Aspect Ratio Converters & polyphase filters

Dr Richard Porges

InSync Technology

Version 1.1 April 2020

Abstract

Dr Richard Porges, Head of embedded software at InSync technology, provides an introduction to video Aspect Ratio Convertors (ARCs) used for changing the size or sampling structure of video pictures. It starts with an introduction to general interpolation theory and an explanation of why filtering theory is needed and how it can be implemented with a hardware phase accumulator. Following this is an explanation of polyphase filters that discusses how they are generated, what their spectral properties are, and how they may be optomised numerically given a particular set of hardware limitations on coefficient bit-width. Finally comes a section discussing multiple field apertures used for vertical interpolation, with particular relevance to interlace video signals.

Contents

1 Introduction **2**
1.1 Simple interpolation . . . 2
1.2 Video interlace and multiple field apertures . . . 2
1.3 Aspect Ratio Converters . . . 4
1.4 The phase accumulator . . . 7

2 Polyphase filters **9**
2.1 Super sampling . . . 9
2.2 Generating individual phases . . . 11
2.3 Aperture length . . . 14
2.4 More examples . . . 15
2.5 Symmetric filters . . . 17
2.6 Symmetric or non-symmetric? . . . 22
2.7 Magnitude and phase responses . . . 24
2.8 Numerical evaluation of filter coefficients . . . 33
2.9 Least squared error optimisation . . . 33
2.10 Quantisation . . . 36

3 Vertical interpolation **41**
3.1 Quincunx interlace spectrum . . . 41
3.2 Single and multiple field interpolators . . . 44
3.3 An example of a multiple field aperture . . . 46

1 Introduction

A very brief general introduction to polyphase filters and multiple fi eld filter apertures is followed by a more detailed introduction to video aspect ratio converters.

An Aspect Ratio Converter (ARC) processes a stream of video data in real time to change the size of the video image (or slide it horizontally or vertically across the screen). To do this to a high quality requires the use of polyphase filters. To complicate things further, the nature of video interlace creates the need for inputs from not just the current field, but from adjacent ones too. This is referred to as having a filter with a multiple field aperture. These two topics are introduced very briefly here before the main body of this introduction which is devoted to ARCs in general.

1.1 Simple interpolation

Simple interpolation is achieved using polyphase filters. A polyphase filter is a digital filter that is used to create extra samples in a sequence of data samples. They can be used to change the sampling frequency of a sequence, as in several audio applications. Polyphase filters can also be used to resample a sequence at the same frequency but with a slight offset. One example of this would be keeping the size of a video image the same but moving it by a sub-pixel amount (i.e. moving it less than one whole sampling interval); another example might be a time base corrector.

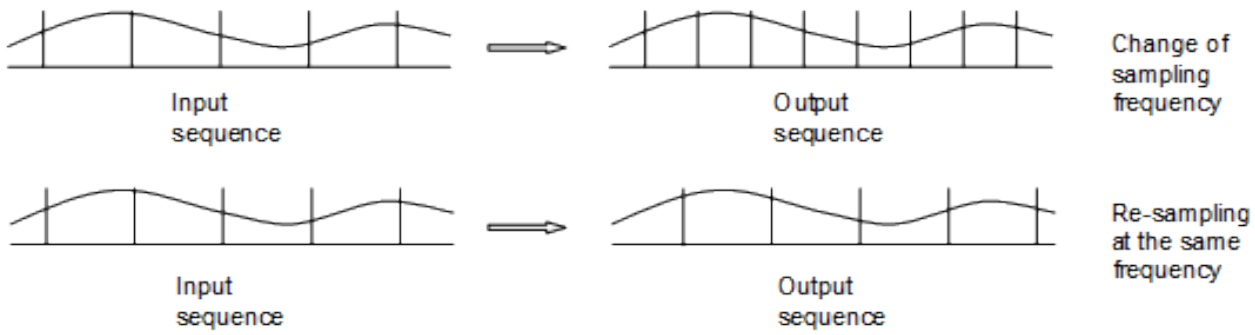

Figure 1: Examples of re-sampling, either at a different frequency or at the same frequency.

Although polyphase filters crop up in a variety of situations and for a variety of purposes, this document is written with digital video in mind, specifically ARCs.

1.2 Video interlace and multiple field apertures

The video signal has horizontal, vertical and temporal components. The electron gun scans horizontally across the screen, then moves down (vertically) to scan another horizontal line. When it

reaches the bottom of the screen the gun moves from the bottom of the screen back to the top of the screen and starts again.

Strictly speaking, the horizontal samples are not independent of time since there must be a finite time difference between one sample and the next one, however small that difference may be. In practice the difference is small enough to be ignored and it is generally stated that the horizontal component is independent of the vertical and temporal components. For this reason most, if not all, of the examples in the main polyphase section of this document refer to the horizontal processing of a picture.

This is not the case for the vertical component. For historical reasons the horizontal scan lines of each video frame are *interlaced*[1] split into two fields of approximately 300[2] lines which are then transmitted sequentially. If the lines are numbered consecutively then the odd numbered lines form one field and the even numbered lines form the other.

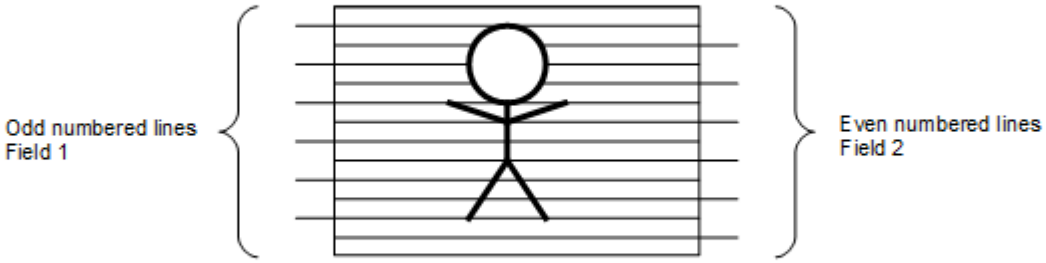

Figure 2: An example of an interlaced image.

The fields are sampled, and displayed, at different moments in time. Instead of each video frame appearing every $1/25^{th}$ second, i.e. at frame rate (for PAL), half the picture appears every $1/50^{th}$ second, i.e. at field rate. This means that a single field only contains half the picture, which obviously affects the vertical resolution of the picture severely. Single field vertical interpolators are simply not adequate for decent quality products. The full vertical resolution can only be obtained from including the adjacent field, however this is from a different moment in time, i.e. $1/50^{th}$ second later (or earlier). This is not a small enough time difference to ignore and thus, unlike the horizontal component, the vertical component of a picture is related to the temporal components of the picture. As a consequence the vertical processing involves several lines from several fields making it a two dimensional problem instead of a one dimensional one. It is essentially more complicated.

[1]This was for SD; most HD standards are now progressive, i.e. not interlaced.

[2]This figure depends on whether the standard is PAL or NTSC. It can also be confusing because the number of lines per field may be stated as being exactly half the number of frame lines, or alternately as the number of active lines per field, i.e. not including the vertical blanking period. This document uses the latter so, for example, a digital PAL D1 stream has 288 active lines per field, written as 288 lines / active picture height (or 288 lines/aph).

1.3 Aspect Ratio Converters

A television image is formed by sending a varying voltage to the electron gun at the back of the Cathode Ray Tube. To stretch, or squash, the image it is necessary to stretch, or squash, the time varying voltage. Although the voltage is an analogue signal, all processing is performed in the digital domain with a sampled representation of the signal. (The voltage is represented using pulse code modulation, effectively sending the numeric value of the voltage rather than the voltage itself.) There are equal time intervals between successive samples.

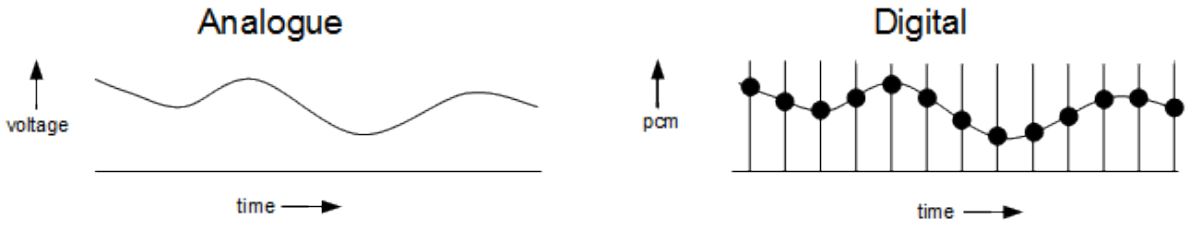

Figure 3: Digital sampling of an analogue signal.

It is therefore necessary to stretch, or squash, the way this digital signal varies with time, without affecting the underlying sampling interval. The output clock rate of an ARC must be the same as the input clock rate (27MHz for D1 format), regardless of any change in the shape or size of the picture content.

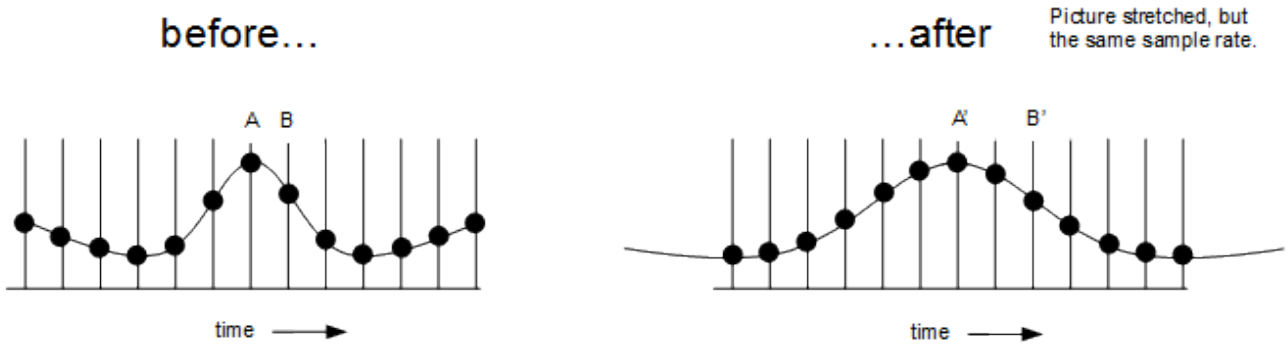

Figure 4: Re-sampling, but playing out the new samples at the same rate gives the impression of stretching (or squashing) the signal.

This process is complicated by the fact that, except for special cases, output samples are required that correspond to locations on the original input sequence where there is no sample present. In other words, it is necessary to estimate what the value of the input would be if it were sampled between existing input samples. This is shown on the diagram above. Two consecutive samples on the input sequence are labelled A and B. The corresponding points on the stretched output are labelled A' and B'. They are not consecutive anymore, with a sample between them that corresponds to a location on the input sequence where no sample existed, i.e. mid-way between A and B. The value of the input at this location is not available and therefore needs to be estimated.

Producing extra samples is generally called *interpolation*, and it can be achieved by averaging the values of the nearest samples on either side of the desired output location. For the example above, the new point half way in time between samples A and B can be formed by taking the average value of A and B, i.e. $\frac{(A+B)}{2}$.

It is possible to estimate the value of the input sequence at different locations, other than exactly mid-way, by using weighted averages instead of the normal [1/2, 1/2] average. For example, a [3/4, 1/4] average will estimate the output value three quarters of the way towards one of the two input samples and, conversely, a [1/4, 3/4] average will estimate the output three quarters of the way towards the other input sample. Luminance and chroma have different sampling rates, data ranges and offsets, and are therefore interpolated separately. For simplicity, figure 5 shows the luminance signal and assumes that sample A represents white and sample B represents black.

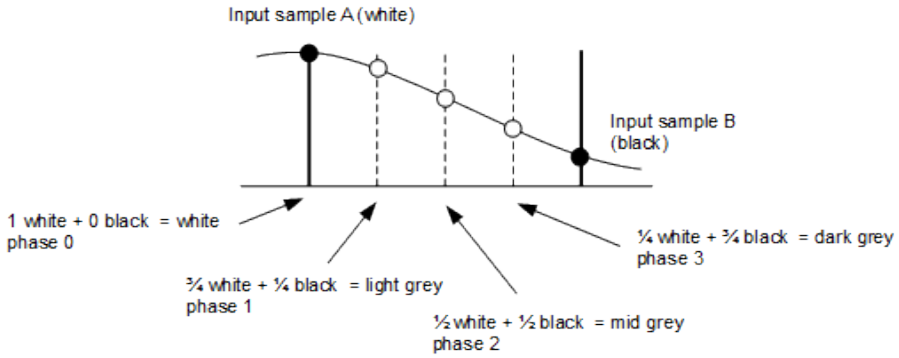

Figure 5: Example of interpolating at several intermediate positions.

These sets of differently weighted averages are usually called *phases*, and (together with the zero phase) comprise a *polyphase filter*. For this example the four phases are:

Phase 0: [1, 0]
Phase 1: [3/4, 1/4]
Phase 2: [1/2, 1/2]
Phase 3: [1/4, 3/4]

In general, the number of phases will be significantly more than four in order to produce smoother, less coarse, aspect ratio conversions[3]. Also, the newly generated points will take contributions from not just the two nearest input samples, but from up to a dozen nearby points - horizontally, vertically and temporally too. The set of input points is usually called the *aperture*. An example of a vertical aperture might be a total of 12 samples from 4 consecutive lines on each of 3 consecutive fields. An example of a horizontal aperture might simply be 10 samples from the same line (for luminance) and 5 (for chroma) - reflecting the different sample rates of luminance and chroma. For apertures larger than two it is not possible to use simple averages as in the example above. The weights that are to be applied to each input sample are collectively called the *filter coefficients*. The exact choice of filter coefficients is the art of making a quality aspect ratio converter.

[3]As an example, the modular IQDARCS has 64 horizontal phases and 32 vertical ones. They are usually powers of two, as a consequence of using an FFT (Fast Fourier Transform) in the filter generation process.

Having considered how the value of the input sequence is estimated at a particular location we can now view the overall process of changing the size of an image. To make an image bigger, often called *up conversion*, new points are estimated at a finer sampling resolution than the input sequence, i.e. more closely spaced. These new points are then released at the correct sample rate (i.e. the same as the input sampling rate) which has the visual effect of stretching the picture out. This is shown in figure 6.

To make an image smaller, often called *down conversion*, new points are estimated at a coarser sampling resolution than the input sequence, i.e. further apart. When these new points are released at the correct sample rate (i.e. the same as the input sampling rate) the visual effect is one of squashing the picture.

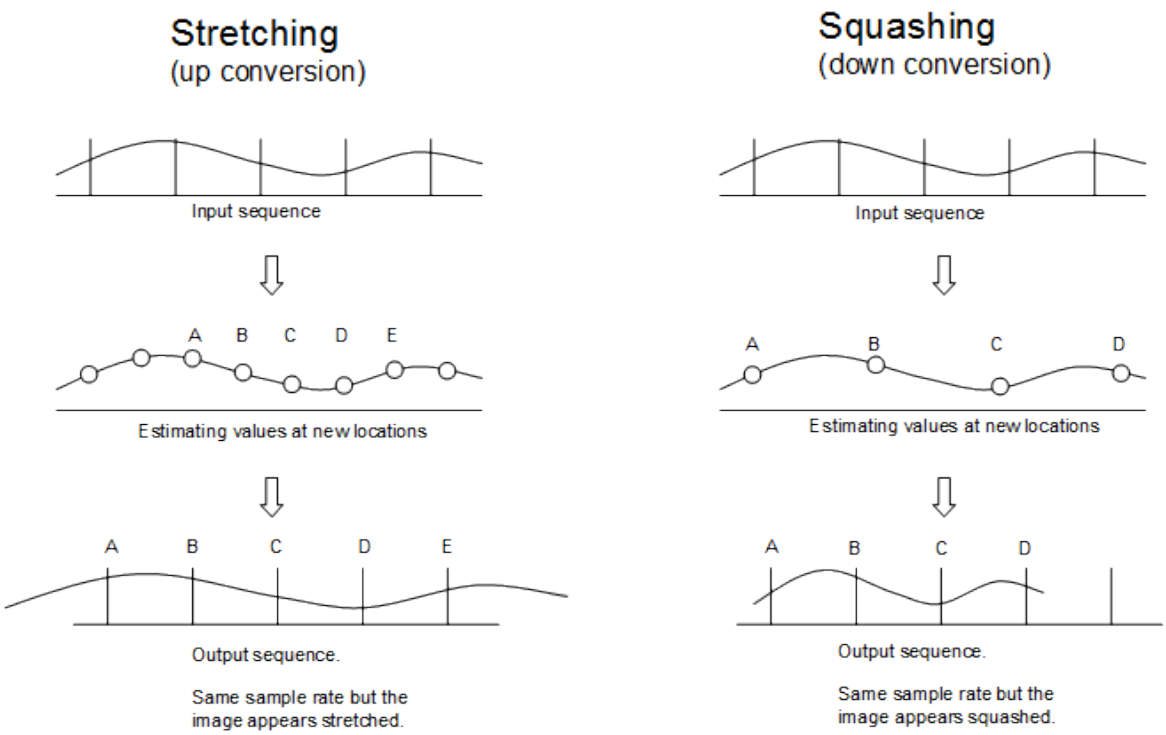

Figure 6: Representation of up conversion and down conversion.

With more samples (up conversion), the full image would not fit the screen, so cropping is required. This can be entirely at the beginning of the sequence, or entirely at the end, or any mixture of the two.

With less samples (down conversion) the full image is now smaller than the screen, and so extra padding is required, usually black bars. For purely horizontal interpolation the bars appear at either side of the screen (called *pillar box* format), and for purely vertical interpolation at the top and/or bottom of the screen screen (called *letter box* format). Many down conversions involve both vertical and horizontal interpolation and so will produce bars all round the image, giving the impression of a box within a box. Some aspect ratio conversions require stretching in one direction and squashing in the other, and thus the final picture needs both bars and cropping.

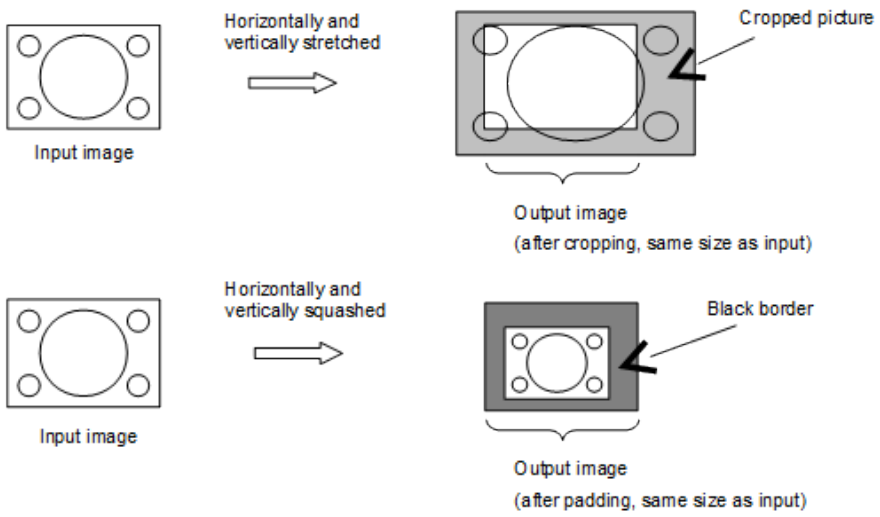

Figure 7: Examples of up and down conversion of TV pictures.

1.4 The phase accumulator

This section provides a more detailed and technical explanation of the process of altering the size of a video picture, including the workings of the *phase accumulator*.

The example that follows considers a 4-phase filter operating on a picture to make it one third wider, i.e. to be stretched horizontally by 4/3. For simplicity, only luminance is considered. (Although down conversion is not explicitly mentioned the process is approximately the same.)

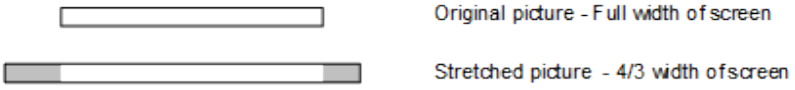

Figure 8: Representation of a TV line being stretched so that it would be wider than the screen width.

As shown in figure 8 the stretched picture is too wide to be displayed in its entirety on the screen, only 3/4 of it will fit (because $3/4 \times 4/3 = 1$). The output sample rate must remain the same as the input sample rate, and thus each displayed picture line must have a total of 720 luma samples, regardless of what the image content is. In our case, therefore, 3/4 of the picture line will need to have 720 samples instead of the 540 that it had originally (3/4 of 720 = 540). This can be achieved by reducing the spacing between successive samples to 3/4 of its original value.

The first desired output sample (labelled A in the diagram below) is simply the input sample A, and is produced by using the zero phase on the input pair$A$ and B. As explained previously, the third phase of a 4-phase filter will estimate the value of the input sequence 3/4 of the way towards the second of the two input samples, and therefore this phase is used on the same input pair A and

B to produce the second output sample, b. The third output sample is an estimate of the input half way between two samples and thus phase 2 is required but this time operates on the next input pair, B and C. The diagram below shows that the next required phase is 1, operating on the inputs C and D, followed by phase 0 operating on D and E, and then by phase 3 still operating on D and E. The phase sequence is thus 0,3,2,1,0... with the input pairs sometimes being used for only one sample and sometimes for two.

Making the sampling interval 3/4 of the original value is equivalent to advancing the phase for each output sample by 3 out of a total of 4 possible phases. This phase counting is performed by the phase accumulator, in this example producing the sequence 0,3,6,9,12... If the phases are considered to group naturally into batches of 4 that correspond to a pair of input samples, then phase 6 corresponds to the 2^{nd} phase of the next pair of inputs, and phase 9 to the 1^{st} phase for the next pair after that. Thus with a 3 bit phase accumulator, advancing by 3 each sample, the bottom 2 bits (counting from 0 to 3) specify the required phase (i.e. produce the sequence 0,3,2,1,0...). When the top bit toggles it signifies that the next pair of input samples is required. For larger picture stretches the phase accumulator increments slowly and thus a particular pair of input samples may be used for many samples until the phase accumulator finally wraps round (the MSB toggles) and the next input pair is brought on. This stopping and starting of the input sequence (output sequence for down conversion) explains why a store is needed. The process is depicted below in figure 9 and table 1.

