Filters for Common Resampling Tasks

Ken Turkowski, Apple Computer

Continuous, Sampled, and Discrete Signals

Signals or functions that are *continuous* are defined at all values on an interval. When these are then *sampled*, they are defined only at a given set of points, regularly spaced or not. When the values at these sample points are then quantized to a certain number of bits, they are called *discrete*. A sampled function may or may not be discrete.

In computer graphics, we deal with all three of these representations, at least in our models of computation. A function such as sin(x) is considered to be continuous. A sequence of floating-point values may be considered to represent a sampled function, whereas a sequence of integers (especially 8-bit integers) represent a discrete function.

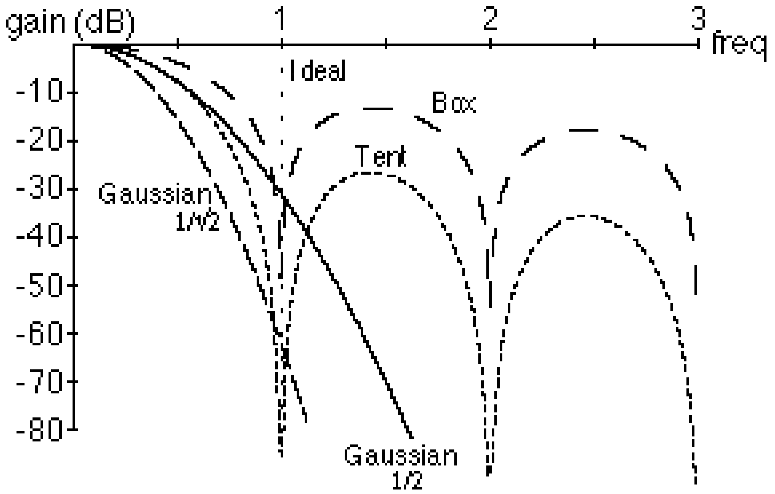
Interpolation and Decimation

Even though a signal is sampled, we may have certain rules about inferring the values between the sample points. The most common assumption made in signal processing is that the signal is *bandlimited* to an extent consistent with the sampling rate, i.e. that the values change smoothly between samples. The Sampling Theorem guarantees that a continuous signal can be reconstructed perfectly from its samples if the signal was appropriately bandlimited prior to sampling [Oppenheim 75]. Practically speaking, signals are never perfectly bandlimited, nor can we construct a perfect reconstruction filter, but we can get as close as we want in a prescribed manner.

We often want to change from one sampling rate to another. The process of representing a signal with more samples is called *interpolation*, whereas representing it with less is called *decimation*. Examples of interpolation are: zooming up on an image, correcting for non-square pixels, and converting an image from 72 dpi to 300 dpi to feed a high resolution output device. Applications of decimation are: reducing the jaggies on an supersampled image, and correcting for non-square pixels.

Choices of Filters

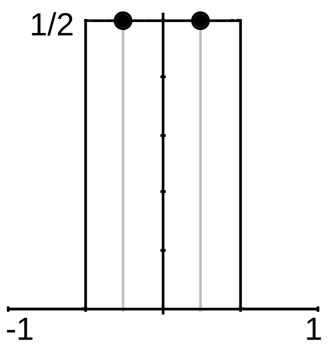
Several types of filters are more popular than others: box, tent, Gaussian, and sinc. To the right, we show the frequency response of a few of the continuous versions of these filters. The ideal filter would have a gain of 0 dB between frequencies of 0 and 1 (the passband), and -∞ beyond 1 (the stopband). The rolloff in the passband is responsible for blurriness, and the leakage in the stopband is responsible for aliasing (jaggies). One generally has to make the tradeoff between sharpness and aliasing in choosing a filter.



We will be sampling some of these filters, specifically for use in interpolation and decimation ratios of integer amounts, such as 2, 3 and 4.

Box

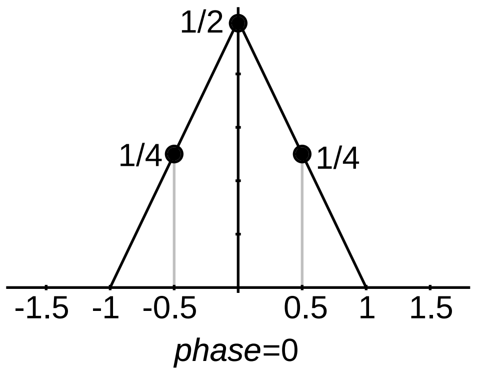
The box filter for interpolation merely picks the closest value. For decimation, it is simply an average of the input samples. With an even number of samples, the filter produces an output that is situated between two input samples (half phase), whereas with an odd number, it is situated at the same location as the middle sample (zero phase). With other filters, you can select the phase of the filter, but not so for the box filter. To the right, we show the half phase box filter for decimation by 2. Higher decimation ratio filters just have coefficients with weights that sum to one.

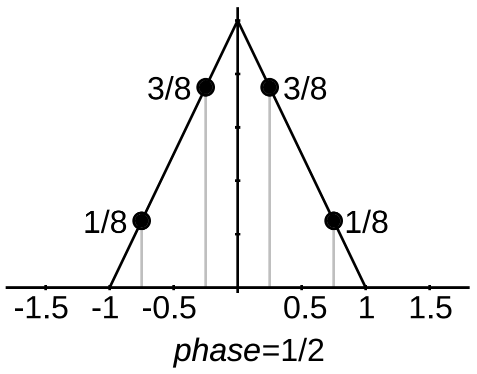


Tent

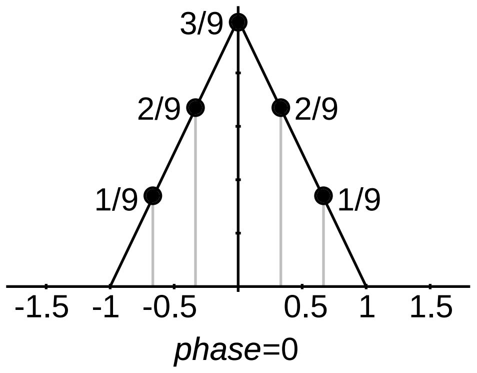
The tent filter is a generalization of linear interpolation, and *is* so when interpolating. Unlike the box filter, this can accommodate arbitrary filter phases; we show the zero phase and half phase filter for decimation by two, three and four below:

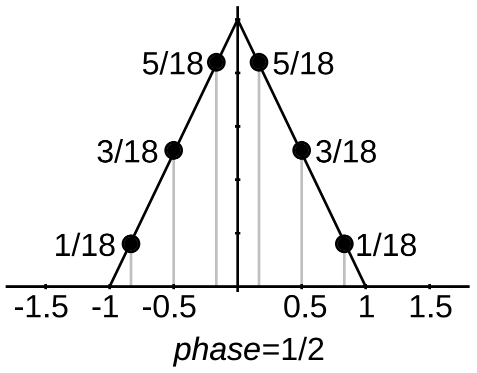
Decimation by a factor of two with the Tent function



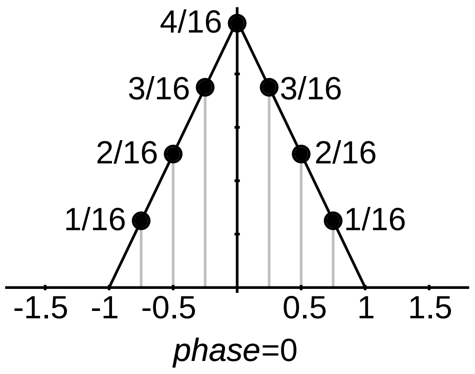


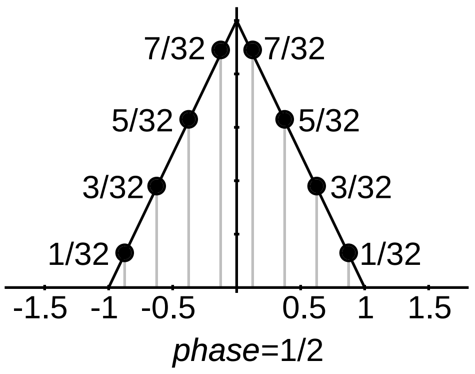
Decimation by a factor of three with the Tent function





Decimation by a factor of four with the Tent function

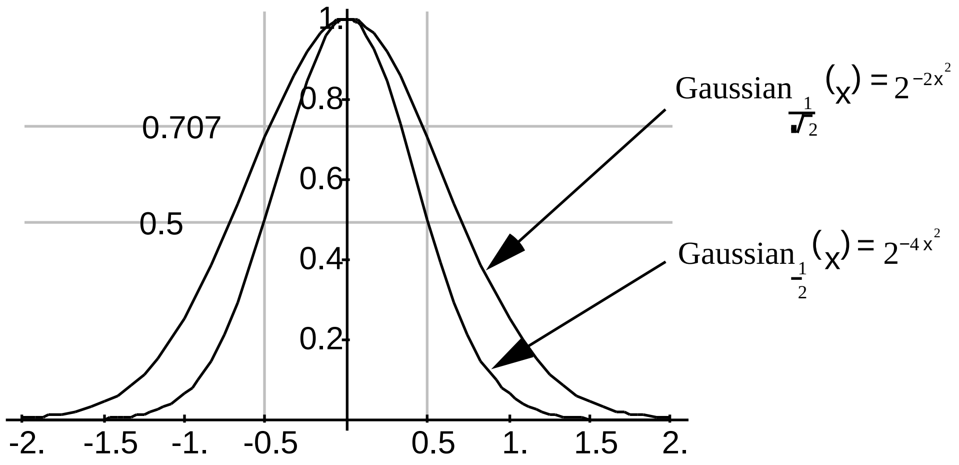




Gaussian

The Gaussian function is popular for its many elegant analytical properties. It is entirely positive, it is the limit of probability density functions, and it is its own Fourier transform. Here, we give rationale for choosing an appropriate width, or variance, for filtering in graphics.

We choose Gaussian filters here whose variances have physical and computational significance.



The first is the narrowest that we would probably ever want to use, and has a half-amplitude width of $\frac{1}{2}$, i.e. it has the value $\frac{1}{2}$ a distance $\frac{1}{2}$ from its center. Its value gets negligible $1\frac{1}{2}$ samples away from the center, so it can be considered to have a support of 3.

Energy, in general terms, is the square of the magnitude. If the eye is more linear in energy than in magnitude, then a more appropriate Gaussian might be one in which the square of the magnitude is $\frac{1}{2}$ at a distance $\frac{1}{2}$ from the center, or that the magnitude itself has a value of $\frac{1}{\sqrt{2}}$ at that point. This is a wider Gaussian than the first, and its magnitude doesn't become negligible until 2 samples from the center, so that it may be considered a filter with support 4.

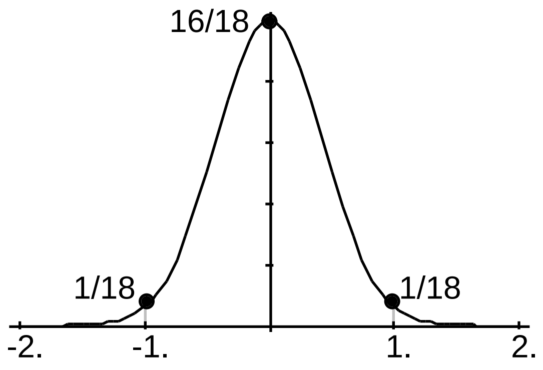
In the previous frequency plot, we compare the box, tent, and these two Gaussians. The box filter captures more of the passband (freq < 1) than the others, but it also lets through more of the stop-band (freq > 1). It is the leakage in the stopband that is responsible for aliasing artifacts, or “jag-gies”. The tent filter is 15 dB better at eliminating aliasing in the stopband, but does so at the ex-pense of losing more features in the passband. The Gaussian $\frac{1}{2}$ filter matches the tent for a good portion of the passband, but continues to attenuate the stopband. The Gaussian$\frac{1}{\sqrt{2}}$ filter does an even better job at attenuating the aliases, but does so at the expense of losing additional detail in the passband.

A comparison of the tent and the narrow Gaussian in the time (space) domain will show that they look very similar, except that the Gaussian is smooth at the peak and the base, whereas the tent has slope discontinuities there. It is these discontinuities that cause the ringing and ineffective alias suppression in the stopband.

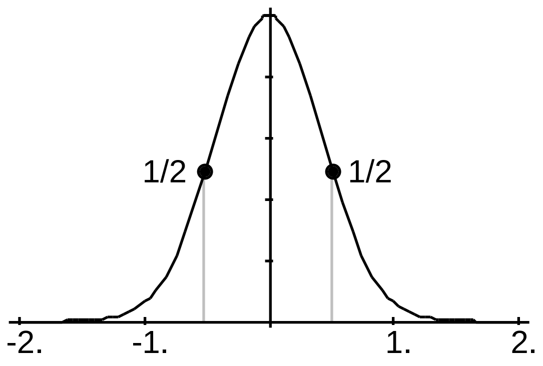
One of the side effects of our particular choices of Gaussian variance is that many of their coefficients at interesting locations are scaled powers of two, which makes way for faster computation. We will see this in the following filters, specialized for certain interpolation and dec-imation tasks.

