Investigations in Upsampling from 48 kHz to 96 kHz and applied Filters

With an upcoming discussion at the audio user forum The Crooked Path about filters used for SRC (samplerate conversion) I have tried to understand the discussed points in a better way. Thus I have started to investigate in this area by running my own tests. These tests are just done in a theoretical way, readers of this report may conclude by themselves about the results, especially when it comes to listening (practice).

But IMO the findings are also a bit de-mystifying the applied filters for samplerate conversion.

The test have been carried out with Acourate. The software is also intended as an audio toolbox and thus it allows to run several functions combined with the immediate display in time and frequency domain. All calculations are done in 64 bit double floating point precision which ensures a good accuracy of the calculations.

Ok, let's start.

1. Upsampling by inserting samples with value 0

Here we like to upsample from 48 kHz to 96 kHz samplerate. This simply means that we will get a double count of samples for the same time of a signal. A new sample must be in the center between two existing samples along the time axis. Its value (amplitude) has to found by a proper interpolation method. There are several methods existing. Linear interpolation, spline interpolation and others. One interpolation method is the sinc interpolation.

According to e.g. http://en.wikipedia.org/wiki/Upsampling the method is described:

Upsampling by integer factor

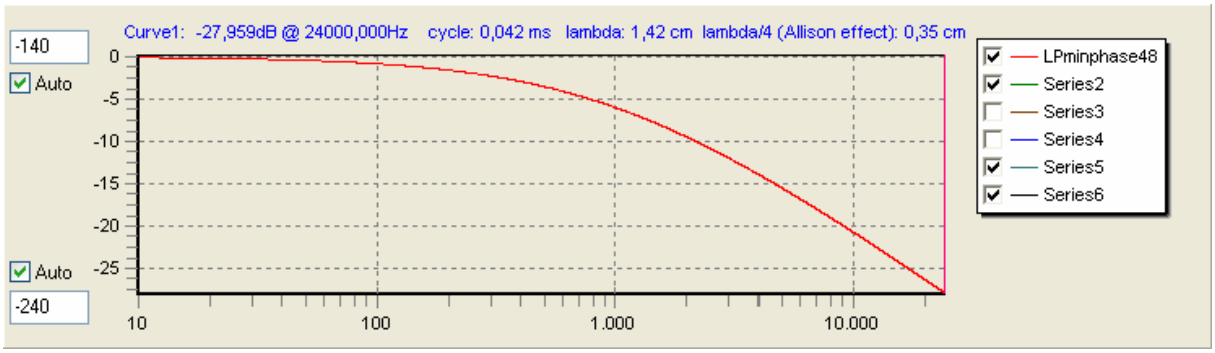
Let L denote the upsampling factor.

1. Add *L-1* zeros between each sample in *f*(*k*). Or, equivalently define
   $$g(k) = \begin{cases} f\left(\frac{k}{L}\right) & \text{if } \frac{k}{L} \text{ is an integer} \\ 0 & \text{otherwise} \end{cases}$$
2. Filter with a low-pass filter which, theoretically, should be the sinc filter with frequency cut off at $\frac{\pi}{L}$

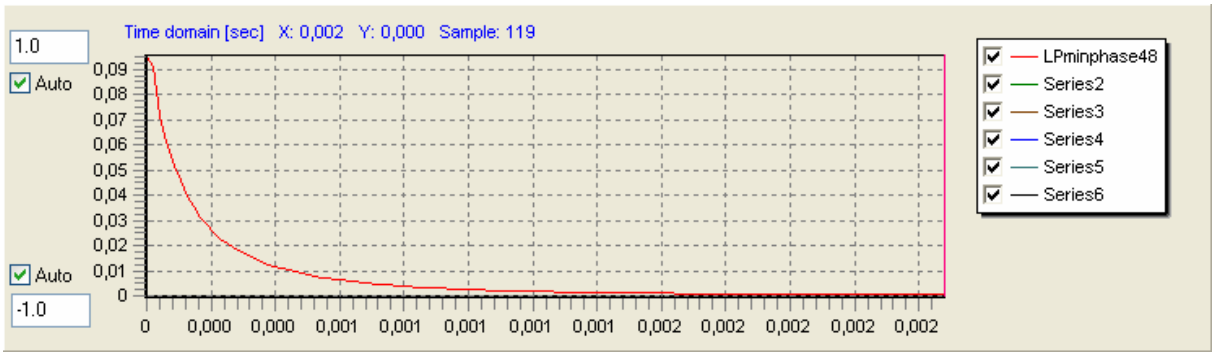
The second step calls for the use of a perfect low-pass filter, which is not implementable. When choosing a realizable low-pass filter this will have to be considered and it will have aliasing effects. These aliases can be removed to a reasonable extent by a finite impulse response low pass filter.

In our case (48 kHz -> 96 kHz) the value of L is simply 2. So in the first step we have to insert a new sample with value 0 between each existing sample of our 48 kHz signal.

The upsampling must be valid for all signals. Therefore I have not used a music signal but a test signal which is short and easy to understand. The signal is a lowpass filter, created as a minimum phase crossover filter of type Linkwitz-Riley 1st order with a corner frequency of 1000 Hz. The filter length is 65536 samples.

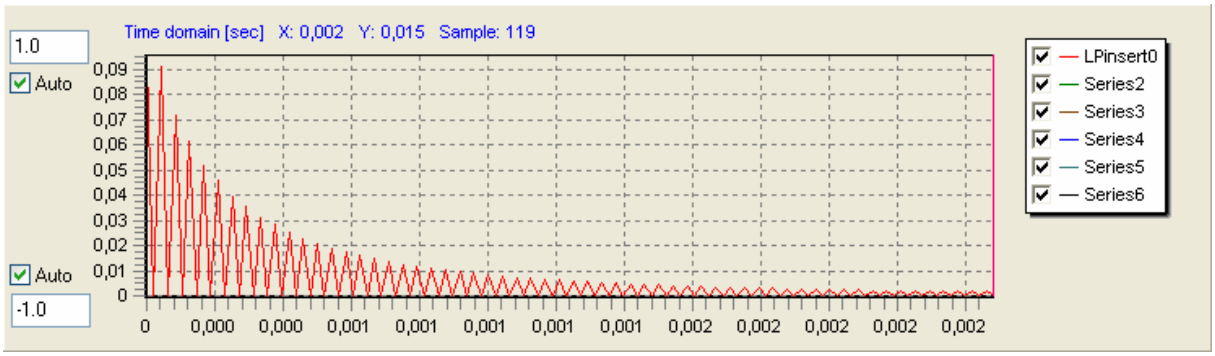


Frequency response of the lowpass filter. Max. frequency is 24 kHz according to Nyquist



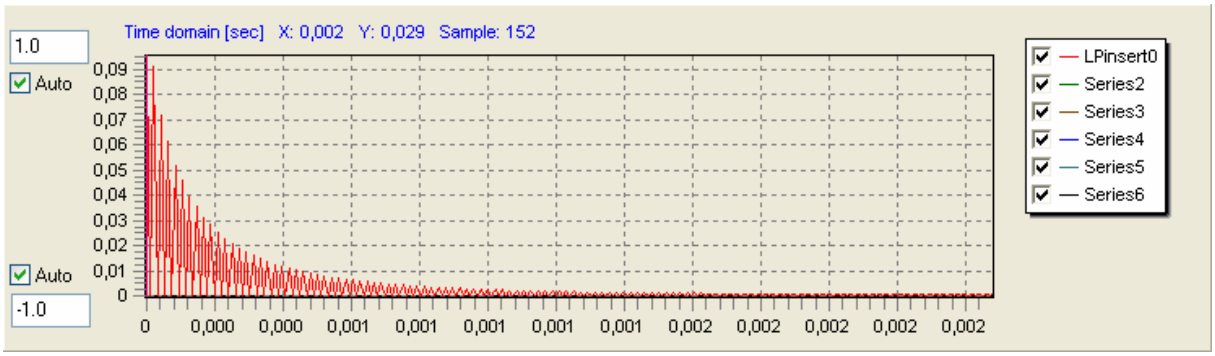
Pulse response of the lowpass filter. Shown are the first 120 samples of the minimum phase filter.