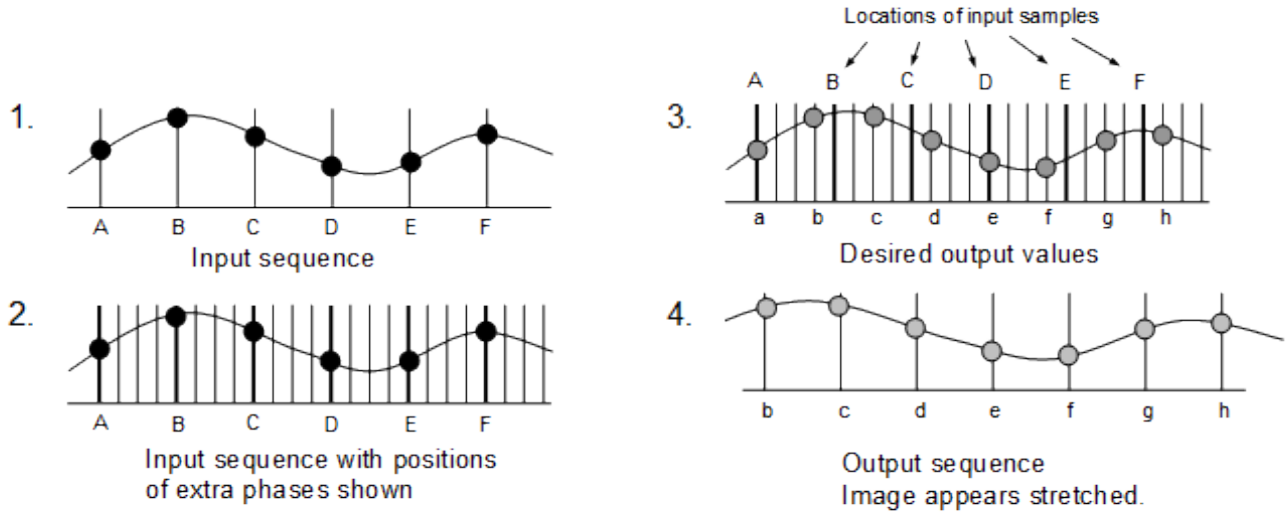

Figure 9: Representation of how the phase accumulator is used to stretch a signal.

This introduction has used a simple example with a small filter aperture for which the coefficients were chosen by considering the required contributions from neighbouring samples to produce weighted averages. This "weighted average" approach is good for an introduction, but does not provide clues about how to lengthen the aperture (indeed, why it would be desirable to do so), nor does it describe the full behaviour of the polyphase filter adequately. To do so requires a view of the filter in the frequency domain.

new sample	phase	input samples
a	0	AB
b	3	AB (note: inputs held for 2 cycles)
c	2	BC
d	1	CD
e	0	DE
f	3	DE (note: inputs held for 2 cycles)
g	2	EF
h	1	FG

Table 1: Relationship of input to output samples for the example of a 4/3 stretch.

2 Polyphase filters

This section shows how to derive the individual phases of a polyphase filter. The aim is to be able to specify the coefficients of each individual phase of a polyphase filter. The approach here is to generate a data sequence that represents the original data sequence sampled at a higher frequency. This super-sampled sequence is made up of all possible extra samples, so, for example, a four-phase filter will give rise to a super-sampled sequence that has four times as many samples as the original sequence. The super sampling is achieved by zero padding followed by low pass filtering. The low pass filtering can be implemented in the time domain as a convolution, from which it is possible to extract the coefficients that correspond to each of the individual phases. Following this comes more frequency analysis of both the magnitude and phase responses of polyphase filters.

2.1 Super sampling

In this section (and most of the following ones) we continue with the four-phase example of the introduction. If the original samples in a sequence are spaced Dt seconds apart so that the sampling frequency is $f_s = 1/Dt$, then the input spectrum folds (has a line of reflection) at $f_s/2$ and repeats at integer multiples of f_s.

Figure 10: A sampled signal has a spectrum that repeats at integer multiples of the sampling frequency.

The original data sequence was sampled at f_s and thus had spectral components up to $f_s/2$. Adding extra samples to simulate a higher sampling frequency should not add any extra spectral

components, although the new spectrum will now repeat at the super sample rate, denoted by f_{ss}, and will fold at half this frequency. Once the super sampled sequence has been generated it is a simple enough process to pick off those samples required. The desired signal and its spectrum are shown below.

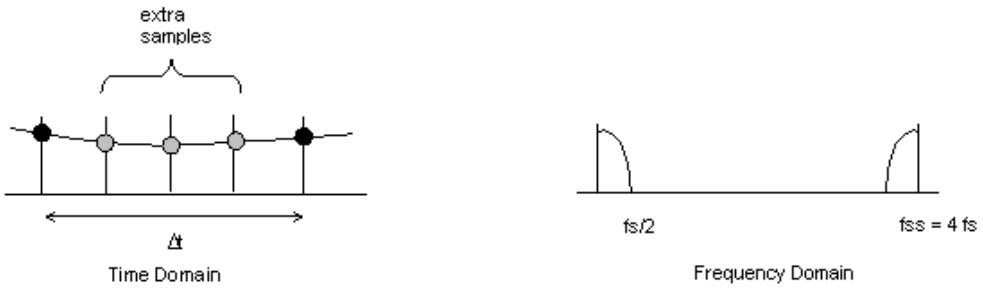

Figure 11: Interpolation causes the sampling rate to increase, thus affecting the aliases in the frequency domain.

Creating a super-sampled version of the original input data can be thought of as a two-stage process, zero-padding between the samples, followed by low pass filtering. Zero padding creates a data sequence with a higher sample rate, the super sampled frequency, by interpolating extra zeros between the original samples. The spectrum of this signal now folds at $f_{ss}/2$ and repeats at f_{ss}. However, the spectrum of the original data repeated at f_s, and therefore also at every integer multiple of f_s. Since the super sample frequency is an integer multiple of the original sample frequency, the original spectrum already repeated at f_{ss}. Similarly, the original spectrum folded at $f_s/2$ and therefore at every integer multiple of $f_s/2$ including $f_{ss}/2$. Thus zero padding does not actually change the spectrum of the original signal at all, which is as expected since no information is being added, but it is useful to re-plot the spectrum up to the new sampling frequency, f_{ss}, as shown in figure 12.

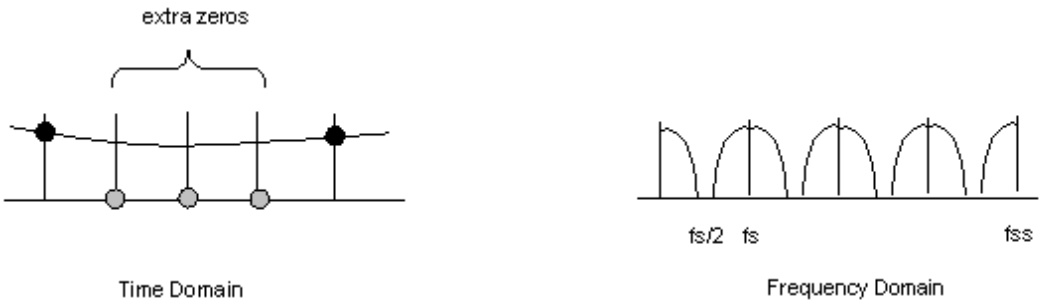

Figure 12: Zero-padding of a signal causes spectral aliasing.

This is not what is desired. To achieve this the signal must be low pass filtered to remove the high frequency components representing the data's sudden transitions to and from zero. The ideal filter for this example has a flat pass band to $f_s/2$ and is then zero until the folding frequency ($f_{ss}/2$), as shown below in figure 13.

Note that this is not always the ideal filter. If the final sample rate (not the super sample rate) is being reduced, rather than increased, then the new signal's spectrum will fold lower than did the original one. This means that spectral components that previously caused no trouble will cause

Figure 13: The ideal filter response for interpolation.

aliasing after the re-sampling process, and thus these components should be filtered out. The pass band may therefore be less than $f_s/2$, depending on the exact application. Also note that whilst the pass band should be shortened for sample rate reduction (usually called down conversion), the converse is not true for an increase in sample rate (up conversion). Firstly, it is impossible to increase the extent of the pass band above $f_s/2$ without including aliasing due to the original sampling process. Secondly, any spectral component below half the original sample frequency will always be below half the new one, so will never cause an aliasing problem. In fact, up conversion does suffer from the associated problem of producing signals lacking high frequency components.

Re-sampling at exactly the same frequency (neither up conversion nor down conversion) can be used to shift the data sequence by an amount equal to less than a sampling interval. This can be achieved by always using the same phase for each output sample, and the ideal super sampled filter response would have a flat pass band up to $f_s/2$, as shown in the diagram above.

2.2 Generating individual phases

This section starts with an ideal low pass filter and converts it to an impulse response function in the time domain. It is shown how the individual filter phases are constructed from the super sampled impulse response function, and used to produce arbitrary points on the super sampled version of the original input sequence.

In this example the ideal super sampled filter frequency response has a flat pass band up to $f_s/2$, as shown in figure 14.

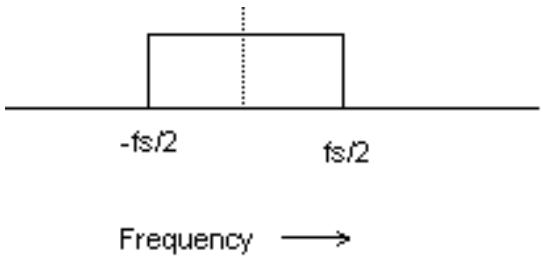

Figure 14: The ideal filter response for interpolation, plotted symmetrically about the origin. Since the spectrum of a sampled signal is periodic (in this case with a requency of f_{ss}), this figure is equivalent to figure 13.

The relationship between rectangular and sinc functions with regard to the Fourier transform is shown in figure 15.

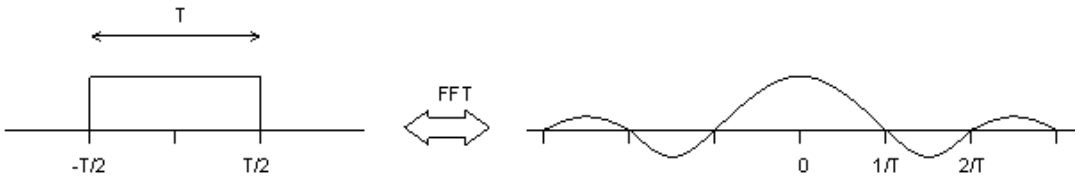

Figure 15: A rectangular pulse and a sinc function are mapped to/from each other by the Fourier Transform.

Thus, a rectangular function in the frequency domain corresponds to a sinc function in the time domain with the first zero crossing at $t = 1/f_s$. This is the continuous version of the impulse response function of the super sampled filter. (As the Fourier transform of a continuous and non-periodic signal, it is itself non-periodic and continuous.) This is shown in figure 16.

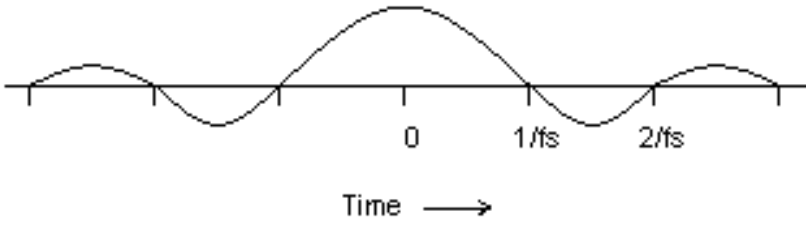

Figure 16: Time domain representation of an ideal (i.e. rectangular frequency response) interpolation filter.

The actual impulse response function has to be applied to the input data as a digital filter, so it cannot be a continuous function. Continuing the previous example in which the super sampled frequency, f_{ss}, is four times the original sample frequency, f_s, the impulse response function now becomes discrete with a sample interval of $1/f_{ss}$. This has the effect of making the spectrum periodic, repeating at integer multiples of f_{ss}.

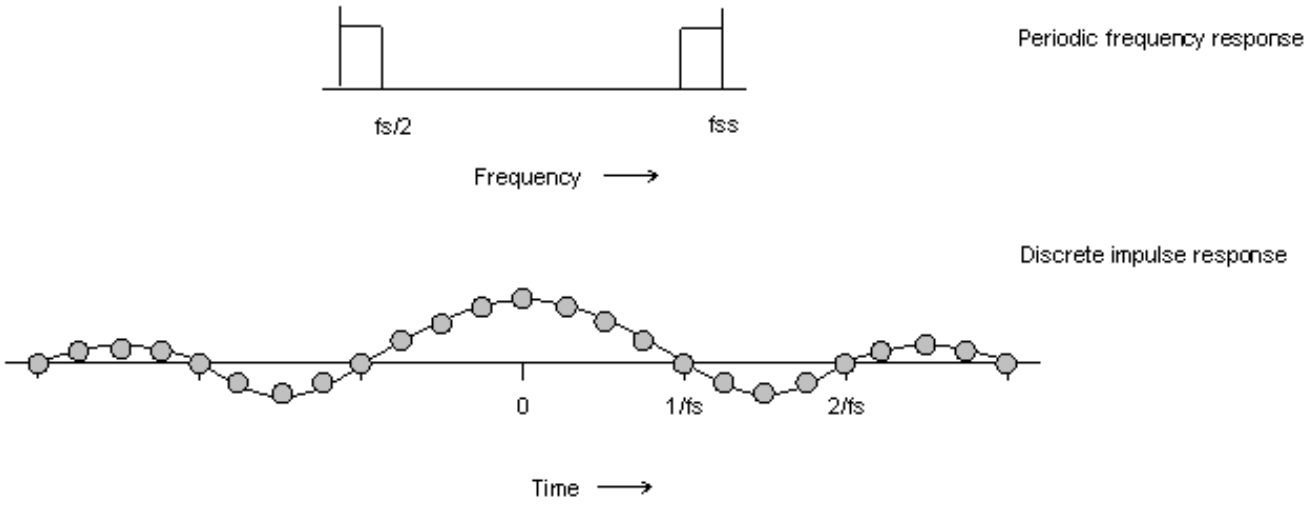

Figure 17: The actual time domain filter must be sampled when applied to digital data.

The application of the filter in the time domain requires that the discrete super sampled impulse response function must be convolved with the zero padded input sequence, shown in figure 18.

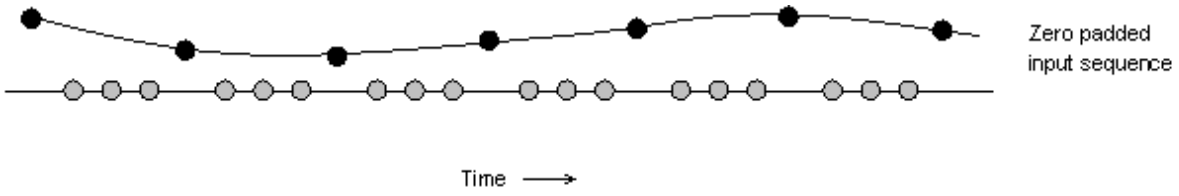

Figure 18: Zero-padded super-sampled input sequence, before filtering.

The input data sequence is denoted by $x(t)$, where $x(0)$, $x(4)$, $x(8)$... $x(4n)$ represent the original samples, i.e. non zero samples. The impulse response function is denoted by $h(t)$ for integer values of t. The instantaneous response of the filter at time t (denoted by $y(t)$) is made up from a component equal to the input at that exact moment multiplied by $h(0)$, i.e. equal to $x(t).h(0)$. The output at time t also comprises a component due to the input one sample before multiplied by the response function one sample later, i.e. $x(t-1).h(1)$. The overall response at time t is made up from the sum of all such components,

$$y(t) = \Sigma x(t-n).h(n)$$

where values of n equal to integer multiples of 4 correspond to original data samples, and all other values of n correspond to interpolated samples. If we consider the output y(0), then because it occurs at exactly the same moment as the input (i.e. no interpolation is required), it should be exactly equal to x(0). Substituting into the equation above gives

$$y(0) = \Sigma x(0-n).h(n)$$

but because x(n) = 0 for all values of n that aren't an integer multiple of 4, the output becomes

$$y(0) = \ldots x(-8)h(8) + x(-4)h(4) + x(0)h(0) + x(4)h(-4) + x(8)h(-8)\ldots$$

The impulse response is a sinc function, which has zero crossings at integer multiples of $1/f_s$. In this example the super sample frequency is four times f_s and so every fourth sample of the super sampled impulse response function is zero, as can be seen from the diagram on the previous page. Thus the output y(0) does indeed equal the input $x(0)$ as required. The output of a genuinely new interpolated sample, for example $y(1)$ is likewise expressed as an infinite sum,

$$y(1) = \Sigma; x(1-n).h(n)$$

Again, the input samples with an index that is not a multiple of 4 are zero valued, and when these are removed from the expression the result is

$$y(1) = \ldots x(-8)h(9) + x(-4)h(5) + x(0)h(1) + x(4)h(-3) + x(8)h(-7)\ldots$$

The remaining two interpolated points, y(2) and y(3), that lie between the original data samples x(0) and x(4) are similarly expressed as

$$y(2) = \ldots x(-8)h(10) + x(-4)h(6) + x(0)h(2) + x(4)h(-2) + x(8)h(-6)\ldots$$
$$y(3) = \ldots x(-8)h(11) + x(-4)h(7) + x(0)h(3) + x(4)h(-1) + x(8)h(-5)\ldots$$

The relationship between the super sample filter response and the individual phases is clearly seen in figure 19, showing how they resemble "comb" subsets of the overall response. Note that since the super sampled sinc function was infinitely long, then so too are all the individual phases.

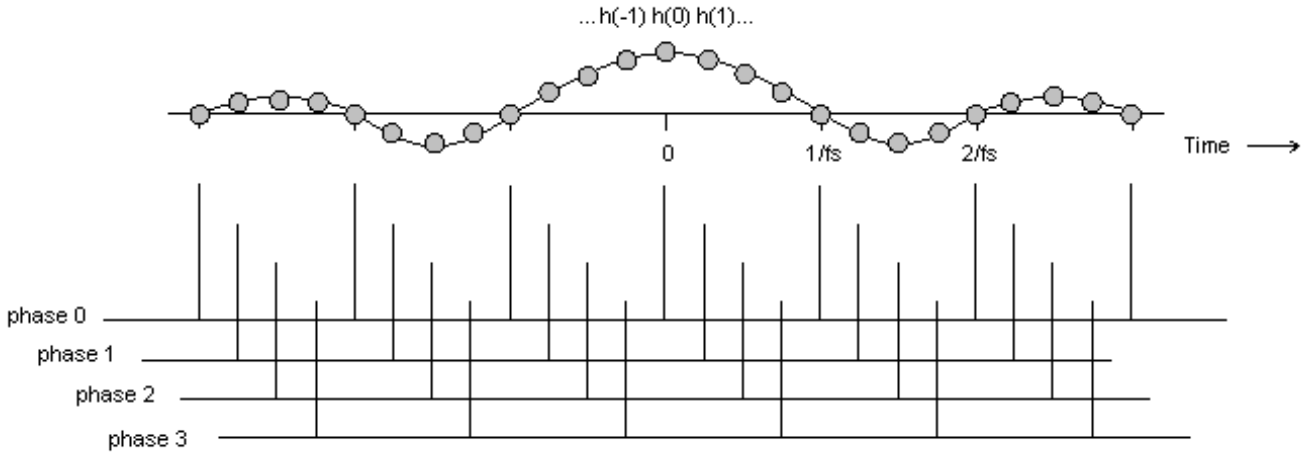

Figure 19: The super-sampled filter is applied to the data in separate phases.

2.3 Aperture length

The previous example considered a super sampled filter impulse response that was a sinc function, i.e. had infinite extent. This meant that all four phases were infinitely long as well (although the zero phase consisted of infinitely long tails of nothing but zeros). Obviously it is not possible to implement such a filter and this section shows the effects of truncating the phases.

Truncation of a signal can be thought of as multiplying it by a rectangular window (value one in the window and value zero outside). This makes it relatively easy to see the effects of truncation since a product in one Fourier domain corresponds to a convolution in the other domain. Thus multiplication by a rectangular function in the time domain corresponds to convolution with a sinc function (the transform of a rectangular function) in the frequency domain. The size of the rectangular window determines in a reciprocal manner where the zero crossings of the sinc function are. For an infinitely long window (equivalent to no truncation) the sinc function is infinitely narrow, existing as a delta function which, when convolved with the desired frequency response function, causes no distortion. As the window is made shorter, the sinc function becomes broader, and so the convolution causes progressively more distortion. Examples of this are shown on the diagram below.

The top plot shows four sinc functions, with no truncation, mild, severe and extreme truncation. The latter case leaves nothing but the central lobe of width $2/f_s$. With 4 samples every $1/f_s$ seconds this portion contains 8 samples and thus each of the 4 individual phases would only have 2 taps. Obviously the first or the four functions corresponds to phases with an infinite number of taps.

The four corresponding frequency responses are shown in the lower plot. The top most is perfect[4]

[4]Perfect, that is, apart from some slight ripple due to the numerical process of plotting spectra, and also a touch

as it should be, corresponding to infinitely long phases. As the length of the aperture is reduced the pass band and stop band develop progressively severe ripple and the cut off becomes increasingly soft. This clearly demonstrates the cost in terms of performance of having short apertures.

