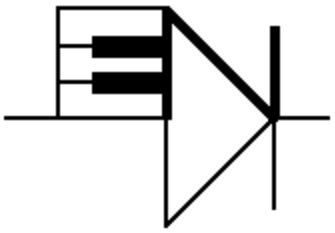


ELECTRONOTES 205

Newsletter of the Musical Engineering Group

1016 Hanshaw Road, Ithaca, New York 14850

Volume 22, Number 205 March 2008

GROUP ANNOUNCEMENTS

CONTENTS OF EN#205

Page 3 Sample-Rate Changing by Polyphase On Demand

Page 17 Recovery from Bunched Sampling

It has been a long time since we produced a new issue of Electronotes. True enough, there have been a passel of Application Notes issued, belying any claim to complete laziness, but no EN's. In fact, in the early days, there was a big difference between the EN's and the AN's: the EN's being more news while the AN's, originally just two pages, were reference materials with less urgency. So there was often some question as to whether a particular item should be an article in an EN or a separate, long AN. This issue contains two articles that seemed to want to be part of an EN, as they were extensions of previous articles. Another hold-up was that we were supposed to do the second part of the finite wordlength presentation, material that was not finished, and still needs to be finished, although it all exists with messy handwritten figures, used for several years for instructional purposes

Another thing was that we stopped making paper copies of new issues and new AN's. That is, there were no actual subscriptions, all the new materials being posted online for free, while we simplified the paper business by selling only back issues. This meant that when we finished an order, we were done. What an improvement. At the same time, making an electronic file is different from preparing an original for hard-copying. In many ways, it is much harder. On the other hand, some things are easier since we don't need to worry so much about empty space, fronts and backs, and so on.

But by far, the biggest thing that has happened in the last few years was the loss of three of our heroes: Bob Moog, John Simonton, and Doug Curtis. Every time I thought about a new EN I knew I had to address this loss, and it was something I did not know how to do, and I still don't. Let's just say that Bob was the pioneer and leader in the commercial synthesizer field, John was the person who made the kit synthesizer a reality (thereby getting a lot of EN builders jump started), and Doug, through his CES chips, made both commercial and home building much simpler. And, as much as there was to admire about their work, these were very good people; good friends.

There are so many Bob Moog stories that have been told which are available. Here are two others, probably not previously told, and not terribly important perhaps, but both of which fit the pattern. In 1971, I took a course, Electrical Engineering 4436 which Bob taught at Cornell. (I got an A+ which delighted me no end, one of very few grades I actually cared much about.) As one of the projects, there was an analysis of a spring reverb. Bob had given out the project and a reference during one class, and suggested how to approach it. What he said seemed wrong to me! But I thought I needed to think about it before saying anything. By the next class I was waiting outside his office for him to arrive, and when he saw me, I said something like "About the moment of inertia of the coil…." I don't know if he had thought about it between the classes, or if his physics training kicked in instantaneously. "You're absolutely right," he said, without my going further. He was very much aware of the way engineering and physics could be so obvious that true practitioners might only need to grunt.

The second story involves sitting with him at a meeting somewhere (probably AES) when a guy came over with a sketch a friend of his had made of Bob; pencil or charcoal I think. The guy asked if Bob would autograph it! That was perplexing enough – why should Bob sign a picture someone else drew? The sketch wasn't all that bad, but, the man in the sketch had a prominent mustache - and Bob did not! This was a problem. Bob looked at it for perhaps a full minute, and then did autograph it. Handing it back he said "Be a good kid and ask your friend to take the mustache off."

And a John Simonton story. John's kits were linear controlled of course, and at one point some other kit maker criticized John's product as being something like: incapable of musical expression. There was some kind of a dust-up as a result, and somehow I got involved – not by way of taking sides – but by suggesting that the issue itself was somewhat silly and unworthy of our time. I got a call from John. His first question was: "Are we still friends?" That was what he wanted to know. We were still friends, and remained so.

This electronic music business has been a great endeavor, blending art and technology, and exposing us to so many wonderful people.

Sample-Rate Changing by Polyphase On-Demand

-by Bernie Hutchins

A standard joke relate to the notion that those who can – do; while those who can't – teach. A possible corollary to this is that when it gets down to "really" doing something, the "textbook examples" just don't apply. Typically, for sampling rate changes, only sampling rates at a ratio of small integers are presented as examples, along with a presentation of the efficiencies of a "polyphase" realization. In our recent presentation [1] we added slightly to this basic view by pointing out a way to derive polyphase structures intuitively rather than formally.

Here we want to examine a notion we can call "polyphase-on-demand." In its most general form, it is the basis of the intuitive approach [1] and was mentioned in a potential real-time rate-changing example [2]. The most basic form of the idea is useful for applications ranging from the generation of figures for this note (see below) to the writing of actual Matlab (or other) code to convert a file at one rate into a file at a new rate.

RATE CHANGING FOR RATIOS OF LARGE INTEGERS A CASCADE OF RATE CHANGERS

Before we move on to the polyphase-on-demand, we need to discuss the complications that come up when the ratio of integers that describes the rate change becomes one of larger and larger integers. A small-integer rate change of 3/2 (for example, going to 30 kHz from 20 kHz) presents no major problems, and perhaps polyphase in any form is not efficient for such a case. It is in cases of larger integer ratios such as 11/5 or 6/13 that we become more inclined to embrace polyphase. Eventually, the integers in the ratio may become so large that a polyphase approach, in the sense of pre-computed polyphase coefficients, may become impractical. In such a case, the first thing to consider (in a standard approach) is to see if the rate change can be done in steps. The second thing is the polyphase-on-demand to be discussed below.

So before looking at the polyphase-on-demand, let's review the rate change in steps. As an example, consider a rate change from a CD rate of 44,100 Hz to a DAT rate of 48,000 Hz. This is a ratio of 480/441 which can only be reduced by a common factor of 3 to a 160/147 ratio.

Here we could achieve the desired rate change by upsampling by a factor of 160 to 7.056 MHz and then downsampling by 147 to get our desired 48,000 Hz. This intermediate sampling rate is very high to work with, but perhaps more importantly, it would involve many, many needless computations [1].

[Note that our considerations should involve the actual bandwidth of the signal. We must assure that all intermediate frequencies are at least twice the bandwidth. We might actually know a signals bandwidth, but in the absence of this information, we need to assume it might be as large as half the sampling rate of the signal as given to us. Generally this means that in order to avoid aliasing we need to always do an upsampling first, and any intermediate sampling frequencies must always be greater than the original. Otherwise, we need to do a pre-decimation filtering (reduce the bandwidth prior to downsampling) and be prepared to accept some loss of information as a result of this filtering.]

There are of course additional prime factors of the numerator and the denominator that can be considered separate from any common factors we have removed. The numerator 160 can be factored into 5 x 2 x 2 x 2 x 2 x 2 and the denominator 147 has factors 3 x 7 x 7. There are various ways to group the factors, and it is clear that we probably would want to do a rate change in terms of alternating upsamplings and downsamplings. This suggests several ways to group the six prime factors of the

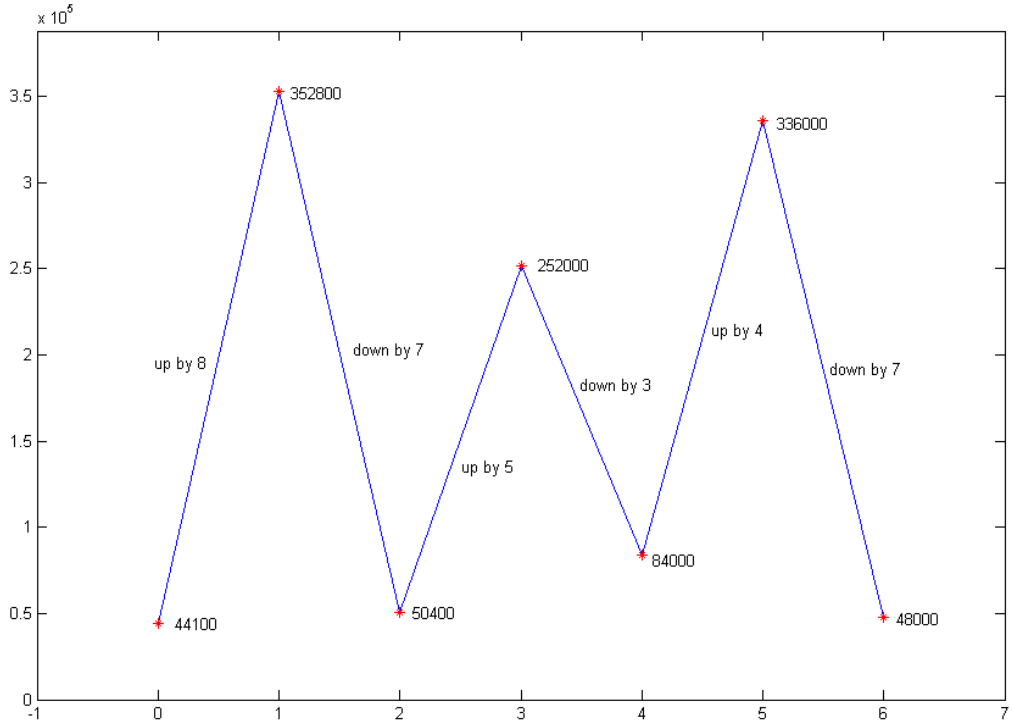


Fig. 1 Three upsampler and three downsampler stages from 44.1 kHz to 48 kHz.