Now we insert new samples with value 0 between the existing samples:

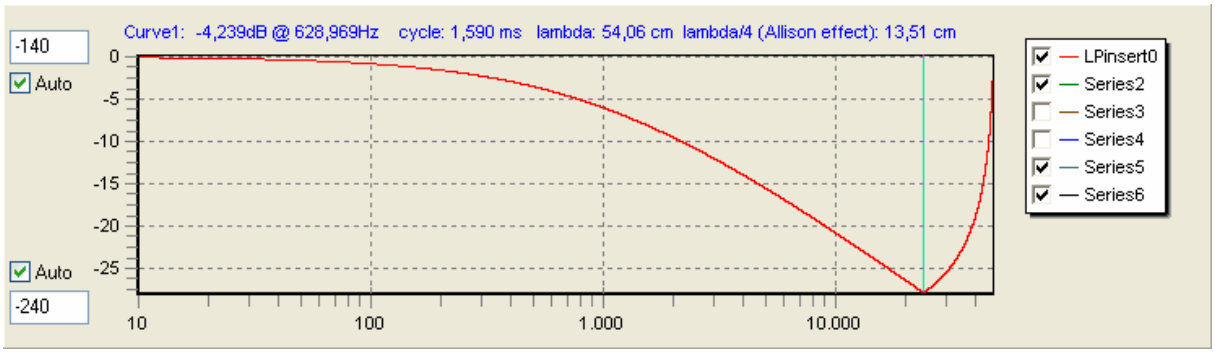


Acourate does not show a smooth interpolation but just a simple line interpolation between the points. Thus each sample point can be easily identified.

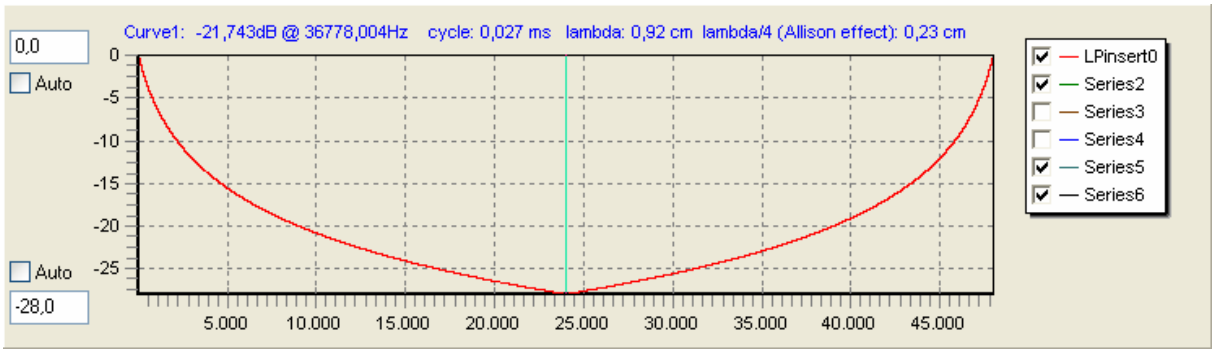
The new signal now has a length of 131072 samples. And as we upsample to 96 kHz we now have to consider the result as a 96 kHz signal. So we load this signal after switching to 96 kHz samplerate:



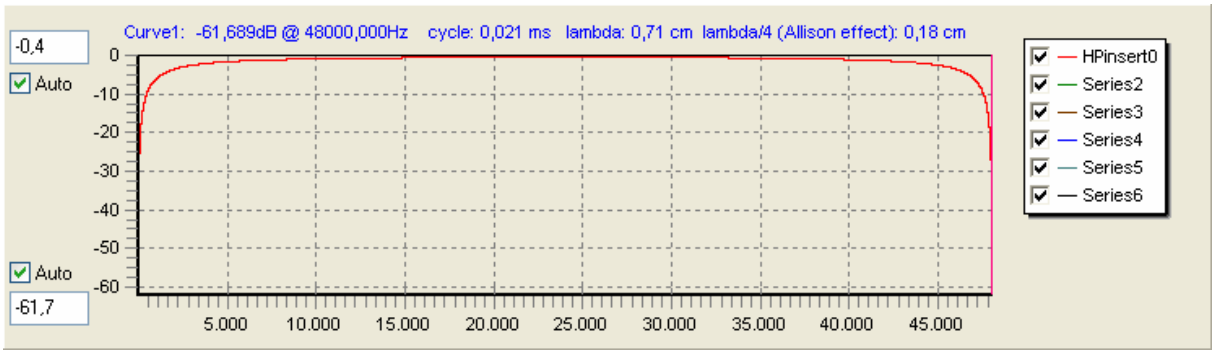
So with the same timescale in the bottom axis the signal looks compressed. We have double samples in the same time.
The frequency response now must go up to 48 kHz:



We must wonder. Before the zero insertion the response just reached 24 kHz but now we can see the curve expanded to 48 kHz with a rising slope at the end. ???
We can see more when we switch the frequency axis from logarithmic to linear display.



24 kHz now is in the center of the chart and it seems the left side of the center is mirrored to the right side.



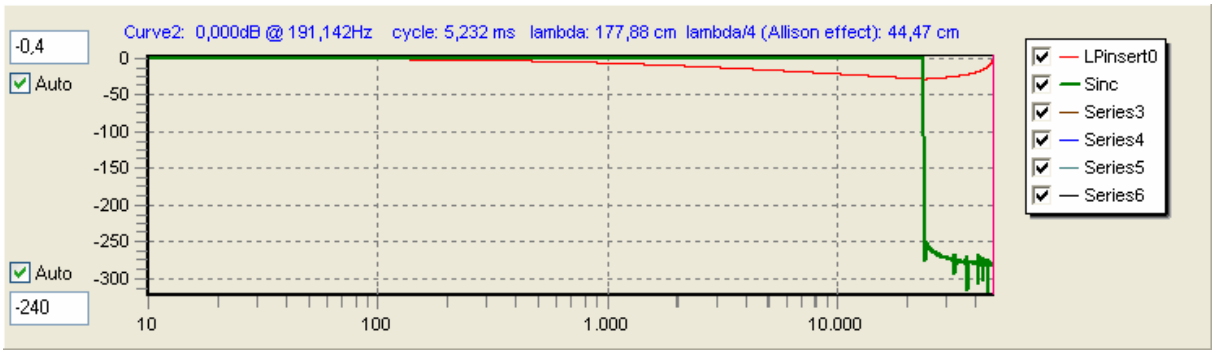
For better understanding of the mirror effect the previous picture in addition shows the frequency response of the corresponding highpass filter (Linkwitz-Riley 1st order 1 kHz) after zero insertion.
It is now very clear: the insertion of the new samples with value 0 will create new frequencies from 24 kHz to 48 kHz. The frequency response of this range is exactly the mirror of the frequency response between 0 and 24 kHz.

What does this mean? We have created new frequencies which have not existed BEFORE the zero insertion. But we also know that the amplitudes for the new frequencies are **definitely wrong**. We have no information in our original signal about the behaviour at frequencies above 24 kHz.

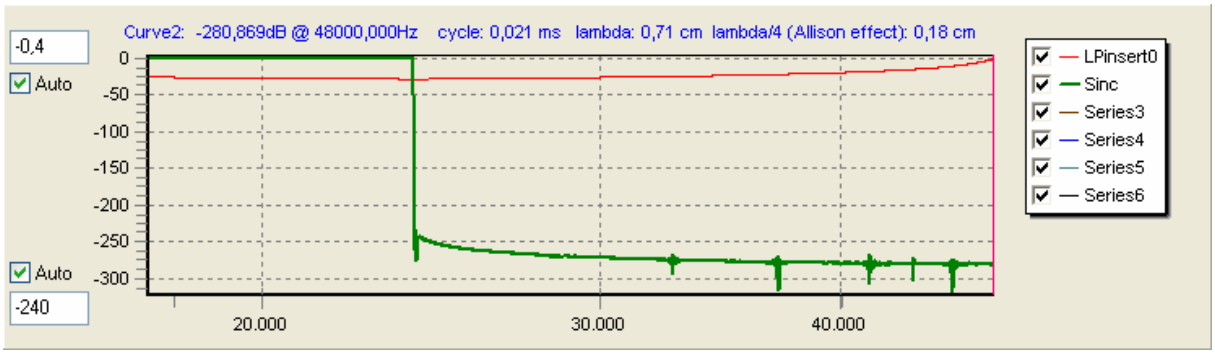
This clearly demonstrates us that it is necessary to apply a new filter after zero insertion. As there is no frequency above 24 kHz before upsampling the filter has to cut all frequencies above 24 kHz. Below 24 kHz the original signal should be ideally untouched. This leads us to the question which filter we can apply.