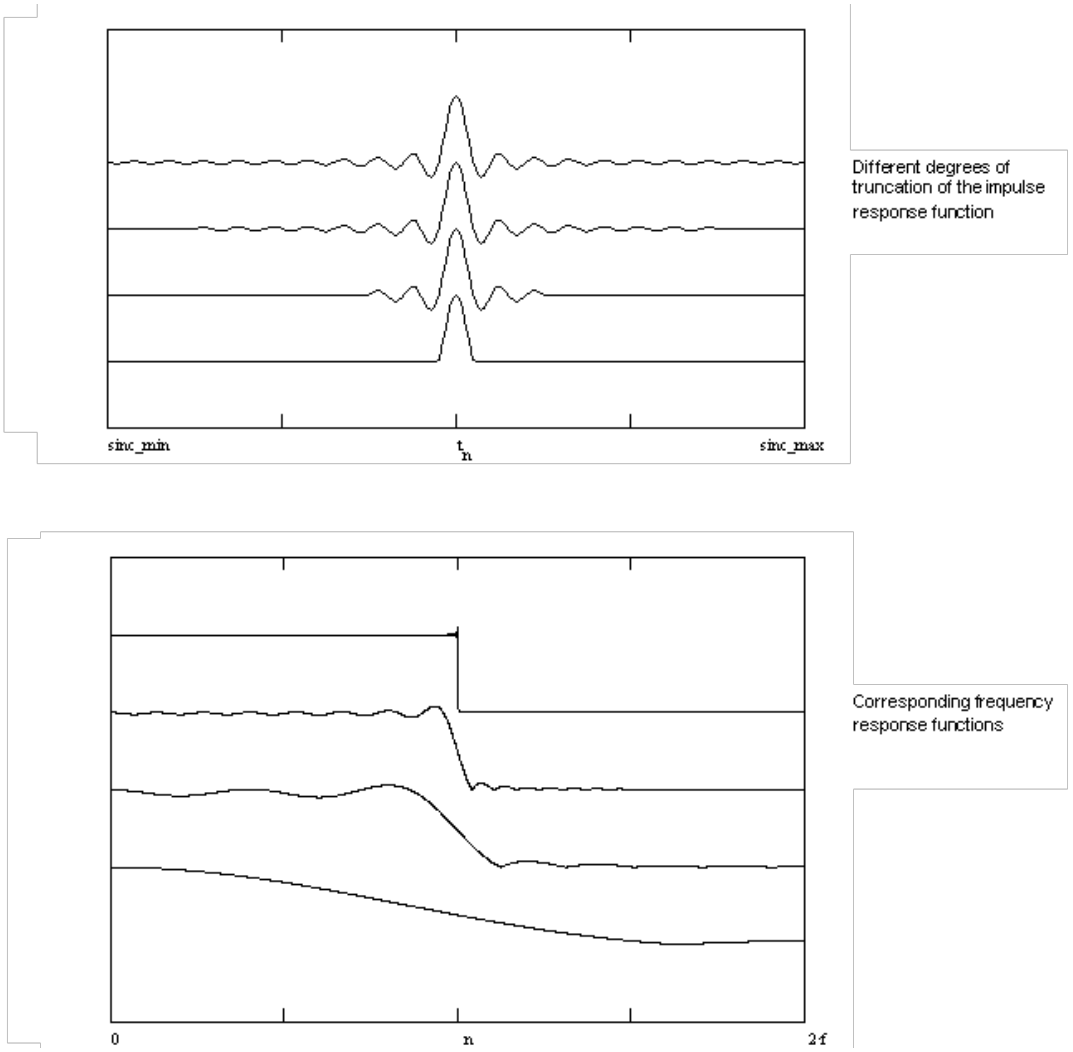

Figure 20: A perfect filter requires an infinitely long aperture. As the aperture is shortened, so the frequency response of the filter deteriorates becoming less sharp.

2.4 More examples

This section looks at another example, that of a simple video de-interlacer, and also returns to the example of the simple 4-phase "weighted average" interpolator of the introductory section.

A common way of de-interlacing two video fields is to use a 2-phase filter to generate extra samples half way between the existing ones. The two phases are [0, 1, 0] which passes the original samples unchanged, and [1/2, 0, 1/2] which creates the extra samples. The overall super sampled filter impulse response is therefore [1/2, 1, 1/2], more conveniently written so as to have unity gain[5], i.e.

of Gibbs phenomenon.

[5]The actual filter [1/2, 1, 1/2] has a gain of two because the number of samples is being doubled yet the amplitude

[1/4, 1/2, 1/4]. Using the same notation as previously, the super sample frequency, f_{ss}, is twice the original sample frequency, f_s, and so the ideal and actual frequency responses are as shown in the following diagram.

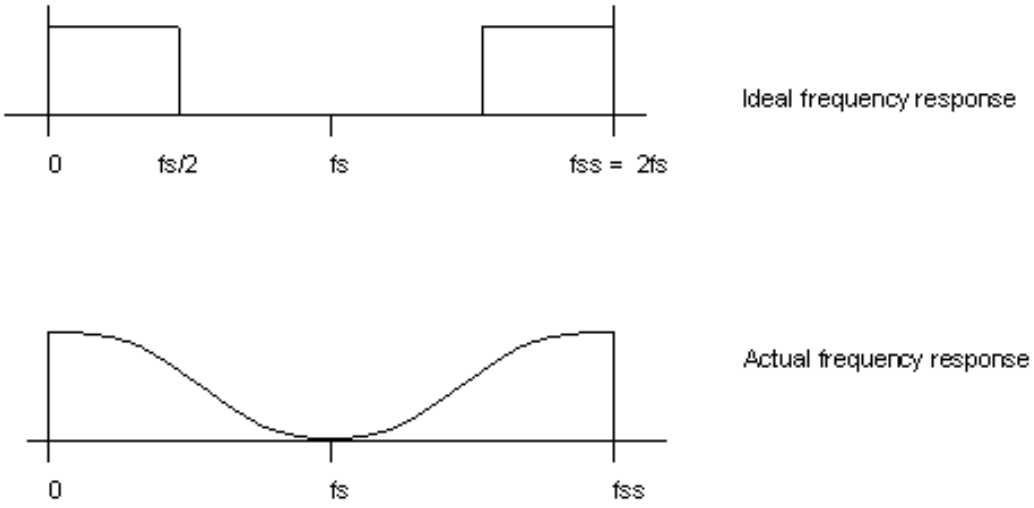

Figure 21: The fequency response of a 2-phase filter is significantly different from the ideal response.

The second example here is the 4-phase "weighted average" filter mentioned in the introduction.

$$\begin{array}{ll} \text{Phase 0:} & [1,\ 0\ ] \\ \text{Phase 1:} & [3/4,\ 1/4] \\ \text{Phase 2:} & [1/2,\ 1/2] \\ \text{Phase 3:} & [1/4,\ 3/4] \end{array}$$

These phases combine to form a super sampled impulse response of

$$[1/4 \quad 1/2 \quad 3/4 \quad 1 \quad 3/4 \quad 1/2 \quad 1/4 \quad 0]$$

which can be Fourier transformed into a sinc squared function. A possibly easier method is to consider the impulse response as the sum of four filters, each of which has a simple transform[6]. The transform of the sum is then the sum of the transforms (the Fourier transform being a linear operator).

$$\begin{array}{rl} [1/4 \quad 1/2 \quad 3/4 \quad 1 \quad 3/4 \quad 1/2 \quad 1/4 \quad 0] = & [1/4 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1/4] \\ + & \quad [1/2 \quad 0 \quad 0 \quad 0 \quad 1/2] \\ + & \quad\quad [3/4 \quad 0 \quad 3/4] \\ + & \quad\quad\quad [1] \end{array}$$

The spectra of these four components are shown in the next diagram, along with their sum (which equals the actual frequency response of the overall filter) and, finally, the ideal frequency response. It is now clear why this example was described in the introduction as a crude filter.

is required to stay the same.

[6] A [1 1] filter has a Fourier transform that is a Cosine with a period equal to twice the sampling frequency.

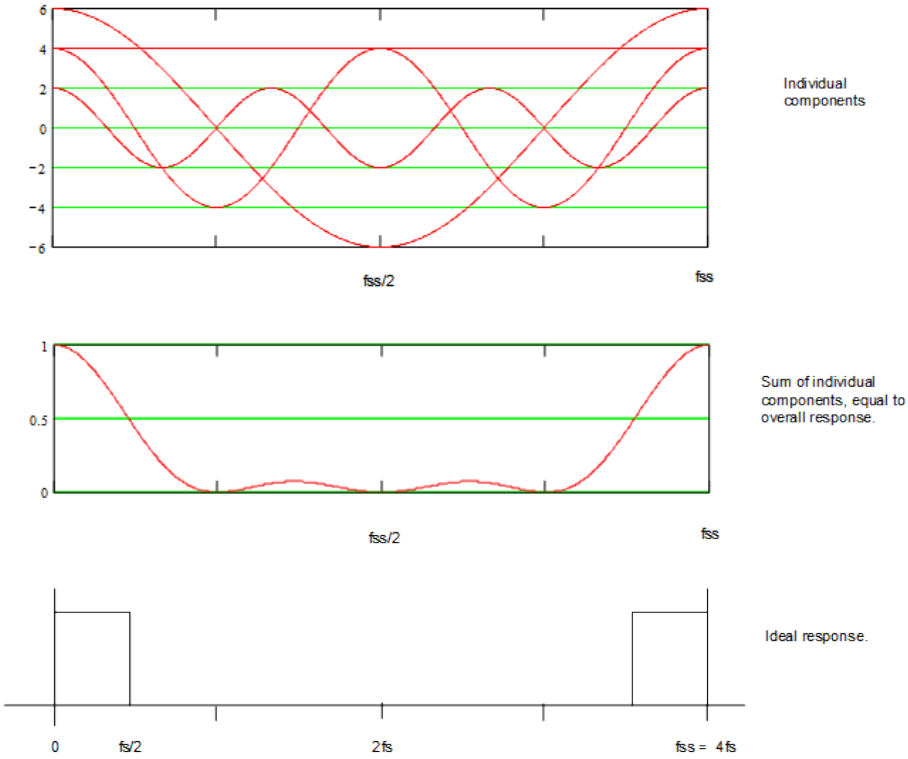

Figure 22: The spectrum of the super-sampled interpolation filter, and those of its separate components (phases) as well as the ideal frequency response.

2.5 Symmetric filters

This section explains the importance of filters having symmetric coefficients and relates this to polyphase filters that clearly have non-symmetric phases.

A real signal that is symmetric about the origin can be synthesised from purely cosine terms. This is because cosine is an even function and sine is an odd function[7]. The Fourier transform is therefore purely real since it contains no sine term, and the phase of the spectrum is thus zero for all frequencies. This means that although the magnitudes of frequency components may be altered, the phases are not. A time delay corresponds to a phase shift and because there are no phase changes it follows that no frequencies are delayed.

In practice, a filter cannot be symmetric about the origin because this requires a response before t=0, so the entire signal is delayed to be entirely after $t = 0$, for example given a delay of τ;. It is relatively simple to show that such a time delay corresponds to a phase shift in the frequency domain. Consider a signal f(t) that has a Fourier transform $F(\omega)$, and a signal $g(t)$ that is a delayed

[7]An even function is defined as one for which $f(-x) = f(x)$ and an odd function one for which $f(-x) = -f(x)$.

copy of $f(t)$.

$$\begin{aligned} g(t) &= f(t-\tau) \\ G(\omega) &= \int g(t)e^{-j\omega t}dt \\ &= \int f(t-\tau)e^{-j\omega t}dt \\ &= \int f(T)e^{-j\omega(T+\tau)}dT \;\; (where T = t-\tau) \\ &= e^{-j\omega\tau}\int f(T)e^{-j\omega T}dT \\ &= e^{-j\omega\tau}F(\omega) \end{aligned}$$

This tells us that the phase shift at a particular frequency is proportional to that frequency, and thus corresponds to a constant time delay[8], which is hardly surprising since the impulse response function was delayed by t to make it start after the origin. The relevance is that, in general, a filter having such a frequency phase response (called linear phase) will have a constant delay for all frequency components, regardless of the magnitude response. Non linear phase responses correspond to filters that delay different frequencies by different amounts causing what is usually deemed unacceptable distortion.

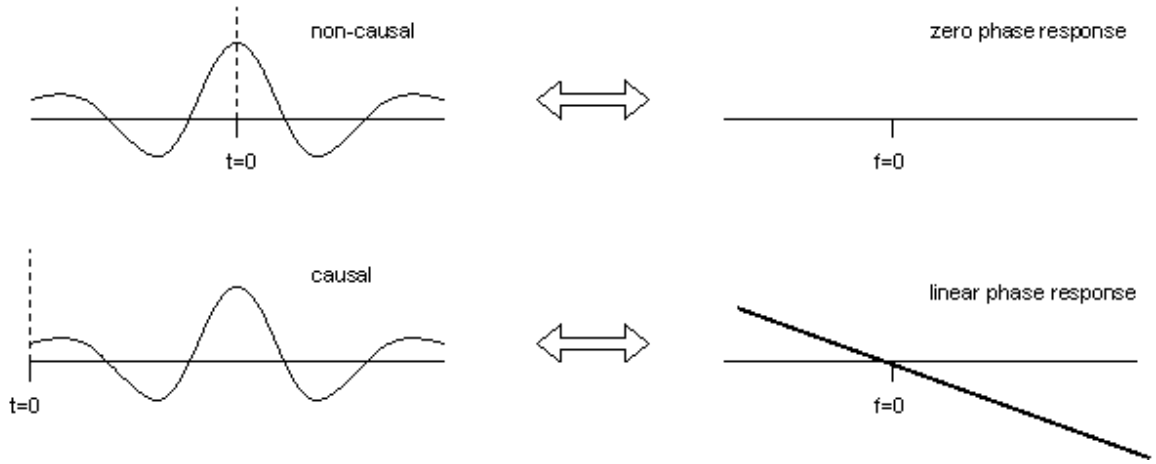

Figure 23: Phase responses of symmetric and asymmetric filters.

Now we consider the symmetry of the individual phases of a polyphase filter, and for simplicity our super sampled impulse response is the sync function detailed above. Clearly the zero phase is symmetric, as the sync function itself is symmetric. Equally clearly, it is certain that any individual phase will not be symmetric[9]. This necessarily implies that the frequency responses do not have a linear phase relationship. The responses for an arbitrary example of an 8 phase filter are plotted below in figure 24. The first of the individual phases is the zero phase that does nothing to the input sequence. This is a unity gain tap surrounded by zeros and it obviously has

[8]The phase of $sin(\omega t)$ advances 2π over one period, i.e in a time interval of 1/f. Thus it advances $2\pi f$ in one second or $2\pi ft(=\omega t)$ in t seconds.

[9]With the exception of peculiar phases, for example those that sample the centre of the main peak and the zero crossings on either side.

unity magnitude response and zero phase response. The next filter phase, the second of eight, has the effect of delaying the input signal by 1/8 of a sample, and so we might expect it to have a linear phase response if we didn't know that this was impossible because it has a non symmetric impulse response. However, the phase function is approximately linear, apart from the kink at the end, and apart from the minor numerical fluctuations due to the small number of samples used to evaluate the spectrum. Subsequent filter phases show a steeper gradient in their phase responses, representing a greater delay of the original data sequence. The kink at the end is correspondingly sharper.

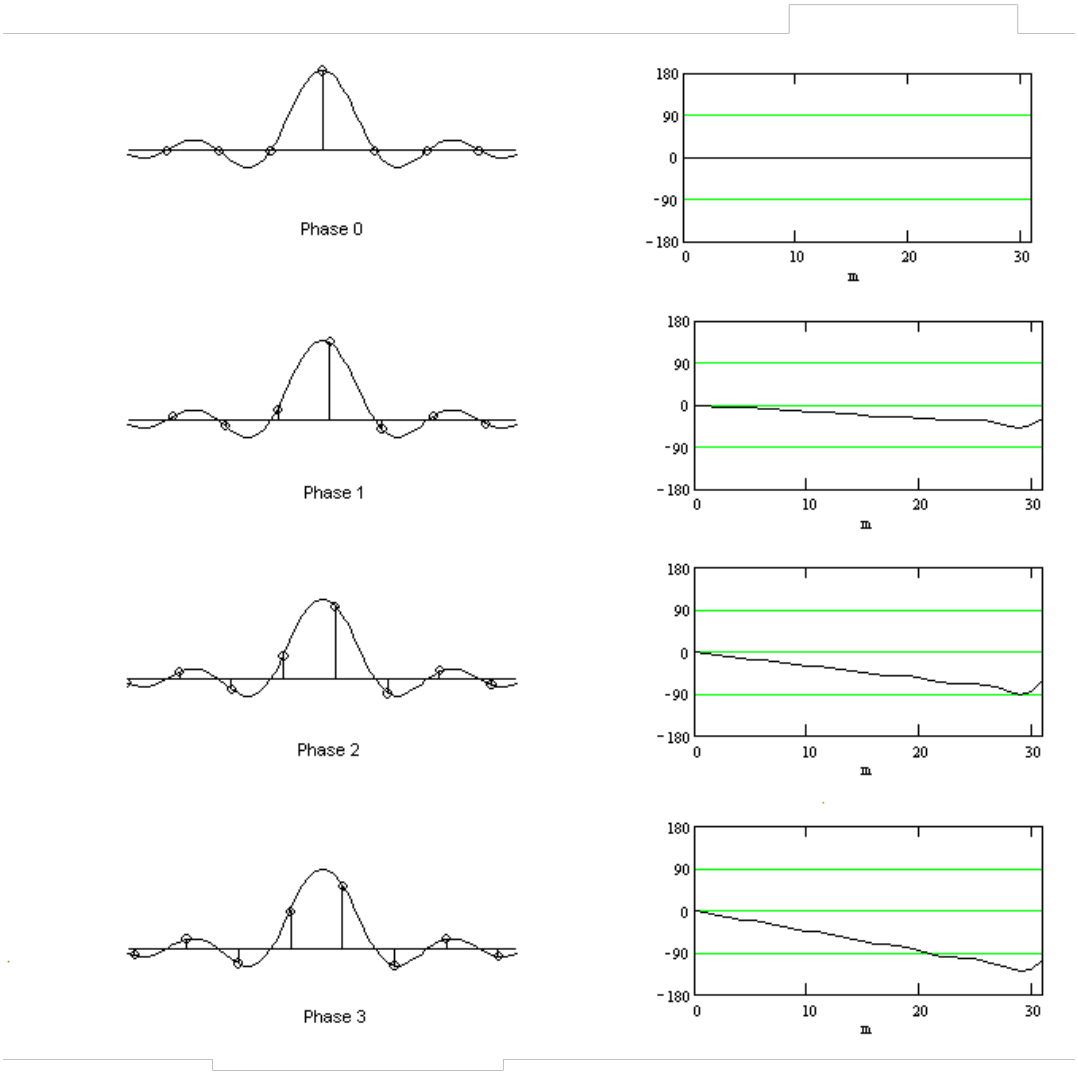

Figure 24: Phase responses for individual phases of a polyphase filter.

The phase response of a real signal is an odd function[10], thus disregarding the numerical crudeness of the above plots and that they are only evaluated up to half the sampling frequency, they would look approximately as shown in figure 25.

[10]This is a corollary of the result that if a signal $h(t)$ is real then its Fourier transform $F(\omega)$ satisfies

$$F(-\omega;) = F*(\omega)$$

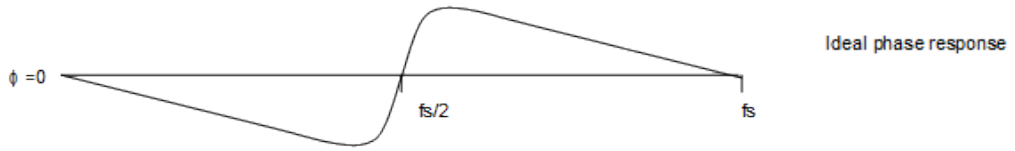

Figure 25: Ideal phase response for a linear-phase filter.

It is not immediately obvious why the phase responses look like this. Why are they not completely linear, or in other words, where does the kink at half sampling frequency come from? It was explained at the beginning of this section how a time shift of the input data corresponds to a phase shift proportional to frequency, i.e. a linear phase relationship. This is certainly the case for the very simple filter that delays the input by one sample[11]. It has unity magnitude response and a phase response, shown in figure 26, that passes through 360° at sampling frequency because the phase response is expressed as $e^{j\omega t}$ where the time delay, t, is equal to $1/f_s$.

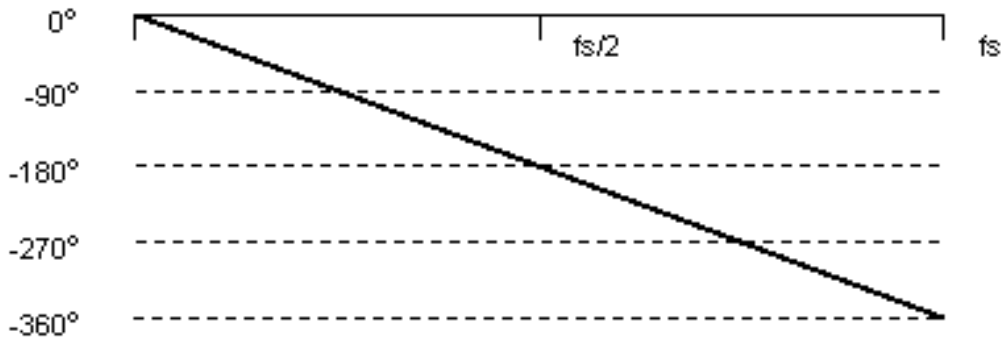

Figure 26: Linear phase response.

This is, though, a special case. The spectrum of any sampled sequence is necessarily periodic, and thus the response at sampling frequency, f_s, must equal that at zero. In addition to being periodic, the phase response is also an odd function, and the two together imply that the phase response is odd about $f_s/2$.

$$\begin{aligned} F(-\omega) &= F(f_s - \omega) - F(\omega) \\ &= F(f_s - \omega) \end{aligned}$$

Therefore the phase response plotted above would perhaps be better plotted as below in figure 27. The two are equivalent because of the way phase wraps every 360°.