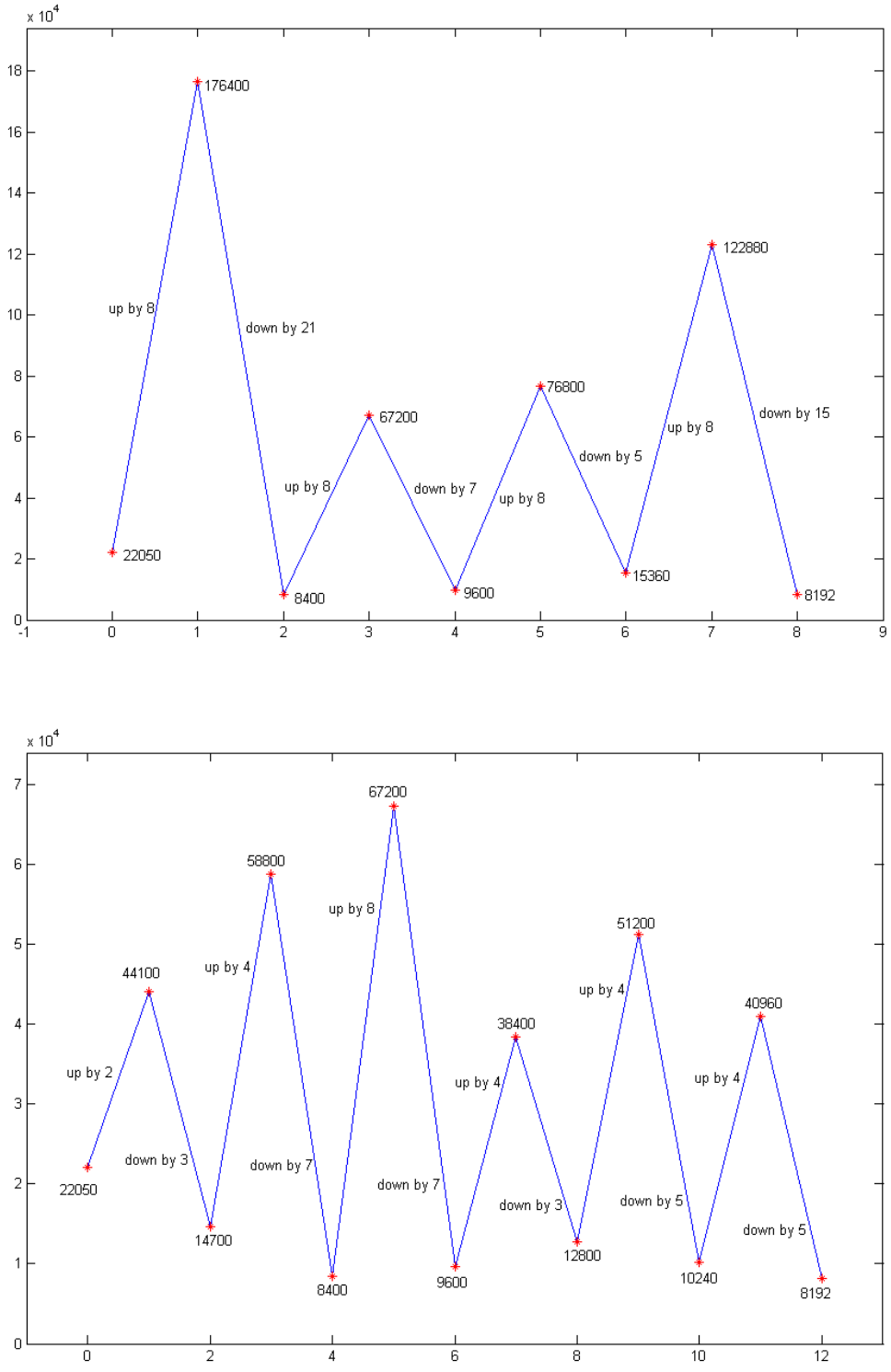


Fig. 2 Two possible stagings of a 8192/22050 rate changes, the lower example having a lower intermediate frequency.

numerator into three upsamplers. One choice would be 8, 5, and 4. For example (see also Fig. 1) we could upsample by 8 to go from 44,100 to 352,800, downsample by 7 to 50,400, upsample by 5 to 252,000, downsample by 3 to 84,000, upsample by 4 to 336,000, and finally downsample by 7 to our desired final sampling frequency of 48,000. This has reasonable values for intermediate sampling rates, and we know how to do even better by using three efficient polyphase structures for rate changes of 8/7, 5/3, and 4/7. We keep in mind that the polyphase structure is really a "recipe" for doing only what is necessary rather than a filter network.

As a second example, consider a downsampling from 22,050 Hz to 8192 Hz. Here we assume that the signal was bandlimited (even at the 22,050 rate) to 4096, OR we are going to filter it to a bandwidth of 4096 Hz. In the first example, the rate change was not that far from unity (480/441) while here it is more like 1/3 (8192/22050). The ratio 8192/22050 gives up only a common factor of 2 to 4096/11025. The numerator 4096 is of course 2^{12}, while 11025 factors into 3 x 3 x 5 x 5 x 7 x 7. Fig. 2 shows two ways (of many!) to arrange a cascade of sample rate changers for this case.

The point in looking at these examples is that when the rate change ratio involves large integers, both the direct polyphase, and the cascade of polyphase realizations at ratios for smaller integers get tedious. That is, the “bookkeeping” gets out of hand.

THE RATE-CHANGE COMPUTATION CYCLE

In our intuitive polyphase [1], we introduced the notion of a Rate-Change-Computation-Cycle (RCCC) as a means of identifying the component filters for the polyphase structure. Fig. 3 shows an example of the RCCC for a 4/3 rate changer. Here the input signal is represented by the four samples A, B, C, and D located at times 0, 1, 2, and 3. Because we want to increase the sampling rate by 4/3, we need to find samples for the output that are spaced at an interval of 3/4. We start with the sample A at time 0 and place there a sample a=A at time 0 for the new sequence. The next sample of the new sequence needs to be at time 3/4 – for example, the sample b as shown. Yet another sample of the output would be needed at time 2 x 3/4 = 1.5, and this is the position of sample c. Likewise we get sample d at time 2.25. Then we get to one more sample at time 3, and this is the position of sample D of the original sequence and sample e of the new sequence. The RCCC is composed of the calculation of samples a, b, c, and d (shown as stars), but not of sample e which is the first sample of the next cycle. Note that the dashed vertical lines identify the time positions of the RCCC output sequence. In this example, we suggest that the output samples are to be established as a linear interpolation of the input samples on either side of the needed positions. This is a very simple method of interpolation, but we could well employ a more sophisticated technique in a particular case. The point is that the RCCC is short in this case.

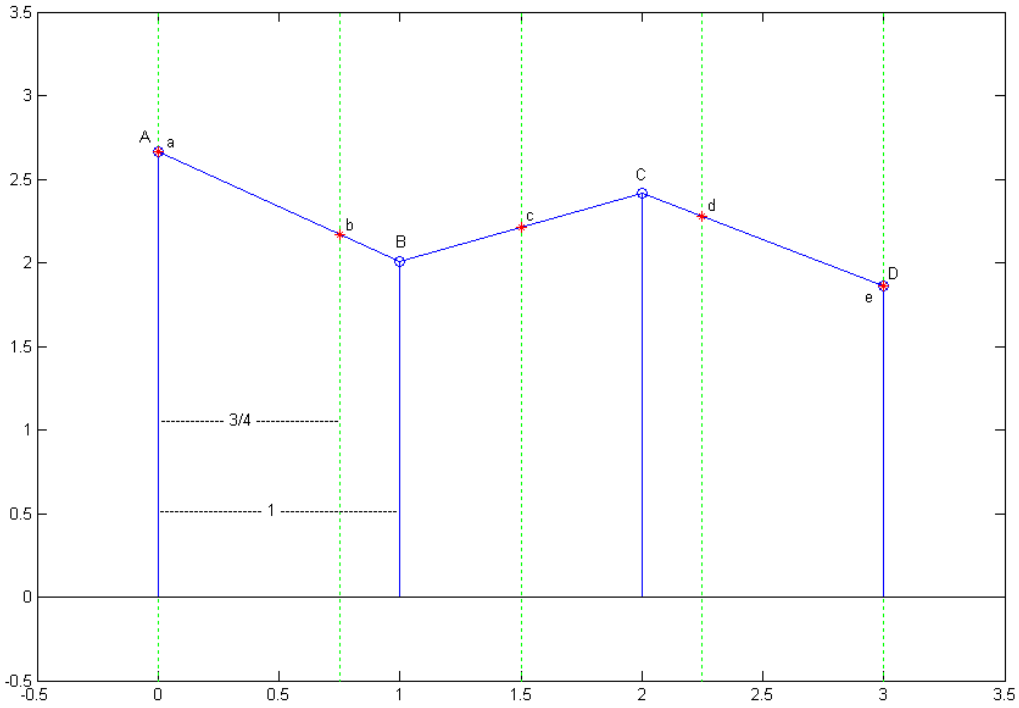


Fig. 3 The RCCC for a 4/3 upsampler (originals=o, new=stars)

In our intuitive approach, we used this RCCC as a means of determining the component polyphase filters. We saw that the output sample a was derived trivially from the input sample A by a one-tap (first polyphase) filter with weight 1. The sample b was obtained with a second polyphase filter which gave a weighted sum of 3/4 of B and 1/4 of A (assuming our linear interpolation model). The sample c was 1/2 of B plus 1/2 of C for the third polyphase filter. The fourth polyphase gives d as 3/4 of C plus 1/4 of D. This completes the RCCC as we would then start a new cycle, obtaining e directly from D, and so on. Note that there are 4 output samples for every three input samples, as required. The number of polyphase filters is the numerator of the ratio, and here it is 4, a manageable number it would seem. The numerator (3 here) gives the length of the original sequence involved.

Fig. 4 shows another RCCC example, this time for a ratio of 11/14, so it is different in that the integers are larger, and in that the sampling rate is decreased. This shows that we would need a total of 11 component polyphase filters over a range of 14 original samples. At this point, the polyphase approach is getting tedious.

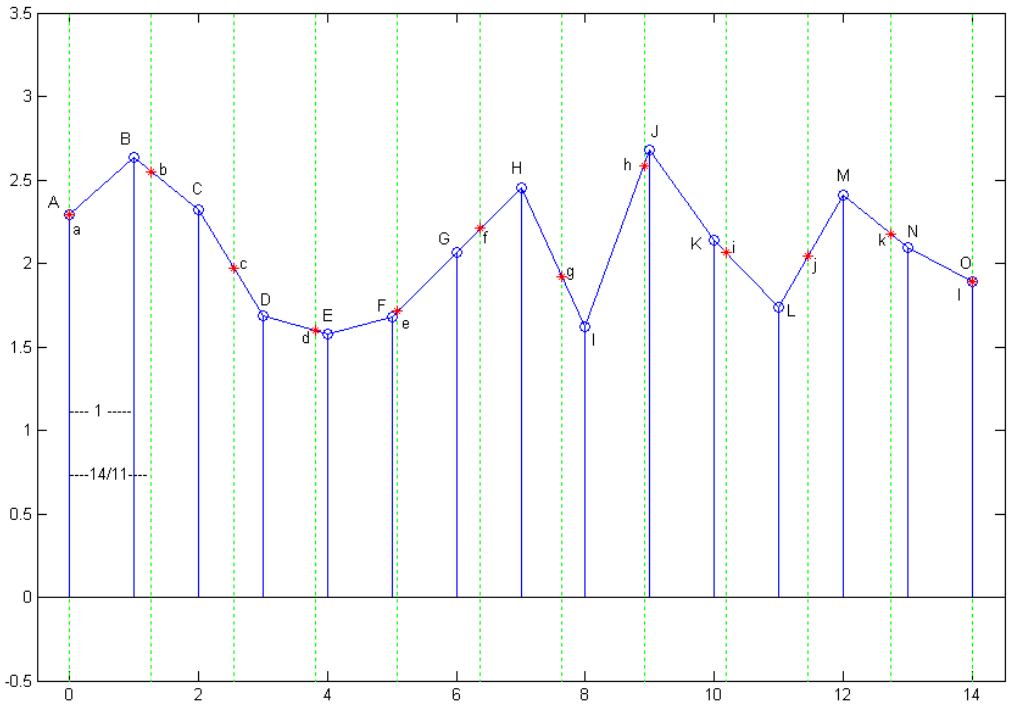


Fig. 4 *RCCC for an 11/14 rate converter*
(input sequence mildly low-passed)

In our next example, we have a rate change that is a decrease from 22050 to 8192 as discussed earlier. This is a ratio of 4096/11025, so there would be a total of 4096 component polyphase filters over an input length of 11025! Fig. 5 shows a small portion of the RCCC for this case. Note that the spacing of output samples is 11025/4096 which is roughly 2.7, but it does take 4095 output samples before we get back to the case of using an actual input directly (not shown!).

In the example of Fig. 5, we have already cheated a bit. This is because we have produced our test sequence not as just a random sequence, but as a low-passed random sequence. It is clear that in going to a lower rate, we have to worry about whether or not the signal is sufficiently bandlimited to be supported at the lower rate. Clearly, the random signal is not – we assume a white spectrum. Accordingly, the signal was low-pass filtered for a cutoff frequency of (1/2)(8092/22050)=0.1835 times the sampling rate. This was done with a length 51 filter designed by Matlab's *firls* function with the frequency response as seen in Fig. 6.

