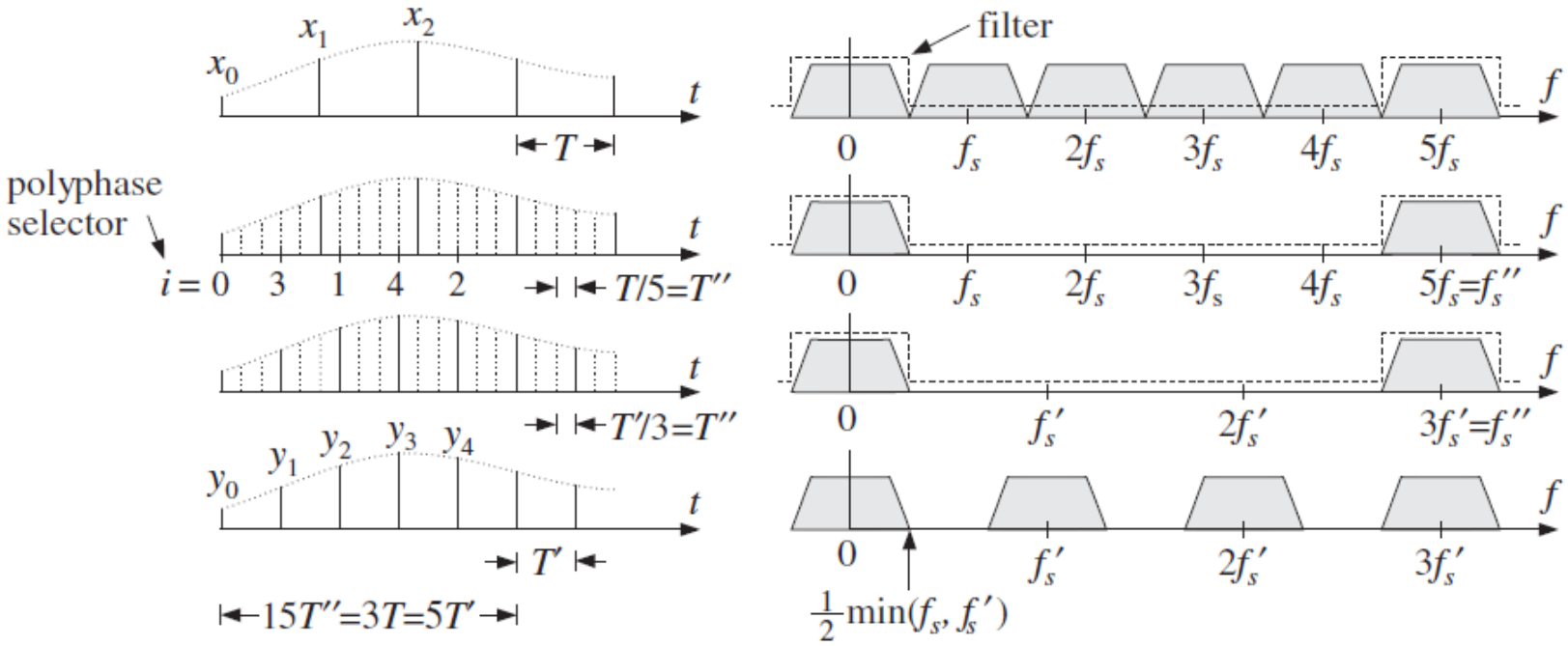
$$R = \frac{Nf_s}{M} = \frac{Nf_s{}'}{L} = 2Kf_s{}'$$

The figure below shows an example with $L = 5$ and $M = 3$, so that $f_s' = 5f_s/3$. The interpolating high rate is $f_s'' = 5f_s = 3f_s'$. The top and bottom figures show the input and output signals and their spectra. The two middle figures show the high-rate interpolated signal, viewed both with respect to the input and output time scales.



Because $f_s' > f_s$, the interpolation filter has cutoff $f_s/2$, and acts as an antialiasing prefilter removing the four input replicas up to $f_s'' = 5f_s$. The downsampling operation then downshifts the replicas at multiples of f_s'.

In the time domain, each block period $T_{\text{block}} = 15T'' = 3T = 5T'$ contains three input samples, say $\{x_0, x_1, x_2\}$, five output samples, say $\{y_0, y_1, y_2, y_3, y_4\}$, and 15 interpolated high-rate samples.



As can be seen in the figure, the first input period from x_0 to x_1 contains two outputs: y_0, y_1. We have time-aligned the samples so that $y_0 = x_0$. The output y_1 is the third ($i = 3$) interpolated value, and therefore, it is obtained as the output of the polyphase filter $\mathbf{h}_3$ with current input x_0. After this operation, the input sample x_0 is no longer needed and the delay-line $\mathbf{w}$ holding the input samples may be shifted and the next input x_1 read into it.

During the next input period from x_1 to x_2, there are two more outputs: y_2, y_3. The output y_2 is the first ($i = 1$) interpolated value, and therefore, it is the output of the filter $\mathbf{h}_1$, whereas the output y_3 is the fourth ($i = 4$) interpolated value, or the output of $\mathbf{h}_4$. After this operation, the delay-line $\mathbf{w}$ may be updated and x_2 read into it.

Finally, the third input period starting at x_2 contains only one output, namely, y_4, which is the second ($i = 2$) interpolated value, or the output of $\mathbf{h}_2$ with input x_2. After this operation, the delay-line may be shifted and the same computational cycle involving the next three inputs repeated.

The above steps may be summarized in the following sample processing algorithm:

for each input block $\{x_0, x_1, x_2\}$ do:

$w_0 = x_0$

$y_0 = \mathbf{h}_0^T \mathbf{w} = x_0$

$y_1 = \mathbf{h}_3^T \mathbf{w}$

delay$(P, \mathbf{w})$

$w_0 = x_1$

$y_2 = \mathbf{h}_1^T \mathbf{w}$

$y_3 = \mathbf{h}_4^T \mathbf{w}$

delay$(P, \mathbf{w})$

$w_0 = x_2$

$y_4 = \mathbf{h}_2^T \mathbf{w}$

delay$(P, \mathbf{w})$

end input block

The outputs $\{y_0, y_1, y_2, y_3, y_4\}$ were computed by the five polyphase filters $\{\mathbf{h}_0, \mathbf{h}_3, \mathbf{h}_1, \mathbf{h}_4, \mathbf{h}_2\}$ corresponding to the sequence of polyphase indices $i = \{0, 3, 1, 4, 2\}$. The input samples that were used in the computations were $\{x_0, x_0, x_1, x_1, x_2\}$, so that the corresponding index of x_n was $n = \{0, 0, 1, 1, 2\}$. When the index n was repeated, the delay line was not updated.

It is easily seen from the above figure that the patterns of i's and n's get repeated for every group of five outputs. These patterns can be predetermined as the solutions of the equations, $5n + i = 3m$, for $m = 0, 1, \ldots, 4$. In general, we can calculate the patterns by solving the L equations:

$$Ln_m + i_m = Mm, \qquad m = 0, 1, \ldots, L-1$$

with solution (where $\%$ denotes the modulo operation):

for, $m = 0, 1, \ldots, L-1$, compute:

$$i_m = (Mm)\%L$$

$$n_m = (Mm - i_m)/L$$

(polyphase selectors)

For example, the solutions of, $5n + i = 3m$, can be verified explicitly

$$\begin{array}{lll} m = 0, & i = 0, & n = 0 \\ m = 1, & i = 3, & n = 0 \\ m = 2, & i = 1, & n = 1 \\ m = 3, & i = 4, & n = 1 \\ m = 4, & i = 2, & n = 2 \end{array}$$

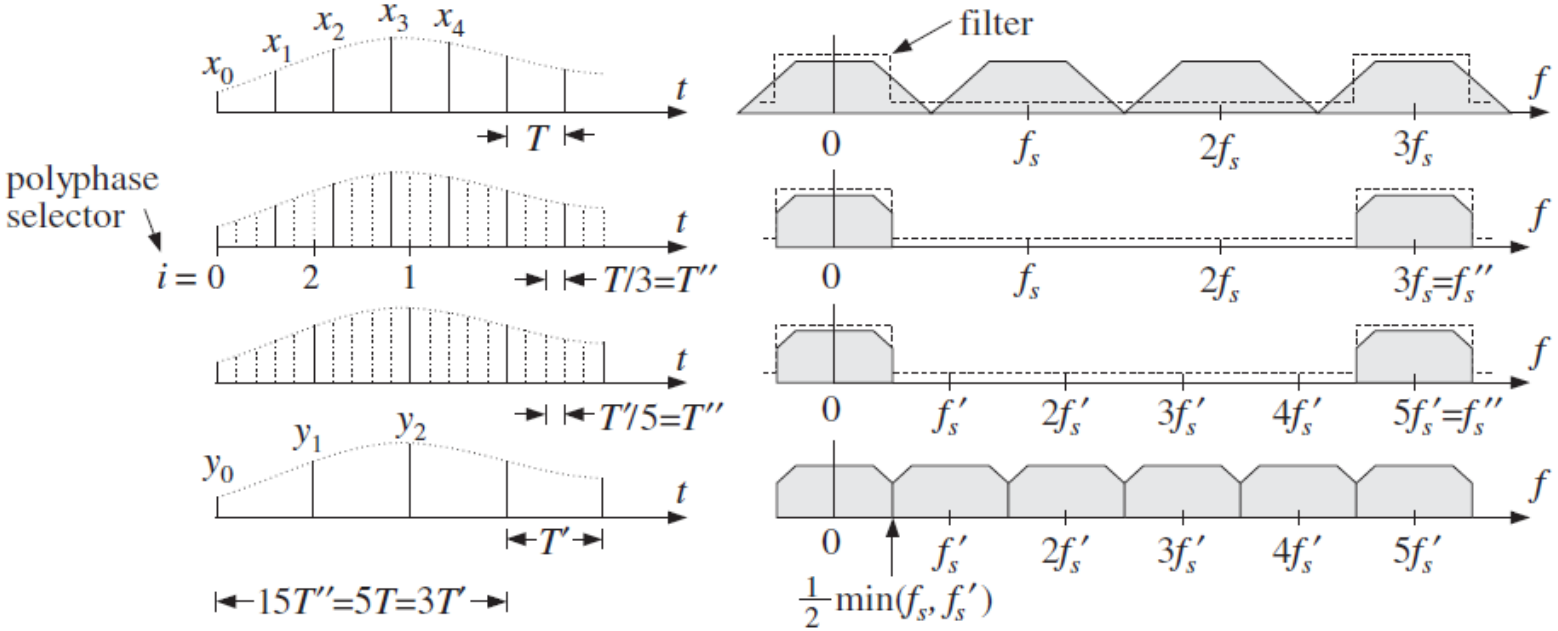
Assuming that the sequences $\{i_m, n_m\}$, $m = 0, 1, \ldots, L-1$, have been *precomputed*, the general sample rate conversion algorithm that transforms each length-M input block, $\{x_0, x_1, \ldots, x_{M-1}\}$, into a length-$L$ output block, $\{y_0, y_1, \ldots, y_{L-1}\}$, can be stated as follows:

```
for each input block, {x_0, x_1, ..., x_{M-1}}, do:
    for, n = 0, 1, ..., M - 1, do:
        w_0 = x_n
        for, Ln/M ≤ m < L(n+1)/M, do:
            y_m = h_{i_m}^T w
        delay(P, w)
    end n–loop
end block
```

The inner loop ensures that the output time index m lies between the two input times $Ln \leq Mm < L(n+1)$, with respect to the T'' time scale. Because $Mm = Ln_m + i_m$, it follows that such m's will have $n_m = n$. The index i_m serves as a polyphase filter selector.

In the special cases of interpolation ($M = 1$), or decimation ($L = 1$), the algorithm reduces to the corresponding sample processing algorithms given in plain interpolation or plain decimation.

Another example is shown below that has $L = 3$, $M = 5$ and decreases the sampling rate by a factor of $3/5$ so that $f_s{}' = 3f_s/5$. The interpolating high rate is now $f_s{}'' = 3f_s = 5f_s{}'$. Because $f_s{}' < f_s$, the filter's cutoff frequency must be $f_c = f_s{}'/2$, and therefore, the filter acts as an antialiasing filter for the downsampler. The filter necessarily chops off those high frequencies from the input that would otherwise be aliased by the downsampling operation, that is, the frequencies in the range $f_s{}'/2 \le f \le f_s/2$.