Note that the above plot satisfies the two requirements that it is periodic (repeating at sampling frequency) and that it is odd. The discontinuity at half sampling rate is only apparent, not real, since it is a consequence of the choice of plot. It is now clear, though, that for other gradients that

[11] As implemented by a simple flip flop.

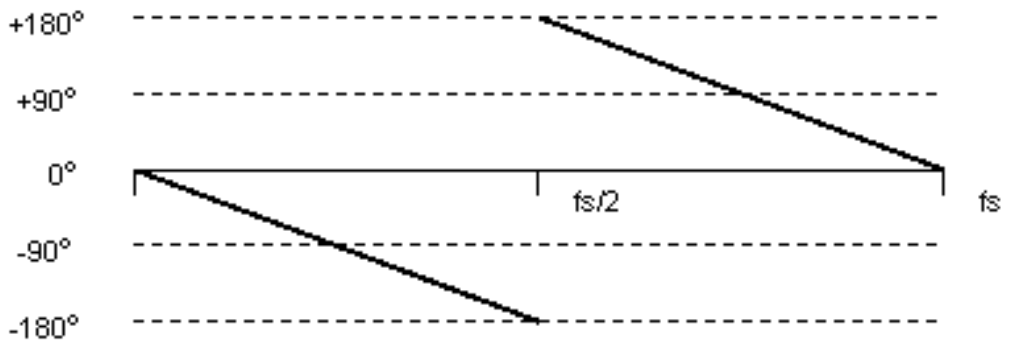

Figure 27: Linear phase response, but plotted with a phase wrap - this is still equivalent to figure 26.

do not have a phase shift of 180° at half sampling frequency there will be an actual discontinuity at this point. Obviously, real systems tend not to have discontinuities, but sharp kinks instead. An example of the ideal and the real are shown below.

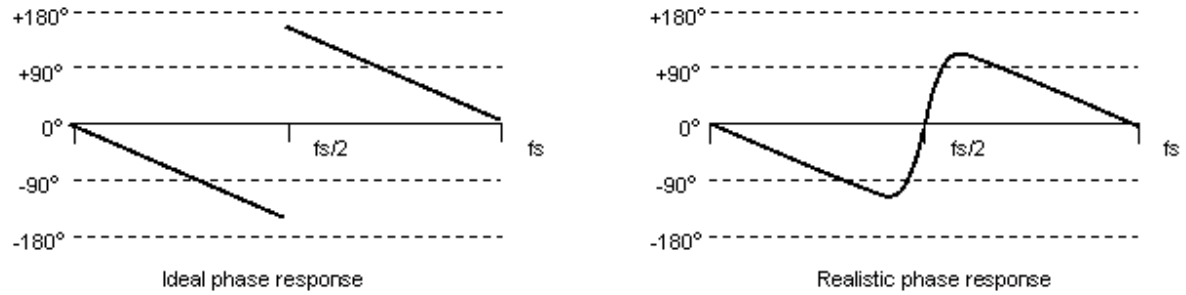

Figure 28: Ideal and actual phase responses.

These phase gradients that do not pass through 180° at half sampling frequency correspond to time delays of the input sequence that are not an integer multiple of sampling frequency. Such delays cannot be implemented by simple chains of flip flops but instead require sub pixel interpolation, i.e. require any of the polyphase filter phases except for the zero phase.

In addition to the phase response changing for the individual filter phases, the magnitude responses also vary. The zero phase filter, [...0, 0, 0, 1, 0, 0, 0,...] has a unity magnitude response for all frequencies. The other phases have magnitudes responses that drop off as they approach half sampling frequency, dropping more severely with each phase until the half way phase is reached (interpolating a point exactly mid-way between two original samples) which has a response of zero at $f_s/2$. This is depicted in figure 29.

In summary, any individual phase of the polyphase filter corresponds to a certain time shift of the input data. Ideally, the frequency response would have unity magnitude and linear phase for all frequencies. In practice this is not possible and there is a region, approaching half sampling rate, at which the magnitude response is not unity and the phase response is not linear. This is explored further in subsequent sections.

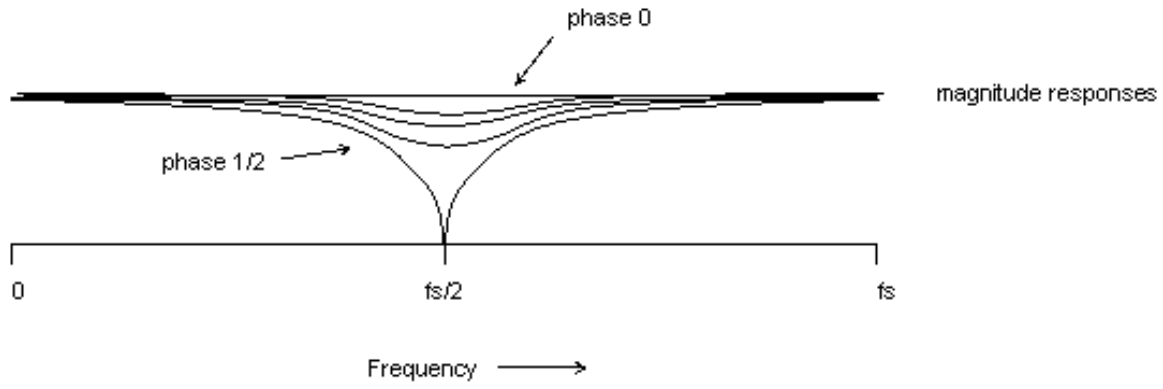

Figure 29: Magnitude responses for different phases of a polyphase filter.

2.6 Symmetric or non-symmetric?

This section attempts to demystify the apparent contradiction that a single individual phase of a polyphase filter can appear to be both symmetric and non symmetric at the same time. The exact nature of the problem is as follows. Again, for simplicity and consistency with previous sections, we shall stick to the example of a sinc function for the super sampled impulse response function. Assuming that the polyphase filter is allowed to have an infinite number of taps allows us to disregard truncation effects for simplicity.

Suppose the aim is to shift an entire sequence by a quarter of a sample, i.e. not to change the sample rate. In order to interpolate a given point a quarter of the way to the next sample, the zero padded super sampled data sequence is convolved with the suitably placed super sampled filter impulse response function, as explained in a previous section. Suitably placed means that it must be centred on the desired output location, as show in the following diagram (for convenience parts of previous diagrams are repeated below). Most of the products are zero, due to the heavily zero padded input. The only non-zero terms correspond to phase 1 of the 4-phase polyphase filter. The non-zero data samples are shown in black, while the zero padded samples are grey. Those points of the impulse response function that line up with non-zero input samples are likewise coloured black. The impulse response function is symmetric and thus has a linear phase response.

In any practical implementation of this sub-sample time shifter, the zero products would not be calculated as they contribute nothing (and also require a high-speed clock). Super sampling is a theoretical construct rather than a matter of implementation. The actual filter would be constructed by multiplying the original data samples by the coefficients of the phase 1 filter (shown in black in figure 30) which are non-symmetric and thus do not have a linear phase relationship. When considered theoretically the filter is clearly symmetric, but in any practical implementation the filter is, equally clearly, non-symmetric. It cannot be both.

If the phase response of the above time shifter is plotted with respect to the super sampling frequency, denoted by f_{ss}, figure 31 is obtained. Note that there are no kinks that correspond to any natural discontinuity.

This shows that at the super sample frequency the phase is being shifted by 360°, or equivalently that the time series is being shifted by one super sample interval, which is as desired. If we tried

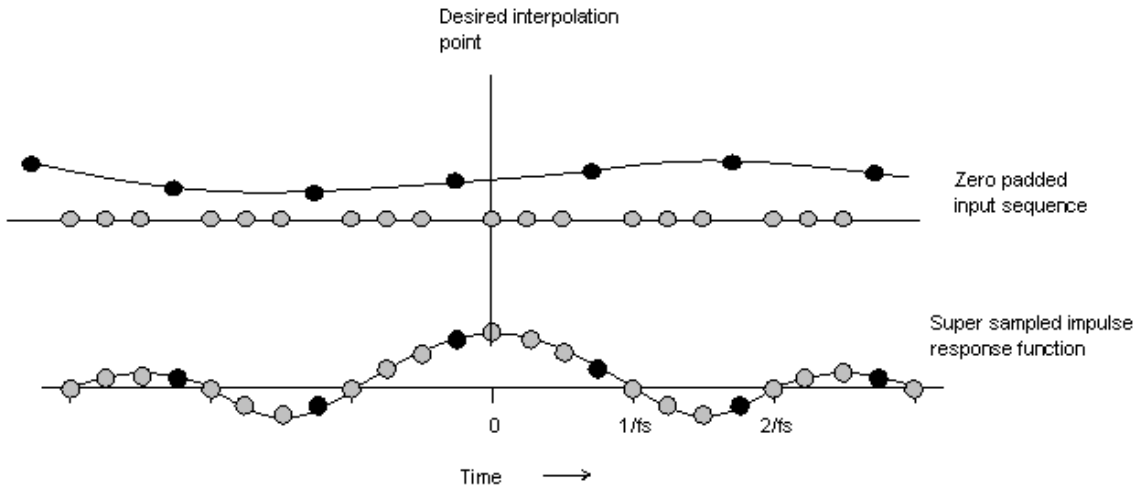

Figure 30: Representation of a time-shifter, showing input samples being multiplied by their respective super-sampled filter coefficients.

Figure 31: Phase response of the example time-shifter.

to plot this with respect to the normal sampling frequency, f_s, the graph would "attempt" to look the same i.e. have the same gradient. At sampling frequency the phase shift is -90^o, equivalent to a time shift of one quarter of the sampling interval, as required. As before, the phase response is an odd function and is also periodic, which is equivalent to the phase being odd centred on half sampling frequency (see the right plot of the diagram above). However, now the sampling frequency is f_s, not $4f_s$, and thus the phase at half the sampling frequency is now -45^o, not -180^o. This produces a genuine discontinuity, or in a real system, a kink as shown in figure 32.

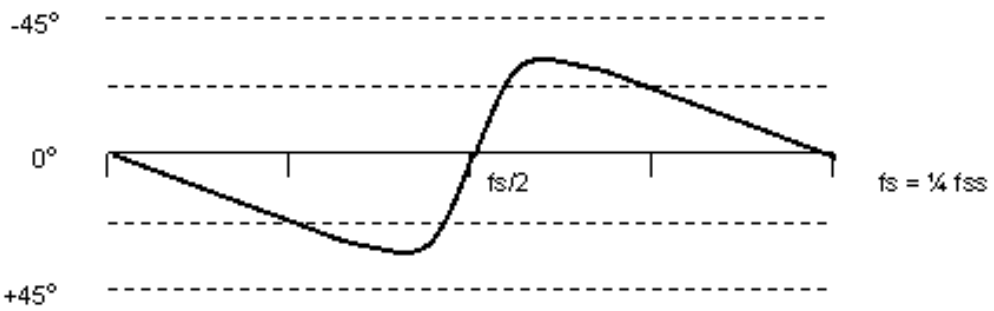

Figure 32: Phase response of a shift of less than 1 sample.

This graph resembles the plots of the phase responses of actual individual filter phases shown in section 3. They are indeed non-linear. They are also nearly linear for all but the middle portion, which is not surprising because they are sub sampled instances of genuine linear responses, and it is the sub sampling that gives them the kink in the middle.

2.7 Magnitude and phase responses

Like section 2.2, this section also derives the individual phases. It does so from a different perspective that is not quite so simple, however, it leads to magnitude and phase responses for the individual filter phases.

It is possible to derive the individual phases of a polyphase filter by a slightly different method that directly shows why they have different phase and magnitude responses. The approach so far has been one of shifting the input sequence, i.e. the sampling grid stays in the same place and the input sequence slides, slightly, to a new position. For example, if a [1/2, 1/2] filter is used for every sample the output sequence will appear to have slid across by half a sampling interval relative to the input sequence. (See figure 33 below.) The emphasis in this section is the other way round; instead of keeping the sampling grid constant and moving the data, the input data sequence stays fixed and the sampling grid moves (i.e. the data is re-sampled on a slightly different grid). The two approaches are equivalent because they produce the same output data sequence.

Our initial example will be, as before, a 4-phase filter. The first step prior to re-sampling is to super-sample and this is achieved (in exactly the same way as previously) by zero-padding followed by low pass filtering. This leaves the original spectrum repeated at integer multiples of the super-sampling frequency, i.e. at $4f_s$. The four filter phases correspond to re-sampling this super-sampled input sequence with an offset of 0, 1/4, 1/2 and 3/4 of the original sampling interval or, in terms of a phase shift, at 0^o, 90^o, 180^o and 270^o as shown in figure 34.

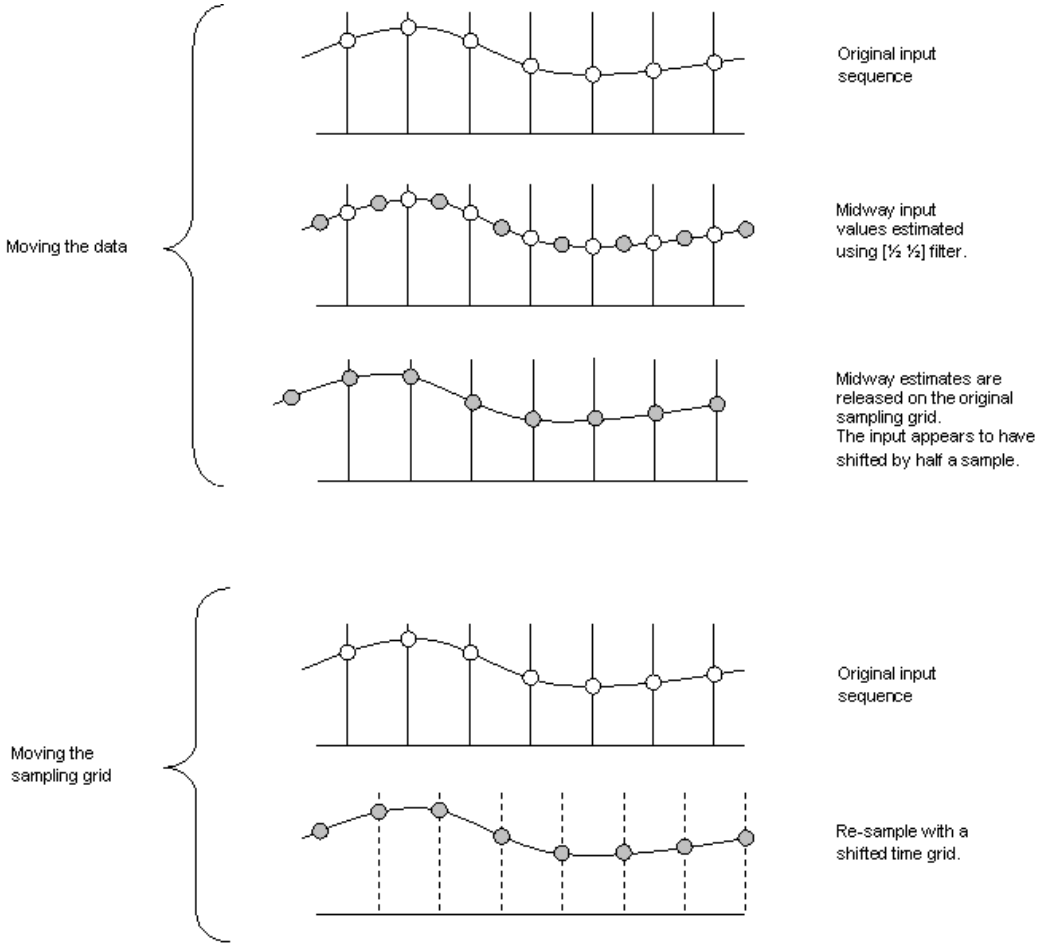

Figure 33: A shift of half a sample can be represented in two ways, either moving the input data against a fixed sampling grid (top plots), or moving the grid against the fixed input sequence (bottom plots).

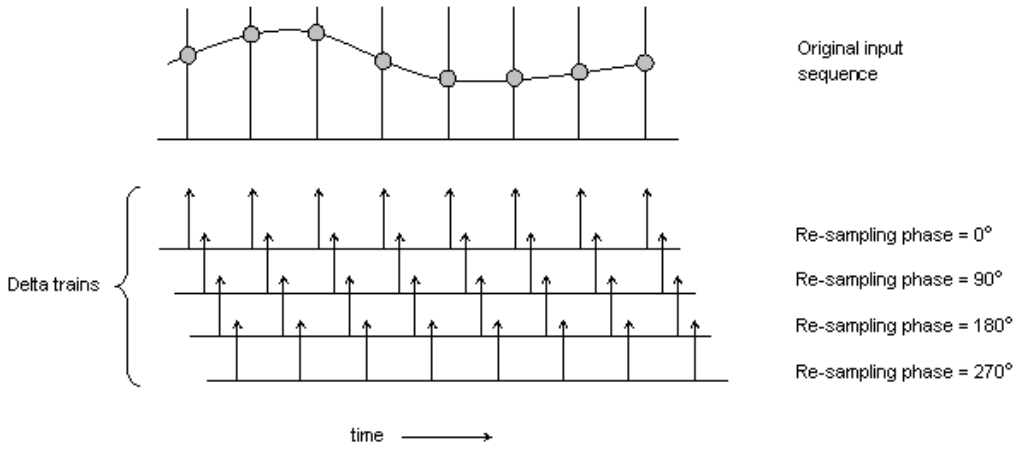

Figure 34: Representation of the 4 individual phases of our super-sampled filter.

The output sequence is formed by multiplying the input data sequence by the sampling sequence of delta functions. This is equivalent to convolving the spectrum of the input with the spectrum of the appropriate delta train, which is itself another delta train[12]. As an example, we consider the first phase, i.e. the 1/4 sample shift that corresponds to a 90° shift. The spectrum of the super-sampled input sequence is shown below in figure 35, along with the spectrum of a delta train that has experienced a 90° phase shift for every interval of f_s along the frequency axis.

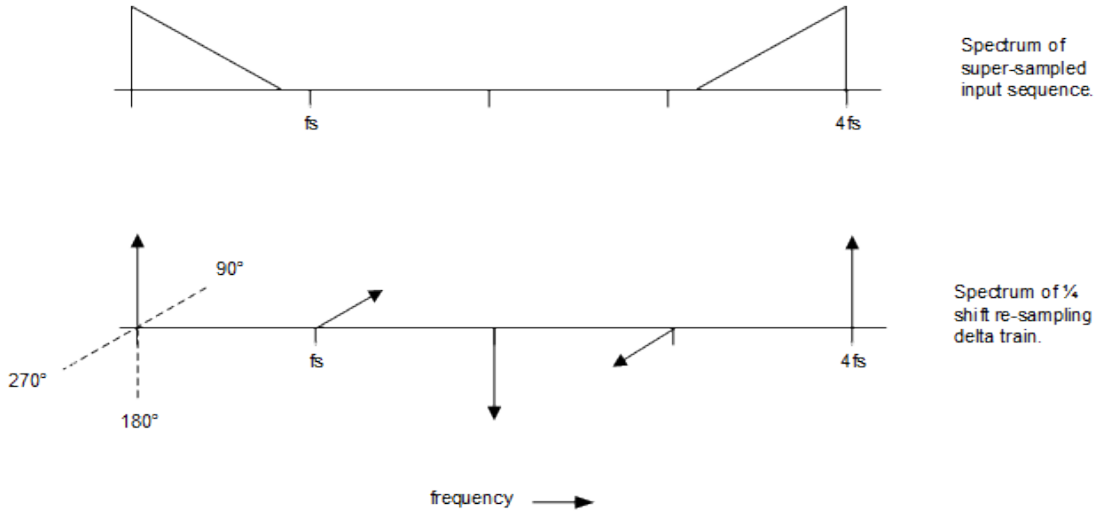

Figure 35: Spectra of super-sampled interpolation filter and a phase shifted delta train.

When these two functions are convolved together the resulting spectrum resembles the tail of a kite (figure 36), the base-band energy of the original spectrum being repeated with 90° twists along the frequency axis. For clarity, figure 36 has shown the data spectrum diminishing to zero before half

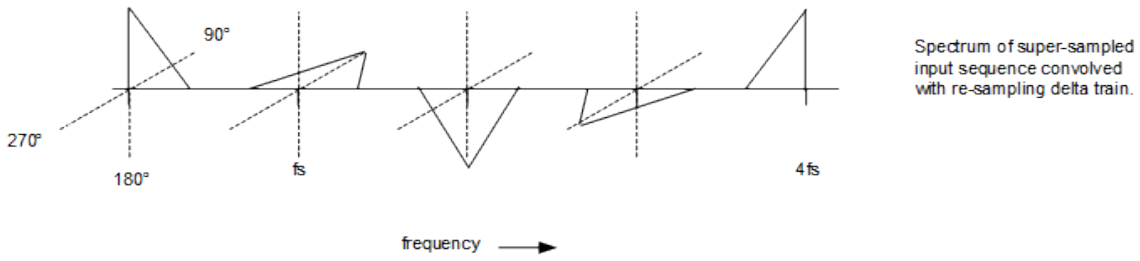

Figure 36: Spectra of super-sampled filter shifted by 1, 2 and 3 super samples.

the sampling frequency. In practice this would not be the case, as the signal and the alias would be progressively overlapped as the frequency approached $f_s/2$.

[12]The Fourier transform of a delta train is an infinite sum of complex exponentials,

$$\ldots + e^{-j\omega 2T} + e^{-j\omega T} + e^{j\omega 0} + e^{j\omega T} + e^{j\omega 2T} + \ldots$$

which can be combined to form an infinite sum of cosines,

$$1 + 2cos(\omega T) + 2cos(2\omega T) + 2cos(3\omega T)\ldots$$

Over the period $2\pi/T$ these cancel everywhere except at the origin where they reinforce each other to form a single delta function. The spectrum is periodic (period = $2\pi/T$) and is thus a train of delta functions.