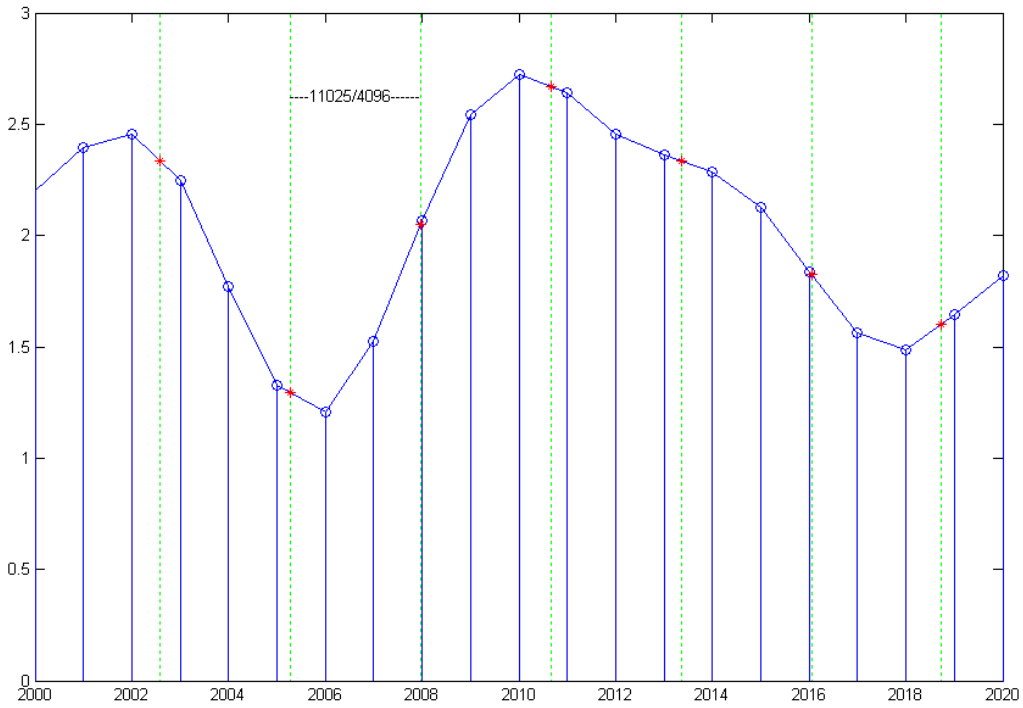


Fig. 5 A small portion of the RCCC for a 11025 to 4096 rate converter.

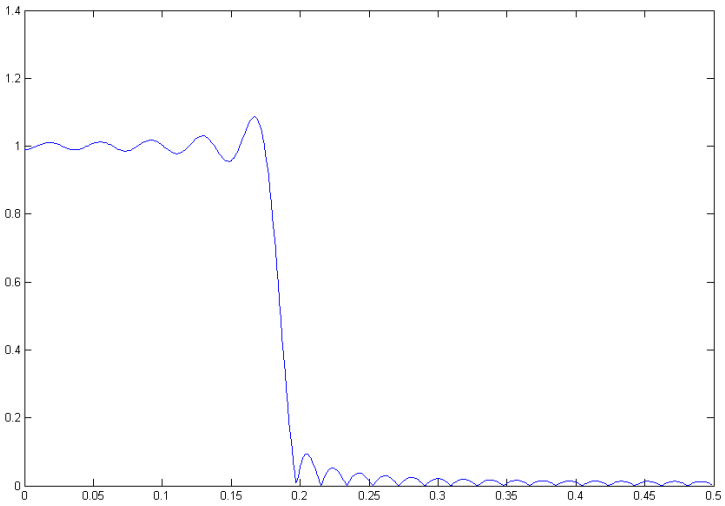


Fig. 6 Filter used for random sequence of Fig. 5

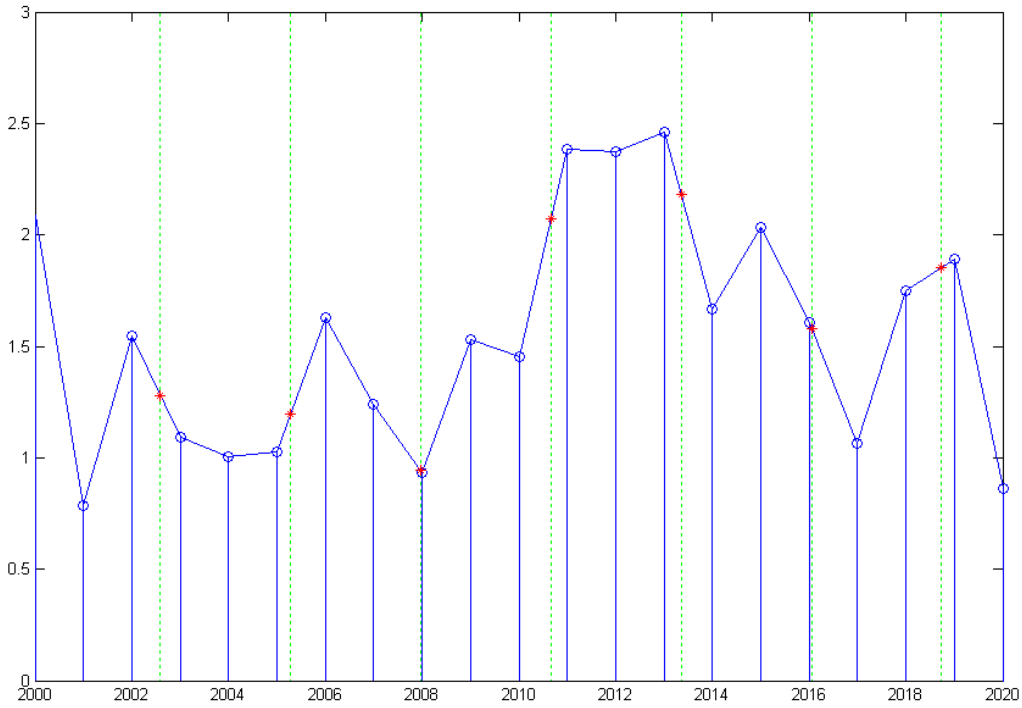


Fig 7 Rate change without predecimation filtering

Fig 7 shows what happens if we don't use the predecimation filtering that was used in Fig. 5. Both Fig. 5 and Fig. 7 show output samples (stars) on lines between two input points (stems), but there are also points that are not involved in any linear interpolation (for example, the points at 2004 and 2012). In Fig. 5 we see that the curve is relatively smooth and and unused points are not too important. In Fig. 7, we see that the unused points may well represent higher frequency information that is never used. That is, we have aliasing.

The situation here is a bit strange in that we are accustomed to doing rate changing in cases where a single filter acts as both an interpolation filter and a predecimation filter (where necessary) [3]. Here we have an interpolation filter – the linear interpolator being essentially low-pass. Why does this filter, not all by itself, provide the predecimation? The answer is because the linear interpolator is acting at the higher rate, so its cutoff is well above what is required. We need the proper cutoff of 0.1835. So, why do we not just use the predecimation filter as an interpolator? Well, in some sense we are, but the problem is that the target locations for the interpolated samples were not output times of the predecimation filter. Hence we use the linear interpolator to help out with the low-pass filtering, and to exactly place the output samples.

POLYPHASE ON DEMAND

Above we saw that rate changes at ratios of small integers using a polyphase approach were practical. As the ratio becomes one of larger and larger integers, the number of polyphase filters (and the RCCC) gets impractically large, and we arrived at the notion that instead of precomputing the filters, we might best just compute them "on the fly" for each interpolated sample. Much as we might think of an irrational ratio as an extension of ratios of very large integers (and vice versa), the irrational case can be handled by "polyphase on demand" (POD). This is straightforward. We just give up the notion that we are using the POD approach because the RCCC is very long, and accept that we are using it because the RCCC becomes infinite. One thing that is different in the infinite case is that there is no prospect of interleaving the various polyphase filters into an overall interpolation filter, although, perhaps, this would amount to a continuous-time filter.

The POD technique is probably obvious from the discussion above, so here, we will just be giving some example Matlab programs. The first program is the one that generates figures such as those in this note above. The second actually converts a trial file.

PROGRAM 1 A FIGURE-GENERATING PROGRAM

```
function rcccfig(N,M)
% rate change by N/M
%     N = New Rate
%     M = Original Rate

x=rand(1,M+1+60)*2+0.5;
% random test signal
% - made 60 longer to accommodate length 51 filtering
%      when filtering is needed

if M>N     % do we need pre-decimation filter ?
   h=firls(51,[0 N/(2*M) N/(2*M) .5]*2,[1 1 0 0])
   % h=[zeros(1,25) 1 zeros(1,25)]  %bypass above filter for study if desired
   x=filter(h,1,x);
end

%  Now take length M after transient
x=x(55:M+55);
```

```
figure(1)
% Original samples separated by 1
plot([0:M],x)
hold on

% New samples to be calcualte at intervals M/N
% vertical lines
for k=0:M/N:M
  plot([k k],[-2 4],':g')
end
stem([0:M],x)

k=1;
y(1)=x(1);
for s=M/N:M/N:M
   k=k+1;
   lw=floor(s);
   up=ceil(s);
   offset=s-lw;
   y(k)=x(lw+1)*(1-offset)+x(up+1)*offset; % linear interp.
end
kk=k;
plot([0:k-1]*(M/N),y,'*r')
axis([-.5 M+.5 -0.5 3])
hold off


figure(2)
plot([0:M],x)
hold on
for k=0:M/N:M
  plot([k k],[-2 4],':g')
end
stem([0:M],x)
plot([0:kk-1]*(M/N),y,'*r')
axis([-.5 M+.5 -0.5 3])
hold off
if M>20
    axis([-.5 20.5 -0.5 3])
end


if M>N
figure(3)
plot([0:.001:.499],abs(freqz(h,1,500)))
end
```

PROGRAM 2 RATE-CHANGER OF AUDIO SIGNAL

This program converts from a 22050 Hz rate to a 8192 Hz rate using polyphase on demand. The test file (x) for this program is generated by the program, but the same code can be used for an actual audio file. The output file is y. The two sould virtually the same in both pitch and quality.

```
t1=(0:22049)/22050;
delt=1/22050;

x=sin(2*pi*500*t1);    % test file here
% This signal is effectively bandlimited since its frequency is
%   500 Hz with a 22050 Hz sampling rate.
% A filter such as used in Program 1 can be used where necessary.

% plot a small segment and listen to the entire signal
figure(1)
plot(t1(200:400),x(200:400),'.')
sound(x,22050)
pause


%determine times for new samples
t2=(0:8191)/8192;
y=zeros(1,8192);


for n=2:8192
  time=t2(n);
    nlow=floor(time*(22050));
    offset=time-nlow/22050;
    L=x(nlow)*(delt-offset)+x(nlow+1)*(offset);
    y(n)=L*22050;
end


figure(2)

% plot small segment and listen to new result
plot(t2(100:200),y(100:200),'.')
sound(y,8192)
```