Interpolation with the Gaussian $\frac{1}{2}$ filter

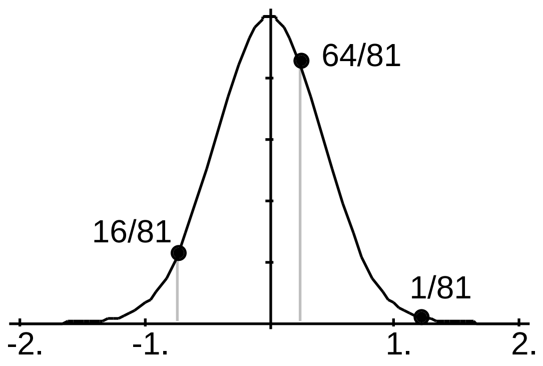
Here, we give the filter coefficients for a set of filters to interpolate between two given samples: halfway between, and a quarter of the way either side of a sample. Notice the nice rational coefficients that are scaled powers of two.



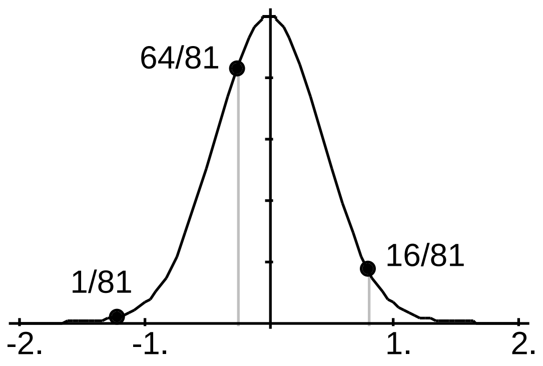
phase=0



phase=1/2



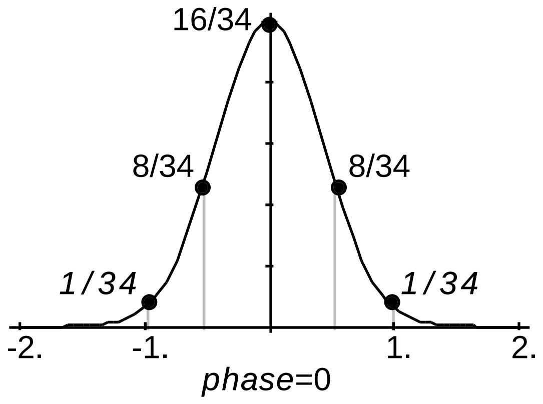
phase= -1/4

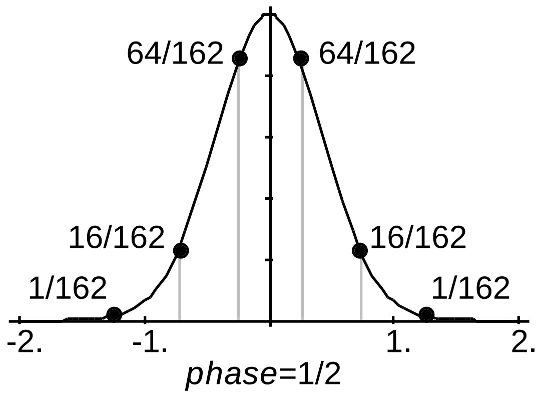


phase=1/4

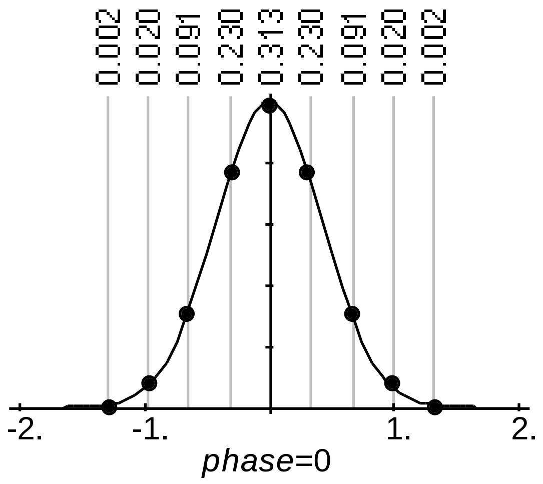
To determine the coefficients for a filter to produce the value at any other point between two samples, we merely sample the Gaussian at a series of locations one sample apart, and normalize them so that their sum equals one. Even though the Gaussian is zero nowhere, we consider this filter's value to be negligible greater than 1.5 samples away from its center.

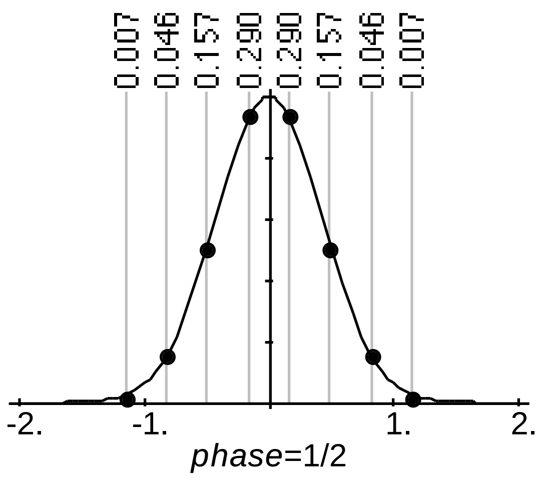
Decimation by a factor of two with the Gaussian $\frac{1}{2}$ filter



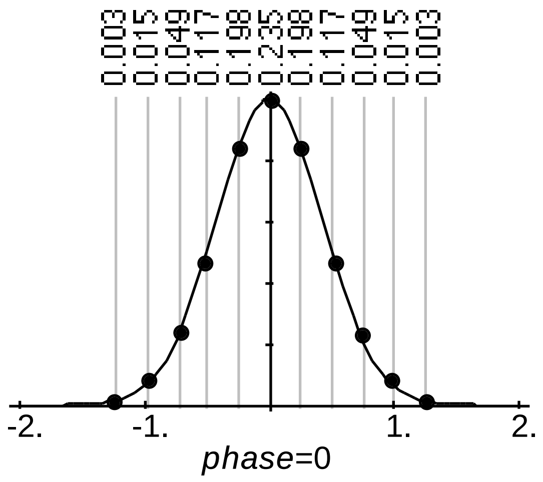


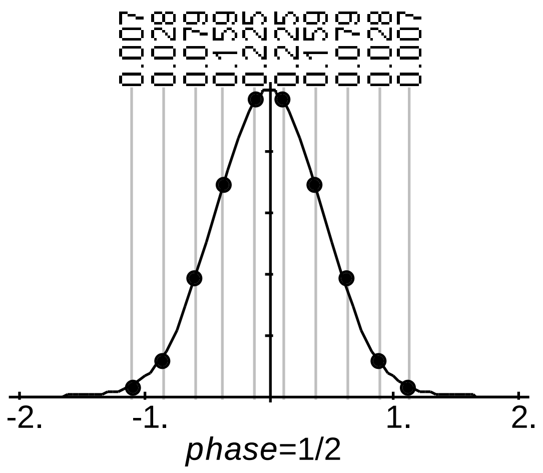
Decimation by a factor of three with the Gaussian $\frac{1}{2}$ filter