In the time domain, each block of five input samples $\{x_0, x_1, x_2, x_3, x_4\}$ generates a block of three output samples $\{y_0, y_1, y_2\}$. The solution of the polyphase selector equations, $3n + i = 5m$, are for $m = 0, 1, 2$,

$$n_m = \{0, 1, 3\}, \qquad i_m = \{0, 2, 1\}$$

Thus, only the inputs $\{x_0, x_1, x_3\}$ will generate interpolated outputs, with the polyphase subfilters $\{\mathbf{h}_0, \mathbf{h}_2, \mathbf{h}_1\}$. The inputs $\{x_2, x_4\}$ will not generate outputs, but still must be shifted into the delay-line buffer. The same conclusions can also be derived by inspecting the above figure. The corresponding sample processing algorithm is now,

for each input block, $\{x_0, x_1, x_2, x_3, x_4\}$, do:

$w_0 = x_0$

$y_0 = \mathbf{h}_0^T \mathbf{w} = x_0$

delay$(P, \mathbf{w})$

$w_0 = x_1$

$y_1 = \mathbf{h}_2^T \mathbf{w}$

delay$(P, \mathbf{w})$

$w_0 = x_2$

delay$(P, \mathbf{w})$

$w_0 = x_3$

$y_2 = \mathbf{h}_1^T \mathbf{w}$

delay$(P, \mathbf{w})$

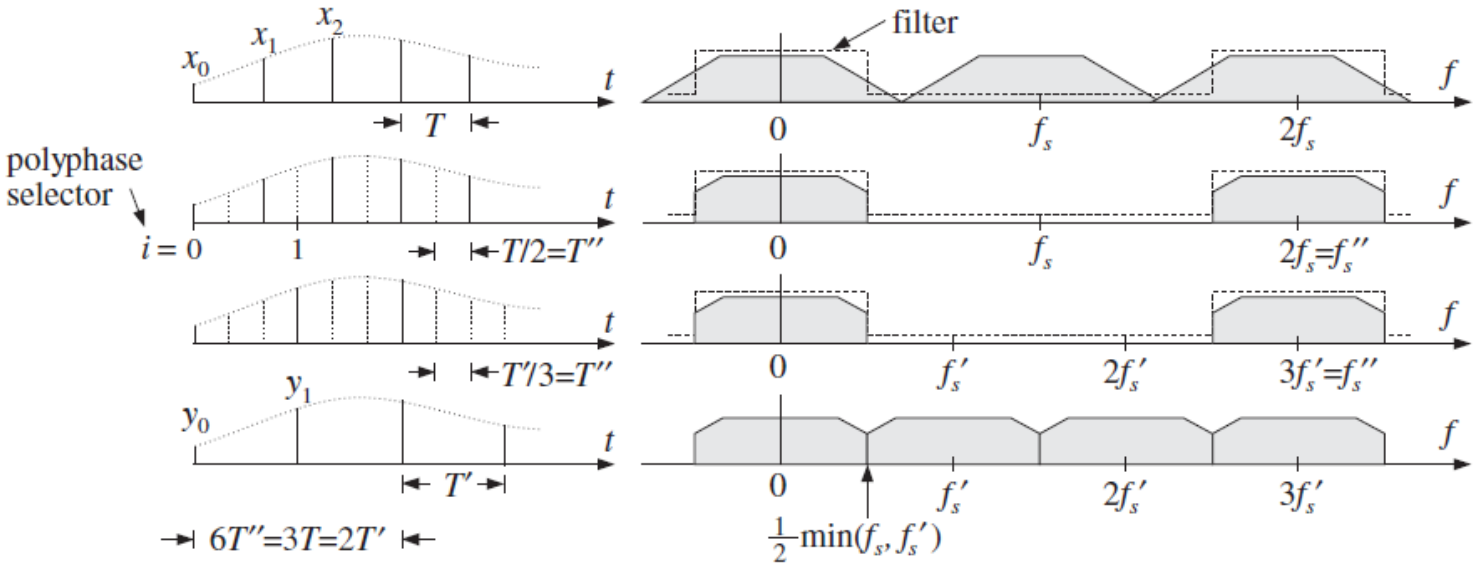
$w_0 = x_4$

delay$(P, \mathbf{w})$

As another example, consider a $2/3$ sample rate converter

$$f_s{}' = \frac{2}{3} f_s$$

Here, the filter operates at the fast rate $f_s{}'' = 3f_s{}' = 2f_s$ and acts as an antialiasing filter for the downsampler, that is, it removes the high frequencies from the input at multiples of f_s, so that the downshifted replicas at multiples of $f_s{}'$ will not overlap.



Because $T = T''/2$ and $T' = T''/3$, it follows that the basic block interval will be $T_{\text{block}} = 6T'' = 3T = 2T'$, containing 3 input-rate samples $\{x_0, x_1, x_2\}$ and 2 output-rate samples $\{y_0, y_1\}$. The interpolated output y_1 that lies halfway between x_1 and x_2 is obtained by the polyphase filter $\mathbf{h}_1$. Indeed, the polyphase selector indices can be precomputed by

$$i_m = 3m \,\%\, 2 = [0, 1]\,, \quad \text{for, } m = 0, 1$$

The sample processing algorithm of the 2/3 sample rate converter will be:

for each input block, $\{x_0, x_1, x_2\}$, do:

$w_0 = x_0$

$y_0 = \mathbf{h}_0^T \mathbf{w} = x_0$

$\text{delay}(P, \mathbf{w})$

$w_0 = x_1$

$y_1 = \mathbf{h}_1^T \mathbf{w}$

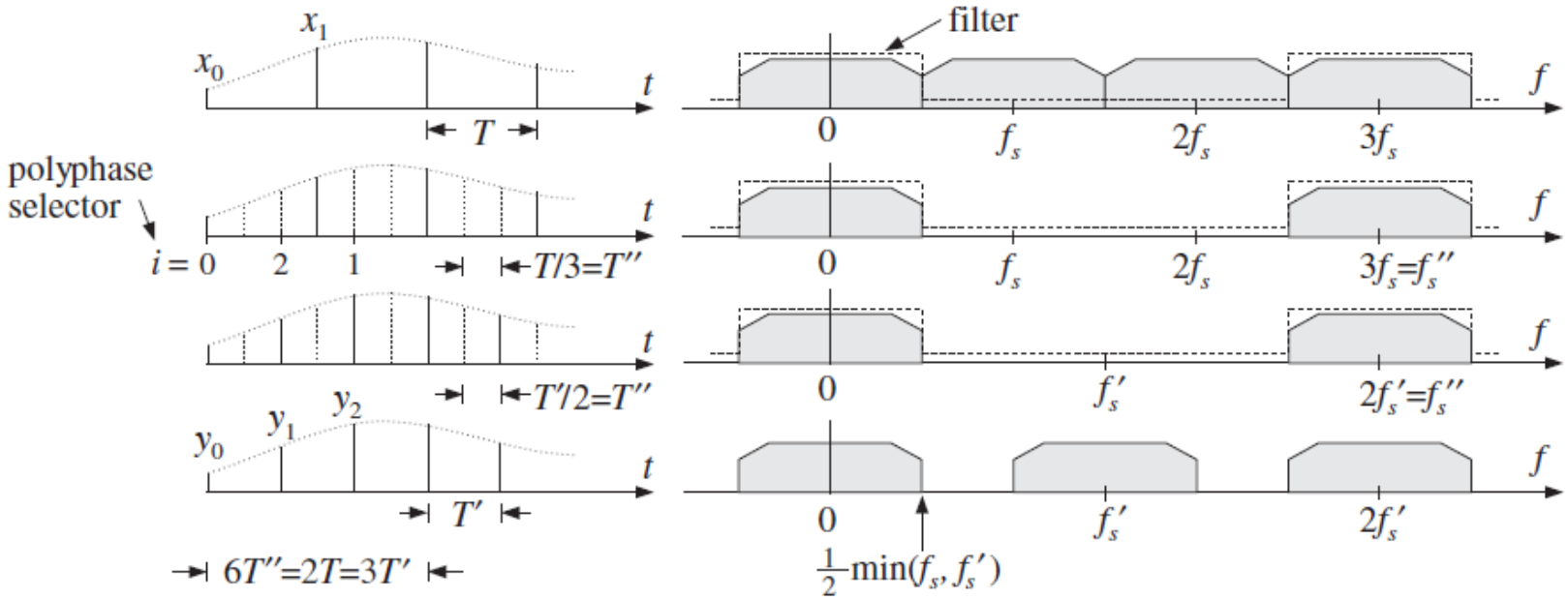
$\text{delay}(P, \mathbf{w})$

$w_0 = x_2$

$\text{delay}(P, \mathbf{w})$

where $P = 2K - 1$ is the order of the polyphase subfilters $\mathbf{h}_0$, $\mathbf{h}_1$, and $\mathbf{w}$ is the length-$(P + 1)$ input-rate delay-line buffer. Note that there is no interpolated output after x_2 (the next two outputs come from the next group of three inputs), and therefore, x_2 is simply read into the buffer $\mathbf{w}$ and the delay line is updated.

The reverse converter by a ratio $3/2$ is shown below. Here, the filter acts as an anti-image filter for the upsampler.



In this case, we have $f_s' = 3f_s/2$. The fast rate is $f_s'' = 2f_s' = 3f_s$ kHz. The input-rate and output-rate sampling intervals are $T = 3T''$ and $T' = 2T''$, and the basic time block, $T_{\text{block}} = 6T'' = 2T = 3T'$. Thus, every group of two input samples $\{x_0, x_1\}$ generates a group of three output samples $\{y_0, y_1, y_2\}$.

As can be seen in the figure, the polyphase selector sequence is

$$i_m = 2m \,\%\, 3 = [0, 2, 1], \qquad \text{for}, m = 0, 1, 2$$

Therefore, the three interpolated outputs $\{y_0, y_1, y_2\}$ will be computed by the three polyphase subfilters $\{\mathbf{h}_0, \mathbf{h}_2, \mathbf{h}_1\}$. The corresponding sample processing algorithm will be:

for each input block, $\{x_0, x_1\}$, do:

$w_0 = x_0$

$y_0 = \mathbf{h}_0^T \mathbf{w} = x_0$

$y_1 = \mathbf{h}_2^T \mathbf{w}$

delay$(P, \mathbf{w})$

$w_0 = x_1$

$y_2 = \mathbf{h}_1^T \mathbf{w}$

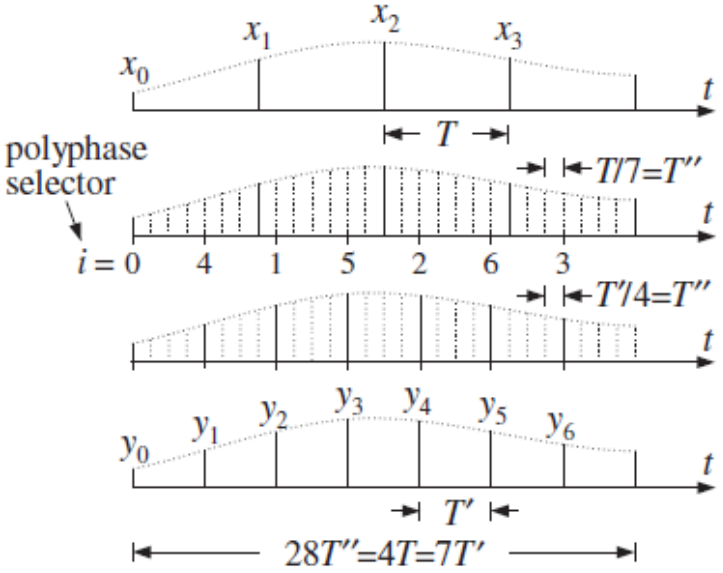
delay$(P, \mathbf{w})$

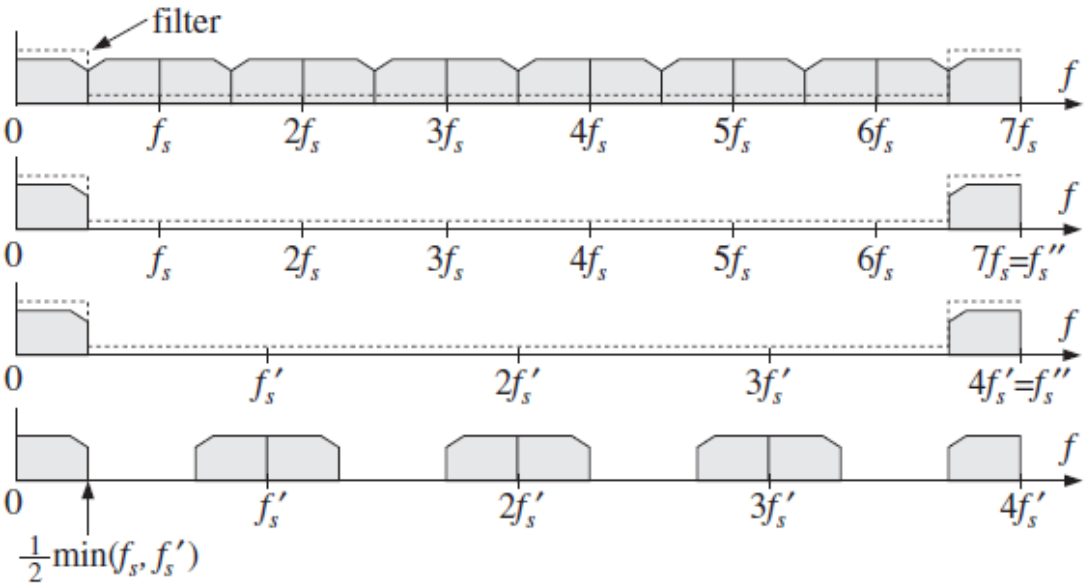
For a converter with $L/M = 7/4$, we have $f_s{}' = 7f_s/4$ and

$$f_s{}'' = 4f_s{}' = 7f_s\,, \quad T'' = \frac{T'}{4} = \frac{T}{7}$$

$$T_{\text{block}} = 28T'' = 7T' = 4T$$

The filter acts as an anti-image postfilter for the upsampler, removing the $L - 1 = 6$ replicas between multiples of the fast rate $f_s{}''$. The downshifting of the replicas caused by the downsampling operation will position replicas at the 3 multiples of the output rate $f_s{}'$, $2f_s{}'$, and $3f_s{}'$.





In the time domain, each input block of 4 input-rate samples $\{x_0, x_1, x_2, x_3\}$ generates 28 fast-rate interpolated samples, out of which 7 output-rate samples $\{y_0, y_1, y_2, y_3, y_4, y_4, y_6\}$ are selected and computed according to their polyphase indices:

$$i_m = 4m \,\%\, 7 = [0, 4, 1, 5, 2, 6, 3], \quad \text{for, } m = 0, 1, 2, 3, 4, 5, 6$$
$$n_m = (4m - i_m)/7 = [0, 0, 1, 1, 2, 2, 3]$$

whenever the index n_m is repeated, the input-rate polyphase delay line is not updated. The sample processing conversion algorithm is as follows:

for each input block, $\{x_0, x_1, x_2, x_3\}$, do:

$w_0 = x_0$

$y_0 = \mathbf{h}_0^T \mathbf{w} = x_0$

$y_1 = \mathbf{h}_4^T \mathbf{w}$

delay$(P, \mathbf{w})$

$w_0 = x_1$

$y_2 = \mathbf{h}_1^T \mathbf{w}$

$y_3 = \mathbf{h}_5^T \mathbf{w}$

delay$(P, \mathbf{w})$

$w_0 = x_2$

$y_4 = \mathbf{h}_2^T \mathbf{w}$

$y_5 = \mathbf{h}_6^T \mathbf{w}$

delay$(P, \mathbf{w})$

$w_0 = x_3$

$y_6 = \mathbf{h}_3^T \mathbf{w}$

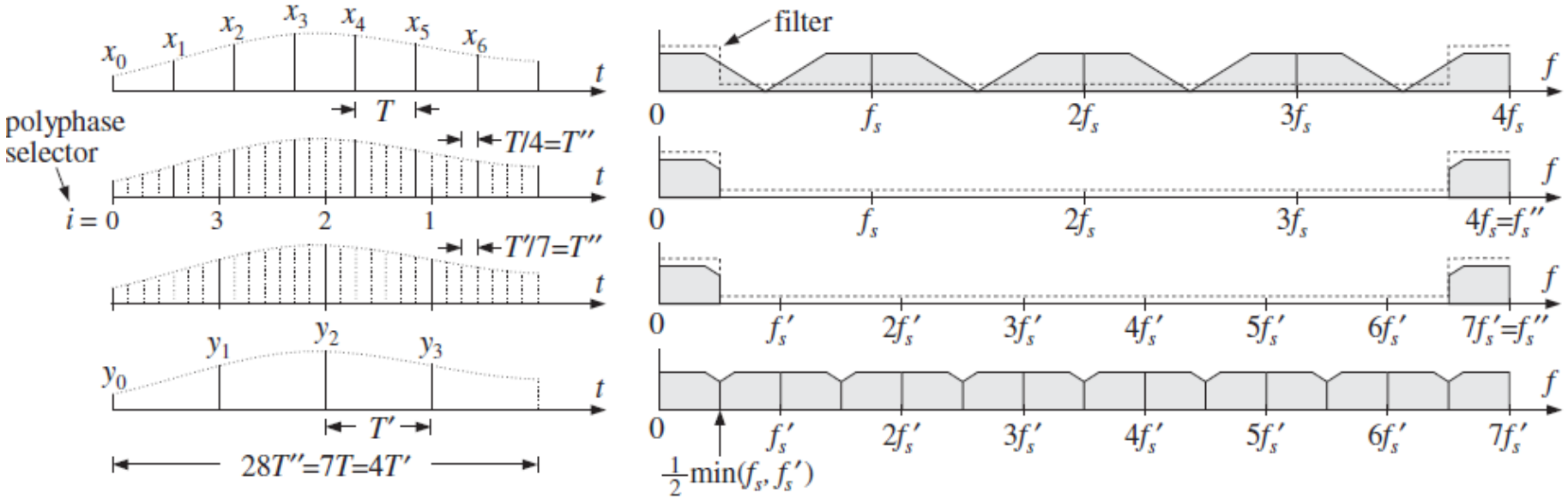
delay$(P, \mathbf{w})$

In the $L/M = 4/7$ case, we have ${f_s}' = 4f_s/7$ and

$$ {f_s}'' = 7{f_s}' = 4f_s\,, \quad T'' = \frac{T'}{7} = \frac{T}{4} $$

$$ T_{\text{block}} = 28T'' = 4T' = 7T $$

The filter acts as an antialiasing prefilter for the downsampler, removing the 3 replicas between multiples of the fast rate ${f_s}''$. The downshifting of the replicas caused by the downsampling operation will position replicas at the 6 multiples of the output rate ${f_s}', 2{f_s}', \ldots, 6{f_s}'$, without overlapping.



In the time domain, each input block of 7 input-rate samples $\{x_0, x_1, x_2, x_3, x_4, x_5, x_6\}$ generates 28 fast-rate interpolated samples, out of which 4 output-rate samples $\{y_0, y_1, y_2, y_3\}$ are selected and computed according to their polyphase indices:

$$i_m = 7m \,\%\, 4 = [0, 3, 2, 1], \quad \text{for, } m = 0, 1, 2, 3$$
$$n_m = (7m - i_m)/4 = [0, 1, 3, 5]$$

Only the input samples x_0, x_1, x_3, x_5 will produce output samples. However, for the remaining input samples the delay line must be updated. Thus, the sample processing conversion algorithm will be as follows:

for each input block, $\{x_0, x_1, x_2, x_3, x_4, x_5, x_6\}$, do:

$w_0 = x_0$

$y_0 = \mathbf{h}_0^T \mathbf{w} = x_0$

delay$(P, \mathbf{w})$

$w_0 = x_1$

$y_1 = \mathbf{h}_3^T \mathbf{w}$

delay$(P, \mathbf{w})$

$w_0 = x_2$

delay$(P, \mathbf{w})$

$w_0 = x_3$

$y_2 = \mathbf{h}_2^T \mathbf{w}$

delay$(P, \mathbf{w})$

$w_0 = x_4$

delay$(P, \mathbf{w})$

$w_0 = x_5$

$y_3 = \mathbf{h}_1^T \mathbf{w}$

delay$(P, \mathbf{w})$

$w_0 = x_6$

delay$(P, \mathbf{w})$

Noise Shaping Quantizers

The main purpose of noise shaping is to reshape the spectrum of quantization noise so that most of the noise is filtered out of the relevant frequency band, such as the audio band. Noise shaping is used in four major applications in DSP:

- Oversampled *delta-sigma* A/D converters.
- Oversampled *requantizers* for D/A conversion.
- Non-oversampled dithered noise shaping for requantization.
- Non-oversampled roundoff noise shaping in digital filters.

In the oversampled cases, the main objective is to trade off bits for samples, that is, increasing the sampling rate but reducing the number of bits per sample. The resulting increase in quantization noise is compensated by a noise shaping quantizer that pushes the added noise out of the relevant frequency band (e.g., out of the audio band) in such a way as to preserve a desired level of signal quality. The reduction in the number of bits simplifies the structure of the A/D and D/A converters.

In the non-oversampled cases, one objective is to minimize the accumulation of roundoff noise in digital filter structures. Another objective is to reduce the number of bits without reducing quality. For example, in a digital audio recording and mixing system where all the digital processing is done with 20 bits, the resulting audio signal must be rounded eventually to 16 bits in order to place it on a CD. The rounding operation can cause unwanted granulation distortions.

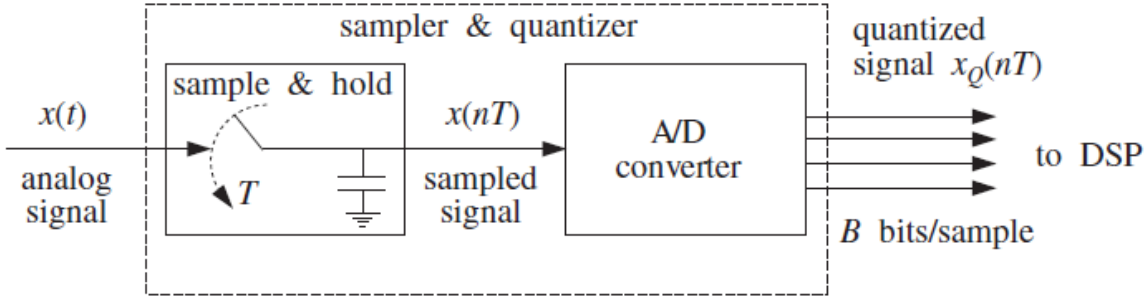
Adding a dither signal helps remove such distortions and makes the quantization noise sound like steady background white noise. However, further noise shaping can make this white noise even more inaudible by concentrating it onto spectral bands where the ear is least sensitive.

A related application in digital audio is to actually keep the bits saved from noise shaping and use them to carry extra data on a conventional CD, such as compressed images, speech, or text, and other information. This "buried" data channel is encoded to look like pseudorandom dither which is then added (subtractively) to the CD data and subjected to noise shaping.

Quantization Process

We review briefly the quantization process and quantization noise shaping, deriving the tradeoff between oversampling ratio and number of bits (see I2SP–Ch. 2). Then, we discuss the actual realization of noise-shaping quantizers.

Sampling and quantization are the necessary prerequisites for any digital signal processing operation on analog signals. A sampler and quantizer are shown below. The hold capacitor in the sampler holds each measured sample $x(nT)$ for at most T seconds during which time the A/D converter must convert it to a quantized sample, $x_{\mathrm{Q}}(nT)$, which is representable by a finite number of bits, say B bits. The B-bit word is then shipped over to the digital signal processor.