2. Sinc filter for the upsampled signal

The citation from Wikipedia is talking about a sinc filter. With Acourate we can create such a filter (remark: to avoid artefacts from signal truncation windowed with a Kaiser window):

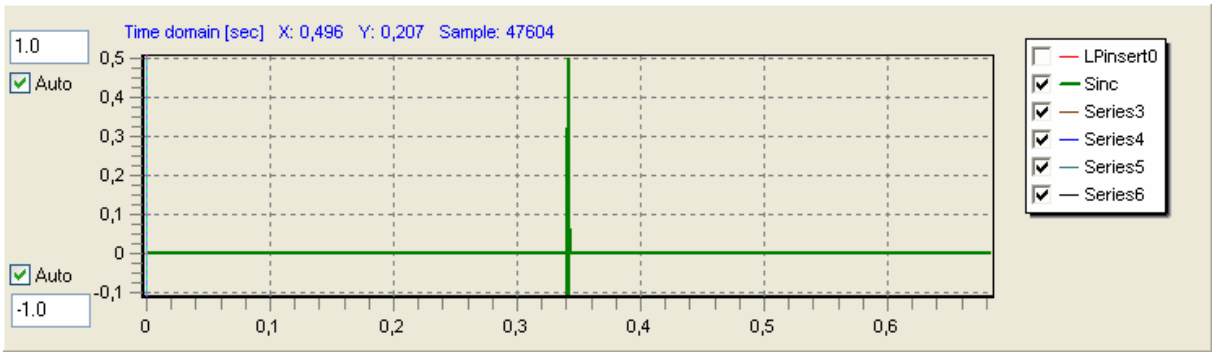


The 24 kHz sinc filter shown as green curve has a steep cut of all frequencies above 24 kHz.

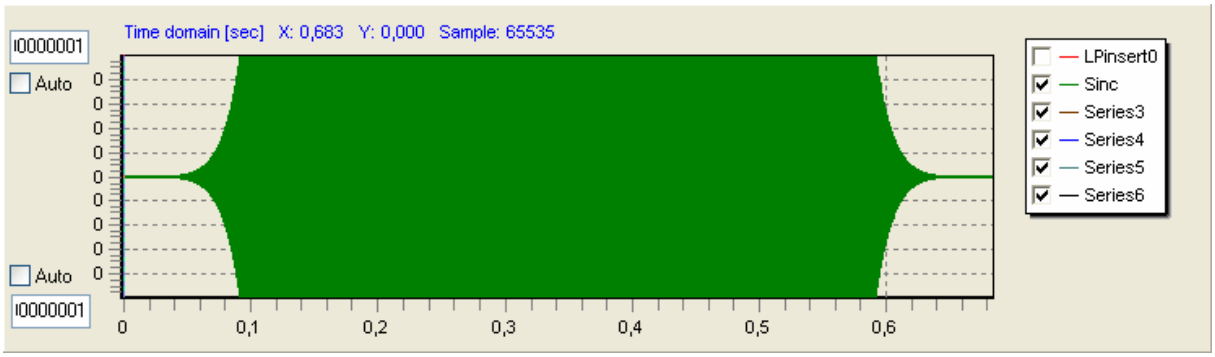


The zoom-in shows this a bit better. All frequencies above 24 kHz have a much lower level, here below -250 dB. The theoretical sinc filter is endless deep. But we must taken into consideration that the number resolution is 64 bit double float and the length of the sinc filter is 65536 samples. A perfect sinc filter indeed would have infinite samples.

The next pictures show the sinc filter in the time domain:



At the first sight the filter does not look endless. But the picture changes when we zoom the amplitude:

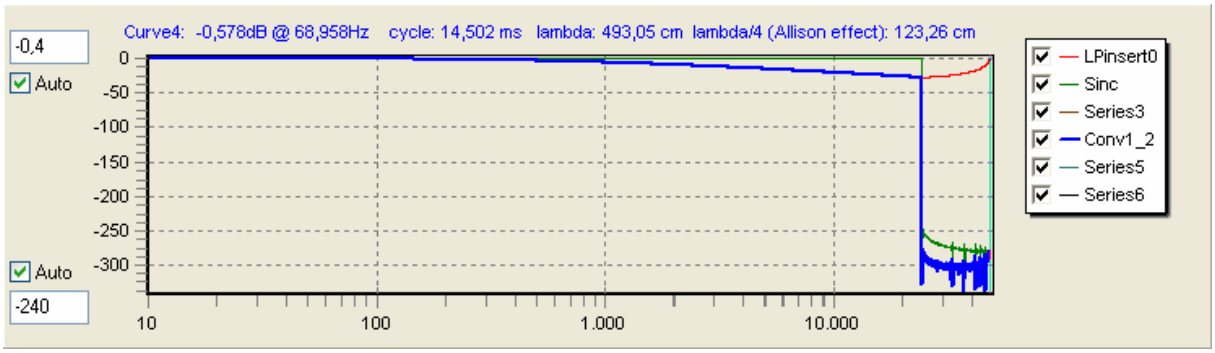


The max. amplitude in the chart is now 0.00000001 = -160 dB !

As the sinc filter is linear phase this means that the filter has a pre-ringing of 32768/96000 = 0.34 seconds. It should be clear that we cannot accept such a long pre-ringing.

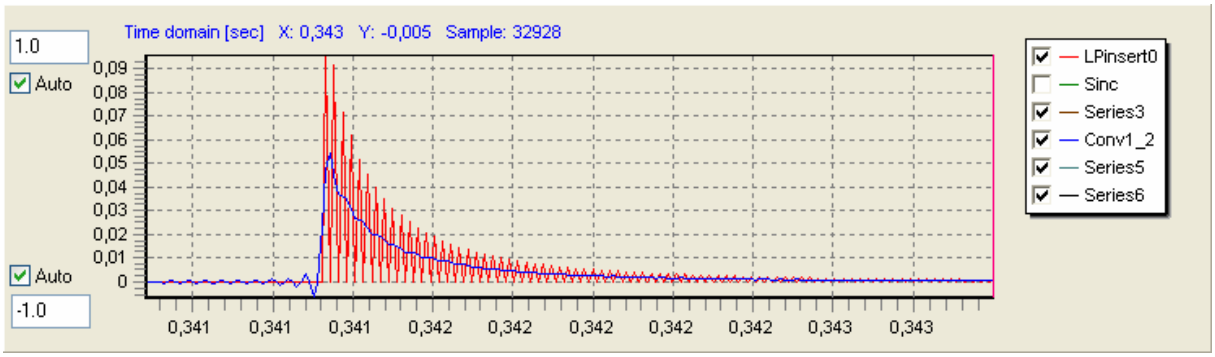
So indeed we know that the sinc filter has no practical value for our audio application. And we can deduct that we need to select another filter and we are forced to make compromises.

Just for completeness the picture of the upsampled signal convolved with the sinc filter:

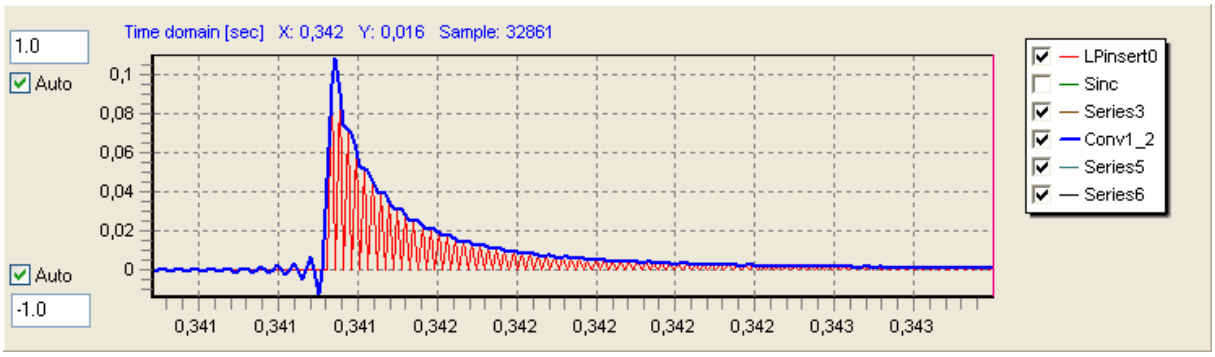


We can still see how the result is following the shape of the mirrored frequencies but anyway below -250 dB.

Let's view the sinc filtered test signal.



The blue curve has lost the big jagging at the first sight. We also notice a lower amplitude of the time domain signal whereas in the previous picture the frequency response of the filtered signal has the same amplitude as the original signal, but only below 24 kHz. Indeed we have to apply a gain of factor 2 to compensate for the energy loss and to get the same signal height in the time domain.



The convolution with the sinc filter has also introduced a time delay of 32768/96000 = 0.341333.. seconds. For better comparison the signal with zero insertion is shifted accordingly by the same amount of delay.