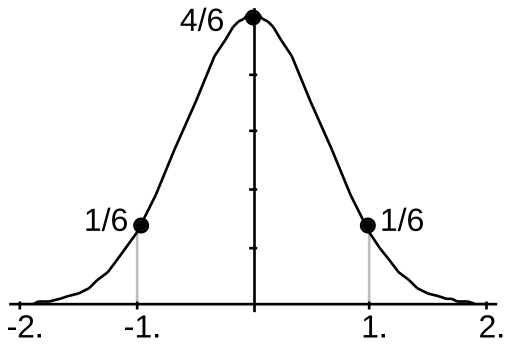
Decimation by a factor of four with the Gaussian $\frac{1}{2}$ filter



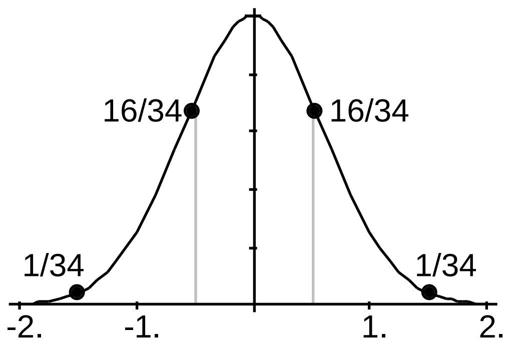


Interpolation with the Gaussian $\frac{1}{\sqrt{2}}$ filter

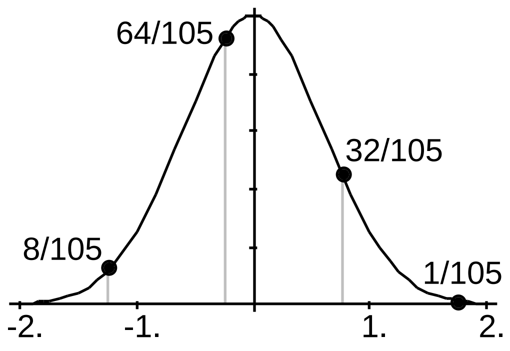
This wider Gaussian becomes negligible greater than 2 samples away from the center.



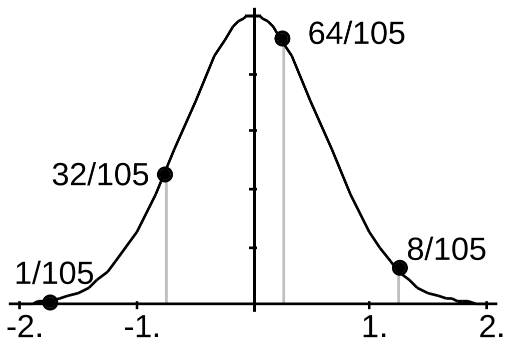
phase=0



phase=1/2

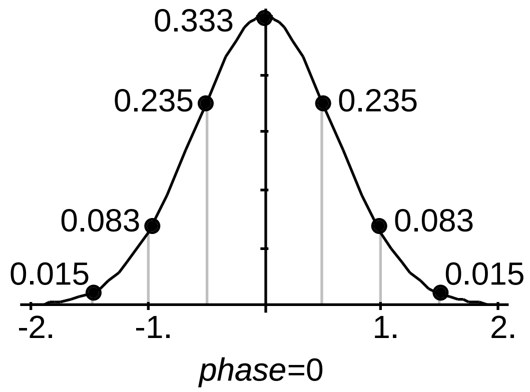


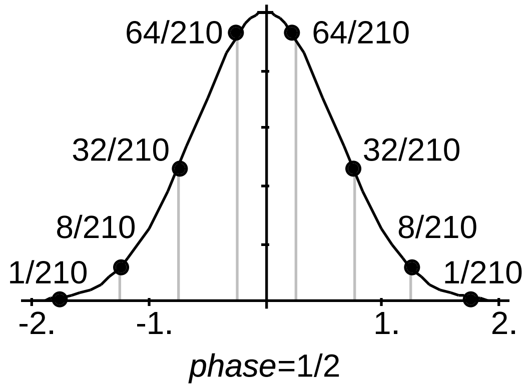
phase=1/4



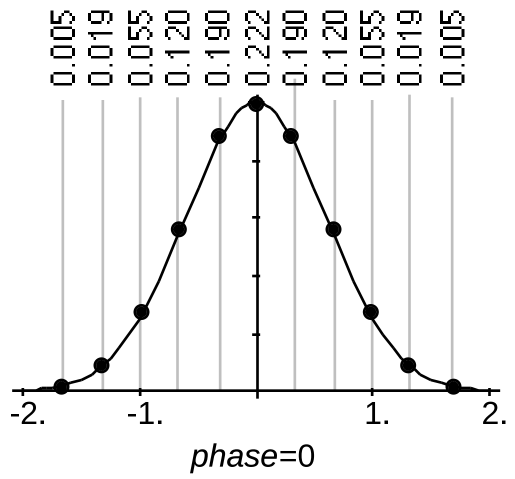
phase=3/4

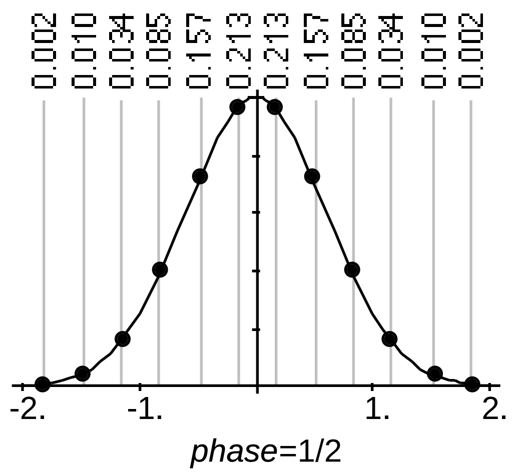
Decimation by a factor of two with the Gaussian $\frac{1}{\sqrt{2}}$ filter