BEYOND LINEAR INTERPOLATION

The presentation of POD above was done using linear interpolation, which was simple to envision and implement. Nothing prevents us from using a higher order polynomial (linear interpolation being a first-order polynomial) or other methods of interpolation. For example, we know how to fit a cubic polynomial to four consecutive points (as in Fig. 8). A third-order cubic polynomial should be of the form [2]:

$$y(t) = at^3 + bt^2 + ct + d \qquad (1)$$

and we suppose we have four given points at t=0, t=1, t=2, and t=3. The actual choice of times is arbitrary as far as the final result is concerned. In this case:

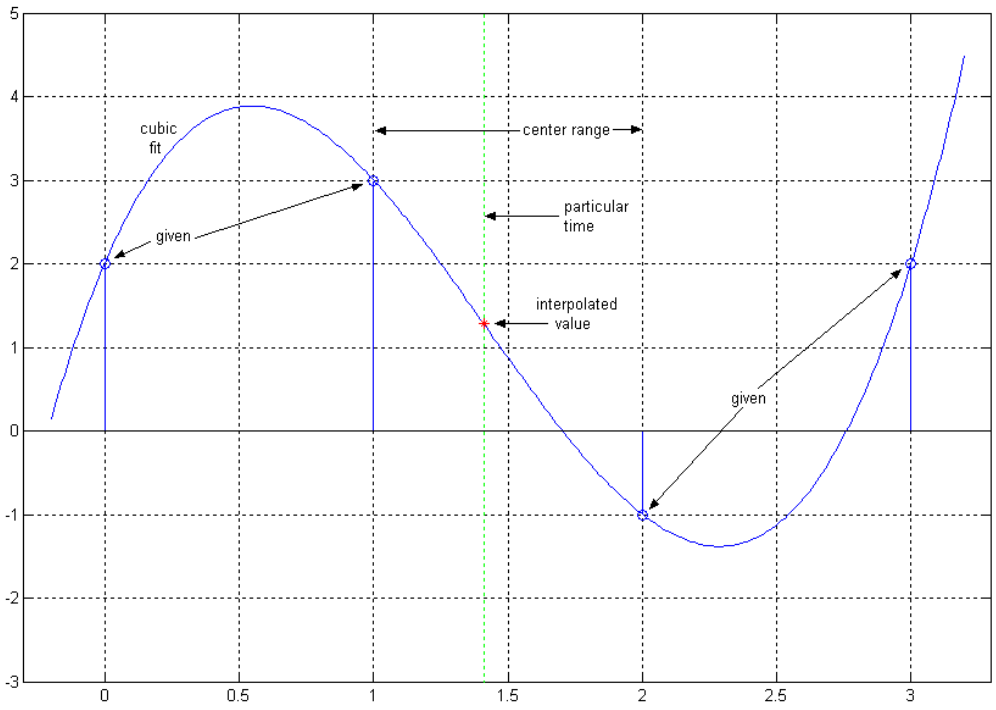


Fig. 8 A cubic polynomial can be fit to four points, as in this example. The usual practice is then to choose value of the polynomial between the second and third points (between 1 and 2 here). Note that the result for this middle region resembles linear interpolation.

$$y(0) = d \quad (2a)$$

$$y(1) = a + b + c + d \quad (2b)$$

$$y(2) = 8a + 4b + 2c + d \quad (2c)$$

$$y(3) = 27a + 9b + 3c + d \quad (2d)$$

Solving these, we get:

$$a = (-1/6)y(0) + (1/2)\,y(1) + (-1/2)\,y(2) + (1/6)\,y(3) \quad (3a)$$

$$b = y(0) + (-5/2)\,y(1) + 2\,y(2) + (-1/2)\,y(3) \quad (3b)$$

$$c = (-11/6)\,y(0) + 3\,y(1) + (-3/2)\,y(2) + (1/3)\,y(3) \quad (3c)$$

$$d = y(0) \quad (3d)$$

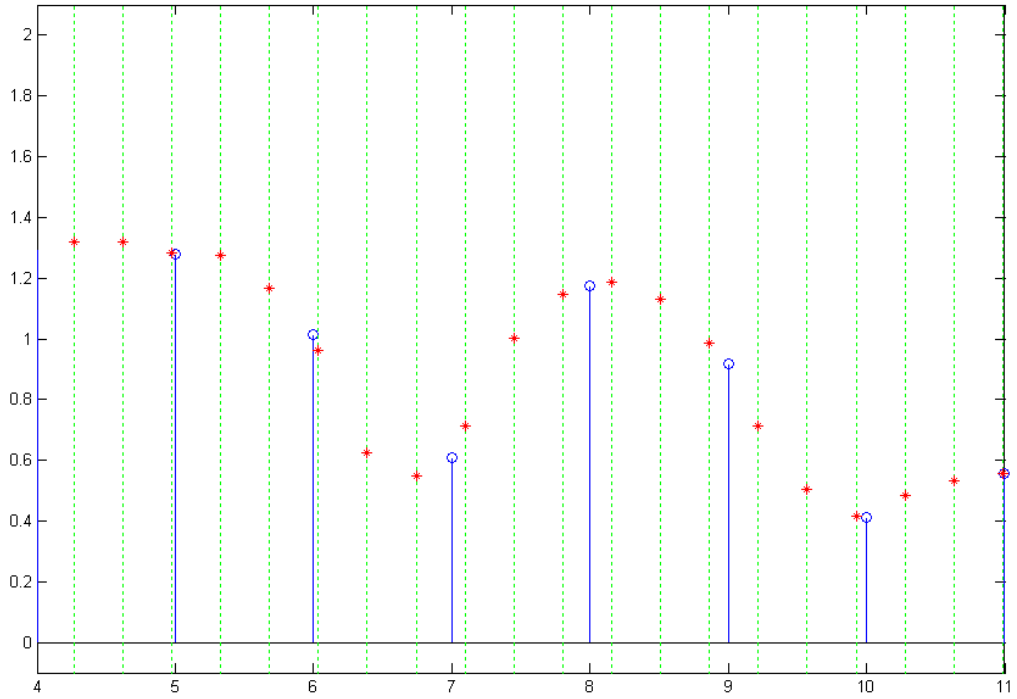


Fig. 9 Example of Cubic Polyphase on Demand

This gives us [substitute back to equation (1)] the polynomial corresponding to the four points, and we can then solve this for any points we need.

Fig. 9 shows a cubic interpolation for a rate change of 14/5. The vertical dotted lines represent the times at which we need to calculate samples, while the stems are the given samples. Note that there are two or (more generally) three vertical lines between each pair of given samples. Between 5 and 6, there are only two points. The average is of course 14/5 =2.8. For each of these times, we look at the samples on either side, plus an extra sample on either side, calculate the coefficients a, b, c, and d, and then calculate the polynomial value with the parameter t being the position + 1. Thus, each of the values represented by stars corresponds on an instance of Fig. 8. Note that we only use the center third of the cubic polynomial fit. Once we calculate the interpolated values for the middle region, we move into the next region, discarding a given point further back and now adding another point moving forward, thus having a different polynomial.

The example shows has given samples chosen at random, and then slightly bandlimited with a length-two moving average, so the spectral content is still relatively high. Thus we notice that the cubic interpolation is in general quite different from the linear interpolation cases (straight lines between the given points).

REFERENCES

[1] B. Hutchins, “An Intuitive Approach to Polyphase Rate Changing,” Electronotes, Vol. 21, No. 203, Jan. 2004

[2] B. Hutchins, “Polynomial Fitting for Sample-Rate Changing at Rational and Irrational Frequency Ratios, Electronotes Application Note AN-317, Jan 1992.

[3] B. Hutchins, “Sample Rate Changing and the Corresponding Effects on Normalized and Unnormalized Spectral Descriptions,” Electronotes, Vol. 18, No. 185, Feb. 1995 (see in particular, Example 4, pp 23-25)

Recovery From Bunched Sampling

- by Bernie Hutchins

1. INTRODUCTION

Previously [1-3] we have looked at a particular case of non-uniform sampling, often called “bunched sampling” (also interleaved sampling [4,5] and sometimes “second-order sampling” [6]). This class of non-uniform sampling examples was comprised of cases where sampling “opportunities” were all equally spaced but where samples were not taken at all such opportunities. Instead, the samples that were taken were in well-defined selected patterns, and these patterns repeated exactly. For example, such a sampling situation would be the case where we take two sampling opportunities and then skip two opportunities, and so on. This we have described previously as a “sampling cell” which would be s=[1 1 0 0] for this example. Thus the “bunch” repeats every four samples.

It is natural to think of a sampling cell such as s=[1 1 0 0] as being composed of the superposition of s_a=[1 0 0 0] and s_b=[0 1 0 0], and this superposition idea is central to our analysis methods. Yet another view of the process is one in which these two components are not immediately superimposed. That is, we can look at the component samplings as “poly-phases” of the signal. Since we expect to recover the full sequence of samples (or, more comprehensively the full continuous-time signal perhaps) from the bunched samples by a filtering (alternative view of filtering: interpolation) scheme, we need to ask an important question. Are we looking to filter the combined components with one filter, or are we looking to filter the separate components, perhaps with different filters, and then combine the results (Fig. 1). Let’s take a closer look at the problem.

Suppose we start with a sequence x(n) known to correspond to a signal x(t) which is suitably bandlimited to some frequency f_B. This is the basis for most sampling discussions. For example, we know that with standard low-pass sampling, that if the samples x(n) are spaced a distance of T apart, that if T is less than $1/2f_B$ than an ideal low-pass filter (or sinc interpolation) leads to complete recovery. The basic ideas can be extended to bandpass sampling and to non-uniform sampling. In our example, we have suggested a length 4 sampling cell. This means that x(n) can be divided into four polyphase components. (Here we will keep the zeros in our notion of polyphase.) Accordingly, the four polyphases are:

$x_a(n)$ = …..x(-4), 0, 0, 0, x(0), 0, 0, 0, x(4), 0, 0, 0, x(8), ….. (1a)

$x_b(n)$ = …. 0, x(-3), 0, 0, 0, x(1), 0, 0, 0, x(5), 0, 0, 0, x(9), ….. (1b)

$x_c(n)$ = 0 (1c)

$x_d(n)$ = 0 (1d)

Here, the last two polyphases, $x_c(n)$ and $x_d(n)$, are all zero because of the sampling cell chosen. In general, they would be defined in the same way $x_a(n)$ and $x_b(n)$ are. Note that the result of sampling with s=[1 1 0 0] , which we can call $x_s(n)$, is:

$$x_s(n) = x_a(n) + x_b(n) \quad (2a)$$

which is a superposition:

$x_s(n)$ = ….x(-4), x(-3), 0, 0, x(0), x(1), 0, 0, x(4), x(5), 0, 0, x(8), x(9), ... (2b)

It can be emphasized that this particular superposition is not at all difficult to undo in the time-domain. (It may well be much more difficult to invert in the frequency domain.) In any cases, we are looking at the possibility of recovering x(n) from $x_s(n)$. If you wish, at this point, we can assume that the bandwidth of x(t) does not exceed 1/4.