After digital processing, the resulting B-bit word is applied to a D/A converter which converts it back to analog format generating a staircase output. In practice, the sample/hold and ADC may be separate modules or may reside on board the same chip.

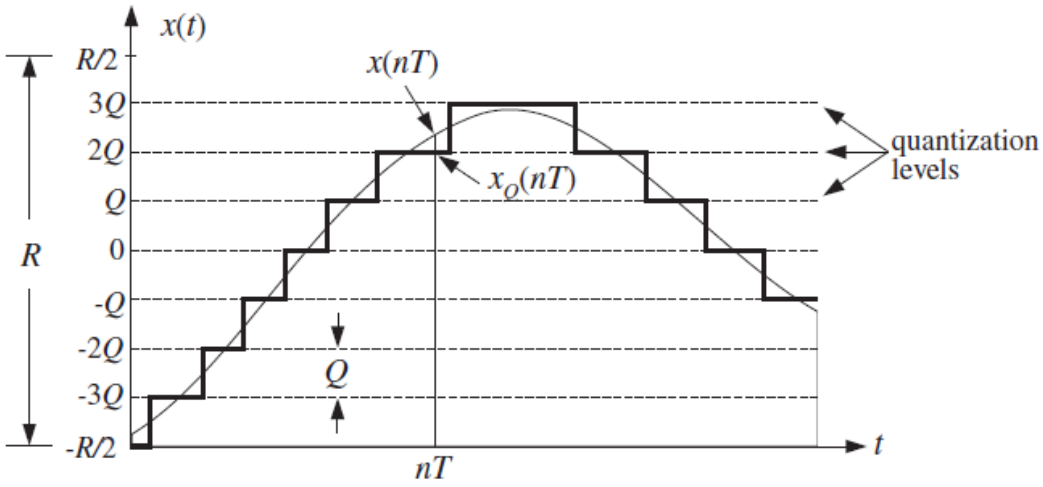
The quantized sample $x_Q(nT)$, being represented by B bits, can take only one of 2^B possible values. An A/D converter is characterized by a *full-scale range* R, which is divided equally (for a uniform quantizer) into 2^B *quantization levels*. The spacing between levels, called the *quantization width* or the quantizer resolution, is given by:

$$\boxed{Q = \frac{R}{2^B}}$$

This equation can also be written in the form:

$$\boxed{\frac{R}{Q} = 2^B}$$

which gives the number of quantization levels.



Typical values of R in practice are between 1–10 volts. The above figure shows the case of $B = 3$ or $2^B = 8$ levels, and assumes a *bipolar* ADC for which the possible quantized values lie within the symmetric range:

$$-\frac{R}{2} \leq x_{\mathrm{Q}}(nT) < \frac{R}{2}$$

The *quantization error* is the error that results from using the quantized signal $x_{\mathrm{Q}}(nT)$ instead of the true signal $x(nT)$, that is,

$$e(nT) = x_{\mathrm{Q}}(nT) - x(nT)$$

In general, the error in quantizing a number x that lies in $[-R/2, R/2)$ is:

$$e = x_{\mathrm{Q}} - x$$

where x_{Q} is the quantized value. If x lies between two levels, it will be rounded up or down depending on which is the closest level. Thus, the error e can only take the values

$$-\frac{Q}{2} \leq e \leq \frac{Q}{2}$$

Therefore, the maximum error is $e_{\max} = Q/2$ in magnitude. This is an overestimate for the typical error that occurs. To obtain a more representative value for the average error, we consider the *mean* and *mean-square* values of e defined by:

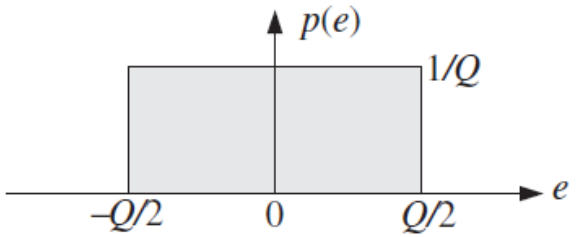
$$\overline{e} = \frac{1}{Q}\int_{-Q/2}^{Q/2} e\,de = 0, \qquad \text{and} \qquad \overline{e^2} = \frac{1}{Q}\int_{-Q/2}^{Q/2} e^2\,de = \frac{Q^2}{12}$$

The result $\overline{e} = 0$ states that on the average half of the values are rounded up and half down. Thus, $\overline{e}$ cannot be used as a representative error. A more typical value is the *root-mean-square* (rms) error defined by:

$$\boxed{e_{\text{rms}} = \sqrt{\overline{e^2}} = \frac{Q}{\sqrt{12}} = \frac{R}{\sqrt{12}}\,2^{-B}}$$

These results can be given a probabilistic interpretation by assuming that the quantization error e is a *random variable* which is distributed *uniformly* over its range, that is, having probability density:

$$p(e) = \begin{cases} \dfrac{1}{Q} & \text{if } -\dfrac{Q}{2} \le e \le \dfrac{Q}{2} \\ 0 & \text{otherwise} \end{cases}$$



The normalization $1/Q$ is needed to guarantee:

$$\int_{-Q/2}^{Q/2} p(e)\,de = 1$$

It follows that $\overline{e}$ and $\overline{e^2}$ represent the *statistical expectations*:

$$\overline{e} = E[e] = \int_{-Q/2}^{Q/2} ep(e)\,de\,, \qquad \overline{e^2} = E[e^2] = \int_{-Q/2}^{Q/2} e^2 p(e)\,de$$

Since R and Q as the ranges of the signal and quantization noise, the ratio, $R/Q = 2^B$, may be considered as a *signal-to-noise ratio* (SNR). It can be expressed in dB:

$$20\log_{10}\left(\frac{R}{Q}\right) = 20\log_{10}\left(2^B\right) = B \cdot 20\log_{10} 2 \approx 6\,B, \qquad \text{or,}$$

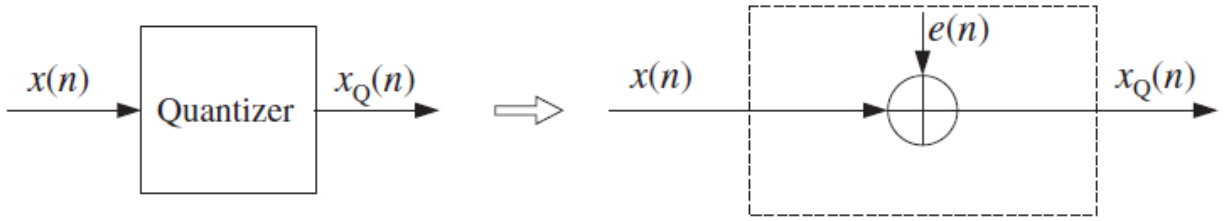
$$\boxed{\text{SNR} = 20\log_{10}\left(\frac{R}{Q}\right) = 6\,B \ \text{ dB}} \quad \text{(6-dB-per-bit-rule)}$$

which is referred to as the 6 *dB per bit* rule. This equation represents the *dynamic range* of the quantizer and can be used to determine the wordlength B if the full-scale range and desired rms error are given. For example, to match the 100-dB range of the human hearing, one needs to use at least $B = 16$ bits, since, $6\,B = 96$ dB.

The probabilistic interpretation of the quantization noise is very useful for determining the effects of quantization as they propagate through the rest of the digital processing system. The quantization noise definition, $e(n) = x(n) - x_Q(n)$, can be rewritten as,

$$\boxed{x_Q(n) = x(n) + e(n)}$$

Thus, we may think of the quantized signal $x_Q(n)$ as a noisy version of the original unquantized signal $x(n)$ to which a noise component $e(n)$ has been added. Such an additive noise model of a quantizer is shown below.

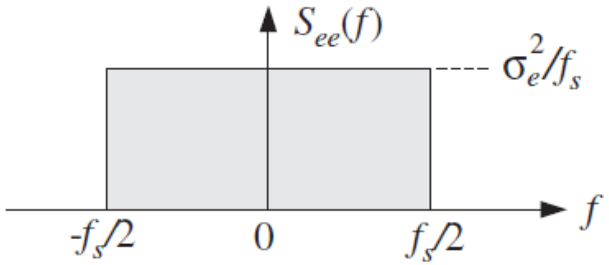


In general, the statistical properties of the noise sequence $e(n)$ are very complicated. However, for so-called *wide-amplitude wide-band* signals, that is, signals that vary through the entire full-scale range R crossing often all the quantization levels, the sequence $e(n)$ may be assumed to be a *stationary* zero-mean *white noise* sequence with *uniform* probability density over the range $[-Q/2, Q/2]$. Moreover, $e(n)$ is assumed to be *uncorrelated* with the signal $x(n)$. The average *power* or variance of $e(n)$ has already been computed above:

$$\boxed{\sigma_e^2 = E[e^2(n)] = \frac{Q^2}{12}}$$

Oversampling and Noise Shaping

In the frequency domain, the assumption that $e(n)$ is a white noise sequence means that it has a flat spectrum. More precisely, the total average power σ_e^2 of $e(n)$ is distributed *equally* over the Nyquist interval $[-f_s/2, f_s/2]$, as shown below.



Thus, the *power spectral density* of $e(n)$ will be,

$$S_{ee}(f) = \frac{\sigma_e^2}{f_s}, \qquad \text{for} \quad -\frac{f_s}{2} \le f \le \frac{f_s}{2}$$

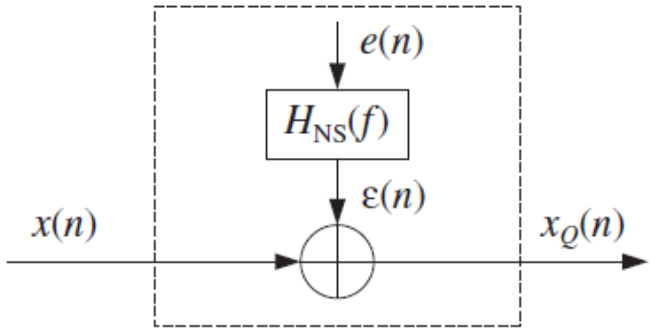
and it is periodic outside this interval with period f_s. The noise power within any Nyquist subinterval $[f_a, f_b]$ of width $\Delta f = f_b - f_a$ is given by

$$S_{ee}(f)\,\Delta f = \sigma_e^2 \frac{\Delta f}{f_s} = \sigma_e^2 \frac{f_b - f_a}{f_s}$$

As expected, the total power over the entire interval $\Delta f = f_s$ will be

$$\frac{\sigma_e^2}{f_s} f_s = \sigma_e^2$$

Noise shaping quantizers reshape the spectrum of the quantization noise into a more convenient shape. This is accomplished by filtering the white noise sequence $e(n)$ by a noise shaping filter $H_{\text{NS}}(f)$. The *equivalent noise model* for the quantization process is shown below.



$$x_{\text{Q}}(n) = x(n) + \varepsilon(n)$$

where $\varepsilon(n)$ denotes the filtered noise. The power spectral density of $\varepsilon(n)$ is no longer flat, but acquires the shape of the filter $H_{\text{NS}}(f)$:

$$S_{\varepsilon\varepsilon}(f) = |H_{\text{NS}}(f)|^2 \, S_{ee}(f) = \frac{\sigma_e^2}{f_s} \, |H_{\text{NS}}(f)|^2$$

The noise power within a given subinterval $[f_a, f_b]$ is obtained by integrating $S_{\varepsilon\varepsilon}(f)$ over that subinterval:

$$\text{Power in } [f_a, f_b] = \int_{f_a}^{f_b} S_{\varepsilon\varepsilon}(f) \, df = \frac{\sigma_e^2}{f_s} \int_{f_a}^{f_b} |H_{\text{NS}}(f)|^2 \, df$$

The concepts of sampling and quantization are independent of each other. The first corresponds to the quantization of the time axis and the second to the quantization of the amplitude axis. Nevertheless, it is possible to trade off one for the other. Oversampling can be used to alleviate the need for high quality prefilters and postfilters. It can also be used to trade off bits for samples.

In other words, if we sample at a higher rate, we may use a coarser quantizer. Each sample will be less accurate, but there will be many more of them and their effect will average out to recover the lost accuracy.