At the beginning we can detect an oscillation. It is caused by the pre-ringing. It looks quite small compared to the pulse height but it is dangerous to assume it has no influence. Also the filtered signal still shows some ripple.

Summary:

We have learnt that a perfect sinc filter would give an optimal filtering. Disadvantages are given in the endless ringing of the sinc filter, in the restricted accuracy of the number representation and simply in the fact that it is impossible to create the perfect sinc filter. Even in case of an acceptable pre-ringing the signal is also delayed by a long non-acceptable time.

The statement of Wikipedia repeated:

The second step calls for the use of a perfect low-pass filter, which is not implementable. When choosing a realizable low-pass filter this will have to be considered and it will have aliasing effects. These aliases can be removed to a reasonable extent by a finite impulse response low pass filter.

3. Upsampling filter of Adobe Audition 1.5

So we are interested in filter designed properly for our purpose. We must expect to get a non-ideal reconstruction of the signal including aliasing effects.

There are many solutions and scientific papers about how to design such filters. All they have the target to create

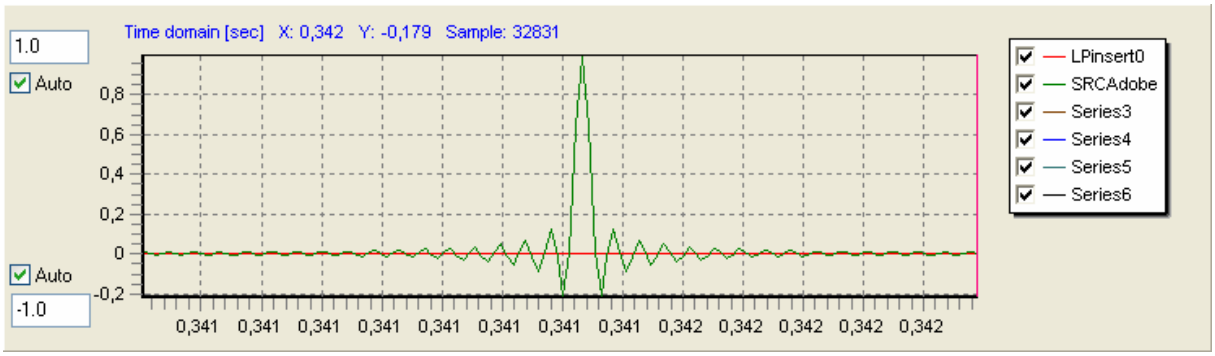
- short or very short filters
- with steep slopes
- less ringing
- pre-defined behaviour in passband, transition area and stop band, e.g. oscillations and ripples

This paper does not present how to design such a brickwall filter. It takes some given filters and demonstrates their behaviour.

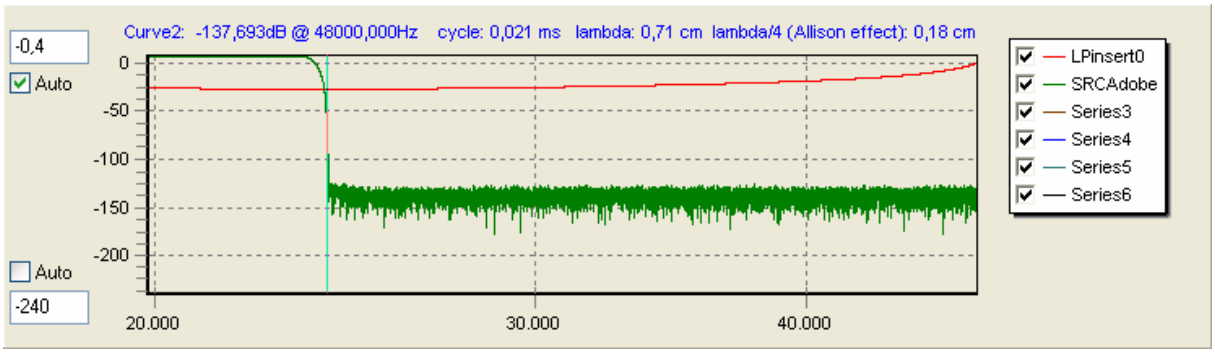
There is a lot of discussion about the filter quality and the resulting sound. Comparisons of different filters are e.g. given at http://src.infinitewave.ca/

To analyze given filters it is necessary to know about the pulse response of the filter. A simple way to do this is to upsample a Dirac pulse. The result simply describes the filter. It can be further used for upsampling of other test signals like music or the given test signal with zero insertion.

The first upsampling filter to be discussed is from Adobe Audition 1.5, created with best setting.

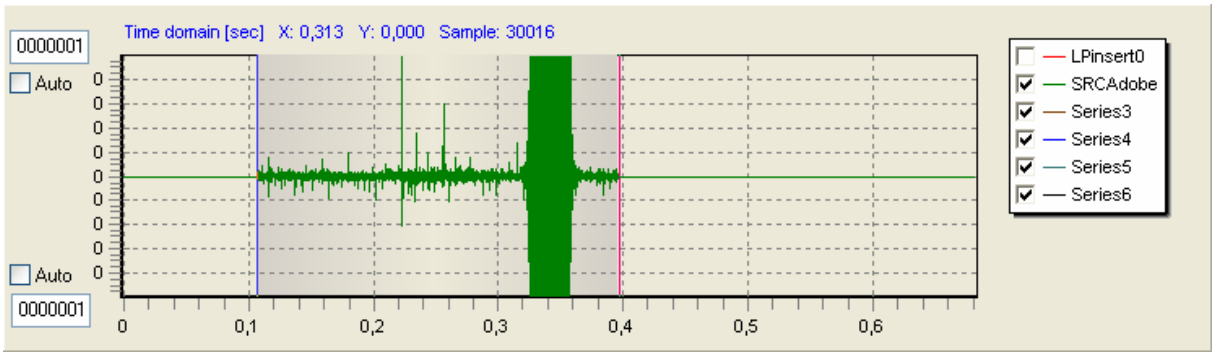


The filter is of type linear phase. The frequency response of the filter shown for the upper frequency area:

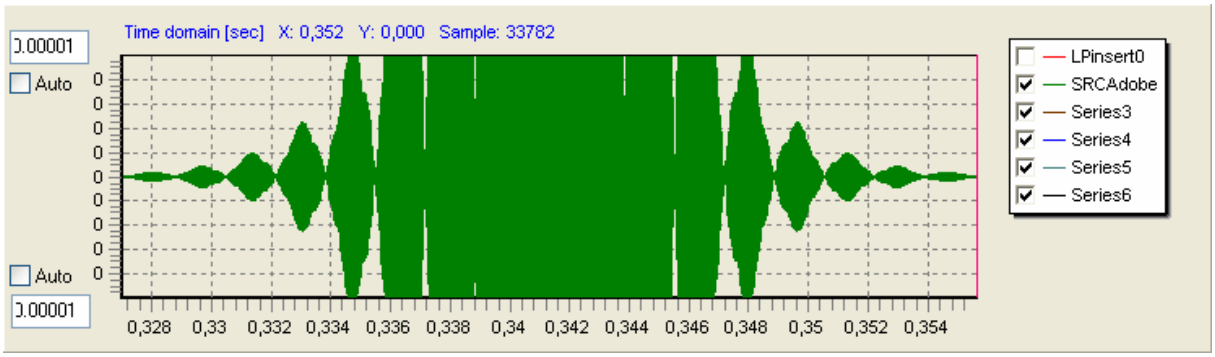


As the filter is given in 32 bit single float precision the noise level is higher, the noise floor is at about -130 dB.

The length of the filter can be found by zooming into the amplitude in the time domain and this gives a first surprise:

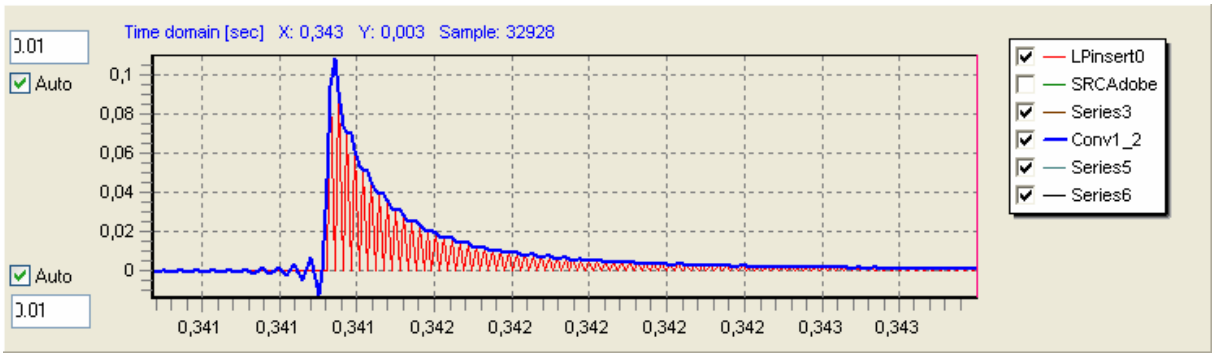


The signal is non-zero over a length of about 27890 samples. This is quite a long filter. There is also some strange part before and behind the main filter part, mainly before.