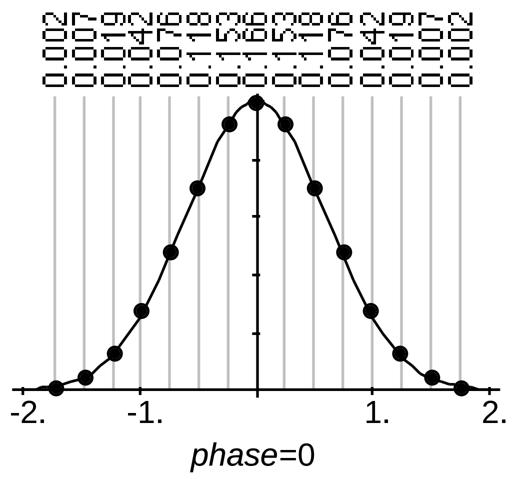


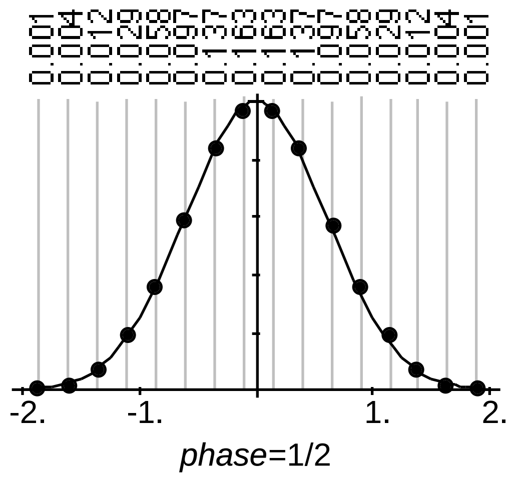
Decimation by a factor of 3 with the Gaussian $\frac{1}{\sqrt{2}}$ filter





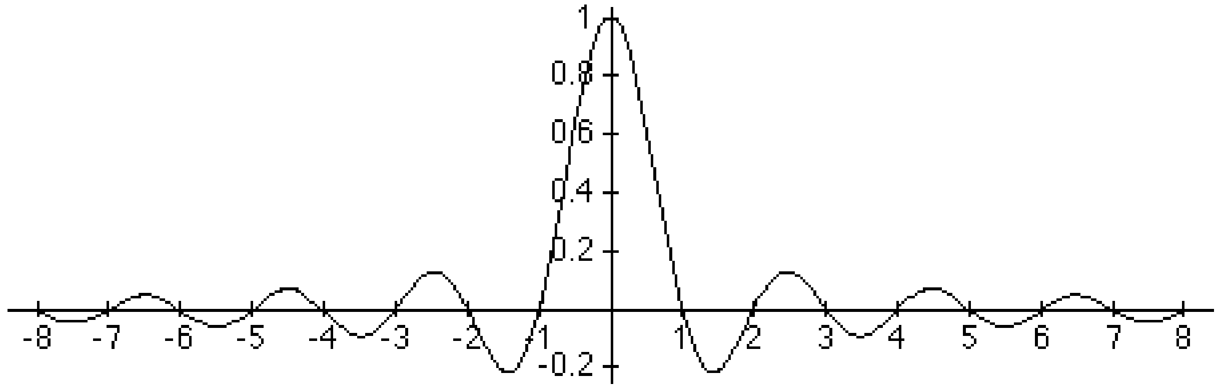
Decimation by a factor of 4 with the Gaussian $\frac{1}{\sqrt{2}}$ filter



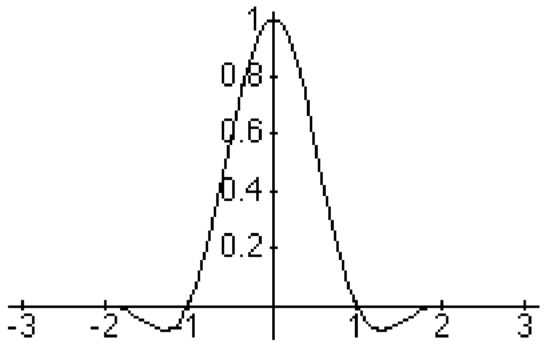


The sinc function

The sinc function is the ideal low-pass filter [Oppenheim 75]:

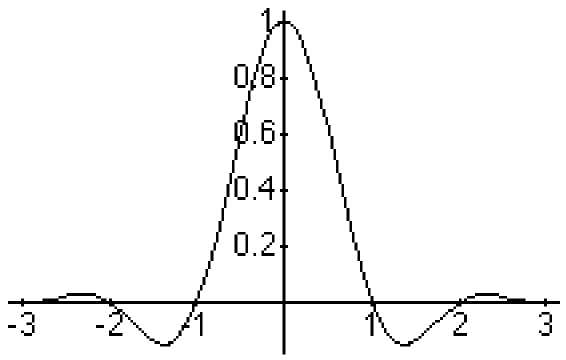


The Lanczos-windowed sinc functions



Since the sinc function never goes to zero, but approaches it slowly, we multiply it by an appropriate windowing function. The two-lobed Lanczos-windowed sinc function is one such windowed sinc function, and is defined as follows:

$$\mathrm{Lanczos2}(x) = \begin{cases} \dfrac{\sin(\pi x)}{\pi x}\dfrac{\sin(\pi \frac{x}{2})}{\pi \frac{x}{2}}, & |x| < 2 \\ 0, & |x| \geq 2 \end{cases}$$



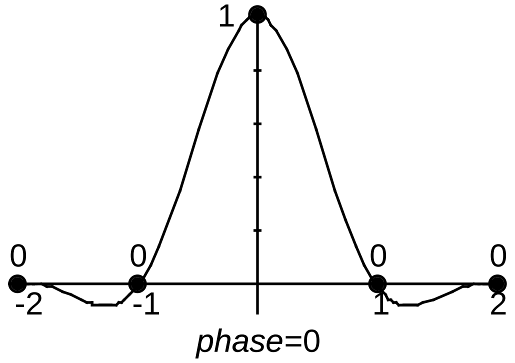
The 3-lobed Lanczos-windowed sinc function is defined similarly:

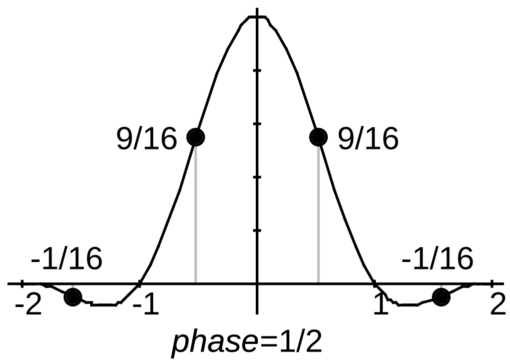
$$\mathrm{Lanczos3}(x) = \begin{cases} \dfrac{\sin(\pi x)}{\pi x}\dfrac{\sin(\pi \frac{x}{3})}{\pi \frac{x}{3}}, & |x| < 3 \\ 0, & |x| \geq 3 \end{cases}$$

The Lanczos-windowed sinc function filters have been shown to be particularly useful for graphics applications[1]. We will concern ourselves here mainly with the two-lobed version, because of its smaller kernel.