Fig. 1 shows a view of the bunched sampling situation along with two possible views to recovering x(n) or x(t):

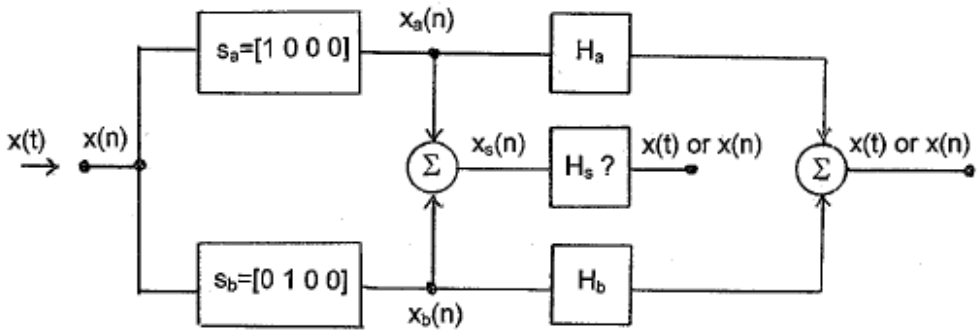


Fig. 1 Here a well-defined signal x(t) is sampled to give x(n) and two of four possible polyphases are separated out as $x_a(n)$ and $x_b(n)$. We want to inquire bout possible filters H_a, H_b, and H_s which allow us to recover x(n) or x(t)

In the figure, we indicate three possible filters (or time-domain interpolators) H_a, H_b, and H_s. The combined use of H_a and H_b as interpolation filters on the sequences x_a and x_b respectively, to return x(t), [and of course, thereby all the samples x(n)], is not totally unfamiliar. In much the same way we interpolate a sequence of equally spaced samples with a sinc function (ideal low-pass filter), we can separately interpolate the $x_a(n)$ and $x_b(n)$ samples with interpolation functions h_a and h_b that are neither sincs, nor identical, for H_a and H_b – although they look something like sincs, and H_a and H_b have symmetric impulse responses. Following these separate interpolations, we add the results. This is a standard procedure [4-7] and will be discussed further below.

The alternative procedure in Fig. 1, combining $x_a(n)$ and $x_b(n)$ to get $x_s(n)$, or just using the sampling cell s=[1 1 0 0], and following it with a single filter F_s is quite a different matter. Yet we do know something about recovering x(n) from $x_s(n)$ in the frequency domain. This is what we did in some previous presentations [2,3]. In these cases, we had the combined spectrum X_s corresponding to $x_s(n)$. Our investigations and demonstrations related these exactly by the DFT. Working from the spectrum, we broke it into segments with ideal filters. In general, these segments were then shifted in frequency, and combined with other segments using appropriate weights, according to a pseudo-inverse matrix [3]. It all worked. It should be clear however that this is not a filtering operation (because of the frequency shifts). That is, the "filter" F_s does not exist (it is not time-invariant) – except in the case where all samples are kept, in which case F_s is just our ideal low-pass filter (sinc interpolator).

2. THE INTERPOLATION FUNCTIONS

Because the notion of interpolation functions for non-uniform sampling are an extension of sinc interpolation, we will find it useful to review the uniform case. We shall use examples that can then be recalculated for the non-uniform case.

2a. UNIFORM SAMPLING AND SINC INTERPOLATION FUNCTIONS

Suppose we have a time sequence x(n) that is bandlimited to $f_s/4$, one quarter of the sampling frequency. We can recover the corresponding signal x(t) in several ways. First, we recognize that the signal bandlimited to $f_s/4$ is necessarily bandlimited to $f_s/2$, so it can be recovered by the ordinary sinc interpolation formula. In general [7] the formula is:

$$x(t) = (2f_cT) \sum_{n=-\infty}^{\infty} x(n) \{ \sin[2\pi f_c(t-nT)] / 2\pi f_c(t-nT)\} \quad (3)$$

Where T is the spacing of the samples ($T=1/f_s$) and f_c is the cutoff frequency of the ideal low-pass filter corresponding to the sinc.

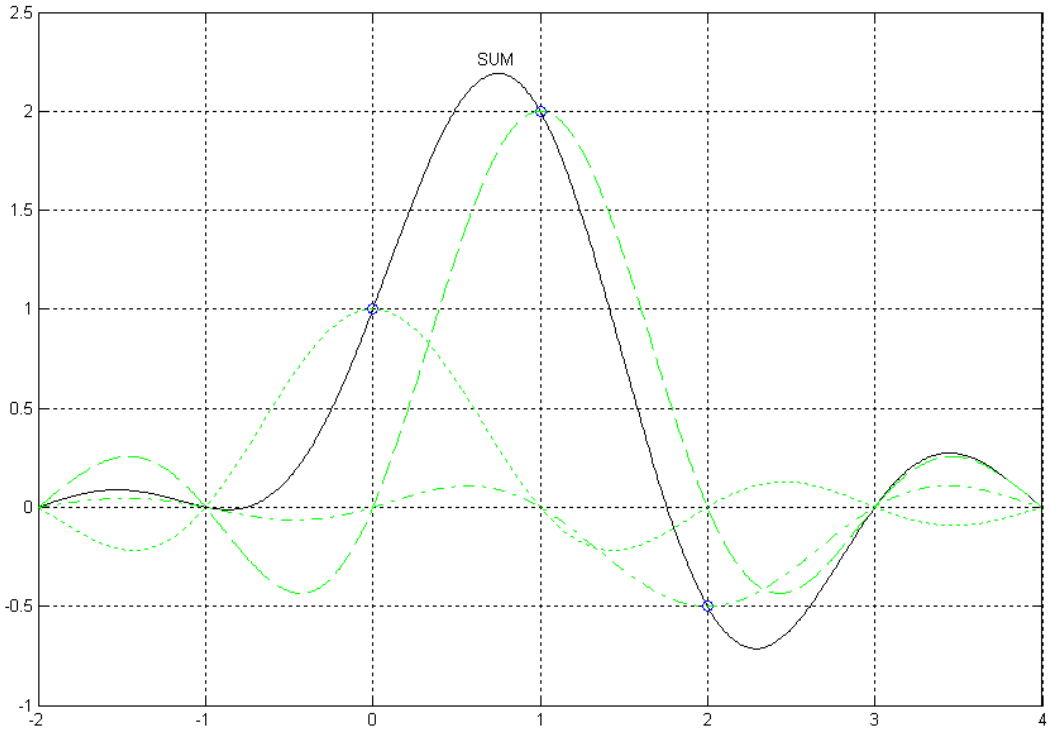
In the special case where $f_c=f_s/2$ this becomes:

$$x(t) = \sum_{n=-\infty}^{\infty} x(n) \{ \sin[(\pi/T)(t-nT)] / (\pi/T)(t-nT) \} \quad (4)$$

And in a second case where $f_c=f_s/4$, it would be:

$$x(t) = (1/2) \sum_{n=-\infty}^{\infty} x(n) \{ \sin[(\pi/2T)(t-nT)] / [(\pi/2T)(t-nT)] \} \quad (5)$$

A third method would be to discard every other sample (in one of two ways) and interpolate by a sinc corresponding to a cutoff $f_s/4$ (twice as wide). Both must give the same x(t).



<u>Fig. 2</u> Sinc interpolation. The solid line is the sum of the three weighted sinc functions. Note that the sum goes through the given samples and through the unspecified (assumed zero) samples at the integer points.

Fig. 2 is a reminder of how sinc interpolation works, illustrated here in the usual "text-book" form of just a few samples and a bandwidth of 1/2 the sampling rate. Note that the sum goes exactly through the original sample points, and is zero at all the non-specified integer points. Here we are convolving the sample points with the sinc, and thus we are multiplying the spectrum of the original samples with the Fourier transform of the sinc – ideal low-pass filtering. Examples such as in Fig. 2 can be thought of as telling us the bandlimited time function that corresponds to the samples. Further, we know we can play a lot of sinc interpolation games with different bandwidths and sampling rates [7].

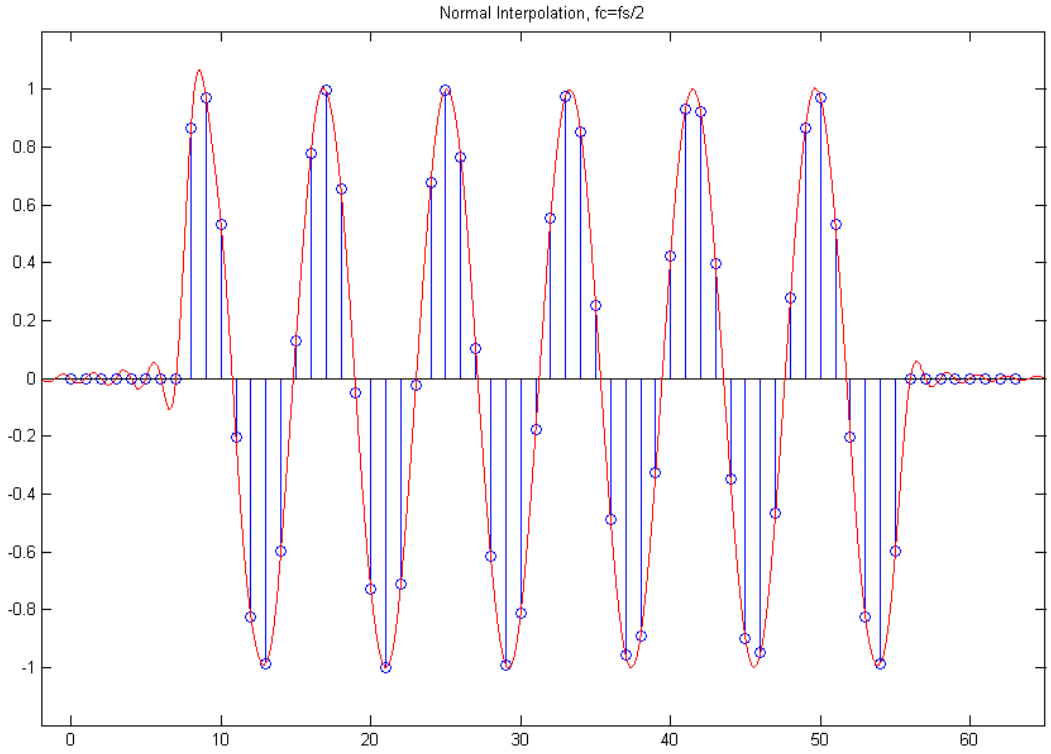


Fig. 3a *Standard sinc interpolation of a sinewave burst.*

In Fig. 3a through Fig. 3d, we show various samplings and interpolations of a "sinewave burst." We have chosen the frequency of the burst to be 1/8.2, a frequency less that 1/4 so in some sense, it is "bandlimited." However, it is not exactly bandlimited because it is timelimited (a finite length burst of about 6 cycles rather than an infinite number of cycles), so recovery is not perfect. The errors are very small near the middle of the burst, and are most prominent at the ends. The choice of a frequency of 1/8.2 rather that of 1/8, and the phase, was chosen specifically as a case which showed significant errors on both ends, so that the reader would not infer results better than those which are typical.