The part with a higher amplitude (see scale at left side) is still 2750 samples long. So the Adobe Audition 1.5 filter isn't a very short filter.

The convolution of the Adobe filter with the test signal results in:

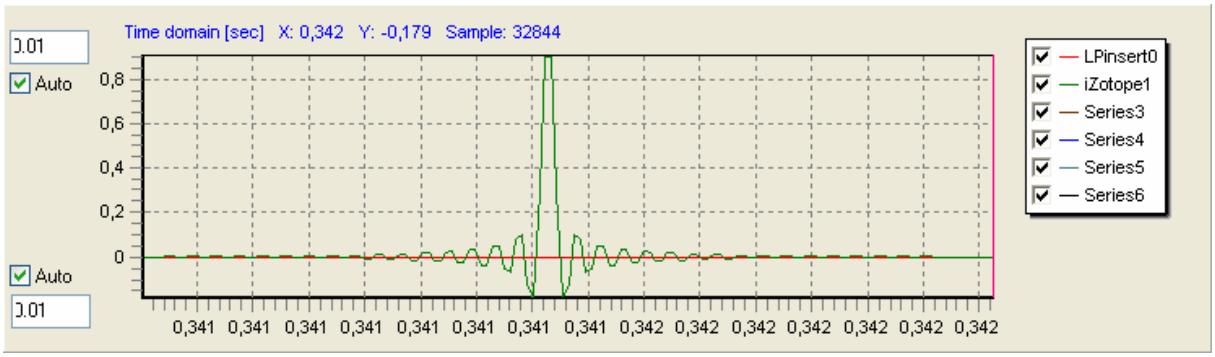


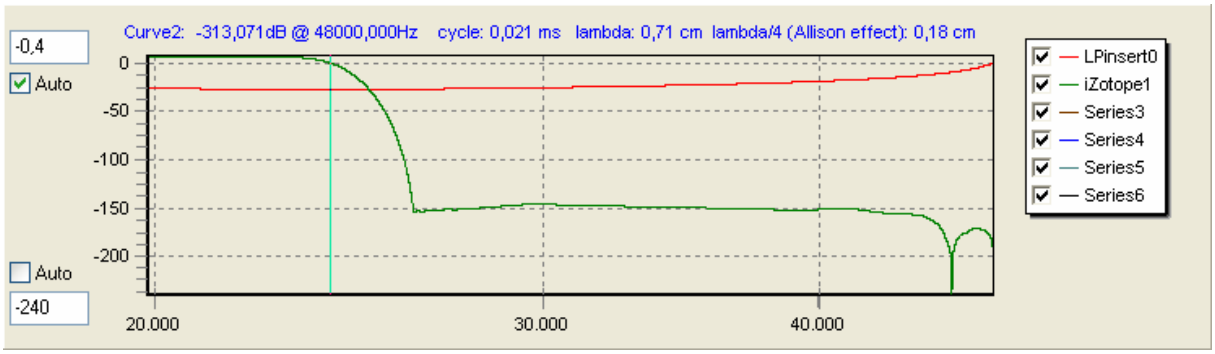
The result looks similar to the convolution with the sinc filter. This is not too bad as the filter is anyway shorter.

4. iZotope filters (RX Advanced) of type linearphase, minimumphase and mixed phase

The pulses of the iZotope filters have been created in the same way like the Adobe filter.

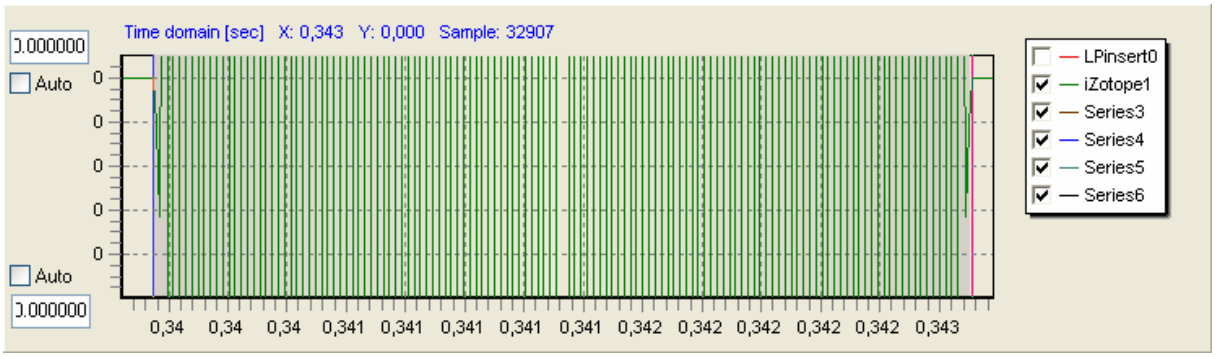
a) iZotope linear phase filter



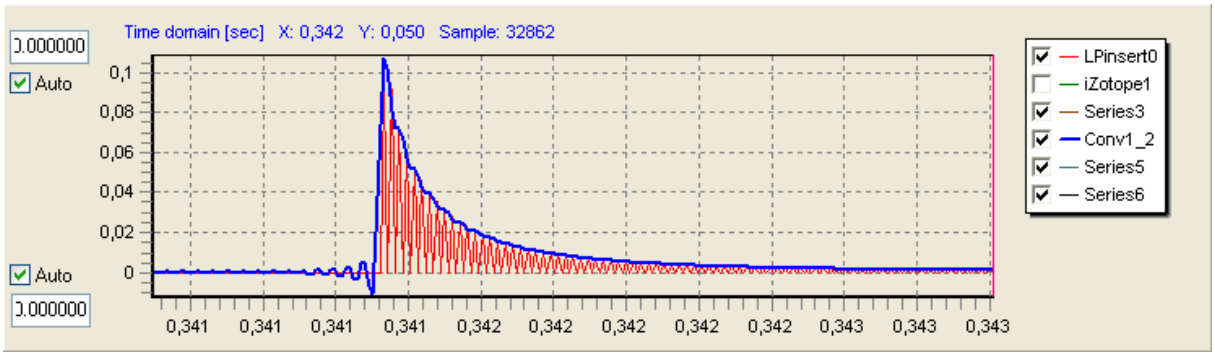


The filter shows some interesting behaviour. It is symmetric. There is no single peak. The max. amplitude is about 0.9, the peak is between two points (intersample peak) but still below 0 dB.
The frequency response shows no noise (clear line). The max. amplitude in the stop band is -146 dB.

But one point is confusing: the transition area is beyond 24 kHz. This will introduce aliasing.

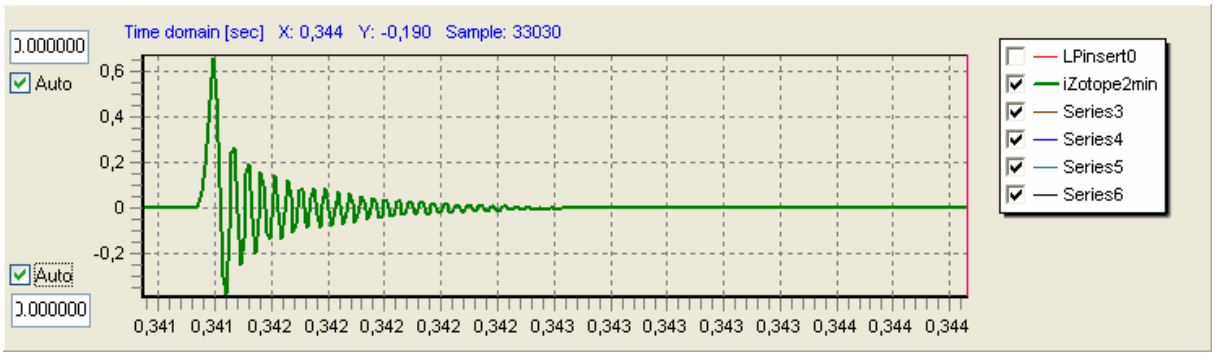


Filter length is 266 samples. A very short filter of roughly 2.8 ms duration, 1.4 ms pre-ringing.