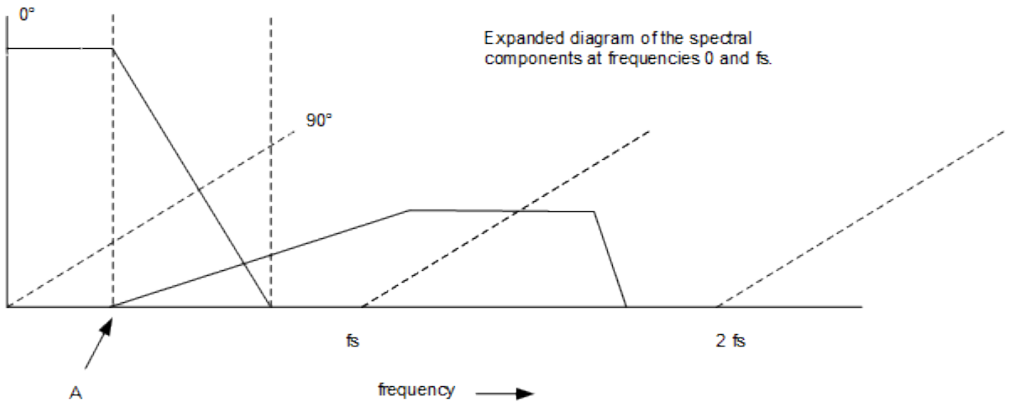

Figure 37: Spectra of super-sampled filter and the same shifted by 1 sample, but with the presence of signal alias, i.e. data that overlaps with the next shifted spectrum.

In the pass band, i.e. at frequencies up to A (marked on figure 37), there are no aliases present and the spectrum is unchanged. At frequencies above A there is a contribution from the alias, but at an angle of 90° to the signal. On a cross section perpendicular to the frequency axis, and cutting the frequency axis at f_0, the signal and alias components will appear as two vectors at right angles to each other. The length of their resultant represents the amplitude of the filter's frequency response at the frequency f_0. This is depicted in figure 38 below. For complementary filters the signal and the alias component will always add up to unity[13], and by considering similar triangles it is seen that the locus of resultants is a straight line. It is also apparent that the minimum magnitude of the resultant occurs at half the sampling frequency for which the signal and alias components are equal in strength.

Interpreting the diagrams (i.e. figure 38) can be confusing. Remember that they are cross sections through the frequency axis, showing in-phase and out-of-phase components of the signal and its aliases. At DC. there is no alias component and the magnitude of the signal is unity (for simplicity). It stays this way until the end of the pass band (A) at which point the alias component becomes non-zero. During the transition band the alias progressively reduces the magnitude of the frequency response function and produces a phase shift. This is depicted in figure 39.

[13]This is the definition of a complementary filter.

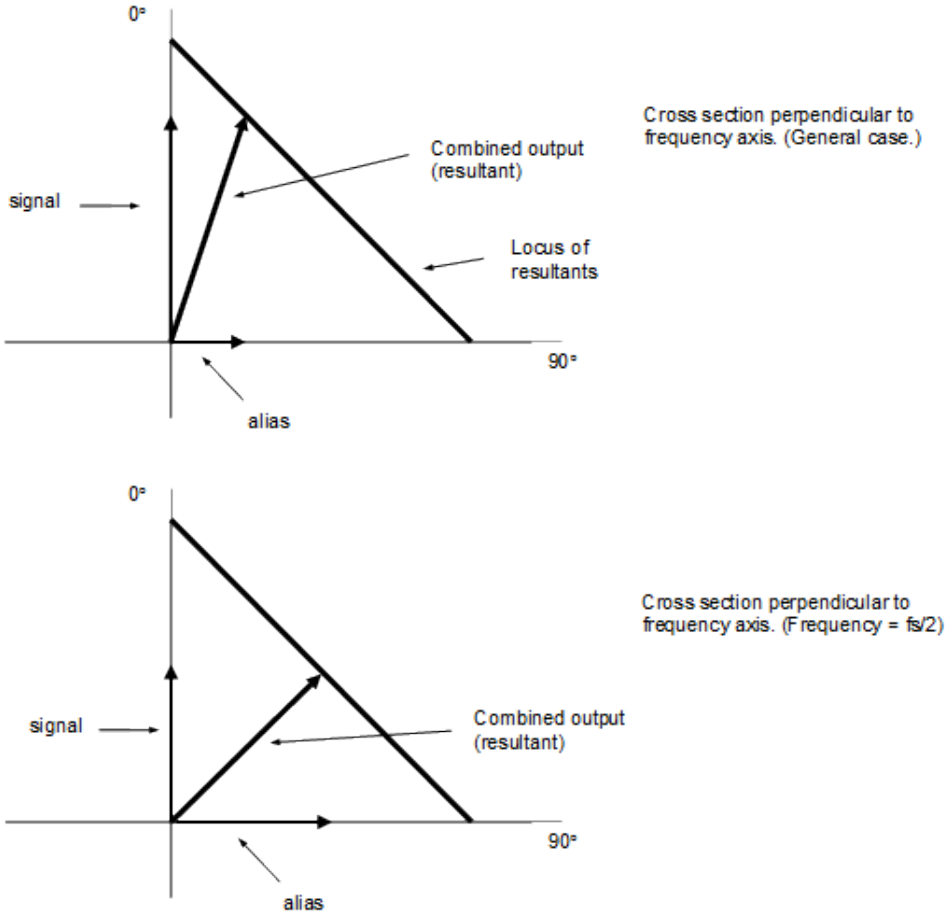

Figure 38: Cross sections through the frequency axis.

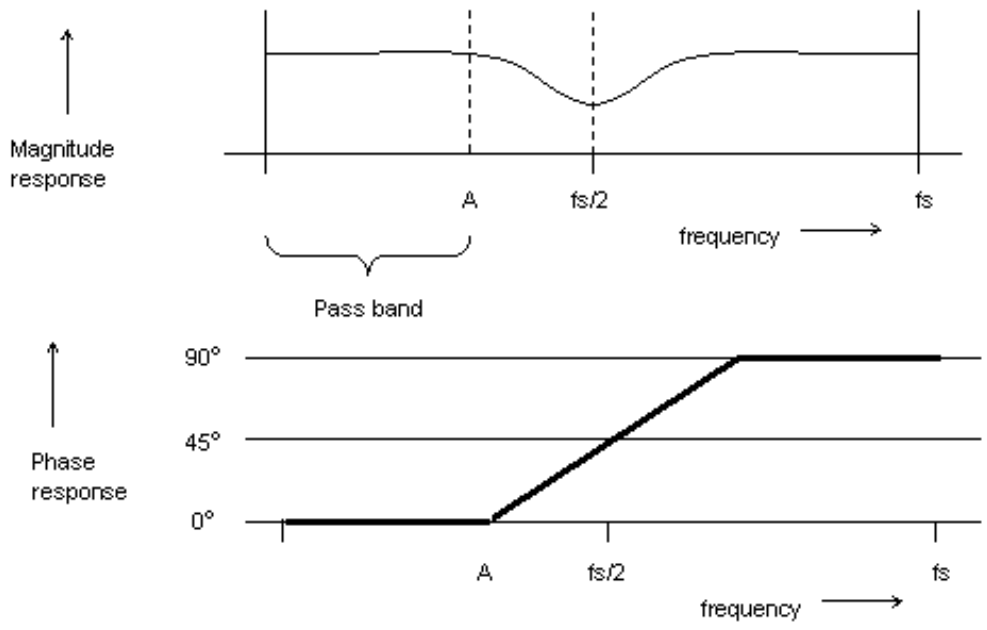

Figure 39: The transition band is characterised by a phase shift and reduction in the magnitude response.

The second phase of the 4-phase filter is formed by re-sampling with a delta train representing a half sample shift, or a phase shift of 180°. This, and the result of convolution with the super-sampled input spectrum, are shown in figure 40.

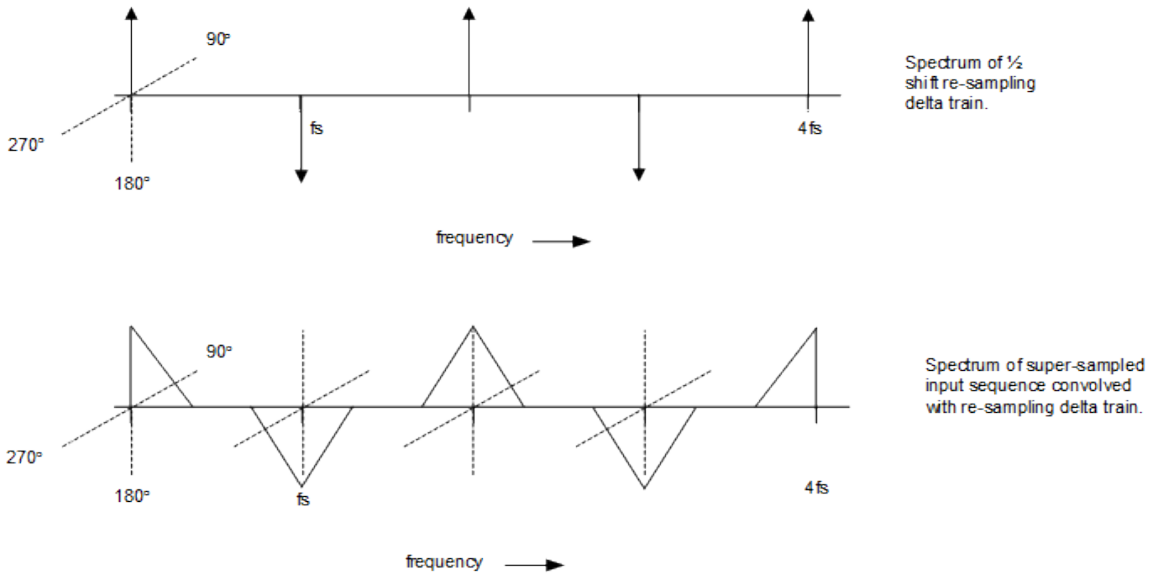

Figure 40: Frequency response of the 2nd phase of the 4-phase filter.

When the cross section through the convolved spectrum is taken, as before, the resultant of signal plus alias vectors is again a straight line since the alias always adds 180° out of phase. At half the sampling frequency, therefore, the magnitude response must be zero.

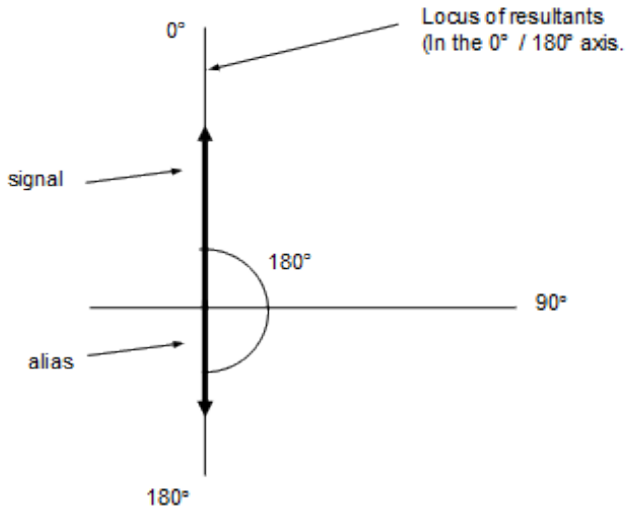

Figure 41: Locus of signal and alias resultant for the 2nd filter phase.

We can now plot the magnitude responses for each of the four individual phases in this example. The zero phase filter is the all pass with unity gain and zero phase. The 1/4 phase and 1/2 phase are discussed above, and the 3/4 phase is equal in magnitude response (but not in phase response) to the 1/4 phase filter. The complete family is shown in figure 42.

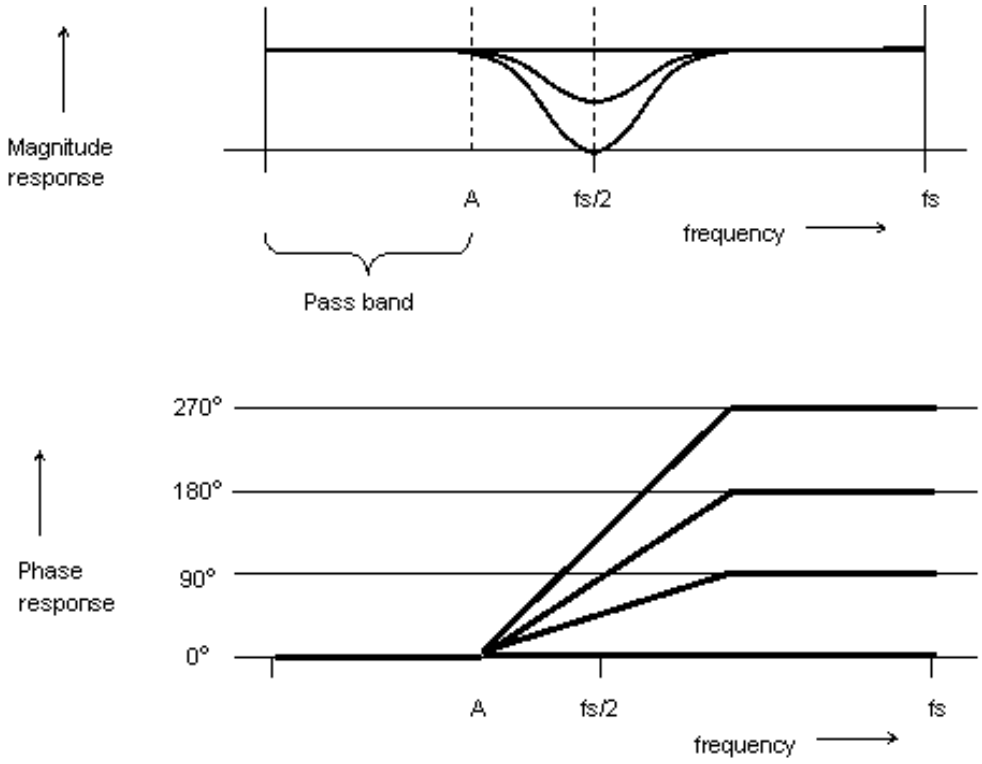

Figure 42: Magnitude and phase responses for each of the 4 phases in the polyphase filter.

The phase responses plotted above do not appear to be as expected, i.e. they do not resemble the plots of linear phase with a kink in the middle that were reproduced in the previous section. This is a consequence of moving the sampling frame of reference. The first of the four phases, the zero phase, represents re-sampling at the same location (i.e. not moving the sampling grid) and has a zero phase response, as expected. The second represents a shift of a quarter sample, or a phase shift of 90°. If the output sequence, shifted by a quarter sample, is referred relative to the original sampling grid it is necessary to impose a $-90°$ shift on the relevant line in figure 42.

This is done by drawing a line through the origin that passes through 90° at sampling frequency. The phase response obtained above is plotted relative to this sloping line to give the phase response relative to the original sampling grid. The third and fourth lines above represent phase shifts of 180° and 270°. If the respective outputs are to be expressed relative to the original frame of reference then it is necessary to impose phase shifts of $-180°$ and $-270°$ to the respective plots. This process yields the expected responses, as shown in figure 43.

It is worth noting that the previous observations about a straight locus and minimum amplitude response at $f_s/2$ are not specific to the right angled aspect of a four phase filter. The argument relies on similar triangles and is applicable to any phase of any complementary polyphase filter. The example of the first phase, 120°, of a three phase filter is shown in figure 44.

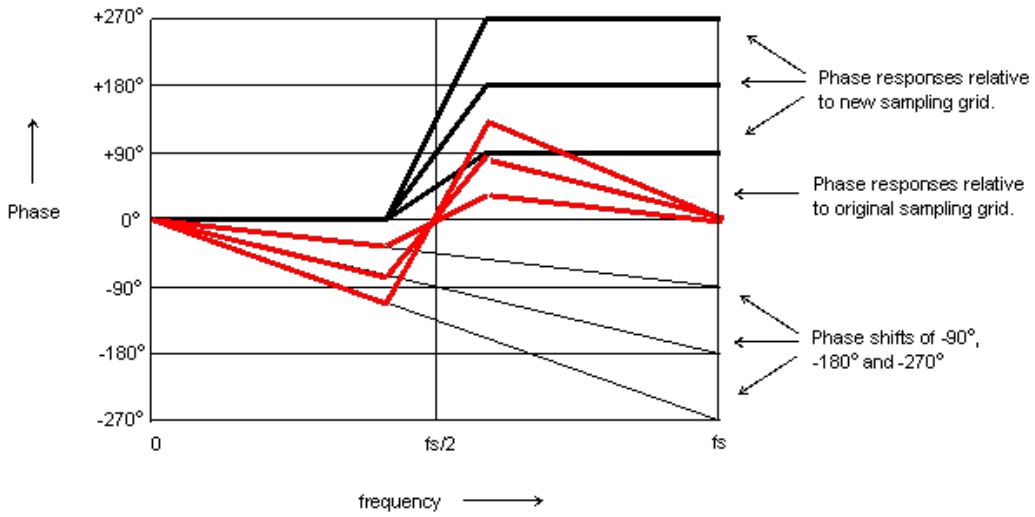

Figure 43: Phase responses for a 3-phase polyphase filter relative to the original and new sampling grids.

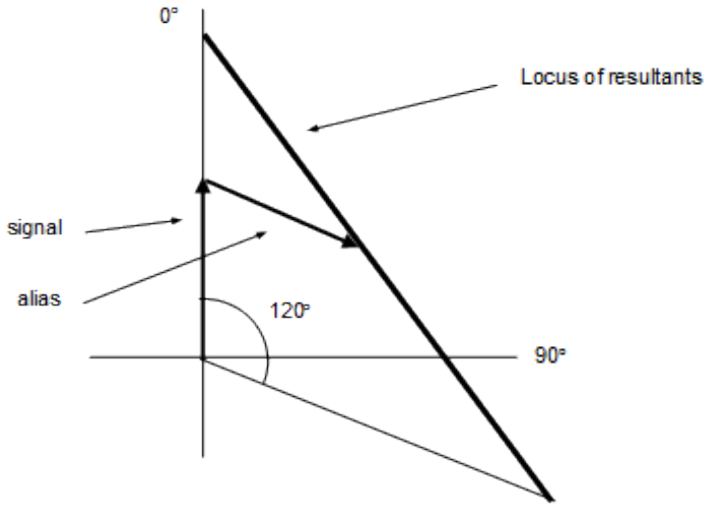

Figure 44: The first phase, 120°, of a three phase filter.

For a complementary filter the magnitude response at half sampling frequency equals unity for a 0° filter and zero for a 180° filter. For all other phases the magnitude response will lie somewhere in between, giving rise to the "womb" diagram shown in figure 45.

Bigger filters (that have longer apertures) give rise to a shorter transition band. However, for all filters there exists some frequency at which the signal is reduced to 90% (for example) of its original value. For a complementary filter the alias contribution at this frequency will be 10%. The amplitude and phase response is calculated by forming the resultant vector, as described above. It is only dependent on the signal to alias ratio (e.g. 90% to 10%) not in any way on the actual filter coefficients. Different filters, specified by different coefficients, have different lengths of transition bands, and so the speed at which the resultant moves along its locus is different too. The path, though, stays the same. This can be represented by marking equally spaced frequencies along the locus. The consequence of this is that the "womb" diagram of magnitude responses for a set of

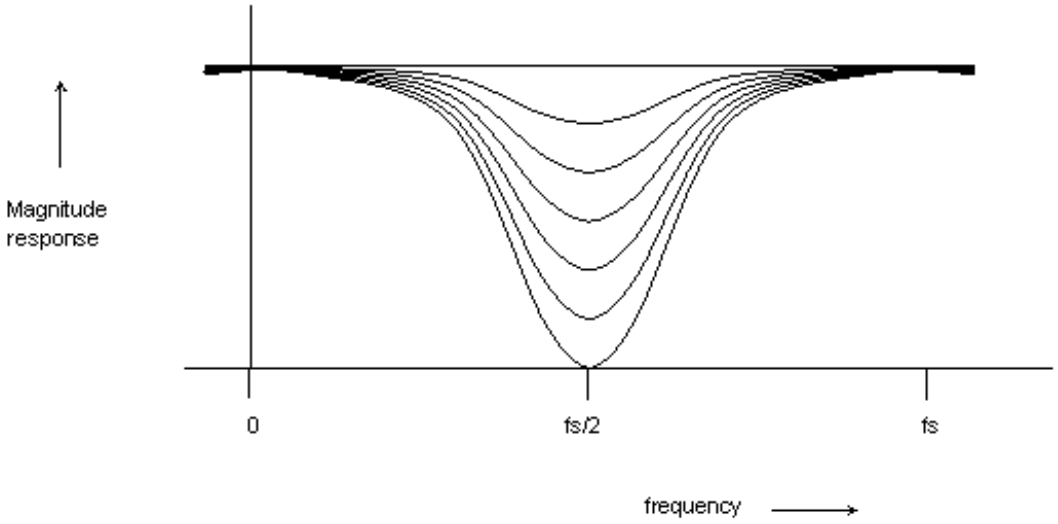

Figure 45: Magnitude responses of individual phases of a polyphase filter.

individual phases is universal, regardless of the precise filter coefficients. The frequency axis will be distorted but the amplitude relationships will be the same. This is shown in figure 46.