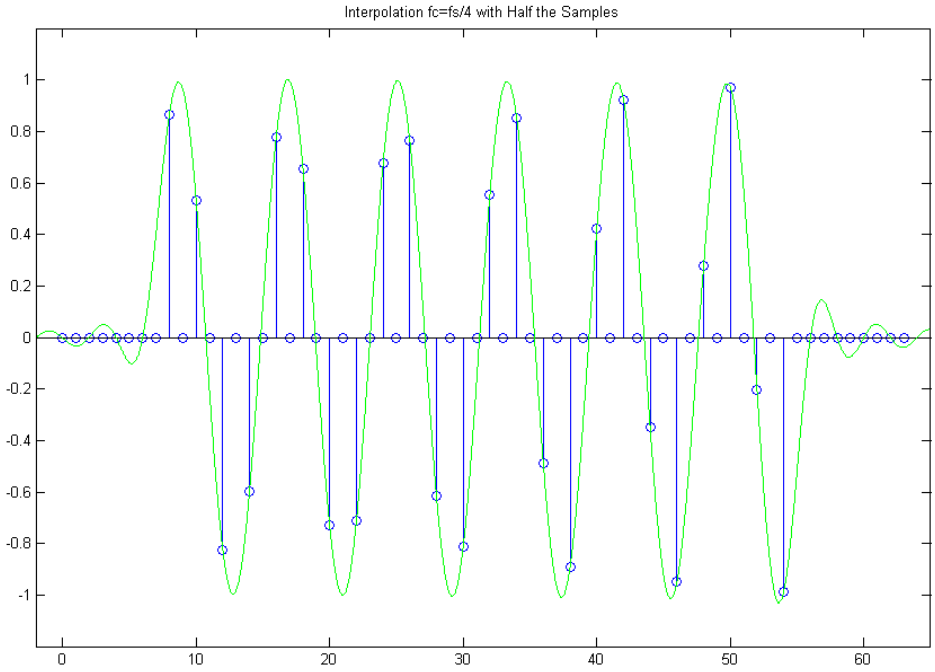


Fig. 3b *Here we keep every other sample, cutoff=1/4*

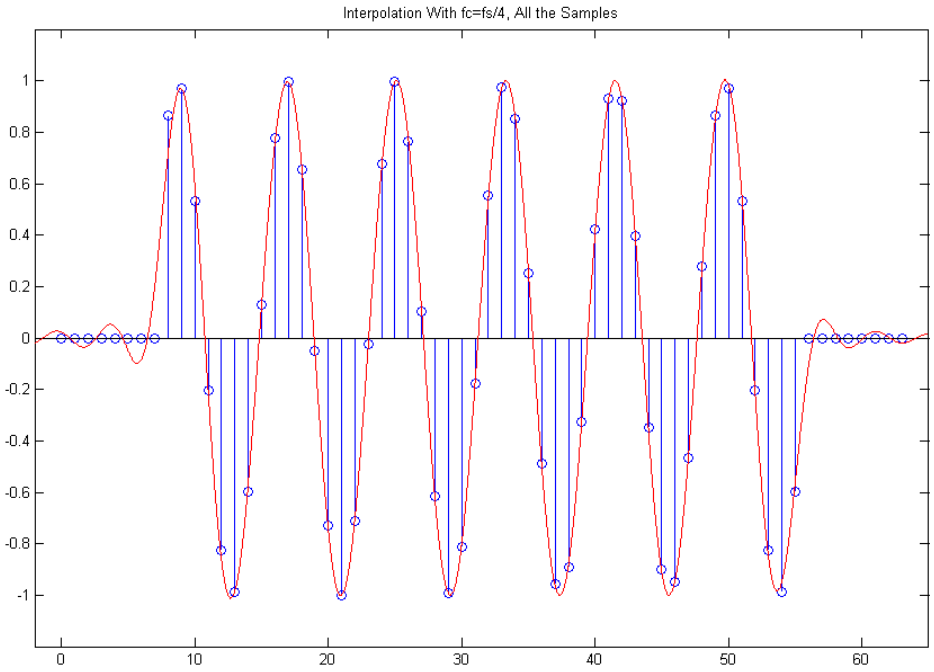
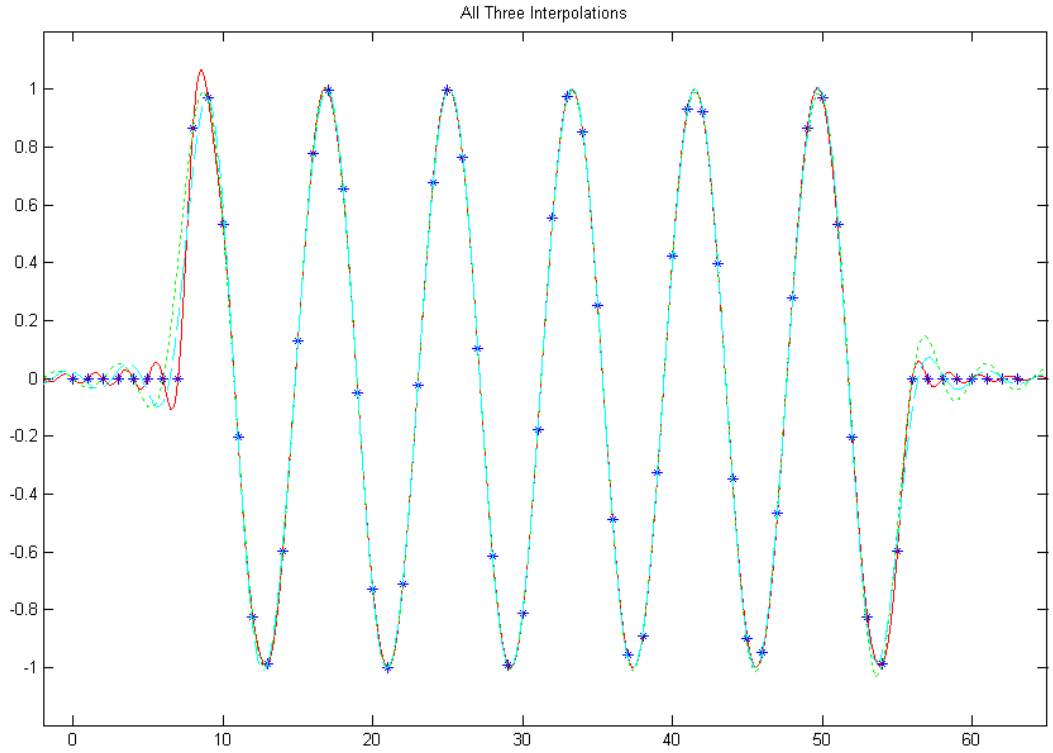


Fig. 3c *We can of course also keep all the samples and recover with a cutoff of 1/4.*



<u>Fig 3d</u> Here we show on one graph the interpolations from Fig. 3a, Fig. 3b, and Fig. 3c, with the original samples shown as stars. We note various end errors. The important thing however is that all three interpolations are virtually the same and near perfect in the center region.

Because the bandwidth of the sinewave burst is less than 1/4 for practical purposes, the situation of Fig. 3b where we have discarded half the given samples is still one where recovery is excellent, here using s sinc corresponding to a cutoff of 1/4. While Fig. 3b is logical enough, nothing prevents us from keeping all the original samples and still interpolating with the cutoff of 1/4, as is seen in Fog. 3c. All these things are normal examples of recovery from uniform sampling of bandlimited signals. By way of sammary, Fig. 3d shows the superposition of all three methods shown in Figures 3a, 3b, and 3c, and we note that all three are virtually identical in the center portion of the burst.

2b. THE INTERPOLATION FUNCTIONS FOR NON-UNIFORM SAMPLING

With non-uniform sampling, the interpolation functions are no longer sincs. Further, there will be different interpolation functions corresponding to each non-zero element of the sampling cell. We proceed here first by just using the published results of Bracewell [4] which lead to the following interpolation functions, also plotted in Fig. 4.

$$h_a(t) = \text{sinc}(2t) - \beta\, t\, \text{sinc}^2(t) \qquad (6a)$$

$$h_b(t) = h_a(-t) \qquad (6b)$$

$$\beta = \pi/\tan(\alpha\pi) \qquad \text{where } \alpha \text{ is the fractional offset } (\alpha = 1/4 \text{ for s=[1 1 0 0] }) \qquad (6c)$$

where sinc(x) is given as $\sin(\pi x)/(\pi x)$. Note that when $\alpha=1/2$ (equal spacing, every other sample), the interpolation functions become sinc(2t) as we expect.

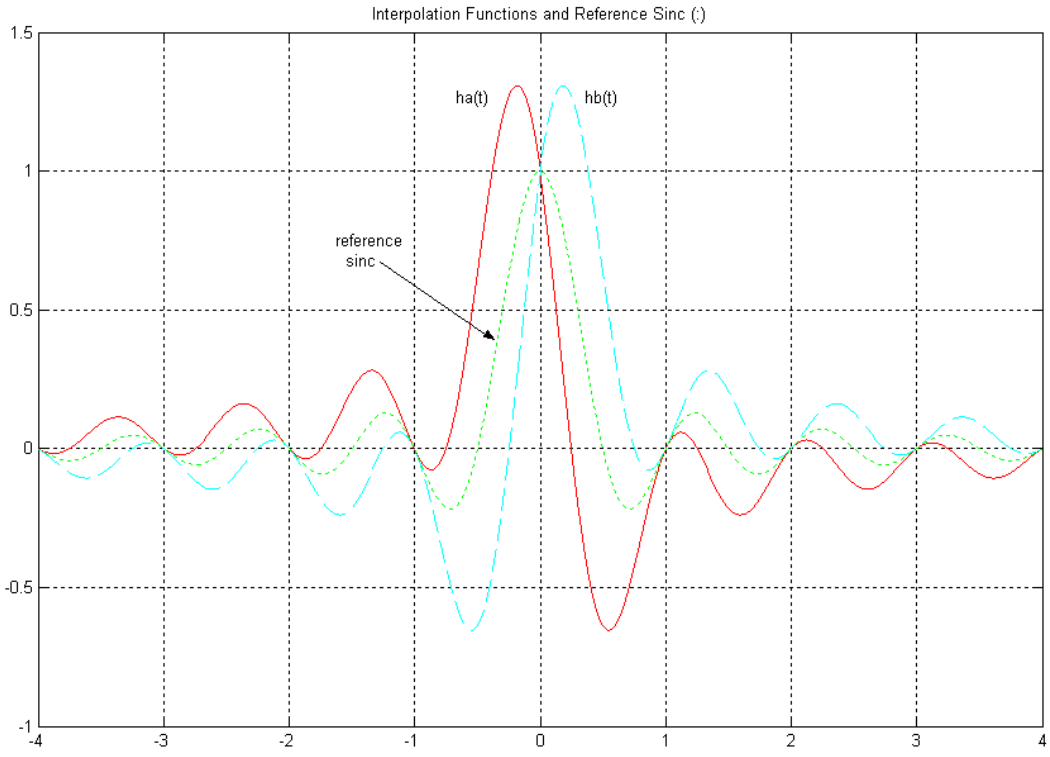


Fig. 4 Interpolation functions for the non-uniform case, along with the sinc function (cutoff 1/4) for reference.

In Fig. 4 we show the plots of $h_a(t)$ and $h_b(t)$ along with sinc(2t) for reference. Note that if the sampling cell were s=[1 0 1 0] we would use sinc(2t) as the interpolation function on all the samples (assuming the bandwidth is less than 1/4). Note that sinc(2t) is the first term in $h_a(t)$. When s=[1 1 0 0] we use $h_a(t)$ with the samples for s=[1 0 0 0] and $h_b(t)$ for the samples corresponding to s=[0 1 0 0].