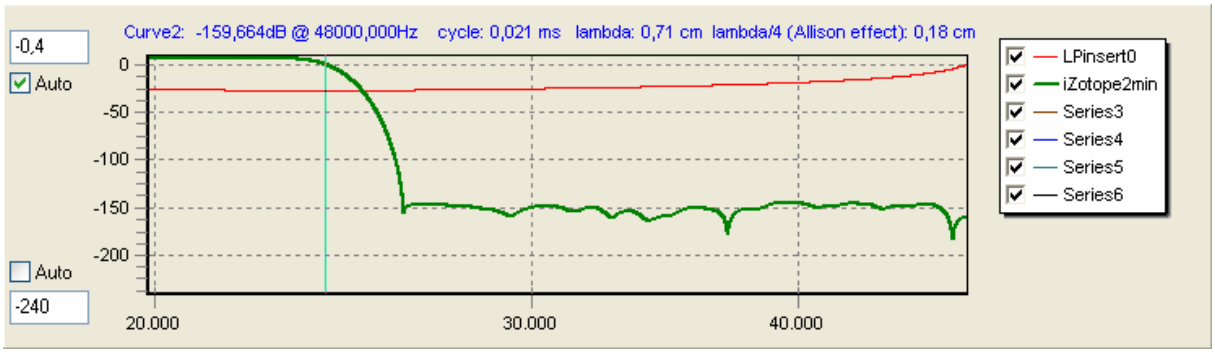


The filtered result also looks a bit more smooth than the Adobe result. This is not a bad filter.

b) iZotope minimumphase filter

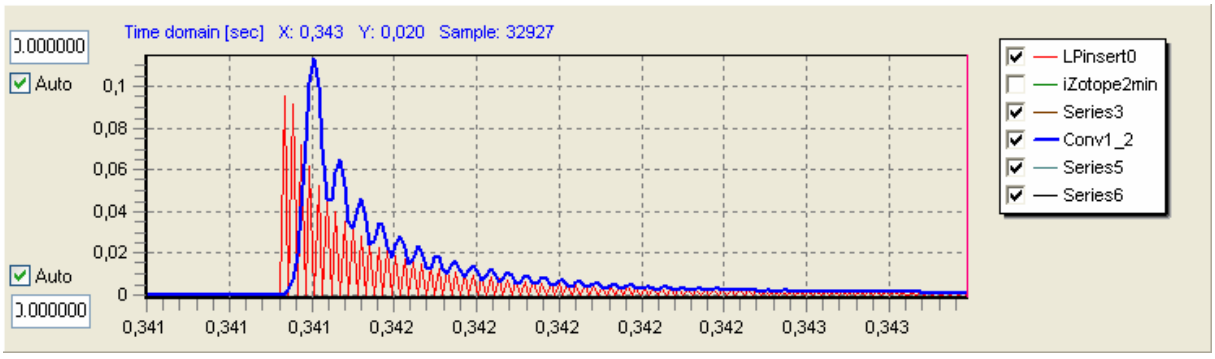


This is clearly a minphase filter. The filter length is also 266 samples.



The frequency response is a bit different in the stop area but also attenuated enough.

As the rising edge of the minimumphase response of the iZotope filter creates a small delay the convolution result shows this too:

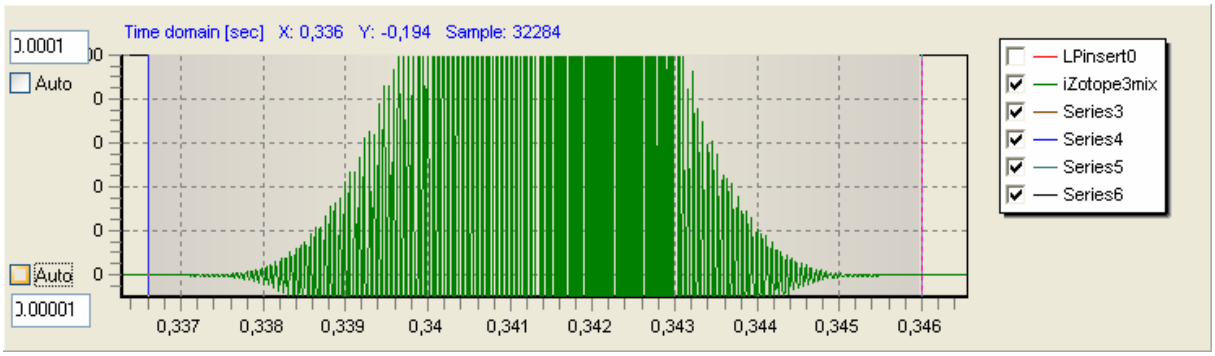


Beside the small introduced delay it can be clearly noticed that the minimumphase filter does not smooth as well as the linearphase filter.

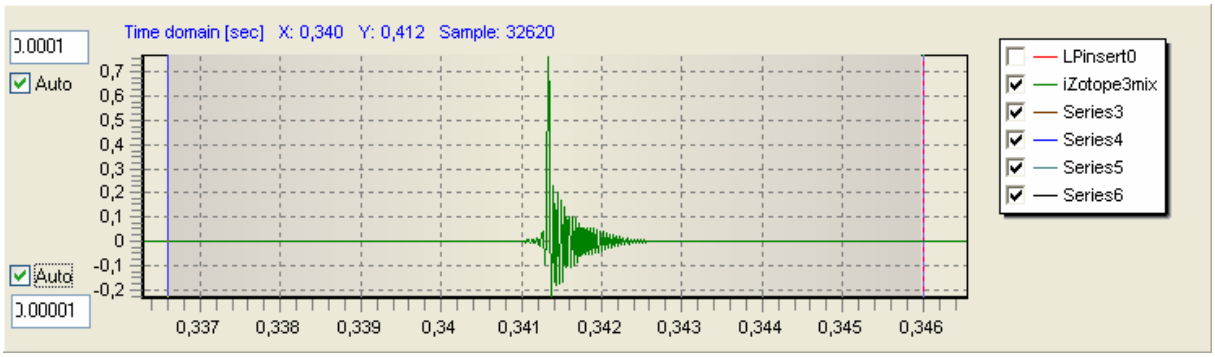
c) iZotope mixed phase filter

It is also possible to create a mixed phase filter between linear phase and minimum phase to find another compromise.

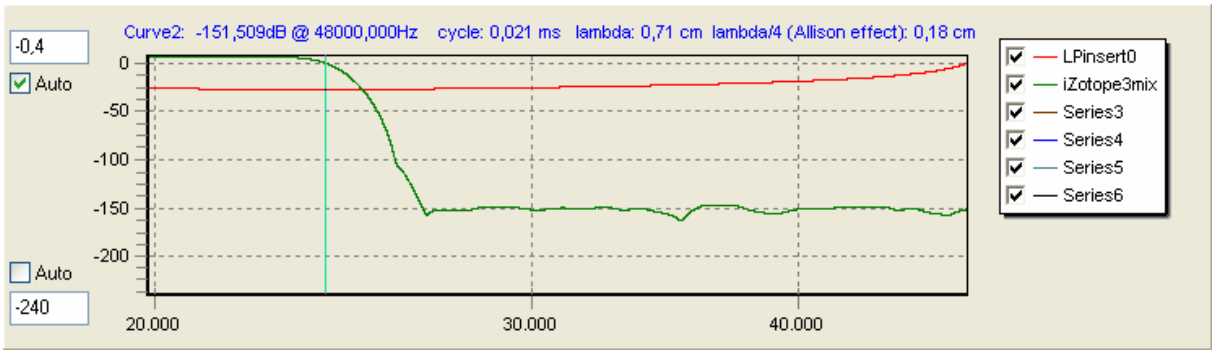
The total filter length here is about 903 samples. Thus this filter is longer compare to the other iZotope filters.