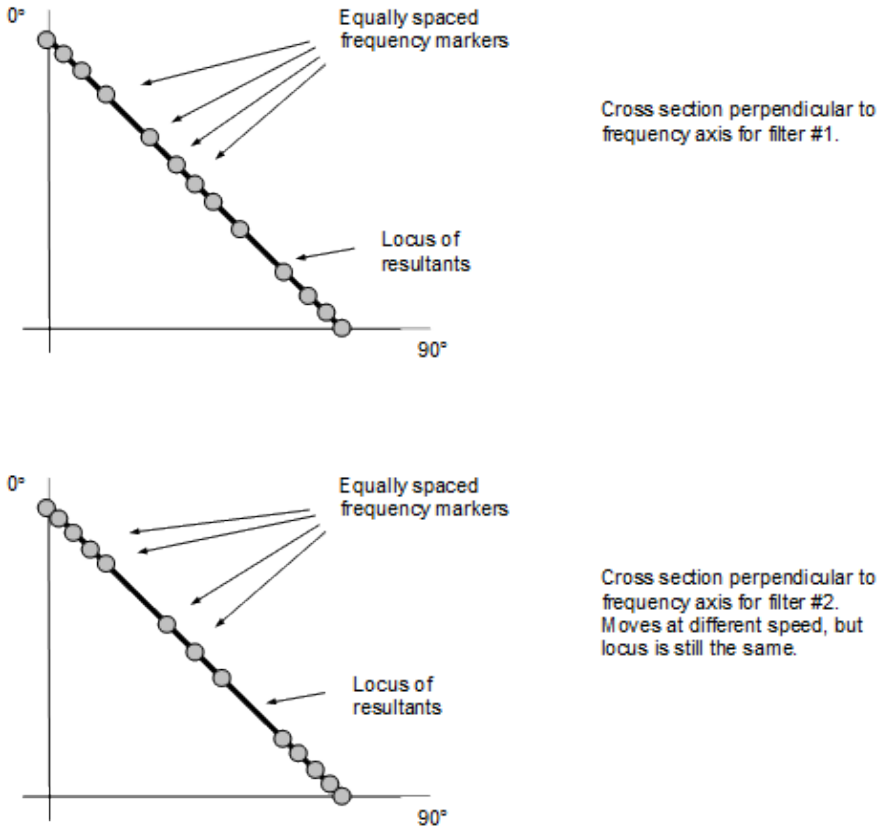

Figure 46: Cross sections through 2 different filters, with different frequency responses.

2.8 Numerical evaluation of filter coefficients

The practical, rather than theoretical, aspects of producing a set of filter coefficients are explained in this section. This provides a brief guide to how most filter coefficient generation software packages work. It also contains a description of error feedback which is an important topic in its own right.

There are different ways of creating a set of polyphase filter coefficients. The method outlined here is a description of a typical software package that is designed to allow a non-mathematician to enter details about a required polyphase filter (number of phases, size of aperture, up or down conversion etc...) and then to produce an optimum set of filter coefficients that comes at least some way near to having the ideal frequency response.

The ideal frequency response is inverse Fourier transformed to produce an impulse response function which can then be split up into sets that correspond to the individual filter phases. Although simple in principle, a few points warrant explanation. Firstly, the Fourier transform produces an output with exactly the same number of samples as were on the input. The final output is the impulse response function, which must have a number of points equal to the number of taps multiplied by the number of phases. This means that the ideal frequency response function must be specified at the same number of points, which often requires a degree of zero padding. In extreme cases the size of the required FFT is prohibitively large and a smaller one is used that produces an impulse response function that is likewise too small. Extra samples are then interpolated to pad the answer up to the correct length.

Secondly, an ideal filter is always perfect, and therefore has an infinite number of taps and coefficients of infinitely fine resolution. This is not possible in practice and thus it is necessary to have some sort of optimisation process, discussed in the first section. Finally, all hardware is constrained to a certain level of precision and so the coefficients must be quantised if they are to be useful. This is briefly explained in the second section.

2.9 Least squared error optimisation

The core of the process is the minimising of the squared error of the frequency response with respect to the ideal frequency response. A finite filter, h(t), has spectrum $H(\omega)$ given by

$$H(\omega) = \Sigma_{-\infty}^{\infty} h(t) e^{-j\omega t} = \Sigma_{-T}^{T} h(t) e^{-j\omega t}$$

In matrix form this is $H = F.h$ where H is an $f \times 1$ vector (for a frequency resolution of f points), h is a $t \times 1$ vector (for a time sequence of t points) and F is a $f \times t$ matrix of Fourier coefficients, i.e. $F_{ft} = e^{-j\omega t}$. If the desired frequency response is $D(\omega)$, or simply written in matrix form as D, an $f \times 1$ vector, then the vector of errors is $E = H - D$. If these errors are squared and summed over all frequencies the resulting expression, the total squared error, can be differentiated with respect to h(t), and set equal to zero to provide an optimum impulse response function, h_{opt} for the desired

frequency response $D(\omega)$.

$$\begin{aligned} E &= H - D \\ &= Fh - D \\ \Sigma_f e_f^2 &= E^H E \\ &= \left(h^H F^H - D^H\right)(Fh - D) \\ &= \left(h^H F^H Fh\right) - \left(h^H F^H D\right) - \left(D^H Fh\right) + \left(D^H D\right) \\ \frac{\partial\left(\Sigma e^2\right)}{\partial h} &= 2F^H Fh - 2F^H D \end{aligned}$$

At the minimum, this is equal to zero and thus

$$\begin{aligned} 2F^H Fh_{\text{opt}} &= 2F^H D \\ h_{\text{opt}} &= \left(F^H F\right)^{-1} F^H D \end{aligned}$$

An examination of the individual elements of $(F^H F)$ reveals that it is simply the identity matrix of order t (see the end of this section). Therefore the above equation tells us that it is the inverse Fourier transform[14] of the required spectrum that is, in fact, the best of all possible time histories in a least squares sense. For example, a desired spectrum could be achieved with a (possibly infinite) time history. If it is decided that the time history must be of a certain length (n), ie. a filter length of n, then the values of those n coefficients should be chosen so that the resulting spectrum is as close as possible to the desired spectrum. If the least squares approach is used the coefficients are simply the first n terms of the inverse Fourier transform of the desired spectrum. This is because the frequency components are all independent, therefore the loss of certain high frequency terms can never be compensated for by altering the lower frequency terms. The best that can be achieved (in a least squares sense) is to throw away the high frequency components and leave the rest untouched.

For many filters the pass and stop bands are of more importance than the transition band, and thus it is acceptable to reduce the error in the former regions at the expense of increasing the error in the latter region. This is achieved by artificially distorting the vector of true errors by multiplying it with a weighting function, $w(\omega)$, that represents the relative importance of different parts of the spectrum. For the weighting function written as an $f \times 1$ vector, w, the weighted error vector becomes

$$\begin{aligned} e_w &= e \cdot w \\ &= (H - D) \cdot w \\ &= (Fh - D) \cdot w \\ &= F_w h - D_w \end{aligned}$$

where the weighted Fourier matrix is formed by multiplying each row of the original Fourier matrix by the respective weight, and the weighted desired spectrum given by doing the same with the

[14]The implementation uses the Fast Fourier Transform routine which is designed to be most effective for sequences of length equal to a power of 2. This explains the comment near the beginning of the introductory section that most polyphase filters tend to have their number of phases equal to a power of 2.

desired spectrum.

$$(F_w)_f = w_f \times F_f$$
$$D_w = w \cdot D$$

Thus, in this case the optimum time function is derived in exactly the same manner as above, resulting in

$$h_{\text{opt}} = \left(F_w^H F_w\right)^{-1} F_w^H D_w$$

The remainder of this section shows how it can be seen that if F is a matrix of Fourier coefficients, then $F^H F$ is the identity matrix, a result used earlier in this section.

F is a $f \times t$ matrix of Fourier coefficients, i.e.

$$F_{f,t} = e^{-j\omega t} = e{-}j2\pi ft$$

so F^H is a $t \times f$ matrix where

$$F_{f,t}^H = e^{+j\omega t}$$

The inner product, $F^H F$, is a $t \times t$ matrix[15] in which each element is the sum of cross products. The element in the m^{th} row and the n^{th} column is the sum of the element by element product of the m^{th} row of the first matrix and the n^{th} column of the second.

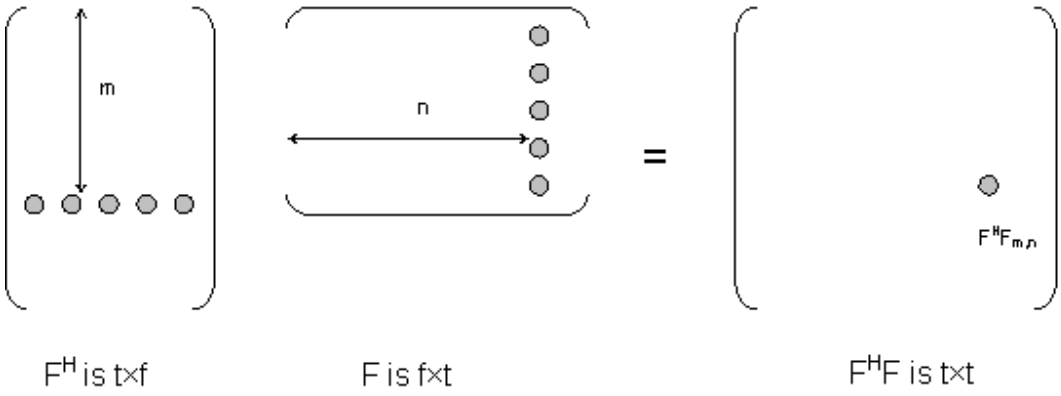

Figure 47: Representation of the vector inner product.

Thus

$$(F^H F)_{m,n} = \Sigma_f e^{+j2\pi fm} e^{+j2\pi fn} = \Sigma_f e^{+j2\pi f(m-n)} = f(ifm = n) = 0(ifm \neq; n)$$

since $e^{j2\pi f(m-n)} = 0$ if $m \neq n$ and 1 if m=n.

This shows that $F^H F$ is a scaled version of the identity matrix of order t.

[15]Matrix multiplication is not, in general, commutative ($a.b \neq b.a$) and it is also possible to form the outer product, FF¡sup¿H¡/sup¿, which, for this example, is an $f \times f$ matrix.

2.10 Quantisation

The desired set of filter coefficients will need to be implemented at a certain level of precision, 10 bit, for example, and thus the coefficients must be quantised to this level. This almost always results in a set that does not quite have unity gain at DC (usually a firm requirement) and so there follows a short process of adjustment to restore unity gain (at DC). There are two methods of adjustment that are used in the filtgui, tiffling and error feedback, both described below. There is necessarily some slight degradation in performance as a result of the quantisation and adjustment operations.

Both methods are described and then the implementation of error feedback is discussed in some detail. This is because of its importance in quantising continuous streams of data (as opposed to a fixed set of coefficients), something which is frequently done in many products including ARCs. Finally, comparisons are made between the two methods.

Tiffling

The process of Tiffling is not complicated. The coefficients are initially between zero and one. First, they are multiplied by the number of quantisation levels, and then they are rounded to the nearest integer. The rounding errors are recorded and sorted by size. Assuming that the original coefficients summed to unity, it follows that when the rounded coefficients are added together the result should ideally be equal to the number of quantisation levels, Q. (It is usually important to preserve unity gain at DC.) If the sum is n greater than Q (for some integer n) the n coefficients with the biggest rounding errors are each reduced by one. Conversely, If the sum is n less than Q, the n coefficients with the smallest rounding errors (i.e. most negative) are increased by one. For example,

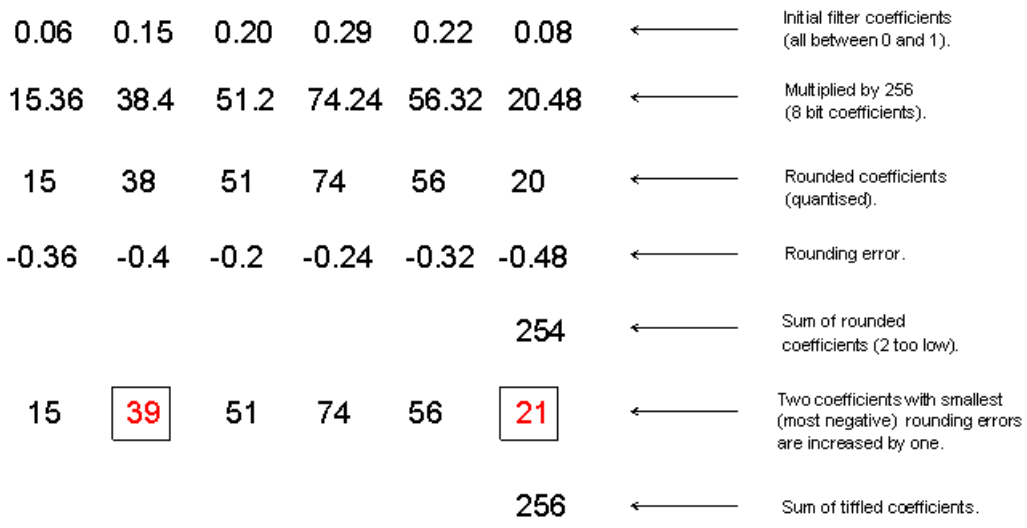

Figure 48: Example of tiffling the coefficients.

Error feedback

This process measures the cumulative error at each application of rounding, and uses it to bias the rounding of the next stage. If, for example, the sum of the first three rounded coefficients is 0.3 less than the sum of the first three raw coefficients, then 0.3 is added to the fourth coefficient before it is rounded to try to keep the cumulative error close to zero. An example is simpler than a wordy explanation, and for ease of comparison the same example is used as was for the tiffling.

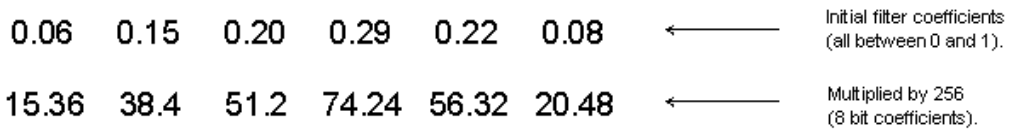

Figure 49: Filter coeficients used in error feedback example.

	Input	Biased input (input - previous cumulative error)	Output (rounded biased input)	Error (output - input)	Cumulative error (error + previous cumulative error)
1^{st} coefficient	15.36	15.36	15	-0.36	-0.36
2^{nd} coefficient	38.4	38.76	39	0.6	0.24
3^{rd} coefficient	51.2	50.96	51	-0.2	0.04
4^{th} coefficient	74.24	74.2	74	-0.24	-0.2
5^{th} coefficient	56.32	56.52	57	0.68	0.48
6^{th} coefficient	20.48	20	20	-0.48	0

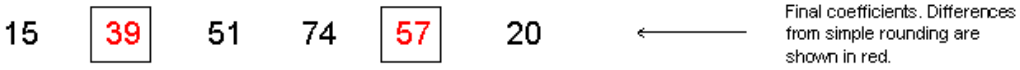

Figure 50: Final coefficients using error feedback.

Error feedback is a useful technique in any situation where quantisation occurs, for example at the output of multipliers or filters. In an ARC, therefore, as in many other products, it is often desired to implement error feedback in hardware to be used with a continuous stream of data samples. For this reason the implementation is described in further detail. In hardware terms the process described above might look something like the diagram below. This is not particularly practical since there are three un-pipelined adders[16].

Using subscripts n and $n-1$ to denote present and previous, we can write the following equations,

$$\text{error}_n = \text{out}_n - \text{in}_{n-1}$$
$$\text{cumulative-erro}_n = \text{cumulative-error}_{n-1} + \text{error}_n$$

[16]In this context a pipeline means registering the output on the next clock cycle. Combinatorial logic takes a finite amount of time which, although small, can still cause problems at high clock rates (i.e. MHz) unless it is broken into manageable chunks and registered at each stage. For the present hardware devices one adder is fine, two is probably fine but three is pushing it a bit.

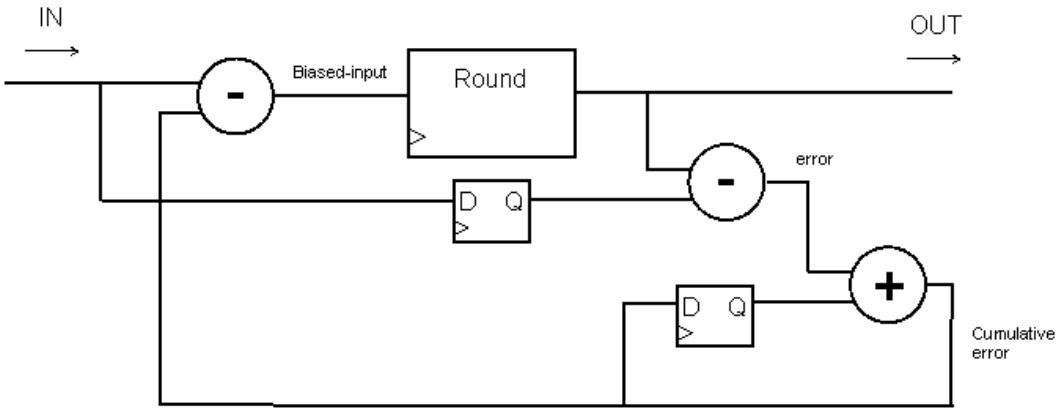

Figure 51: Hardware to implement error feedback.

from which it follows that

$$\text{out}_n = \text{in}_{n-1} + \text{cumulative-error}_n - \text{cumulative-error}_{n-1}$$

Looking at the diagram above it is seen that there is essentially a loop within a loop, one of which can be eliminated whilst still being able to maintain a record of the cumulative error. By changing the input to the error subtractor as shown in figure 52 exactly the same functionality can be obtained, which can be proved by producing the same equation as above.

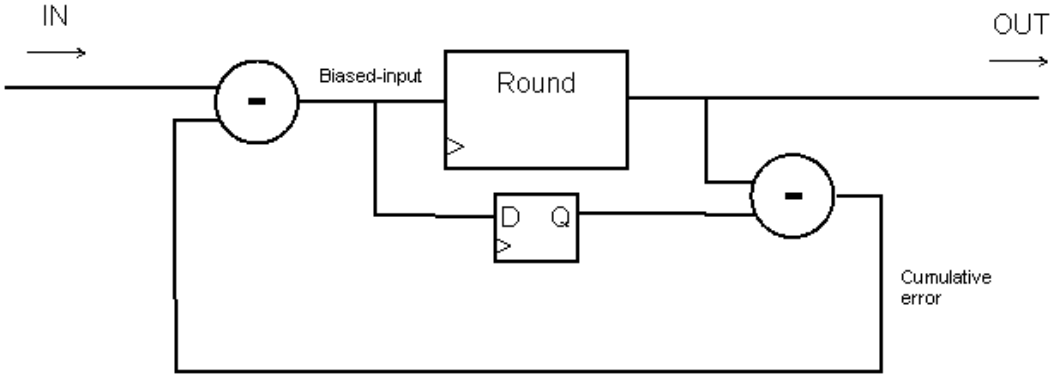

Figure 52: Alternative hardware to implement error feedback.

$$\begin{aligned}\text{cumulative-error}_n &= \text{out}_n - \text{biased-input}_{n-1} \\ &= \text{out}_n - (\text{in}_{n-1} - \text{cumulative-error}_{n-1})\end{aligned}$$

thus

$$\text{out}_n = \text{in}_{n-1} + \text{cumulative-error}_n - \text{cumulative-error}_{n-1}$$

This equation is the same for both circuit diagrams, confirming their functional equivalence. Further simplification is possible by noting that the output does not specifically depend on the rounding block. Different quantisation routines will produce different errors but the overall DC error should be zero in all cases. Simple truncation produces DC errors that rounding does not, but since error

feedback removes DC errors there is no advantage in using error feedback with rounding as opposed to truncation. Truncation is obviously simpler and cheaper by removing the need for an adder[17]. The subtractor (and the complementary flip flop) that works out the error is also un-necessary as the bottom bits that were chopped off *are* the error. Figure 53 below shows an example of a 10 bit input being reduced to 8 bits by truncating with error feedback.

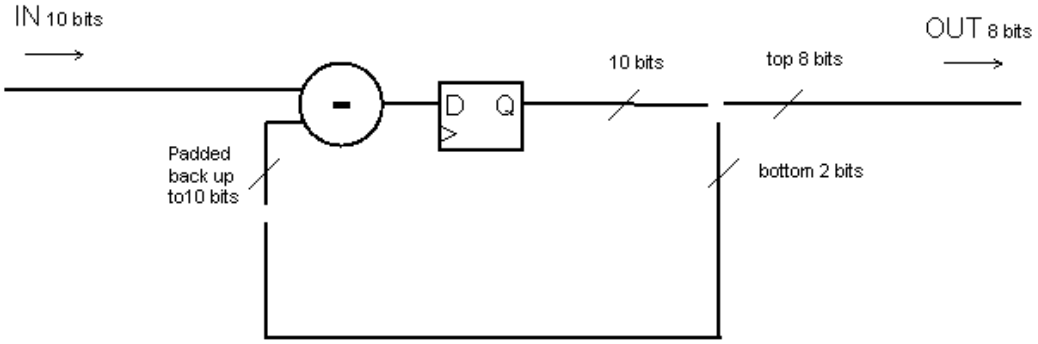

Figure 53: Hardware example of truncation and error feedback.

The error feedback loop has an infinite memory and it might be considered prudent to reset it every so often, for example at the beginning of each line. It is remotely possible that this could cause streaks (due to rounding transitions) that are the same from line to line. Hence it would be preferable to have a random seed for each reset. The accumulated error at the end of each line is more than likely to be effectively random, more so after running through the horizontal ancillary space, so in practice it is best simply to let the circuit run free. For debug purposes it is well worth being able to force the error to zero otherwise confusion may arise.

Differences between error feedback and tiffling

The quantisation errors of normal rounding can usually be assumed to be uniformly random, thus having a flat spectrum. Tiffling forces the DC gain to be zero by brute force, which must necessarily affect the error spectrum at other frequencies. This is shown on the diagram below. The spectrum of the tiffling errors is zero at DC (indicating that there is no error at DC) and elsewhere is slightly more than it would have been without the tiffling process.

The error feedback involves subtracting the previous error from the current one and so can be thought of as somehow involving two times a [1/2, -1/2] filter. The process is described very loosely as follows: if the original error sequence has a flat spectrum, then convolving it with the filter 2 [1/2, -1/2] will result in the spectrum of the error noise being the product of a flat function and a sine function (which is the response of the [1/2, -1/2] high pass filter). The sine function is double the normal magnitude because the filter is [1, -1] not [1/2, -1/2]. This means that error feedback gives not just zero noise at DC, but also very little noise at close to DC, as shown on the diagram above. The price to be paid for this is considerably more noise at higher frequencies, although hopefully such noise is less noticeable, at least for video data.