The idea is similar to performing multiple measurements of a quantity, say x. Let σ_x^2 be the mean-square error in a single measurement. If L independent measurements of x are made, it follows from the law of large numbers that the measurement error will be reduced to σ_x^2/L, improving the accuracy of measurement. Similarly, if σ_x^2 is increased, making each individual measurement worse, one can maintain the *same* level of quality as long as the number of measurements L is also increased commensurately to keep the ratio σ_x^2/L constant.

Consider two cases, one with sampling rate f_s and B bits per sample, and the other with higher sampling rate f_s' and B' bits per sample. The quantity,

$$L = \frac{f_s'}{f_s}$$

is called the *oversampling ratio* and is usually an integer. In such case, B' can chosen to be less than B and still maintain the same level of quality. Assuming the same full-scale range R for the two quantizers, we have the following quantization widths and quantization noise powers,

$$Q = R2^{-B}\,, \quad \sigma_e^2 = \frac{Q^2}{12}\,, \quad \text{and} \quad Q' = R2^{-B'}\,, \quad \sigma_e'^2 = \frac{Q'^2}{12}$$

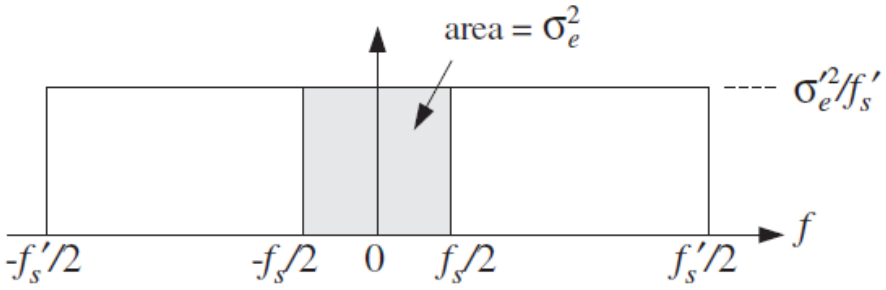
To maintain the same quality in the two cases, we require that the power spectral densities remain the same, that is,

$$\frac{\sigma_e^2}{f_s} = \frac{\sigma_e'^2}{f_s'}$$

which can be rewritten as,

$$\sigma_e^2 = f_s \frac{\sigma_e'^2}{f_s'} = \frac{\sigma_e'^2}{L}$$

Thus, the total quantization power σ_e^2 is less than $\sigma_e'^2$ by a factor of L, making B greater than B'. The meaning of this result is shown pictorially below. If sampling is done at the higher rate f_s', then the total power $\sigma_e'^2$ of the quantization noise is spread evenly over the f_s' Nyquist interval.



The shaded area gives the *proportion* of the $\sigma_e'^2$ power that lies within the smaller f_s interval. Solving for L and expressing it in terms of the difference $\Delta B = B - B'$, we find:

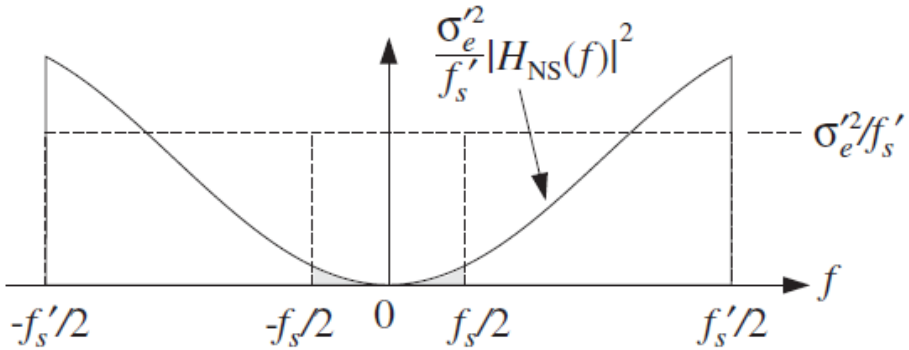
$$L = \frac{\sigma_e'^2}{\sigma_e^2} = 2^{2(B-B')} = 2^{2\Delta B}$$

or, equivalently,

$$\boxed{\Delta B = 0.5 \log_2 L}$$

that is, a saving of half a bit per doubling of L. This is too small to be useful. For example, in order to reduce a 16-bit quantizer for digital audio to a 1-bit quantizer, that is, $\Delta B = 15$, one would need the astronomically large oversampling ratio of $L = 2^{30}$.

A *noise shaping* quantizer operating at the higher rate f_s' can reshape the flat noise spectrum so that most of the power is squeezed out of the f_s Nyquist interval and moved into the outside of that interval. The figure below shows a typical power spectrum of such a quantizer.



The total quantization noise power that resides within the original f_s Nyquist interval is the little shaded area in this figure. It can be calculated by integrating the power spectral density over $[-f_s/2, f_s/2]$:

$$\sigma_e^2 = \frac{\sigma_e'^2}{f_s'} \int_{-f_s/2}^{f_s/2} |H_{NS}(f)|^2 \, df$$

We will see later that a typical pth order noise shaping filter operating at the high rate f_s' has magnitude response:

$$|H_{NS}(f)|^2 = \left| 2 \sin\left(\frac{\pi f}{f_s'}\right) \right|^{2p}, \qquad \text{for} \quad -\frac{f_s'}{2} \le f \le \frac{f_s'}{2}$$

For small f, we may use the approximation, $\sin x \simeq x$, to obtain:

$$|H_{\text{NS}}(f)|^2 = \left(\frac{2\pi f}{f_s'}\right)^{2p}, \quad \text{for} \quad |f| \ll f_s'/2$$

Assuming a large oversampling ratio L, we will have $f_s \ll f_s'$, and therefore, this approximation is justified for, $|f| \leq f_s/2$. This gives,

$$\sigma_e^2 = \frac{\sigma_e'^2}{f_s'} \int_{-f_s/2}^{f_s/2} \left(\frac{2\pi f}{f_s'}\right)^{2p} df = \sigma_e'^2 \, \frac{\pi^{2p}}{2p+1} \left(\frac{f_s}{f_s'}\right)^{2p+1}$$

$$= \sigma_e'^2 \, \frac{\pi^{2p}}{2p+1} \, \frac{1}{L^{2p+1}}$$

Using $\sigma_e^2/\sigma_e'^2 = 2^{-2(B-B')} = 2^{-2\Delta B}$, we obtain:

$$2^{-2\Delta B} = \frac{\pi^{2p}}{2p+1} \, \frac{1}{L^{2p+1}}$$

Solving for ΔB, we find the gain in bits arising from oversampling,

$$\boxed{\Delta B = (p+0.5)\log_2 L - 0.5 \log_2 \left(\frac{\pi^{2p}}{2p+1}\right)}$$

Now, the savings are $(p + 0.5)$ bits per doubling of L. Practical values for the order p are $p = 1, 2, 3, 4, 5$. The table below compares the gain in bits ΔB versus oversampling ratio L for various quantizer orders.

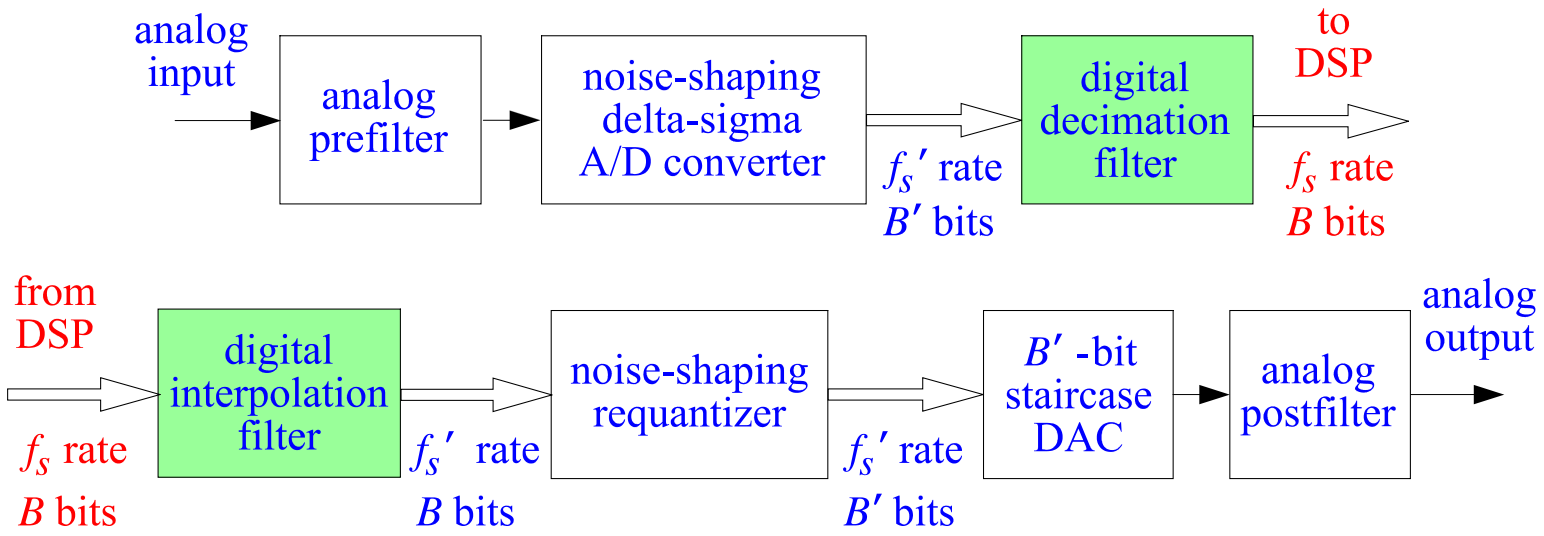
p	L	4	8	16	32	64	128
0	$\Delta B = 0.5 \log_2 L$	1.0	1.5	2.0	2.5	3.0	3.5
1	$\Delta B = 1.5 \log_2 L - 0.86$	2.1	3.6	5.1	6.6	8.1	9.6
2	$\Delta B = 2.5 \log_2 L - 2.14$	2.9	5.4	7.9	10.4	12.9	15.4
3	$\Delta B = 3.5 \log_2 L - 3.55$	3.5	7.0	10.5	14.0	17.5	21.0
4	$\Delta B = 4.5 \log_2 L - 5.02$	4.0	8.5	13.0	17.5	22.0	26.5
5	$\Delta B = 5.5 \log_2 L - 6.53$	4.5	10.0	15.5	21.0	26.5	32.0

The first CD player built by Philips used a first-order noise shaper with 4-times oversampling, that is, $p = 1$, $L = 4$, which according to the table, achieves a savings of $\Delta B = 2.1$ bits. In fact, the Philips CD player used a 14-bit, instead of a 16-bit, D/A converter at the analog reconstructing stage.