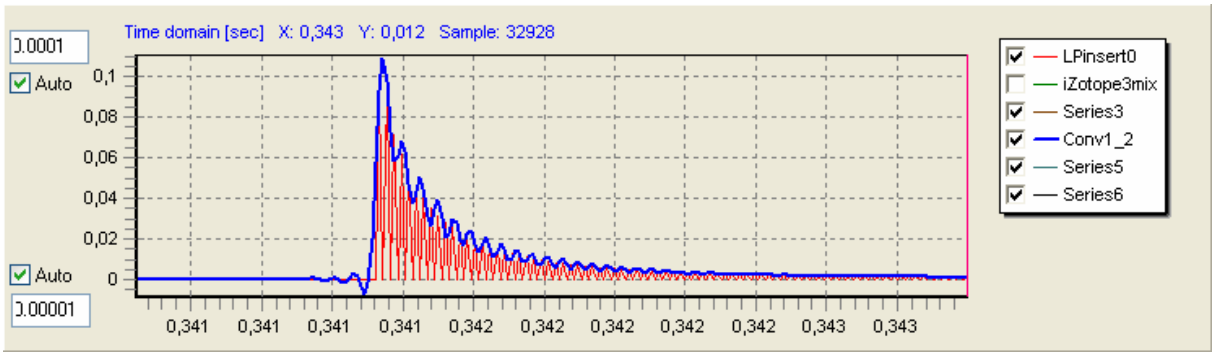
The behaviour of the non-symmetry of the mixed phase filter is shown in the full scale view:



The frequency response is again like the other filters



The falling slope in the transition area looks a bit disturbed whereas the stop band looks pretty nice.



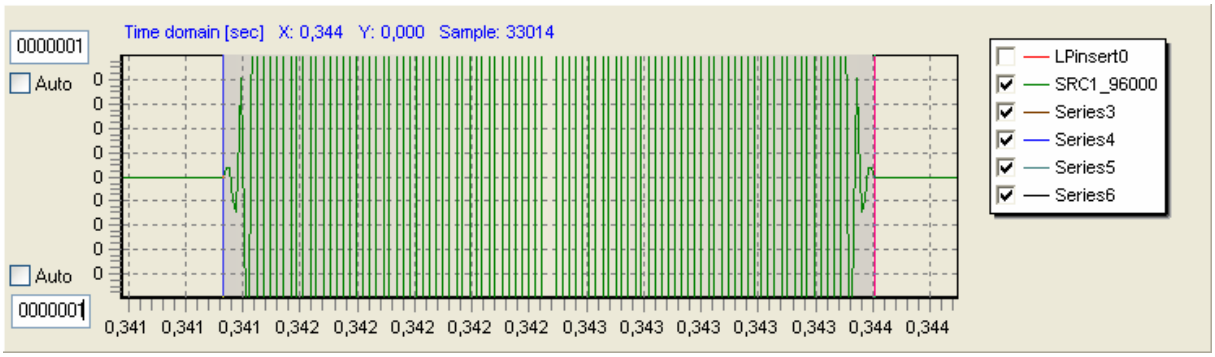
Now the delay of the pure minimumphase filter has gone. The filtered signal looks a bit more smooth too. But the quality of the shape of the linearphase filter is not met.
As already said the mixed phase filter is a compromise.

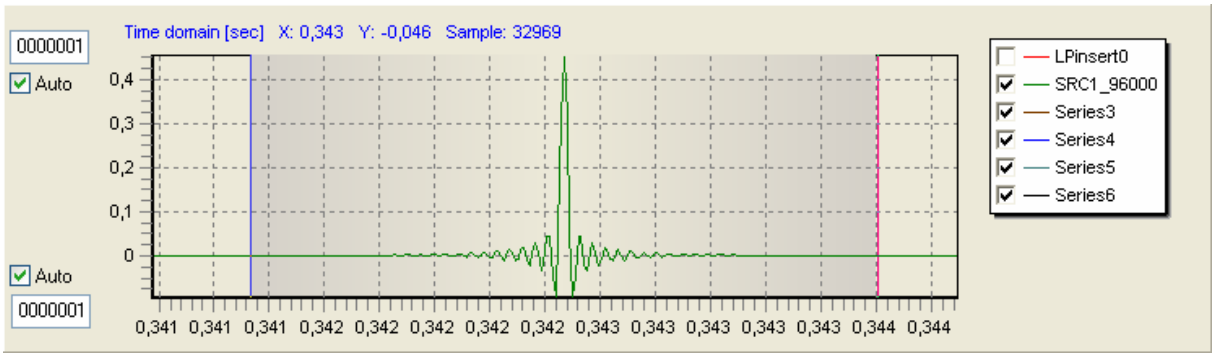
Conclusion:

The iZotope filter definitely shows a very good smoothing with the linearphase filter. Furthermore it is quite short with 266 samples. It is no wonder why it has good reputations. The problem of the transition area beyond 24 kHz is still there but this is correctable. It is also a question if the aliasing can even be noticed as the applied brickwall filters during the 48 kHz recording already suppress frequencies between 22 kHz and 24 kHz. Maybe it is even intended by the iZotope author to have such a frequency response.

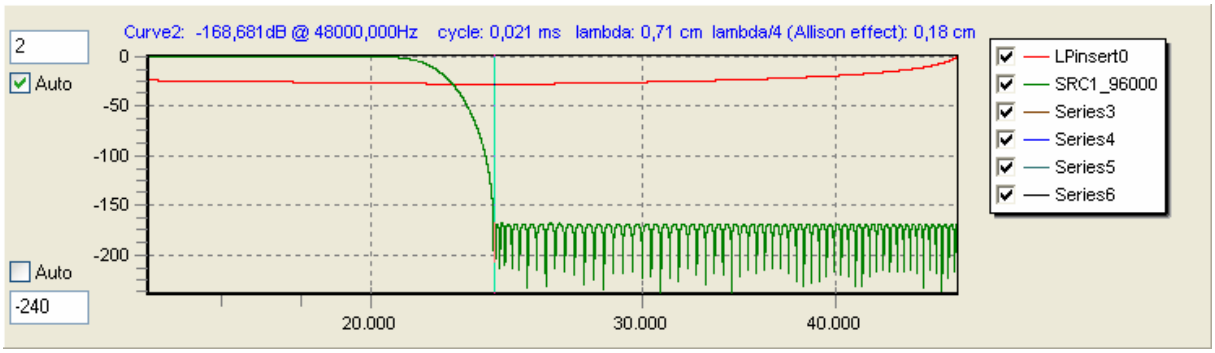
5. Acourate samplerate conversion

Acourate also allows a samplerate conversion. Its filter is a Kaiser filter. The filter is also a linear phase filter with a length of 219 samples.



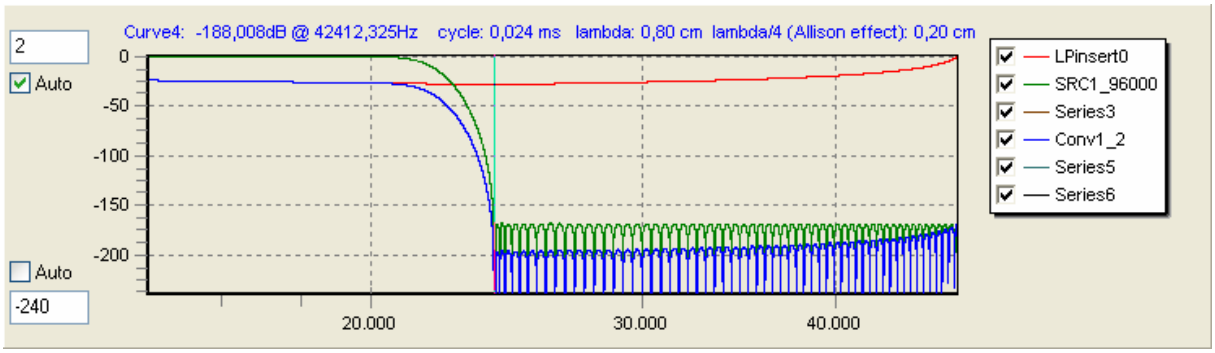


The frequency response:

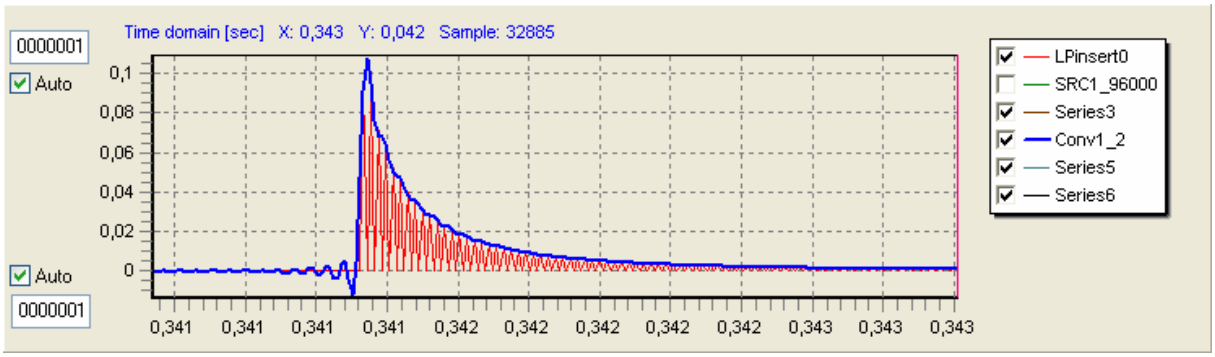


The filter transition area stops at 24 kHz. The stopband shows the equiripple design underlying the filter design.