[17]Rounding is achieved by adding 1/2 and then truncating. The examples here assume that the result is then registered.

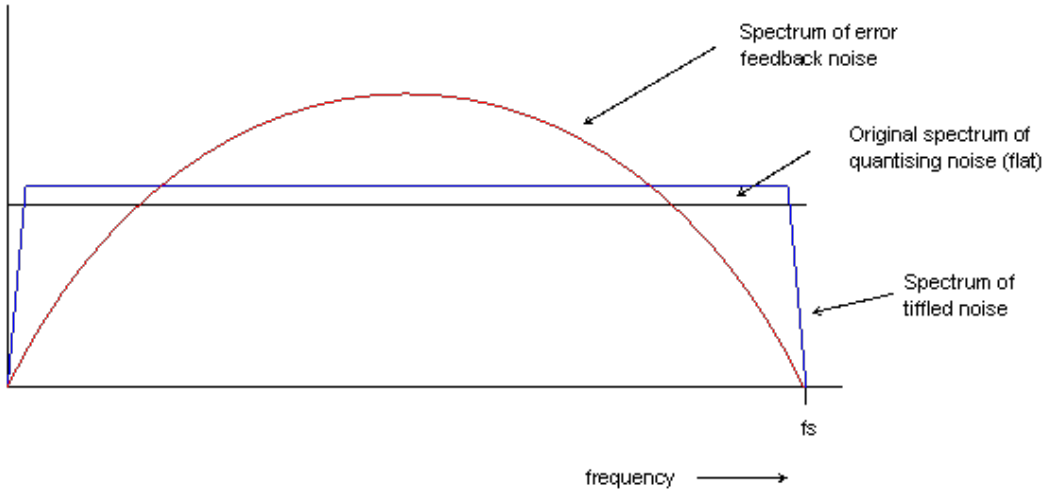

Figure 54: Spectra of quantising noise alongside spectra of tiffling noise and error feedback noise.

This description of error feedback is termed "loose" because in fact it is considerably more complicated than this. Trying to produce an error feedback noise sequence by high pass filtering a normal sequence will not work. The errors are complicated by the feedback loop which means that the individual errors do not fit the description above. The statistical expectation must be used which is itself complicated by the fact that the errors are not statistically independent, again, because of the feedback loop.

The two graphs below in figure 55 illustrate the difference between simple rounding and error feedback when applied to a string of random data. The underlying shape of each spectrum is distorted by random fluctuations so, for clarity, the smoothed versions are also included on the plots.

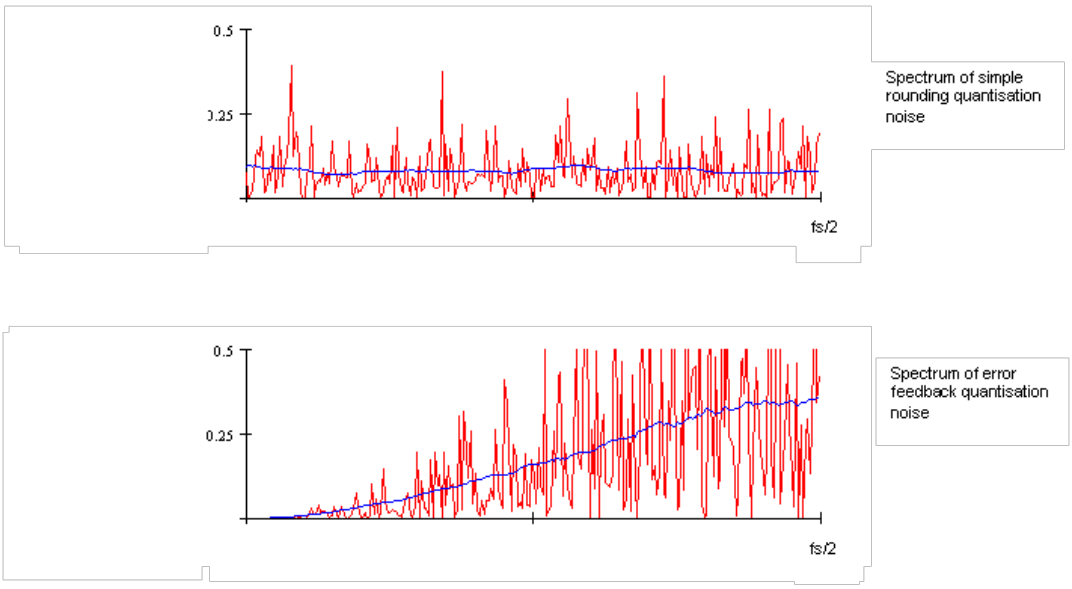

Figure 55: Spectra of real random data showing rounding quantisation noise and error feedback quantisation noise.

3 Vertical interpolation

The process of horizontally sampling a picture can be thought of as a one dimensional issue. It is acceptable to ignore the tiny time difference between adjacent samples and to say that the numerical difference of two samples is a function only of horizontal position. This is not the case for vertical interpolation which is a 2-dimensional problem involving time. Before introducing multiple field filter apertures it is necessary to understand the spectral properties of a conventional interlaced video image.

3.1 Quincunx interlace spectrum

Several text books show the Quincunx diagram of a vertical / temporal video spectrum. Not many explain where it comes from in more than a few sentences.

The scan lines of a conventional television picture are split into two fields. If the lines are numbered consecutively then the odd numbered lines form one field and the even numbered lines form the other. The fields are sampled from different moments in time, and thus so too are consecutive picture lines. This means that unlike the horizontal spatial frequency, the vertical component of a picture is related to the temporal components of the picture. As a consequence it is not usually adequate to represent the vertical spectrum of a video signal on its own. Rather, it must be represented as a two dimensional vertical / temporal spectrum. Several books show the 2-dimensional spectrum of an interlaced video signal as the "Quincunx" diagram[18] but the explanations of its derivation are usually somewhat sparse. Its appreciation is essential as a first step to producing a vertical filter for aspect ratio conversion.

The video signal has horizontal, vertical and temporal components. The horizontal component is independent of the other two and is not discussed here. The video signal is sampled both vertically and temporally and thus the base band spectrum repeats at integer multiples of the respective sampling frequencies. An individual field is sampled every 25^{th} second (for PAL) and thus the spectrum repeats every 25Hz. There are 576 active picture lines (for PAL) with 288 on each field, thus the vertical sampling rate for a single field is 288 cycles per active picture height (c/aph). The 2-D vertical / temporal spectrum for a single field is shown in figure 56.

The second video field occurs $1/50^{th}$ second later. As explained before if $f(t)$ is a signal that has a Fourier transform $F(\omega)$ and $g(t)$ is a signal that is a delayed copy of $f(t)$, then the spectrum of $g(t)$ is a phase shifted copy of the spectrum of $f(t)$.

$$g(t) = f(t - \tau)$$
$$G(\omega) = e^{-j\omega\tau} F(\omega)$$

The second video field therefore has a spectrum that is essentially identical to the first field but which is modified by the phase term $e^{-j\omega t}$.

At frequencies for which ωt is equal to an even multiple of 2π this phase term equals +1, and when

[18] The dictionary defines Quincunx as: *an arrangement of five things in a square or rectangle with one at each corner and one in the middle.*

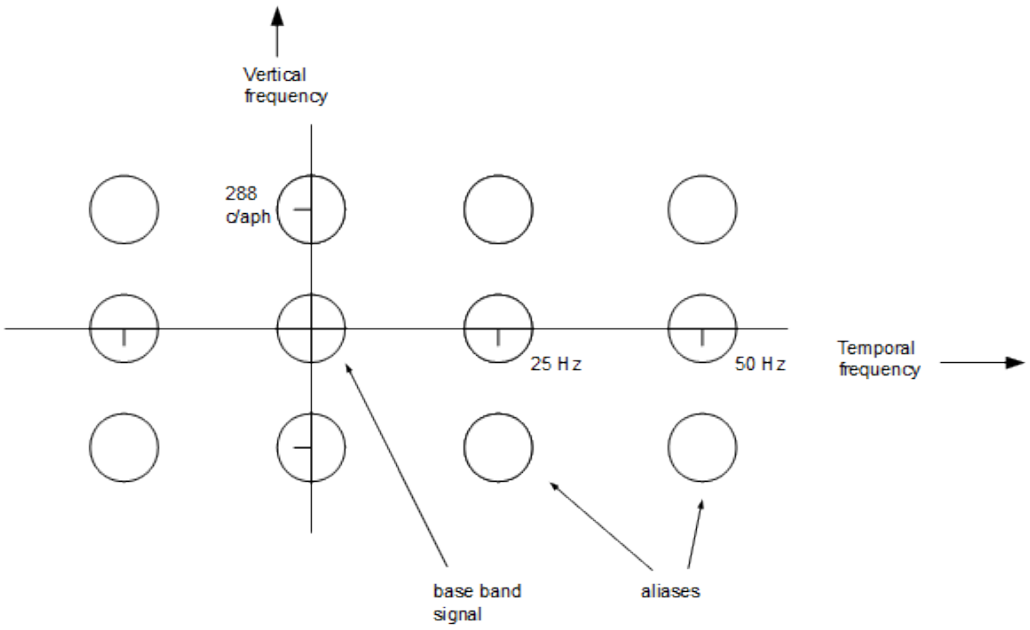

Figure 56: 2-D spectrum of a single field of an interlaced PAL TV signal.

ωt is equal to an odd multiple of 2π this phase term equals -1.

$$\begin{aligned} e^{-j\omega t} &= +1 \\ \omega t &= 2k\pi (\text{for some k}) \\ 2\pi f t &= 2k\pi \\ f t &= k \\ f &= 50k (\text{because t=1/50}) \end{aligned}$$

$$\begin{aligned} e^{-j\omega t} &= -1 \\ \omega t &= (2k+1)\pi (\text{for some k}) \\ 2\pi f t &= (2k+1)\pi \\ f t &= k + 1/2 \\ f &= 50(k+1/2) (\text{because t=1/50}) \end{aligned}$$

When the 2-D spectrum is plotted the aliases at 25Hz, 75Hz, 125Hz etc... therefore have opposite polarity to the base band energy and all the aliases at 50Hz, 100Hz etc... And by exactly the same argument every alternate clump of energy along the vertical axis has inverted polarity. Where the vertical and temporal phase terms are both negative the product is positive. The spectrum of the second field is shown below with the positive and negative polarities indicated by different shading.

The overall spectrum of the entire video frame is the sum of the individual spectra of the two constituent fields (because the Fourier transform is a linear operator), which interact constructively and destructively alternately, as shown in figure 58. This is commonly called the Quincunx diagram.

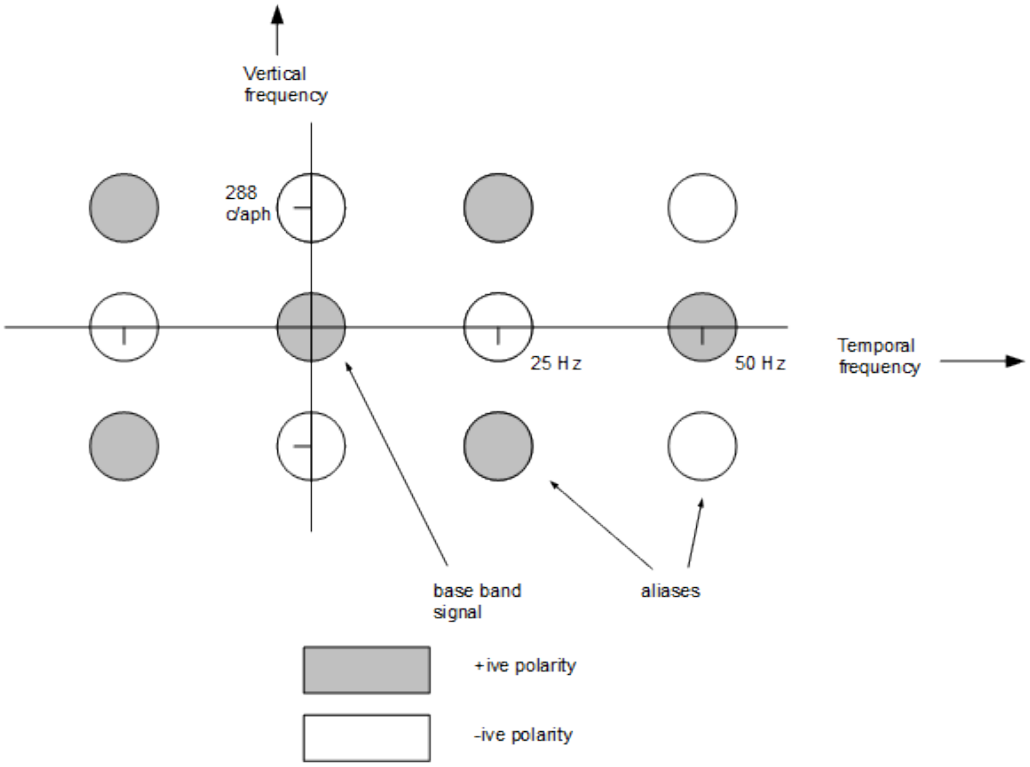

Figure 57: 2-D spectrum of the second field of an interlaced PAL TV signal.

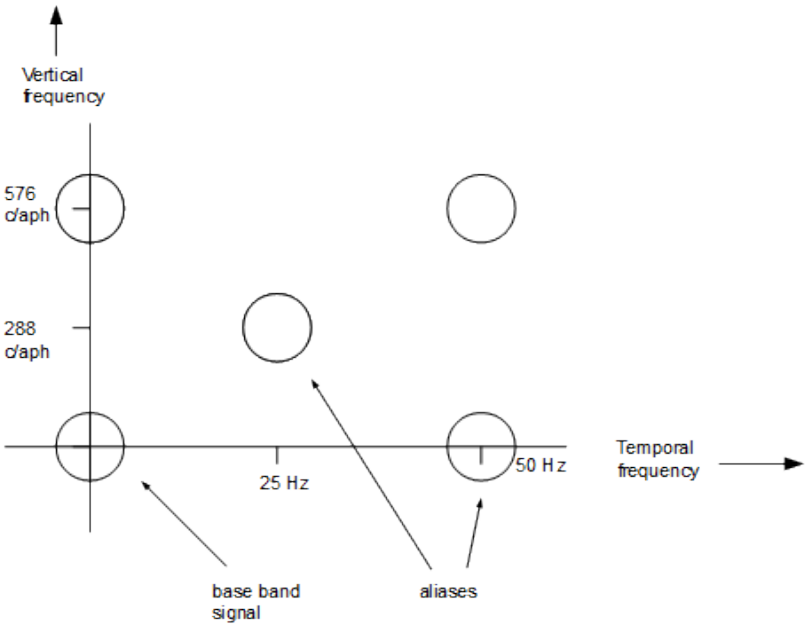

Figure 58: The Quincunx diagram showing the 2-D spectrum of a single interlaced video frame.

3.2 Single and multiple field interpolators

This section explains why a single field aperture is inadequate for vertical interpolation.

The true spectra as opposed to the stylised circular ones of the preceding section look more as shown below (although still stylised, albeit less so). A cross section along the 0Hz line mostly contains signal but does also have some alias energy. Conversely, a section through 25Hz is not all alias, there is also significant signal energy.

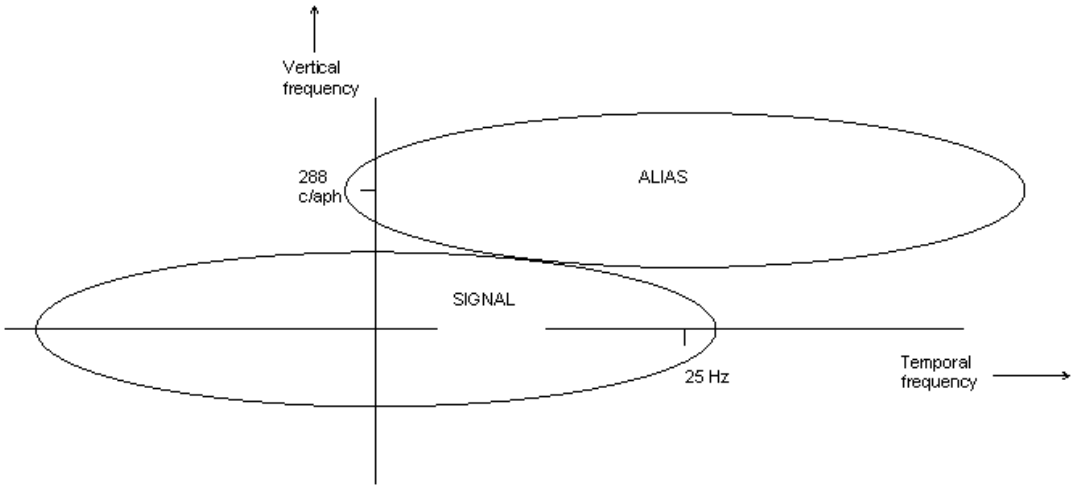

Figure 59: A more realistic representation of a real signal spectrum.

A single field interpolator can have no temporal qualities to it since it comes from a single snapshot in time. Therefore its vertical frequency response is always the same, i.e. independent of time. This means that it is always a compromise. At 0Hz it will lose much of the signal and yet at 25Hz it lets too much of the alias energy through. It is doubly inadequate as shown in figure 60.

The ideal filter has a 6dB cut off at the point where the signal and the alias cross. At 0Hz the crossover is at a much higher frequency than at 25Hz and thus a more advanced interpolator would have different filter cut-offs at different frequencies as shown in figure 61. This is obviously impossible for a single field aperture. It is, however, possible to design a multiple field aperture that can do this to a certain extent. The diagram below shows a representation of a multiple field aperture interpolator with both signals and aliases for 0Hz (in red) and 25Hz (in blue). The design of such a multiple field aperture filter is the subject of the next section.

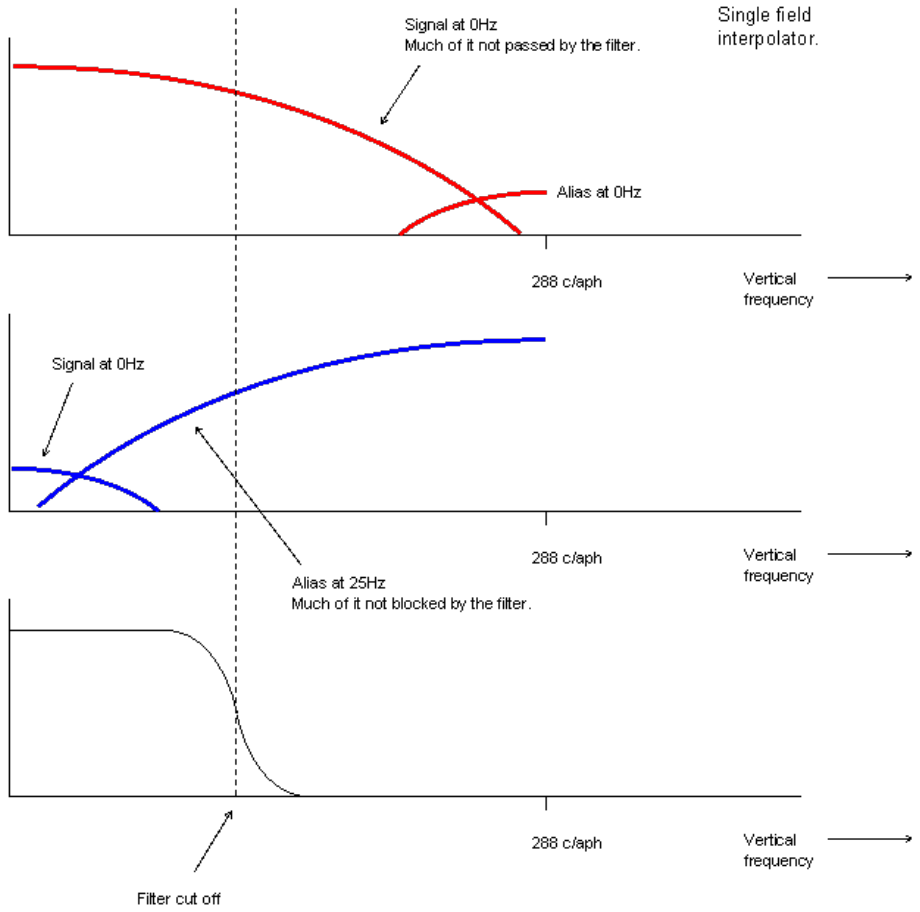

Figure 60: The spectral inadequacies of a single field vertical filter that has the same vertical responses at all temporal frequencies.

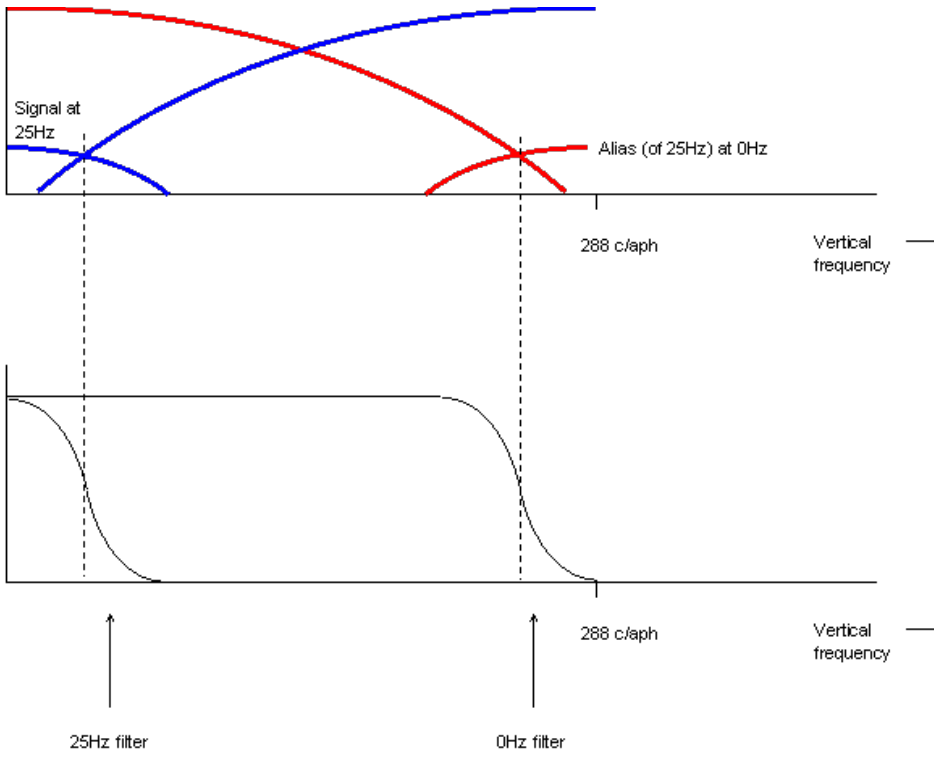

Figure 61: A multi-field vertical filter that can have different vertical responses at different temporal frequencies.