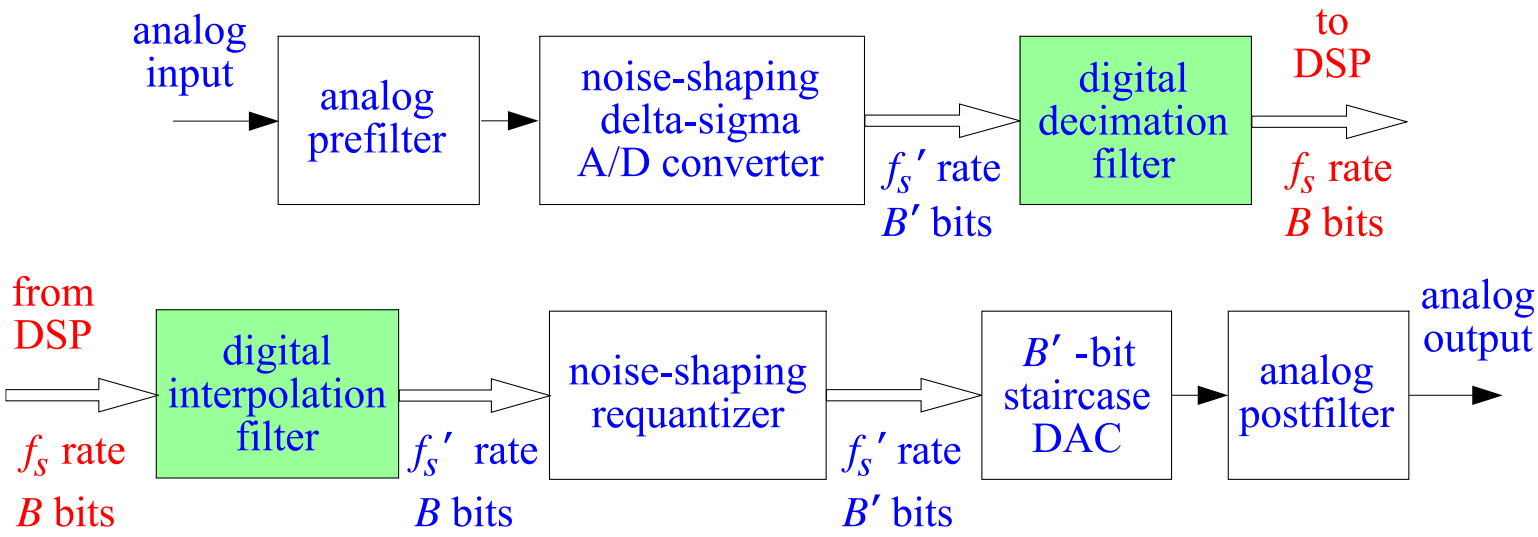
We also see from the table that to achieve 16-bit CD-quality resolution using 1-bit quantizers, that is, $\Delta B = 15$, we may use a second-order 128-times oversampling quantizer. For digital audio rates $f_s = 44.1$ kHz, this would imply oversampling at $f_s' = Lf_s = 5.6$ MHz, which is feasible with the present state of the art. Alternatively, we may use third-order noise shaping with 64-times oversampling.

An overall DSP system that uses oversampling quantizers with noise shaping is shown below. Sampling and reconstruction are done at the fast rate f_s' and at the reduced resolution of B' bits.

Intermediate processing by the DSP is done at the low rate f_s and increased resolution of B bits. The overall quality remains the same through all the processing stages. Such a system replaces a traditional DSP system.



The faster sampling rate f_s' also allows the use of a less expensive, lower quality, antialiasing prefilter. The digital decimation filter converts the fast rate f_s' back to the desired low rate f_s at the higher resolution of B bits and removes the out-of-band quantization noise that was introduced by the noise shaping quantizer into the outside of the f_s Nyquist interval.



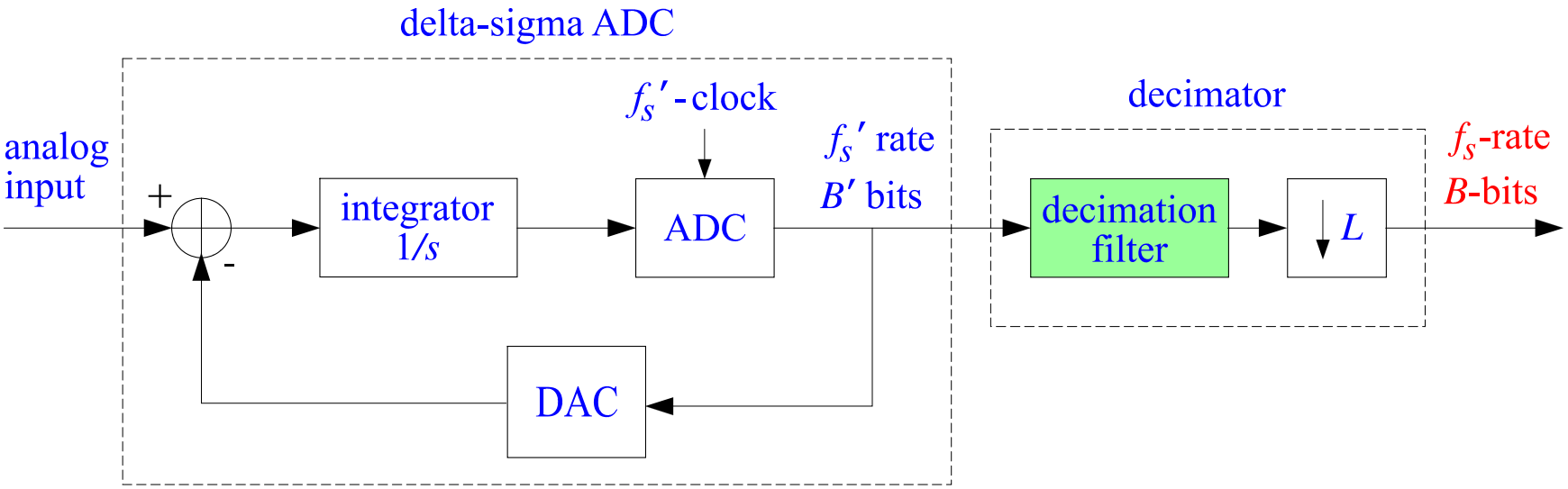
After digital processing by the DSP, the interpolation filter increases the sampling rate digitally back up to the fast rate f_s'. The noise shaping requantizer rounds the B-bit samples to B' bits, without reducing quality. Finally, an ordinary B'-bit staircase D/A converter reconstructs the samples to analog format and the postfilter smooths out the final output. Again, the fast rate f_s' allows the use of a low-quality postfilter.

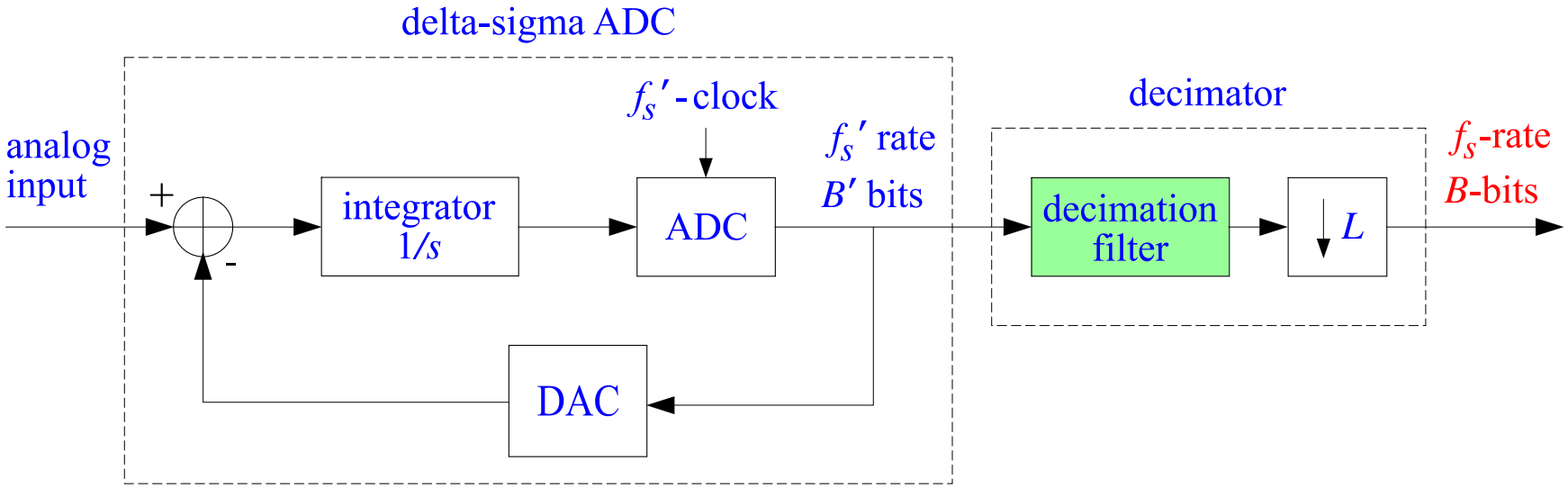
Oversampling DSP systems are used in a variety of applications, such as digital transmission and coding of speech, the sampling/playback of audio systems.

Delta-Sigma Noise-Shaping Quantizers

We discuss the structure of noise shaping quantizers and $\Delta\Sigma$ converters next, beginning with a first-order quantizer.

The figure below shows a typical oversampled first-order *delta-sigma* A/D converter system. The analog input is assumed to have been prefiltered by an antialiasing prefilter whose structure is simplified because of oversampling. The relevant frequency range of the input is the low-rate Nyquist interval $f_s/2$.





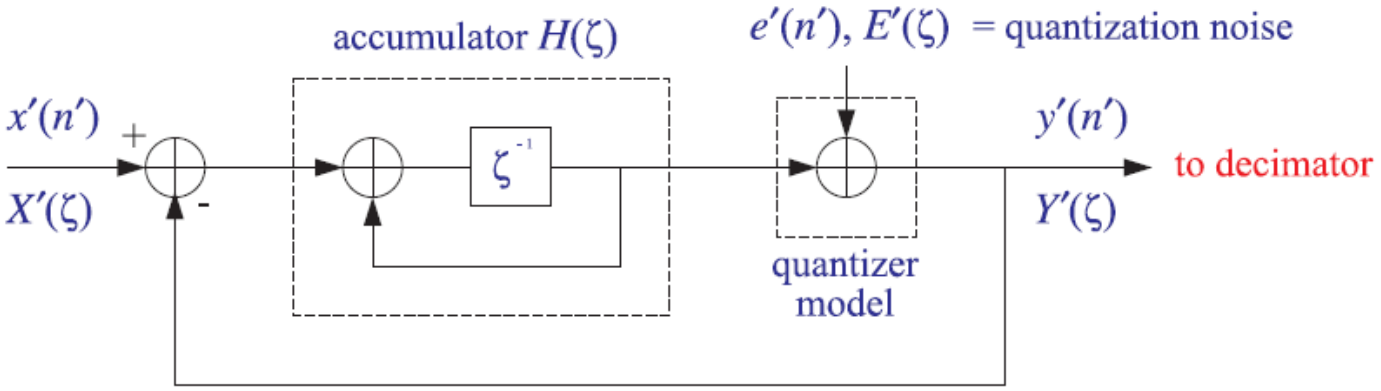
The analog part of the converter contains an ordinary A/D converter operating at the fast rate $f_s' = Lf_s$ and having a small number of bits, say B' bits. The most useful practical choice is $B' = 1$, that is, a two-level ADC. The output of the ADC is reconstructed back into analog form by the DAC (i.e., a two-level analog signal, if $B' = 1$) and subtracted from the input.

The difference signal (the "delta" part) is accumulated into the integrator (the "sigma" part) and provides a local average of the input. The feedback loop causes the quantization noise generated by the ADC to be *highpass* filtered, pushing its energy towards the higher frequencies (i.e., $f_s'/2$) and away from the signal band.

The digital part of the converter contains an L-fold decimator that reduces the sampling rate down to f_s and increases the number of bits up to a desired resolution, say B bits, where $B > B'$. In practice, the analog and digital parts reside usually on board the same chip.

The lowpass decimation filter does three jobs: (1) It removes the high-frequency quantization noise that was introduced by the feedback loop, (2) it removes any undesired frequency components beyond $f_s/2$ that were not removed by the simple analog prefilter, and (3) through its filtering operation, it increases the number of bits by linearly combining the coarsely quantized input samples with its coefficients, which are taken to have enough bits.

To see the filtering action of the feedback loop on the input and quantization noise, we consider a sampled-data equivalent model of the delta-sigma quantizer. The time samples, at rate $f_s{}'$, are denoted by $x'(n')$ in accordance with our notation in this chapter.



The ADC is replaced by its equivalent additive-noise model and the integrator by a discrete-time accumulator $H(\zeta)$ with transfer function:

$$H(\zeta) = \frac{\zeta^{-1}}{1 - \zeta^{-1}}$$

where ζ^{-1} denotes a high-rate unit delay. The numerator delay ζ^{-1} is necessary to make the feedback loop computable.