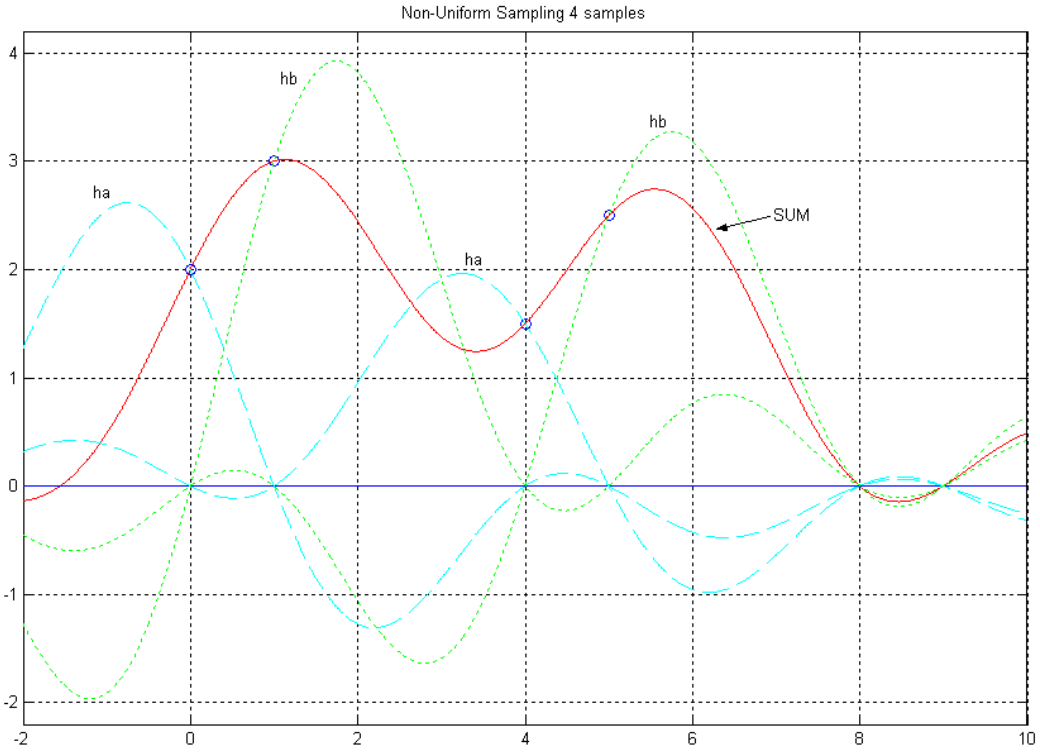


Fig. 5 A simple interpolation of four non-uniform samples.

Fig. 5 shows a case where we have only four non-zero samples total, at times 0, 1, 4, and 5 (open circles). We want to discuss the details of the interpolation, but note here that the result (solid line) goes through the four specified samples, and through zero at the unspecified samples (assumed zero) - times 8 and 9 seen here, and so on. The result does not go through zero at the other integers such as times 2, 3, 6, and 7, etc. The samples at times 0 and 4 are interpolated by functions $h_a(t)$, dashed lines while the samples at times 1 and 5 are interpolated by $h_b(t)$, the dotted lines. The solid line is the sum of these four non-zero components. Compare this result with Fig. 2.

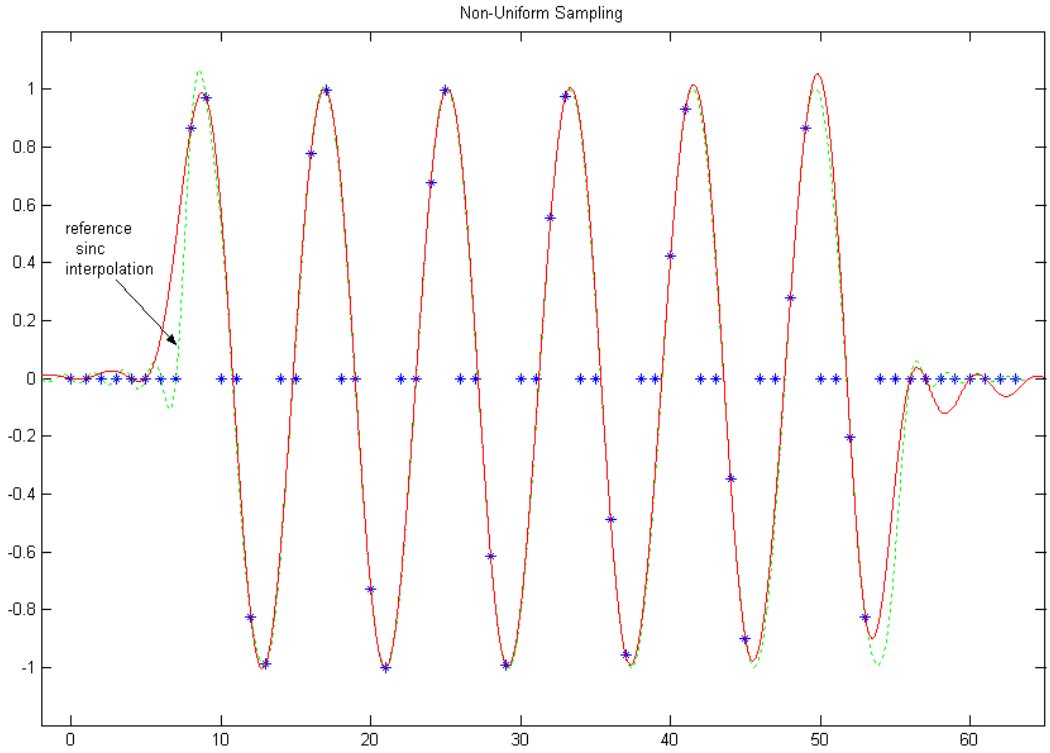


Fig.6 Interpolation of a non-uniform sampling of the sinewave burst.

Fig. 6 shows the exact same interpolation procedure that was used in Fig. 5, except here it is applied to all the samples of the sinewave burst, and not just to the four non-zero samples of Fig. 5. Alas shown for reference is the sinc interpolation of all the samples as the dotted line, which is the same as Fig. 3a. Note that as with the esamples of Fig. 3, the interpolated result is near perfect in the center of the burst, and we only see errors on the ends.

The solid line does in fact pass through all the given samples. This statement needs to be more precise perhaps. Clearly, it goes through all the non-zero samples, which come in pairs. It does not go through the zeros in general. We are not saying that these sample positions are zero, just that they are not defined, and are not involved in the interpolation. Now, outside of the burst, on both ends, where are samples that really are zero (such as positions 56 and 57, 60 and 61) and the interpolation does go through these zeros.

3. FINDING THE INTERPOLATION FILTERS

Above we demonstrated the interpolation using the interpolation functions (impulse responses) using formulas copied from Bracewell and Linden [4,5]. These two presentations give different derivations and different (but of course, equivalent) formulas. Here we want to give yet a third derivation. One overriding principle that many of you have heard from the current author is: "Never underestimate the value of knowing or suspecting the right answer." Here we have the right answer courtesy of Bracewell and Linden. We want to derive the impulse response of some filters. We already have these impulse responses given to us, but the published derivations are "brief". Let's start with the desired filters and use our well-understood filter design procedures. Fine. What is the right filter?

By a sort of "reverse engineering" we can use the Bracewell/Linden formulas and calculate the frequency response. That should give us a good idea what the right filter is! Using the formula(s) we can takes samples of the impulse response (Fig. 3) at intervals of 1/4 to give a FIR filter. We do this over an interval of -200 to +200. This gives us the magnitude response (Fig. 5) and the phase response (Fig. 6) as shown. We note that these are clearly half-band filters, and the phase response for h_a is $-\pi/4$ while that of h_b is $\pi/4$. Note that the phase response between 0.25 and 0.5 jumps rapidly back and forth by π. This is dues to the rapid zero-crossings of the response, but for practical purposes, since the response is very tiny, there is no real meaning to phase response for this region.

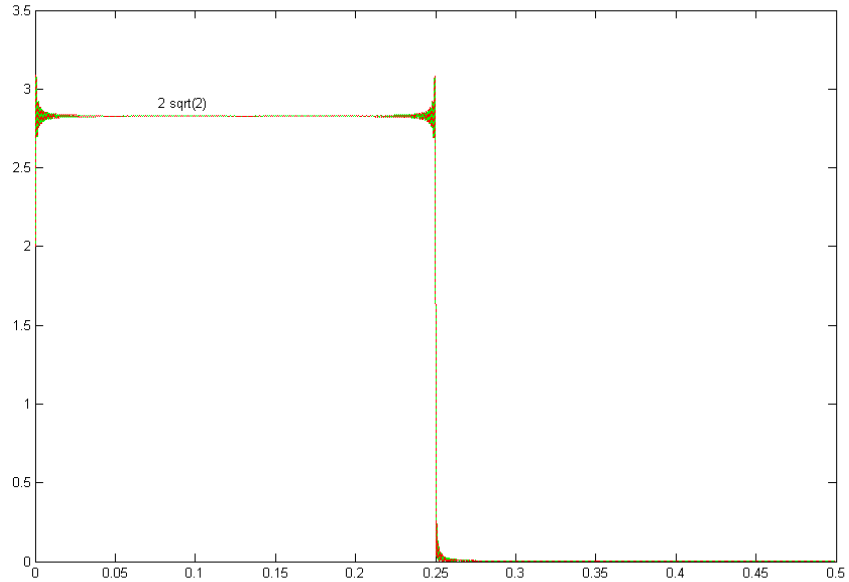


Fig. 7a The magnitude response of Bracewell's ha(t) and hb(t) are essentiallt half-band low-pass filters (cutoff is 1/4, or half of 1/2).

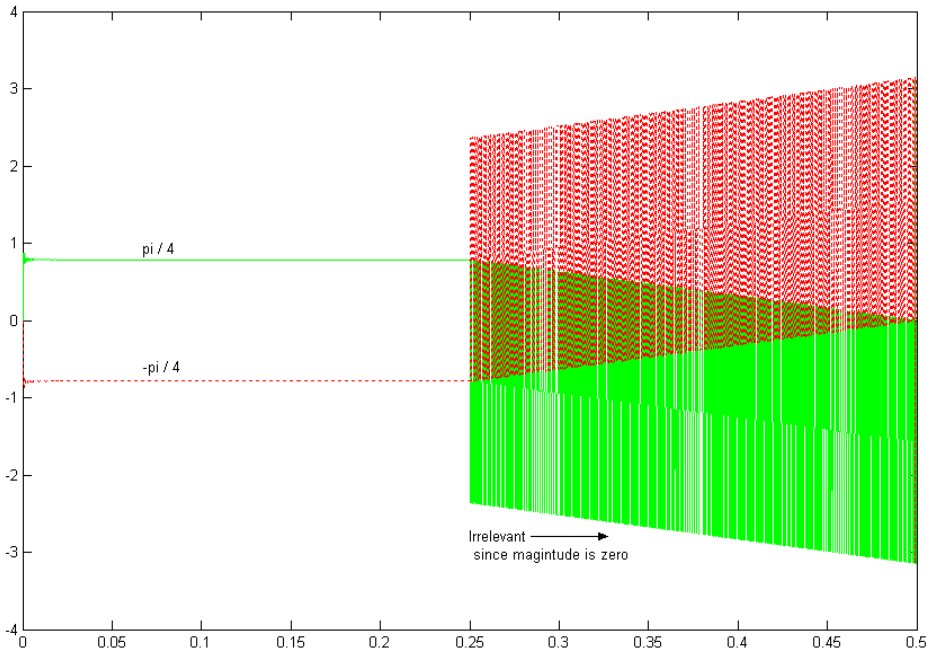


Fig. 7b The phase response of Bracewell's $h_a(t)$ and $h_b(t)$.

So now we know how to specify the correct filter. Our purpose here is not to find the impulse response, which we already have, but rather to see exactly what mathematical form comes out. Is it perhaps Bracewell's formula or perhaps Lindens, or perhaps something else? It's something else.