The frequency response after convolution:



And the time domain view:

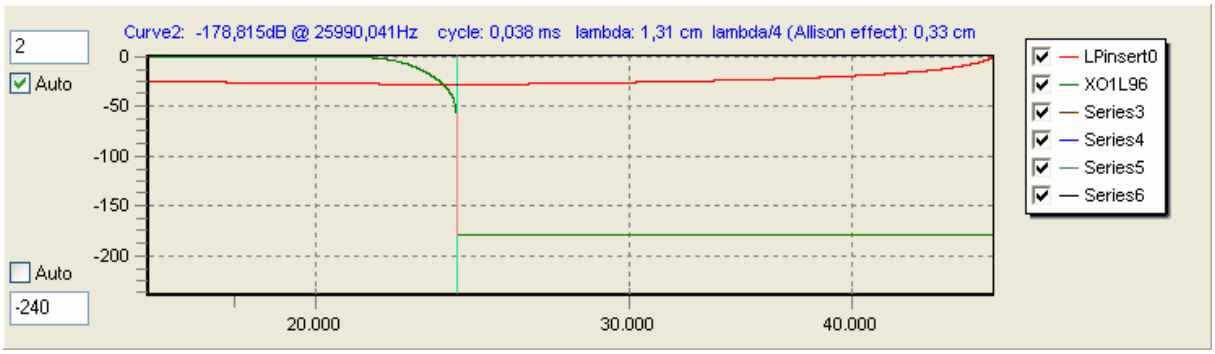


The result is the smoothest result of all. But in the given case we also need to keep in mind that the filter is given in 64 bit resolution whereas the Adobe Audition and iZotope RX Advanced filters are given in 32 bit resolution.

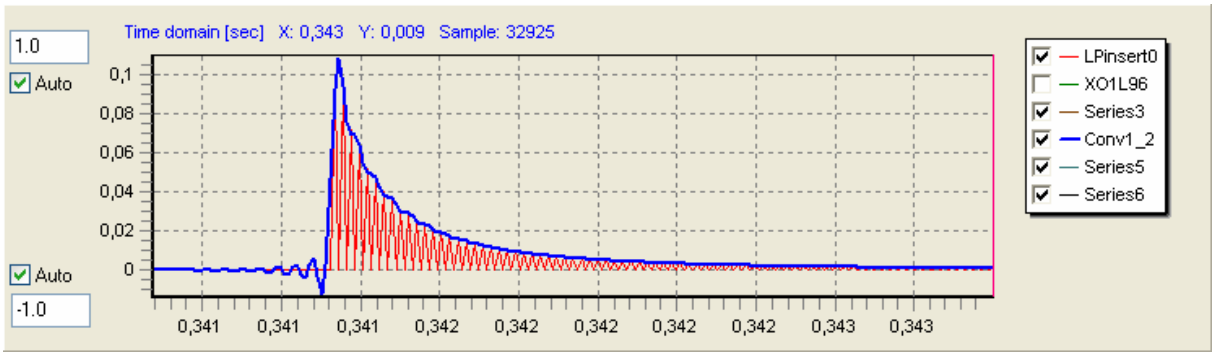
The influence of small numbers shall not be underestimated ! So finally I like to show this effect.

6. Influence of small numbers

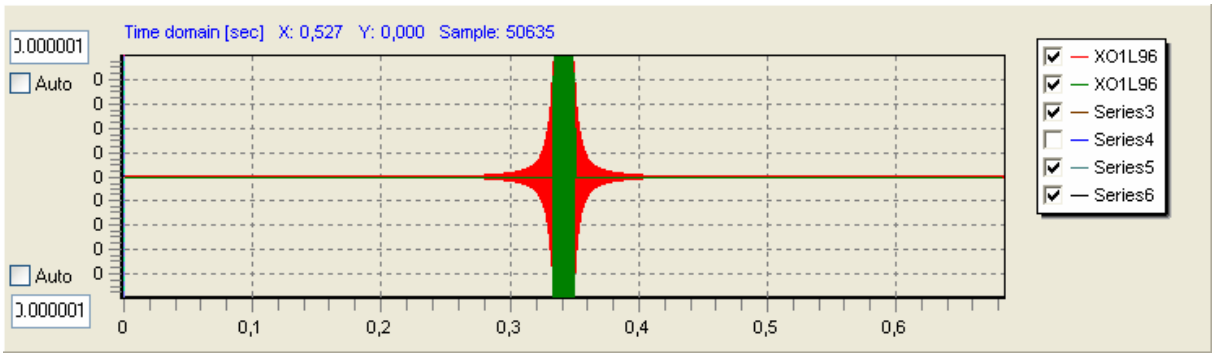
To demonstrate the influence I like to introduce a linearphase Neville-Thiele crossover filter of order 10 at the corner frequency 22388 Hz. Thus the stop band will start at 24 kHz.



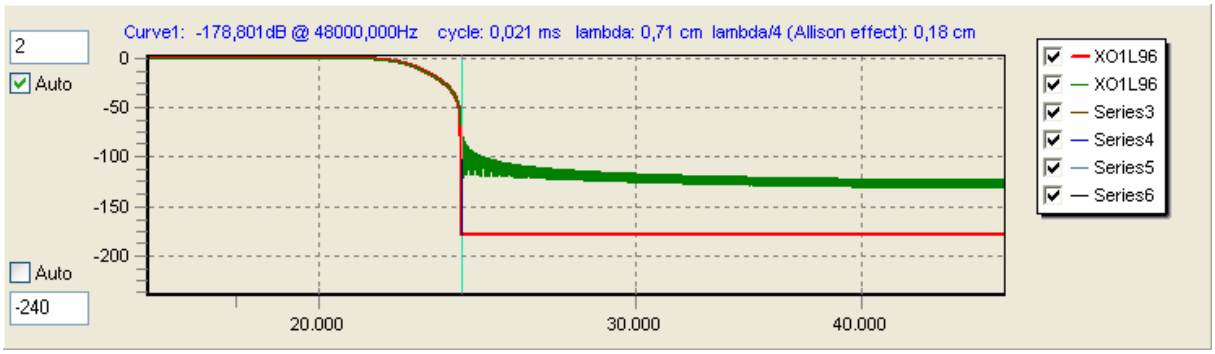
The convolution of the test signal with this nice looking filter will show:



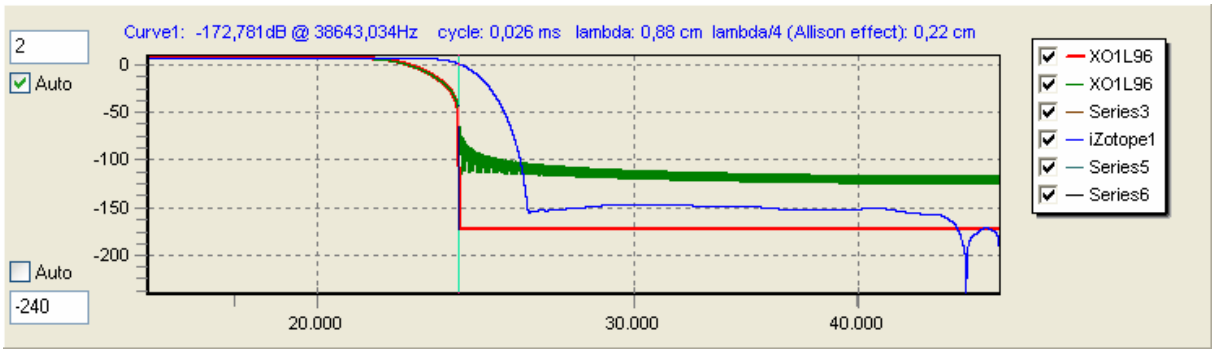
But now we shorten the filter to a length of 1500 taps. Indeed we set all other taps smaller than 0.000001 to 0.



The resulting frequency response compared to the original one:



And to show how this filter with 1500 taps compares against the iZotope filter with 266 taps:



The iZotope filter is the better one.

Summary:
This proves that a brickwall filters for samplerate conversion have to be designed carefully. We have learnt that good designed filters will result in nice curves. And I'm sure we can directly conclude from these curves to the resulting sound. But there is no mystic behaviour in such filters. They are just properly designed under the constraints of allowed filter length, pre-ringing, filter properties as ripple and behaviour in transition area and also finally the number resolution for the filter as well for the signal.