3.3 An example of a multiple field aperture

A simple example of a multiple field aperture is introduced to highlight the nature of the LF and HF contributions.

For simplicity, the necessary de-interlacing is very basic and it is assumed that the samples from consecutive fields are all co-sited after being suitably zero padded. Also for simplicity, this example of a multiple field aperture has three fields although this is not strictly necessary - the IQDARCM has only two fields. Lastly, for simplicity, there will only be 3 lines from each field. In practice this is more likely to be at least 4.

The quincunx diagram shows that the nearest alias to the base band video signal is the one at 25Hz and 288c/aph, shown in the diagram below. If the vertical and temporal components are considered separately it can then be shown how to combine them.

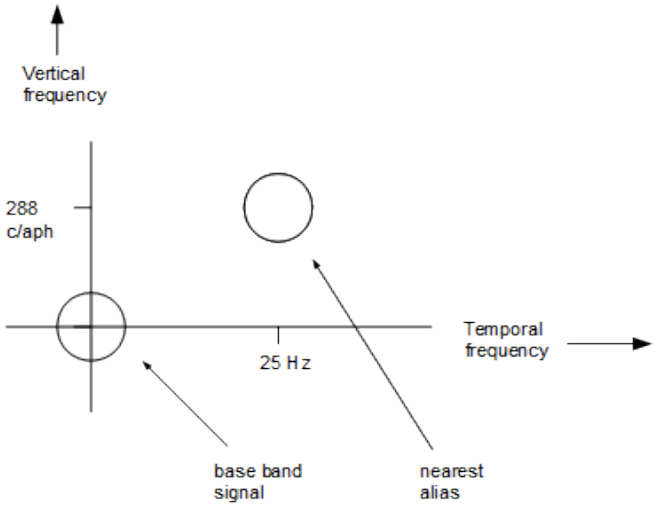

Figure 62: 2-D spectrum of a video frame (only part of 1st quadrant shown).

First of all, the interlaced inputs are zero padded to de-interlace them into sets of co-sited samples, as depicted in figure 63. As mentioned above, this is crude but adequate for an introductory example.

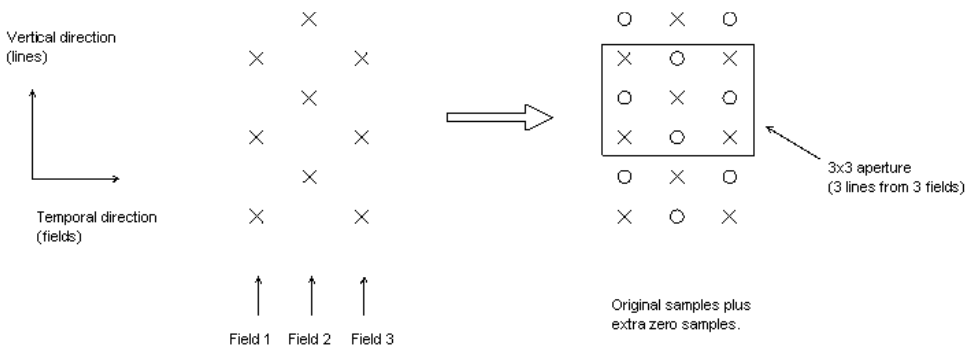

Figure 63: Crude de-interlacing performed by zero-padding.

If all the coefficients in the outer fields are zero then there is no time dependence, thereby making the filter response independent of temporal frequency. For this reason the filter is good for material containing much motion. The vertical frequency response can be evaluated in two ways. It can

$$\begin{matrix} 0 & 1/2 & 0 \\ 0 & 1 & 0 \\ 0 & 1/2 & 0 \end{matrix}$$

either be thought of as a two phase filter, the two phases being [1] and [1/2, 1/2] which are then averaged together, or equivalently it can be thought of as a [1/2, 1, 1/2] filter at twice the sampling rate i.e. 576 lines/aph.

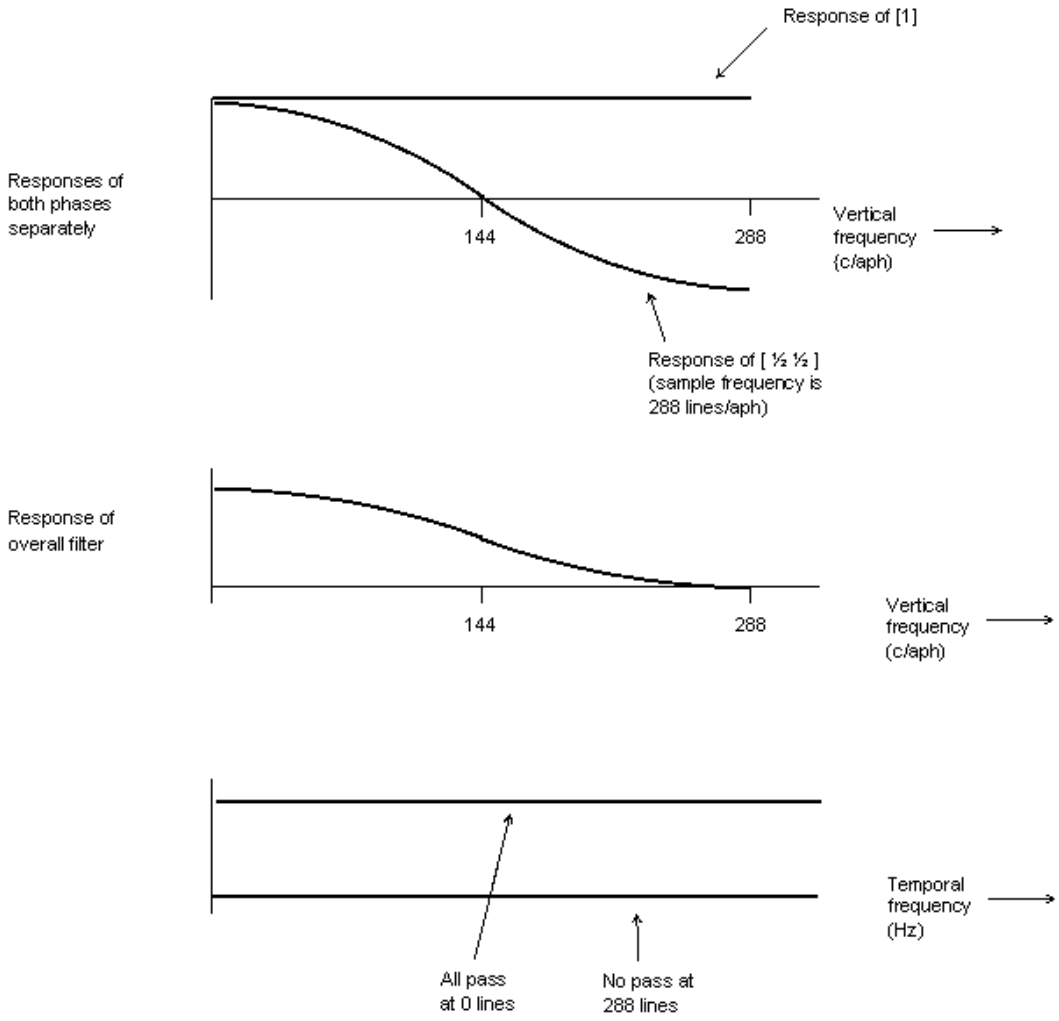

Figure 64: Vertical frequency responses at different temporal frequencies for the example filter in the text.

The converse of this filter is that if all the coefficients in the top and bottom lines are zero then there is no vertical spatial dependence, thereby making the filter response independent of vertical frequency. The filter has a perfect vertical response but relies on the material containing no motion (i.e. stills).

$$\begin{matrix} 0 & 0 & 0 \\ 1/2 & 1 & 1/2 \\ 0 & 0 & 0 \end{matrix}$$

As before, the response can be evaluated in two ways. It can either be thought of as a two phase filter, the two phases being [1] and [1/2, 1/2] which are then averaged together, or equivalently it can be thought of as a [1/2, 1, 1/2] filter at twice the sampling rate. The only difference is that the samples are with respect to time, not the vertical direction.

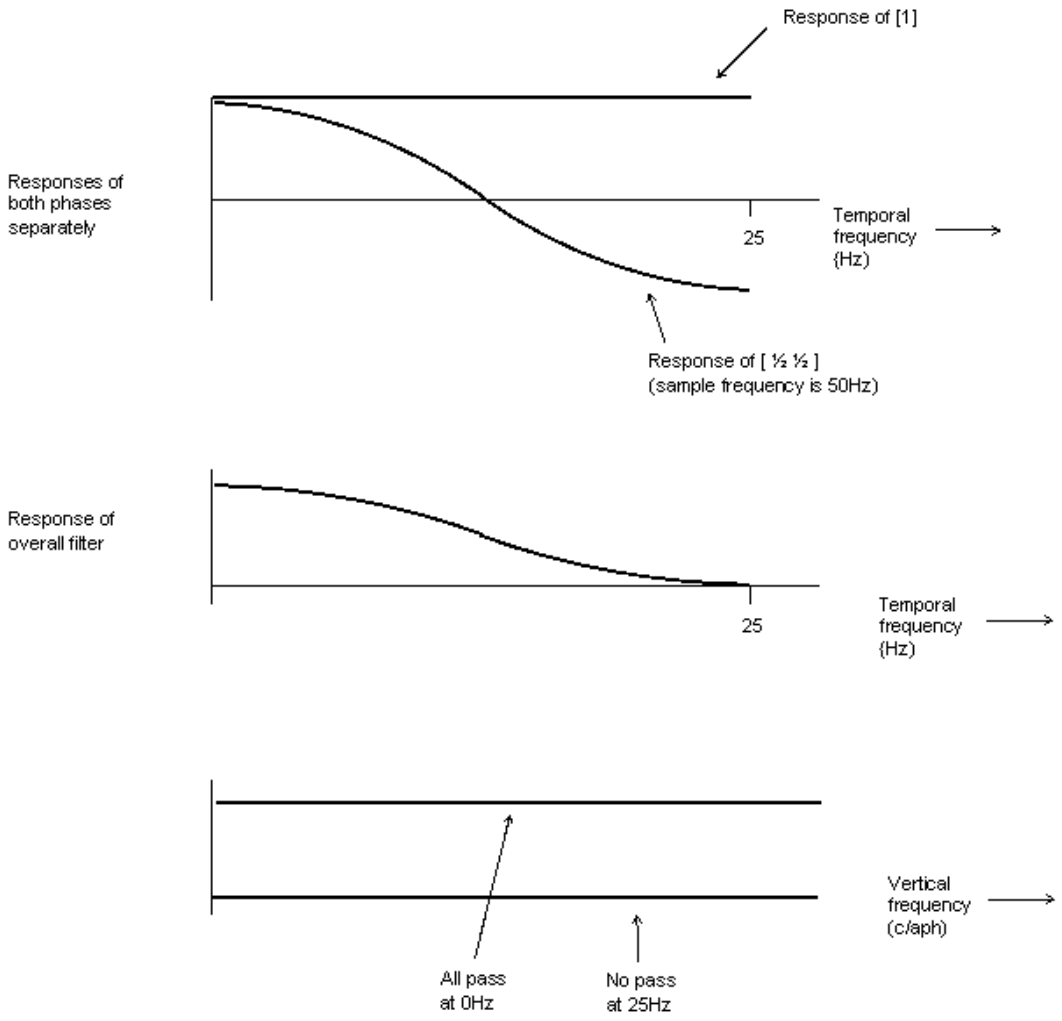

Figure 65: Temporal frequency responses at different vertical frequencies for the example filter in the text.

The 2 dimensional spectra of these two filters are shown in figure 66.

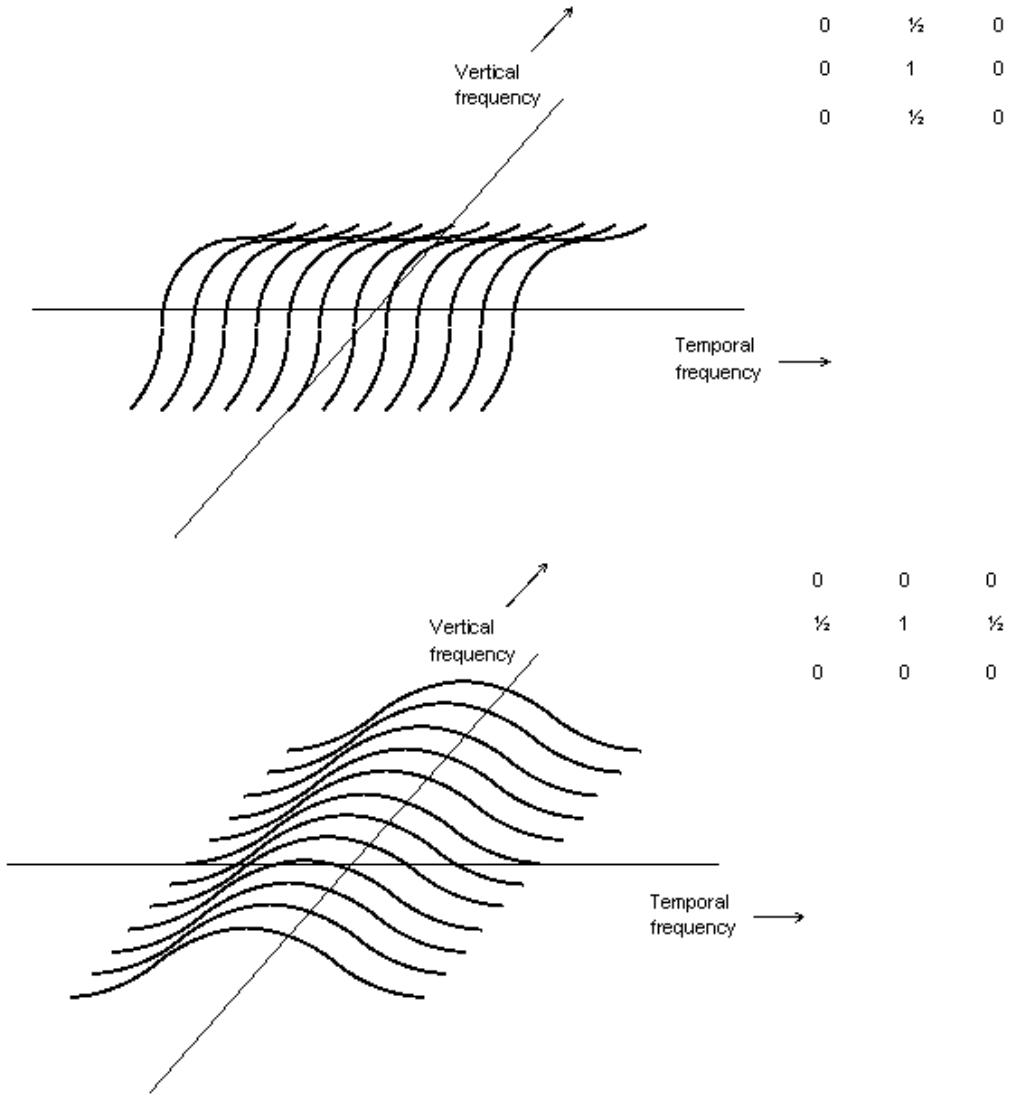

Figure 66: The 2-dimensional vertical-temporal frequency responses for the example filter in the text.

The obvious way to combine these two filters is simply to add them (and divide by two) to give

$$\begin{array}{ccc} 0 & 1/4 & 0 \\ 1/4 & 1 & 1/4 \\ 0 & 1/4 & 0 \end{array}$$

This, however, does not give satisfactory performance. A better solution (simplified) is to use the following filter.

$$\begin{array}{ccc} -1/16 & 0 & -1/16 \\ 0 & 1/2 & 0 \\ 1/8 & 0 & 1/8 \\ 0 & 1/2 & 0 \\ -1/16 & 0 & -1/16 \end{array}$$

Each of the outer vertical fields is a weighted [-1/4, 1/2, -1/4] filter which has a double null at zero frequency and unity gain at half the sampling frequency, i.e. is a high pass filter. With the extra zeros the filter has a peak lower down at one quarter of the sampling frequency, or equivalently it can be regarded as the same filter but with a lower sampling rate. Either way, the response is shown in figure 67.

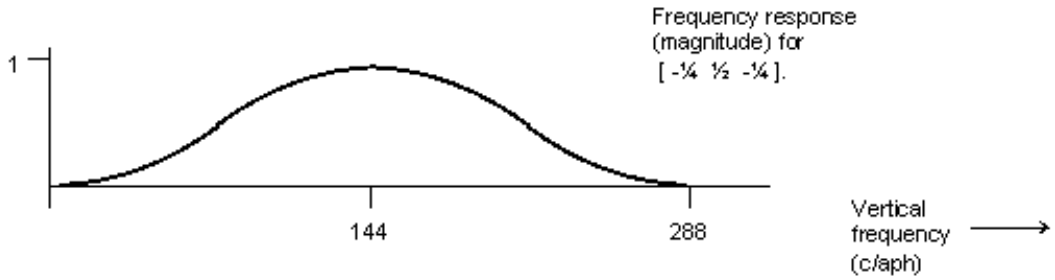

Figure 67: Vertical frequency response of a [1/4, 1/2, 1/4] filter.

The total response of both outer fields is simply their sum, i.e. [-1/8, 1/4, -1/8] or equivalently 1/2 [-1/4, 1/2, -1/4]. The central field is 2 [1/2, 1/2], the factor of two being due to the fact that the de-interlacing involves zero padding which halves the spectral energy. Ignoring this factor of two, the two responses are plotted below.

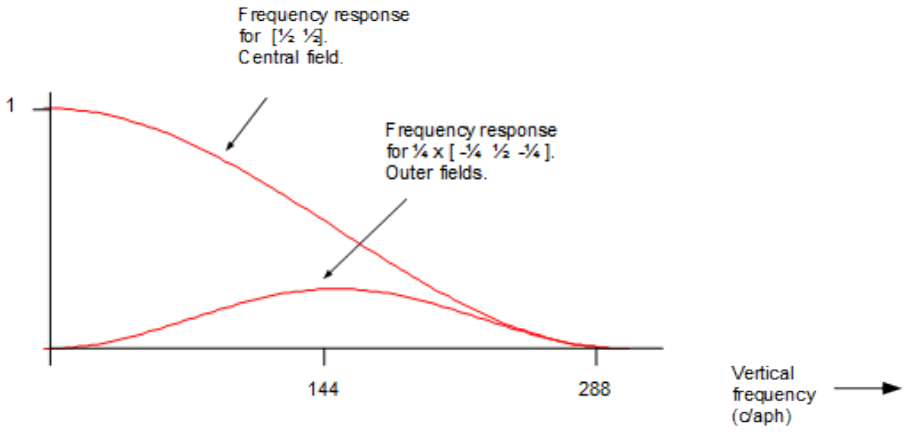

Figure 68: Vertical frequency responses of the central and outer fields.

These two filters are often referred to as the low frequency and high frequency components or, even more often, simply as the LF and HF components of the aperture. (This can be slightly misleading.)[19]

The overall response of the filter at 0Hz is simply the sum of these two individual responses, and as shown in figure 69 it can be seen that the high frequencies are boosted. However, at 25Hz, the time

[19]The low frequency portion of the overall filter response comes exclusively from the central field, thus it is fair enough to refer to the central field as the LF field. However, the high frequency portion of the overall response does not come exclusively from the outer fields but from the central field as well. It is therefore not strictly accurate to refer to the outer fields as the HF fields, though this is often done.

difference between the central and outer fields (i.e. $1/50^{th}$ second) represents half a wavelength, and thus the responses are out of phase with each other. The overall response is still found by adding the two individual responses but out of phase with each other, i.e. subtracting them. The result is that the high frequencies are significantly attenuated.

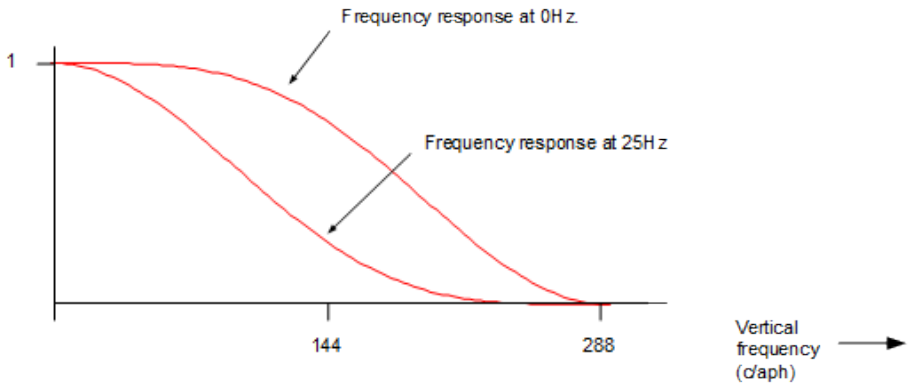

Figure 69: Vertical frequency responses at 0Hz and 25Hz.

This satisfies the initial requirement that at 0Hz the filter has a relatively high cut off and that at 25Hz a relatively low cut off.