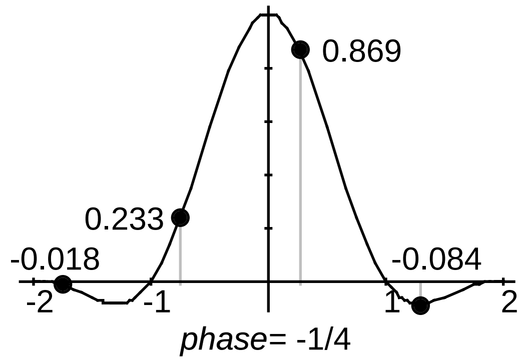
1. Turkowski, Ken and Gabriel, Steve, 1979. Conclusions of experiments done at Ampex, comparing box, Gaussian, truncated sinc, and several windowed sinc filters (Bartlett, cosine, Hanning, Lanczos) for decimation and interpolation of 2-dimensional image data. The Lanczos-windowed sinc functions offered the best compromise in terms of reduction of aliasing, sharpness, and minimal ringing.

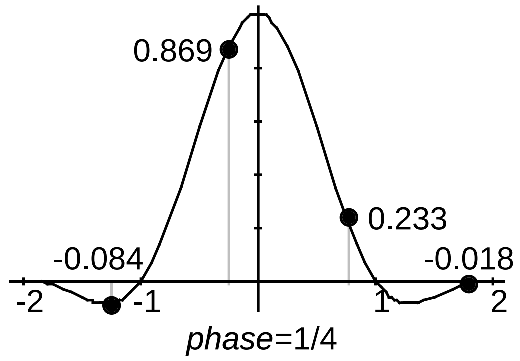
Interpolation by factor of 2 with the Lanczos2 sinc function



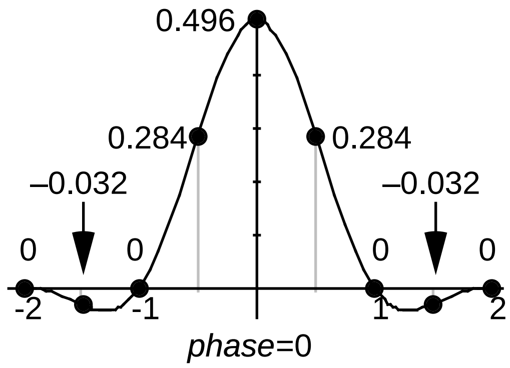


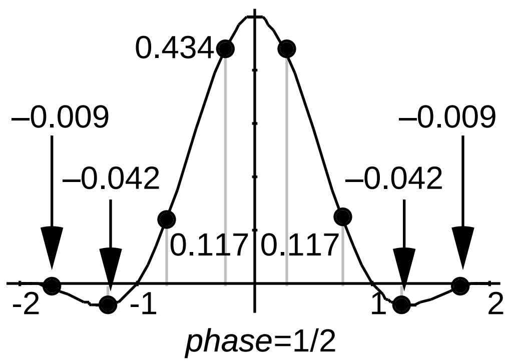
Note that with a zero phase filter, the contributions from other than the central pixel are zero, so that only the central pixel is used.



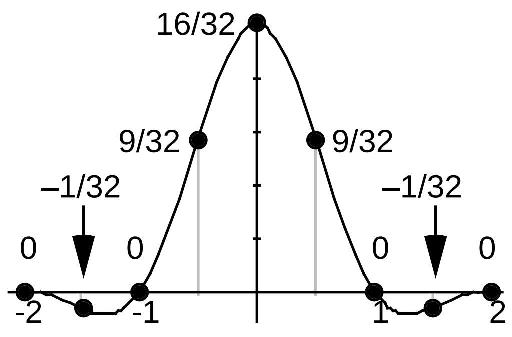


Decimation by factor of 2 with the Lanczos2 sinc function

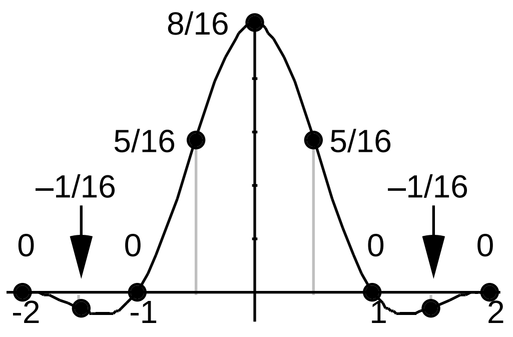




The zero phase filter has coefficients that are nearly rational. If the negative coefficients are scaled so that they are equal to -1, then the remaining coefficients are 9 and 15.7024. This inspired a search for such filters with rational coefficients. This yielded the following two zero phase filters:

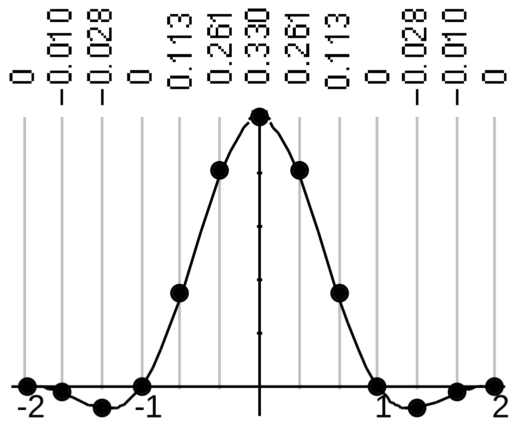


Turkowski decimator

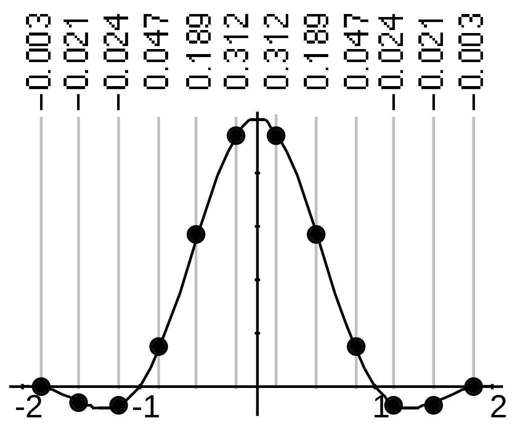


Gabriel decimator

Decimation by a factor of 3 with the Lanczos2 sinc function

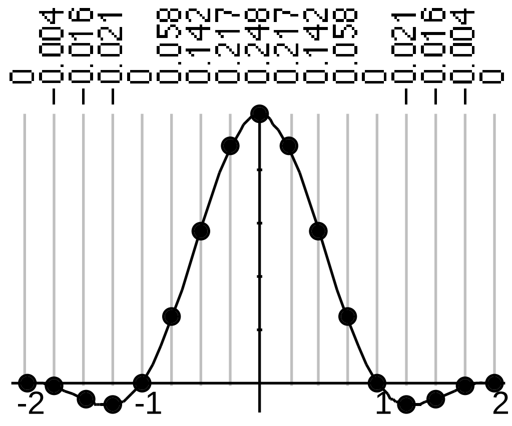


phase=0

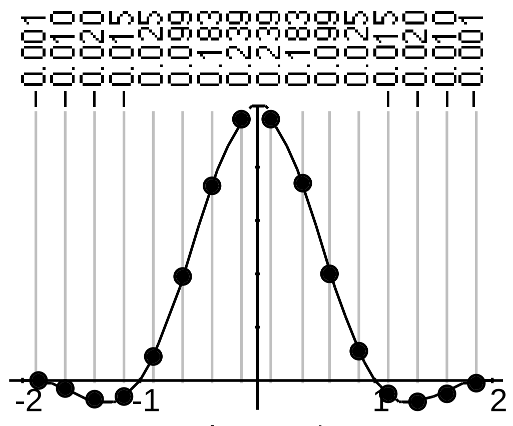


phase=1/2

Decimation by a factor of 4 with the Lanczos2 sinc function



phase=0



phase=1/2

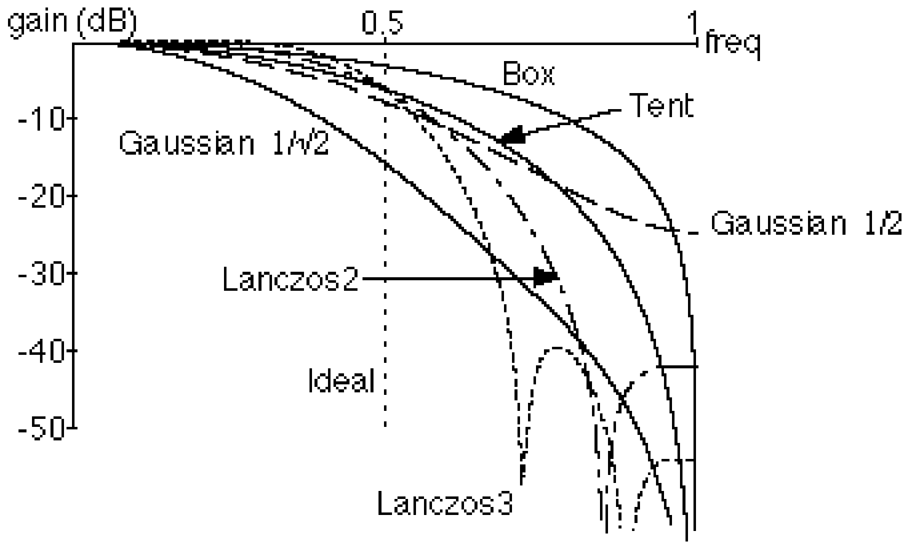
Comparative Frequency Responses

Filters are evaluated on their ability to retain detail in the passband (sharpness is valued more than blurriness) and to eliminate aliasing in the stopband (smoothness is valued more than jagginess).

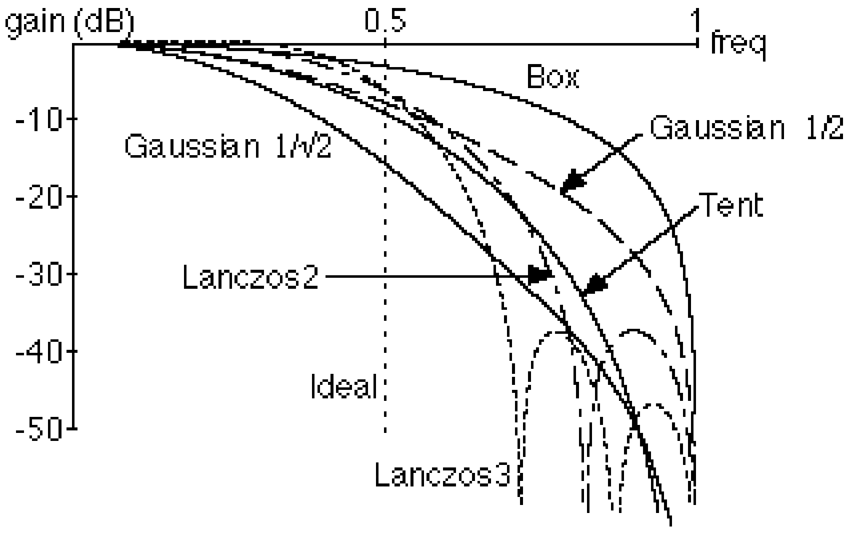
The frequency response of a sampled filter is quite different than the continuous one it was derived from. Instead of taking the Fourier transform as with continuous filters, we take the z-transform and sample on the unit circle.

By the way, one bit corresponds to about 6 dB, so that attenuation beyond 48 dB is irrelevant when working with 8-bit pixels.