Working with ζ-transforms, we note that the input to $H(\zeta)$ is the difference signal $X'(\zeta) - Y'(\zeta)$. Its output is added to $E'(\zeta)$ to generate $Y'(\zeta)$,

$$H(\zeta)\big[X'(\zeta) - Y'(\zeta)\big] + E'(\zeta) = Y'(\zeta)$$

which may be solved for $Y'(\zeta)$ in terms of the two inputs $X'(\zeta)$ and $E'(\zeta)$:

$$Y'(\zeta) = \frac{H(\zeta)}{1 + H(\zeta)}\, X'(\zeta) + \frac{1}{1 + H(\zeta)}\, E'(\zeta)$$

It can be written in the form:

$$Y'(\zeta) = H_x(\zeta)X'(\zeta) + H_{\text{NS}}(\zeta)E'(\zeta)$$

where the noise shaping transfer function $H_{\text{NS}}(\zeta)$ and the transfer function for the input $H_x(\zeta)$ are defined as:

$$H_x(\zeta) = \frac{H(\zeta)}{1 + H(\zeta)}, \qquad H_{\text{NS}}(\zeta) = \frac{1}{1 + H(\zeta)}$$

Replacing $H(\zeta)$, we find for the first-order case:

$$\boxed{H_x(\zeta) = \zeta^{-1}, \qquad H_{\text{NS}}(\zeta) = 1 - \zeta^{-1}}$$

Thus, $H_{\text{NS}}(\zeta)$ is a simple highpass filter, and $H_x(\zeta)$ an allpass plain delay. The I/O equation becomes:

$$Y'(\zeta) = \zeta^{-1}X'(\zeta) + (1 - \zeta^{-1})E'(\zeta)$$

or, in the time domain:

$$\boxed{y'(n') = x'(n' - 1) + \varepsilon(n')}$$

where we defined the filtered quantization noise:

$$\varepsilon(n') = e'(n') - e'(n' - 1) \quad \Leftrightarrow \quad \mathcal{E}(\zeta) = (1 - \zeta^{-1})E'(\zeta)$$

Thus, the quantized output $y'(n')$ is the (delayed) input plus the filtered quantization noise. Because the noise is highpass filtered, further processing of $y'(n')$ by the lowpass decimation filter will tend to average out the noise to zero and also replace the input by its locally averaged, decimated, value. A typical example of a decimator is the hold decimator, which averages L successive high-rate samples.

By comparison, had we used a conventional B-bit ADC and sampled the input at the low rate f_s, the corresponding quantized output would be:

$$\boxed{y(n) = x(n) + e(n)}$$

where $e(n)$ is modeled as white noise over $[-f_s/2, f_s/2]$.

The "design" condition that renders the quality of the two quantizing systems equivalent and determines the tradeoff between oversampling ratio L and savings in bits, is to require that the rms quantization errors and be the *same* over the desired frequency band $[-f_s/2, f_s/2]$. As we saw earlier, the mean-square errors are obtained by integrating the power spectral densities of the noise signals over that frequency interval, yielding the condition:

$$\boxed{\sigma_e^2 = \sigma_{e'}^2 \frac{1}{f_s{}'} \int_{-f_s/2}^{f_s/2} |H_{\text{NS}}(f)|^2 \, df}$$

Setting $f_s{}' = Lf_s$ and $\sigma_e/\sigma_{e'} = 2^{-B}/2^{-B'} = 2^{-\Delta B}$, where $\Delta B = B - B'$, we obtain the desired relationship between L and ΔB.

Higher-order delta-sigma quantizers have highpass noise shaping transfer functions of the form,

$$H_{\text{NS}}(\zeta) = (1 - \zeta^{-1})^p$$

where p is the order. The frequency and magnitude responses of $H_{\text{NS}}(\zeta)$ are obtained by setting $\zeta = e^{2\pi j f / f_s{}'}$, resulting in the expressions used earlier,

$$H_{\text{NS}}(f) = \left(1 - e^{-2\pi j f / f_s{}'}\right)^p , \qquad |H_{\text{NS}}(f)|^2 = \left|2 \sin\left(\frac{\pi f}{f_s{}'}\right)\right|^{2p}$$

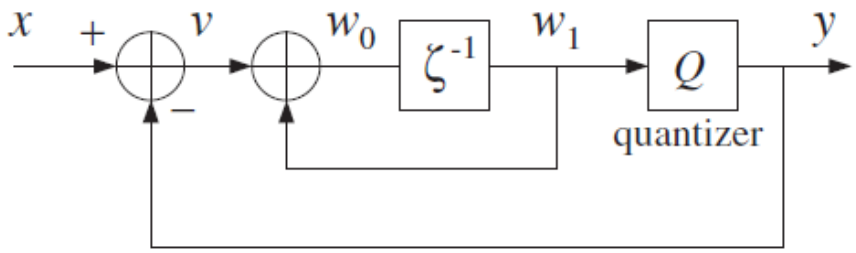
See I2SP–Ch. 12 problems for concrete realizations of second-order ($p = 2$) and third-order ($p = 3$) cases.

Example.

To illustrate the time-domain operation of a delta-sigma quantizer, consider the common 1-bit case that has a two-level ADC. Let $Q(x)$ denote the two-level quantization function defined by:

$$Q(x) = \text{sign}(x) = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases}$$

The corresponding block diagram of the quantizer is shown below, together with the computational sample processing algorithm. The quantity w_1 is the content of the accumulator's delay:



for each input x, do:

$$y = Q(w_1)$$
$$v = x - y$$
$$w_0 = w_1 + v$$
$$w_1 = w_0$$

The following table shows the computed outputs for the two constant inputs, $x = 0.4$ and $x = -0.2$, with the algorithm iterated ten times:

x	w_1	y	v	w_0
0.4	0.0	1.0	−0.6	−0.6
0.4	−0.6	−1.0	1.4	0.8
0.4	0.8	1.0	−0.6	0.2
0.4	0.2	1.0	−0.6	−0.4
0.4	−0.4	−1.0	1.4	1.0
0.4	1.0	1.0	−0.6	0.4
0.4	0.4	1.0	−0.6	−0.2
0.4	−0.2	−1.0	1.4	1.2
0.4	1.2	1.0	−0.6	0.6
0.4	0.6	1.0	−0.6	0.0

x	w_1	y	v	w_0
−0.2	0.0	1.0	−1.2	−1.2
−0.2	−1.2	−1.0	0.8	−0.4
−0.2	−0.4	−1.0	0.8	0.4
−0.2	0.4	1.0	−1.2	−0.8
−0.2	−0.8	−1.0	0.8	0.0
−0.2	0.0	1.0	−1.2	−1.2
−0.2	−1.2	−1.0	0.8	−0.4
−0.2	−0.4	−1.0	0.8	0.4
−0.2	0.4	1.0	−1.2	−0.8
−0.2	−0.8	−1.0	0.8	0.0

The average of the ten successive values of y are in the two cases, $\bar{y} = 0.4$ and $\bar{y} = -0.2$. Such averaging would take place in the decimator.

Example.

To illustrate the capability of a delta-sigma quantizer/decimator system to accurately sample an analog signal, consider the first-order quantizer of the previous example, but with a time-varying input defined with respect to the fast time scale as:

$$x'(n') = 0.5\sin(2\pi f_0 n'/f_s'), \qquad n' = 0, 1, \ldots, N_{\text{tot}} - 1$$

We choose the values $f_0 = 8.82$ kHz, $f_s = 44.1$ kHz, $L = 10$, and $N_{\text{tot}} = 200$ samples. The fast rate is $f_s' = 10 \times 44.1 = 441$ kHz, and the normalized frequency $f_0/f_s' = 0.02$.

We would like to see how the two-level quantized output $y'(n')$ of the delta-sigma quantizer is filtered by the decimation filter to effectively recover the input (and resample it at the lower rate). We compare three different decimation filters, whose frequency responses are shown below, with magnified passbands on the right.

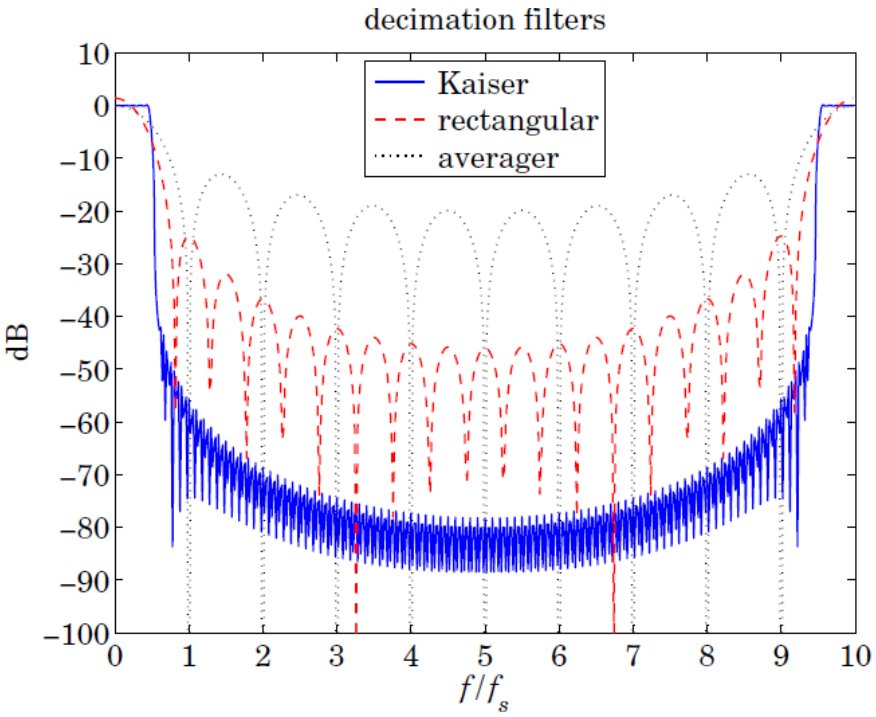
The first one is an L-fold averaging decimator with transfer function,

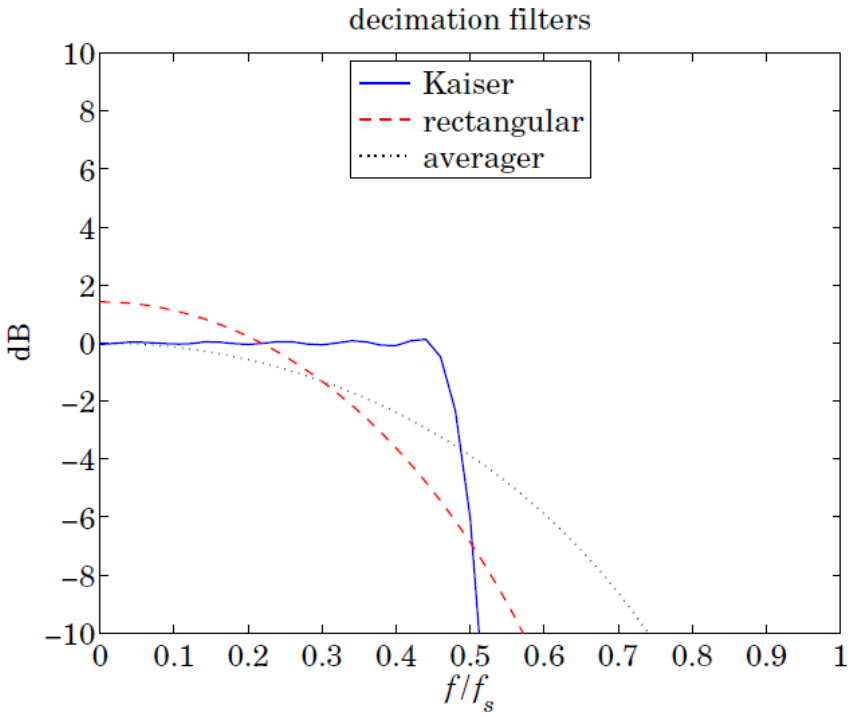
$$H(\zeta) = \frac{1}{L}\,\frac{1-\zeta^{-L}}{1-\zeta^{-1}} = \frac{1}{L}\left[1+\zeta^{-1}+\zeta^{-2}+\cdots+\zeta^{-(L-1)}\right]$$

The other two are designed by the window method, and have impulse responses:

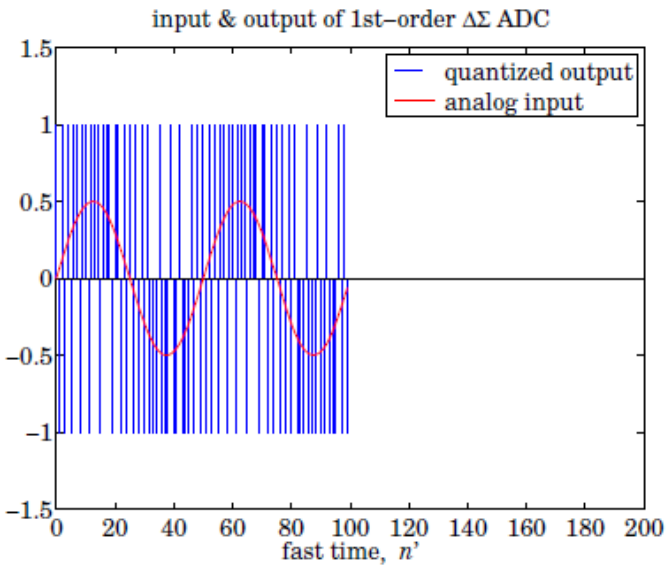
$$h(n') = w(n')\,\frac{\sin\bigl(\pi(n'-LM)/L\bigr)}{\pi(n'-LM)}\,, \qquad n' = 0, 1, \ldots, N-1$$

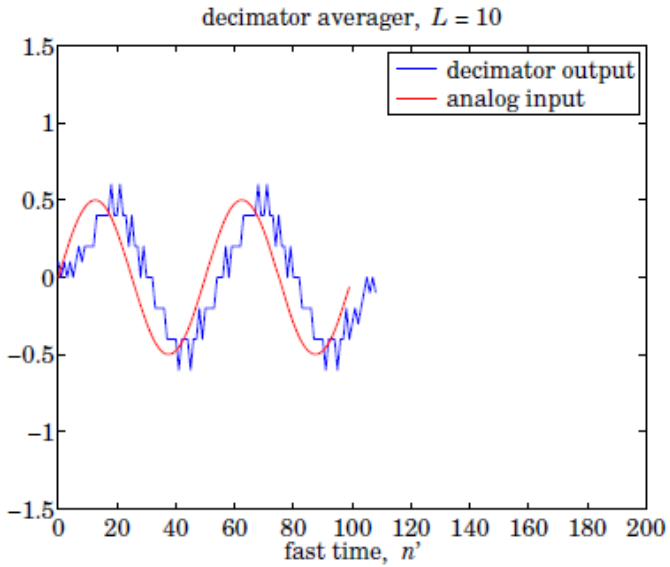
where $N = 2LM + 1$. One has the minimum possible length, that is, $N = 2LM + 1$, with $M = 1$, giving $N = 21$, and uses a rectangular window, $w(n') = 1$. The other one is designed by the Kaiser method using a stopband attenuation of $A = 35$ dB and transition width $\Delta f = 4.41$ kHz, or $\Delta f / f_s = 0.1$ (about the cutoff frequency $f_c = f_s/2 = 22.05$ kHz). It has length $N = 201$, $M = 10$, and Kaiser parameters $D = 1.88$ and $\alpha = 2.78$.


decimation filters
Kaiser
rectangular
averager
dB
f/f_s

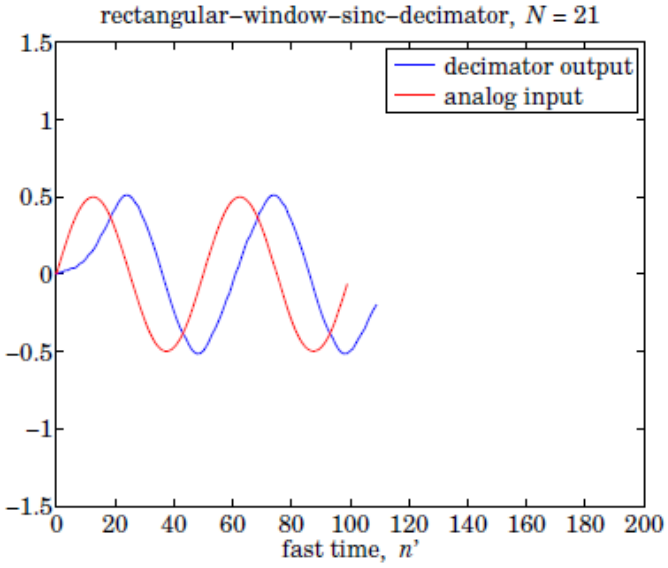

decimation filters
Kaiser
rectangular
averager
dB
f/f_s

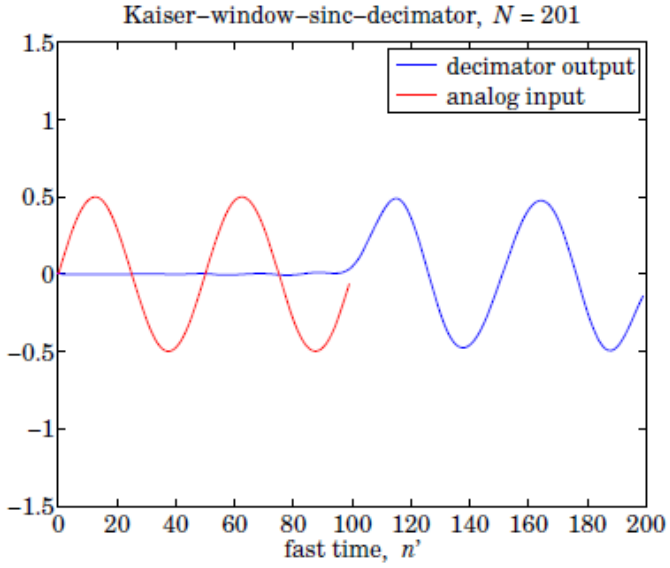
The output of the quantizer $y'(n')$, which is the input to the three decimators, is shown below on the left graph; the output of the averaging decimator is on the right.





The outputs of the rectangular and Kaiser decimators are shown below.





The averager recovers the input sinusoid only approximately and with a delay of $(L-1)/2 = 4.5$. Some of the high frequencies in $y'(n')$ get through, because they cannot be completely removed by the filter. This can be seen from the decimator's frequency response, which does not vanish everywhere between $[f_s/2, 10f_s - f_s/2]$, although it does vanish at the multiples mf_s, $m = 1, 2, \ldots, 9$,

$$|H(f)| = \left| \frac{\sin(\pi f/f_s)}{L \sin(\pi f/10f_s)} \right|$$

The outputs of the window designs are faithful representations of the input sinusoid, with a filter delay of LM samples, that is, $LM = 10$ and $LM = 100$, respectively. The Kaiser decimator gives the best output because it acts as a better lowpass filter.

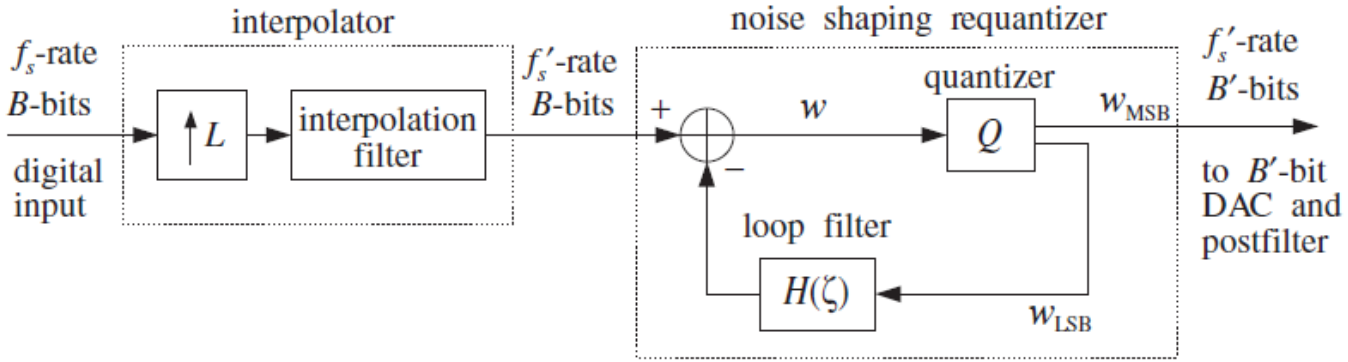
What is being plotted in these graphs is the output of the decimation filter *before* it is downsampled by a factor of $L = 10$. The downsampled signal is extracted by taking every tenth output. The nine intermediate samples which are to be discarded need not be computed. However, we did compute them here for plotting purposes.

See Project-10 for the case of a 2nd-order noise-shaping quantizer realization, with the same input and the same decimation filters plus a 2nd-order comb (whose frequency response is the square of the above averager).

Delta-Sigma DACs

Next, we discuss oversampled noise shaping requantizers for D/A conversion. A typical requantizer system is shown below. The digital input is incoming at rate f_s and B-bits per sample.

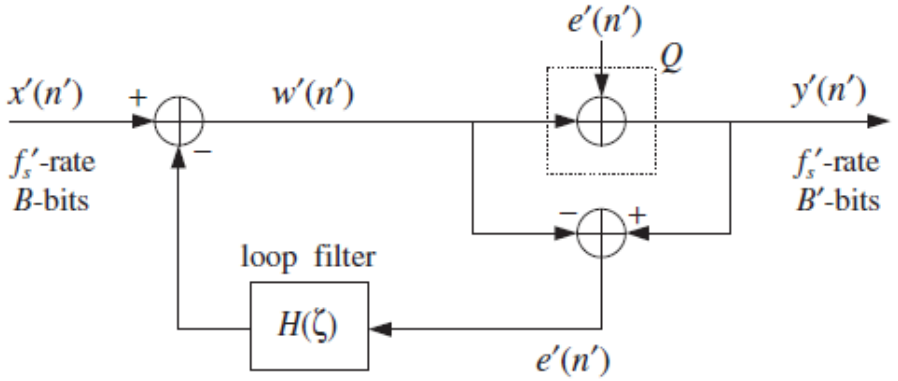
It is upsampled and interpolated by an L-fold interpolator, which increases the rate to f_s'. The noise shaping requantizer reduces the number of bits to $B' < B$. This output is, then, fed into an ordinary B'-bit DAC, followed by an anti-image postfilter (whose structure is greatly simplified because of oversampling).



The quantizer Q rounds the incoming B-bit word w by keeping the B' most significant bits, say w_{MSB}, which become the output, $y = w_{\text{MSB}}$. The requantization error is represented by the $B - B'$ least significant bits of w, that is, $w_{\text{LSB}} = w - w_{\text{MSB}}$, which is fed back through a loop filter and subtracted from the input.

The feedback loop causes the quantization noise to be highpass filtered, reducing its power within the input's baseband by just the right amount to counteract the increase in noise caused by the reduction in bits.

The figure below shows a model of the requantizer in which the quantizer Q is replaced by its equivalent noise model and the difference of the signals around the quantizer generates the LSB signal and feeds it back.



The quantized output is $y'(n') = w'(n') + e'(n')$, so that $y'(n') - w'(n') = e'(n')$. Therefore, the input to the loop filter is $e'(n')$ itself. In the ζ-domain,

$$Y'(\zeta) = W'(\zeta) + E'(\zeta) \quad \text{and} \quad W'(\zeta) = X'(\zeta) - H(\zeta)E'(\zeta)$$

which gives the I/O equation:

$$Y'(\zeta) = X'(\zeta) + \big[1 - H(\zeta)\big]E'(\zeta) = X'(\zeta) + H_{\text{NS}}(\zeta)E'(\zeta)$$

Thus, the effective noise shaping filter is

$$H_{\text{NS}}(\zeta) = 1 - H(\zeta)$$

First-, second-, or higher-order filters $H_{\text{NS}}(\zeta)$ can be constructed easily by choosing the loop filter as $H(\zeta) = 1 - H_{\text{NS}}(\zeta)$, for example:

$$\begin{aligned} H(\zeta) &= \zeta^{-1} \\ H(\zeta) &= 2\zeta^{-1} - \zeta^{-2} \end{aligned} \quad \Rightarrow \quad \begin{aligned} H_{\text{NS}}(\zeta) &= (1 - \zeta^{-1}) \\ H_{\text{NS}}(\zeta) &= (1 - \zeta^{-1})^2 \end{aligned}$$