Frequency response of the zero phase filters for decimation by 2



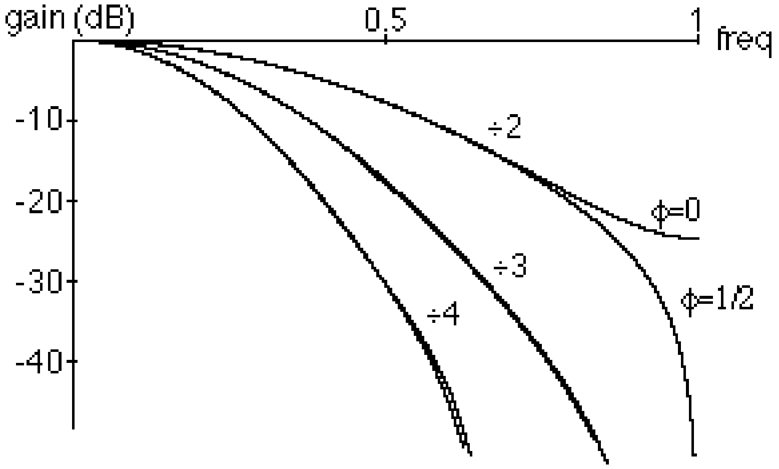
Frequency response of the half phase filters for decimation by 2



In the above diagrams, we see that the filter derived from the Gaussian 1/2 filter doesn't perform as well as the one derived from the tent, although we know that in the continuous case, the Gaussian is much better. What happened? We sampled the filter functions, that's what happened. In the process, we changed the characteristics of the filter. In fact, there are several continuous filters that give rise to the same sampled filters. The labels on each of the filters are actually misnomers, since the sampled filters are not the same as the continuous ones.

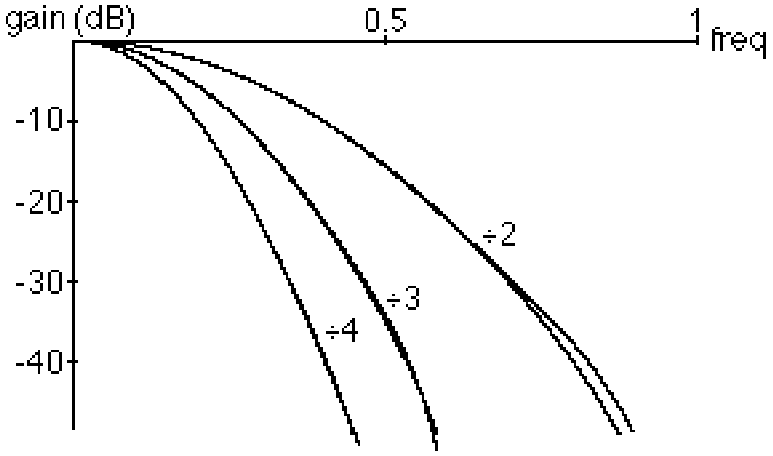
The box filter seems to retain a large portion of the passband, but lets through a tremendous amount of energy in the stopband, resulting in noticeable aliasing. The Lanczos filters keeps more of the passband than the others (except for maybe the box), and they cut off more of the stopband (except for maybe the Gaussian 1/√2), with the Lanczos3 filter coming closest to the ideal filter shape of all the filters evaluated. The Gaussian 1/√2 filter is competitive with the Lanczos3 for stopband response, but does so at the expense of excessive attenuation on the passband.

Frequency response of the Gaussian $\frac{1}{2}$ filter for several decimation ratios



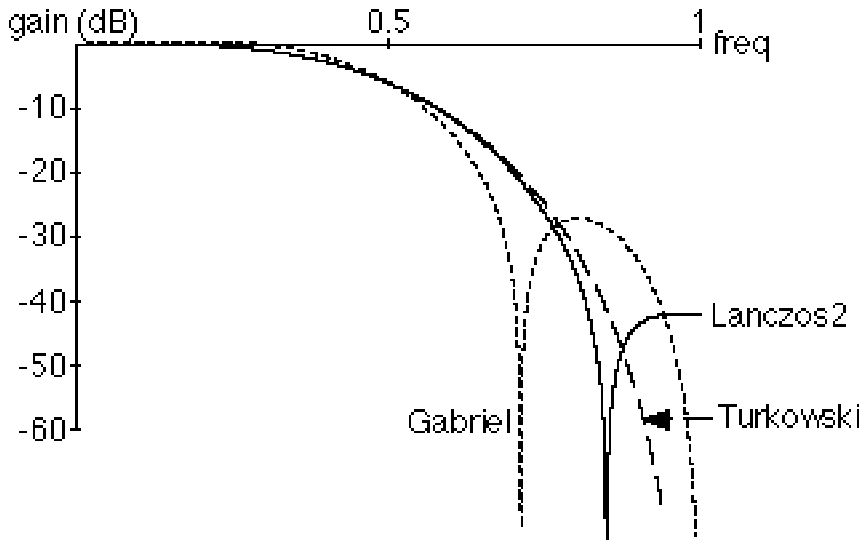
The cutoff frequencies are 0.5 for the ÷2 filter, 0.333 for the ÷3, 0.25 for the ÷4. Note that the zero phase and the half phase filters for decimation by 2 diverge, whereas the higher-decimation filters do not.

Frequency response of the Gaussian $\frac{1}{\sqrt{2}}$ filter for several decimation ratios



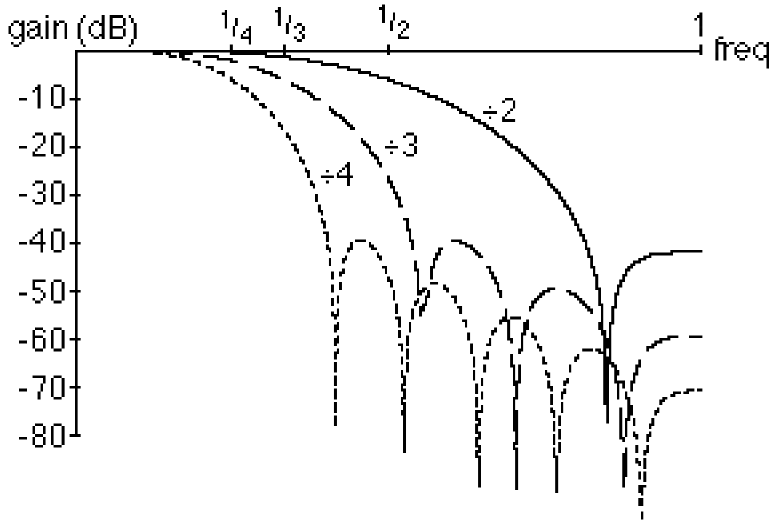
Frequency response of the Lanczos2 sinc functions

Here we show the responses of the decimate-by-2 filters related to the Lanczos2 filter.



Note that the Gabriel decimator lets more of the passband through, and has a sharper cutoff in the stopband, but also bounces back in the stopband at a higher level than that of the Lanczos2. The Turkowski decimator does not bounce back, though, and eliminates more of the highest frequencies than the other two. They all have approximately the same passband response and aliasing energy, but the aliasing energy is distributed differently throughout the spectrum, so they can be considered about equivalent.

Frequency response of the Lanczos2 sinc functions for several decimation ratios



References

Oppenheim 75 Oppenheim, A.V. and Schaeffer, R.W. *Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, N.J., 1975.