So now we know how to specify the correct filter. Our purpose here is not to find the impulse response, which we already have, but rather to see exactly what mathematical form comes out. Is it perhaps Bracewell's formula, or perhaps Lindens, or perhaps something else? It's something else!

We wish to obtain the continuous-time impulse response of a continuous-time filter which has the frequency response:

$$H_a(\Omega) = A(\Omega)e^{j(\theta(\Omega)} = A\, e^{j\theta} \qquad (7)$$

Where A is a constant value from $\Omega=0$ to $|\Omega|=\Omega_c$, and zero outside this range (thus it is an ideal low-pass). The phase θ is also constant, but since we want $h_a(t)$ to be real, the phase is an odd function of frequency.

The impulse response $h_a(t)$ is thus obtained as the Continuous-Time Fourier Transform of $H_a(\Omega)$:

$$h_a(t) = (1/2\pi) \int_{-\infty}^{\infty} H_a(\Omega)\, e^{j\Omega t}\, d\Omega$$

$$= (1/2\pi) \int_{-\Omega_c}^{0} A e^{-j\theta} e^{j\Omega t}\, d\Omega + (1/2\pi) \int_{0}^{\Omega_c} A e^{j\theta} e^{j\Omega t}\, d\Omega$$

$$= (A/\pi t) [\, \sin(\theta + \Omega_c t) - \sin(\theta)\,] \qquad (8)$$

where the integrals and simplification leading to the bottom line are straightforward.

The result does not look very much like equation (6a) but the proof is in the computing, and we can use Matlab (program below) to compute both formulas, and indeed, they give the same result. There is a need to change to the time variable t as shown, but the results are identical [equation (8) need's L'Hospitel's rule at t=0 so that $h_a(0)=1$].

```
t=-12:.01:12;
%  Bracewell - Equation (6a)
alpha=1/4
beta=pi/(tan(alpha*pi))
tt=t/4;
hbrace=sinc(2*tt) - beta*tt.*(sinc(tt).^2);
figure(1)
plot(t,hbrace,'r')
grid

% Equation (8)
theta=pi/4
Wc=pi/2
A=2*sqrt(2)
hnew=(A./(pi*t)).*(sin(theta+Wc*t) - sin(theta));
figure(2)
plot(t,hnew,'g')
grid
```

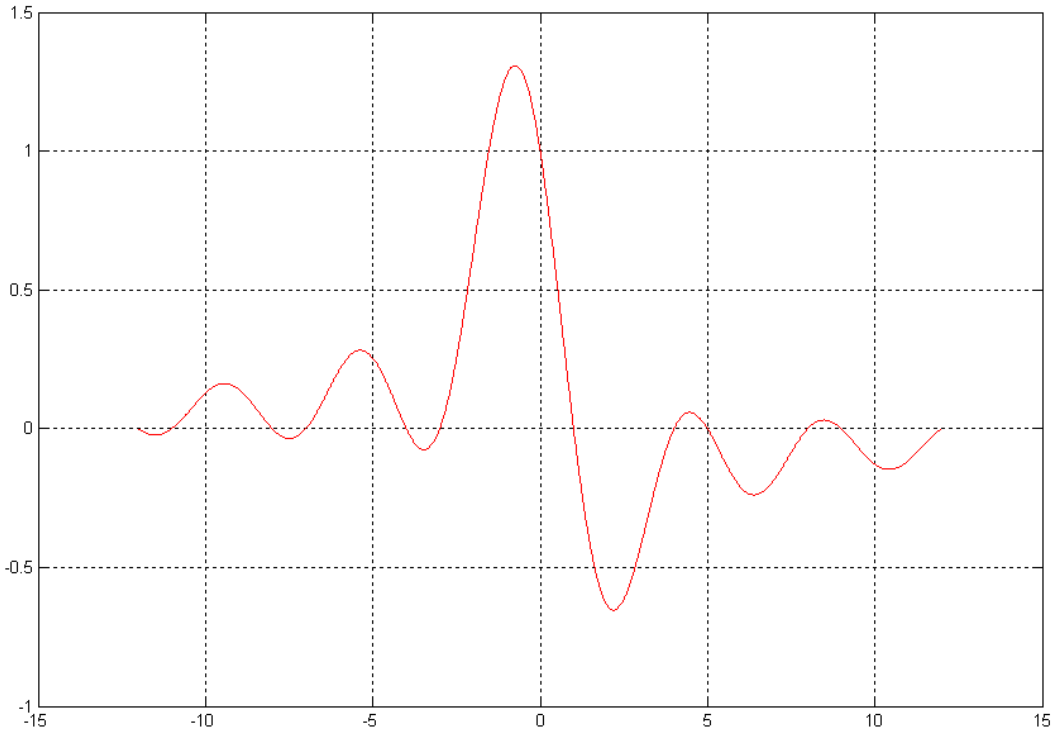


Fig. 8 Equations (6a) and (8) yield the same function for $h_a(t)$

So we see that our results converge. Equation (8) strikes us as a being a simpler formula. While we have mentioned Linden's result [5], we have not given his formula. It is:

$$h_a(t) = \{ \cos(2\pi t - \alpha\pi) - \cos(\alpha\pi)\} / [2\pi t \sin(\alpha\pi)] \qquad (9)$$

which is numerically the same as equation (8) and is easily converted to equation (8) by standard trig. So everyone agrees.

4. INTERPRETATION OF FILTERING – DFT VERIFICATION

We have so far not done all that much. We borrowed Bracewell's result and found the filter it implies, and shown that the impulse response of that filter is the same as the interpolation functions. The remaining question is, should we have been able to guess the correct filter without peeking at the answer first? We can answer this question by doing an experimental verification using Matlab. Here we first need to remember that we knew a lot about the spectra of the two polyphase components [2, 3] which were involved in our frequency domain approaches.

We will work here with a "traditional" triangular spectrum. While we know how to generate a time domain signal that has a triangular spectrum (a sinc^2) it is much quicker to just type in values for the DFT we want. Because the DFT is a discrete set of points in frequency, we naturally think of a "stem" plot, but here we want to represent the DTFT, so we use a continuous plot. Further, we use the triangle as an example because it is easier to see what is going on. Nothing here relies on the spectrum being triangular. In fact, once we work out our program code, we can easily substitute random (but bandlimited) frequency samples instead of the triangle, and have the program show that things still work (although it is hard to see). The actual program here uses a length 4096 DFT.

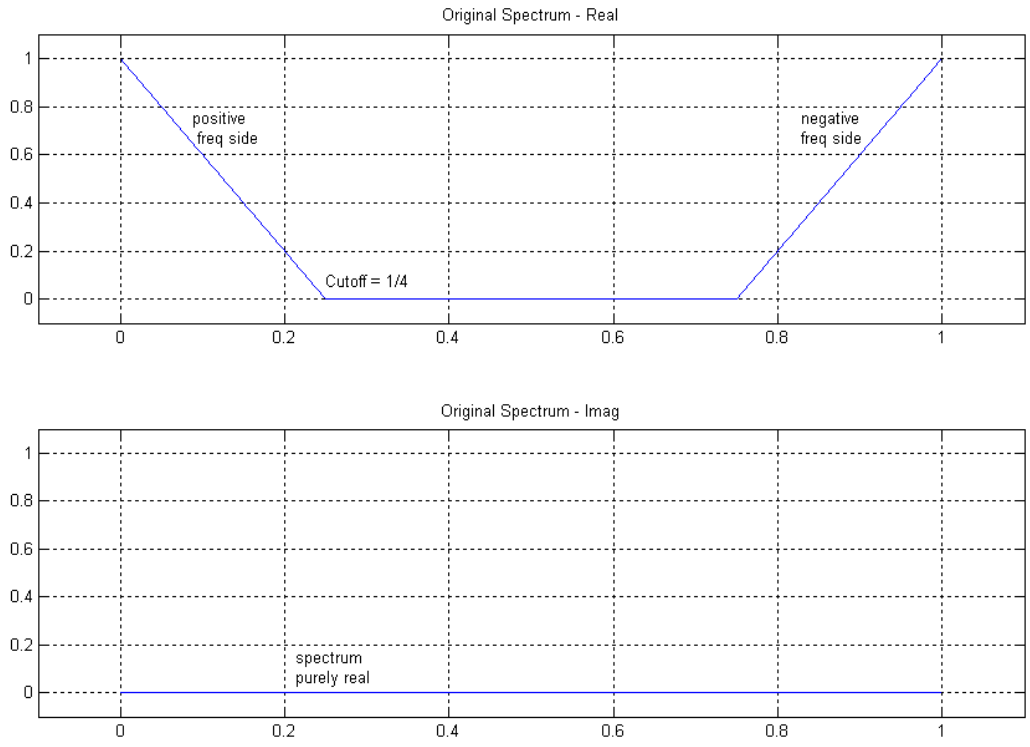
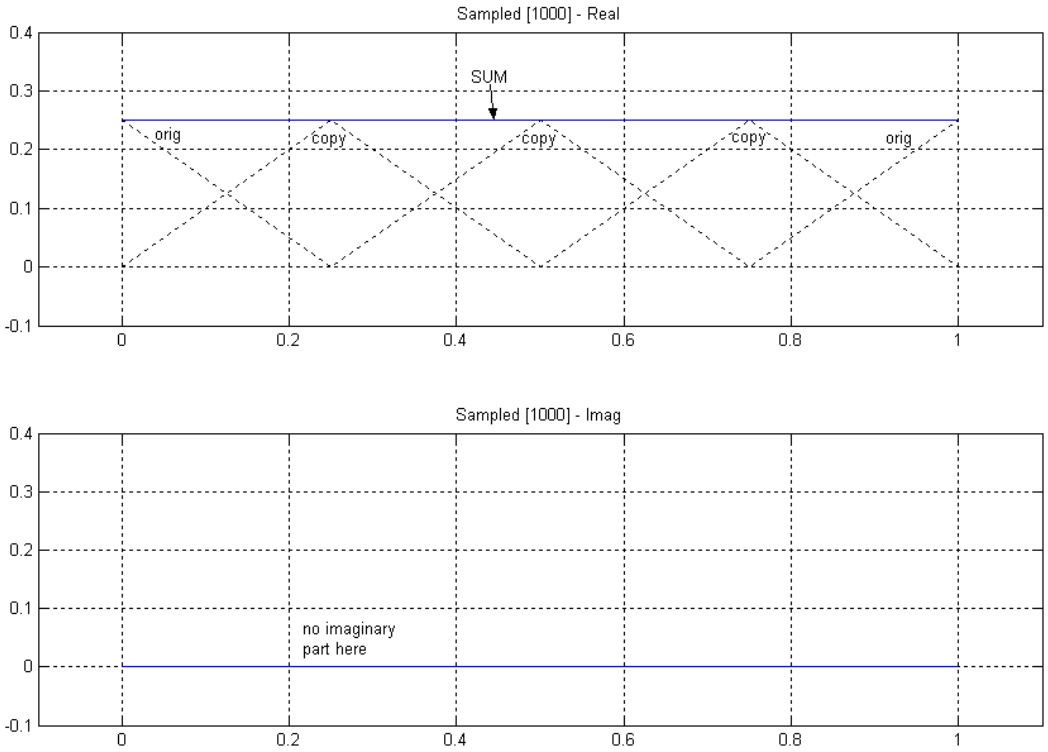


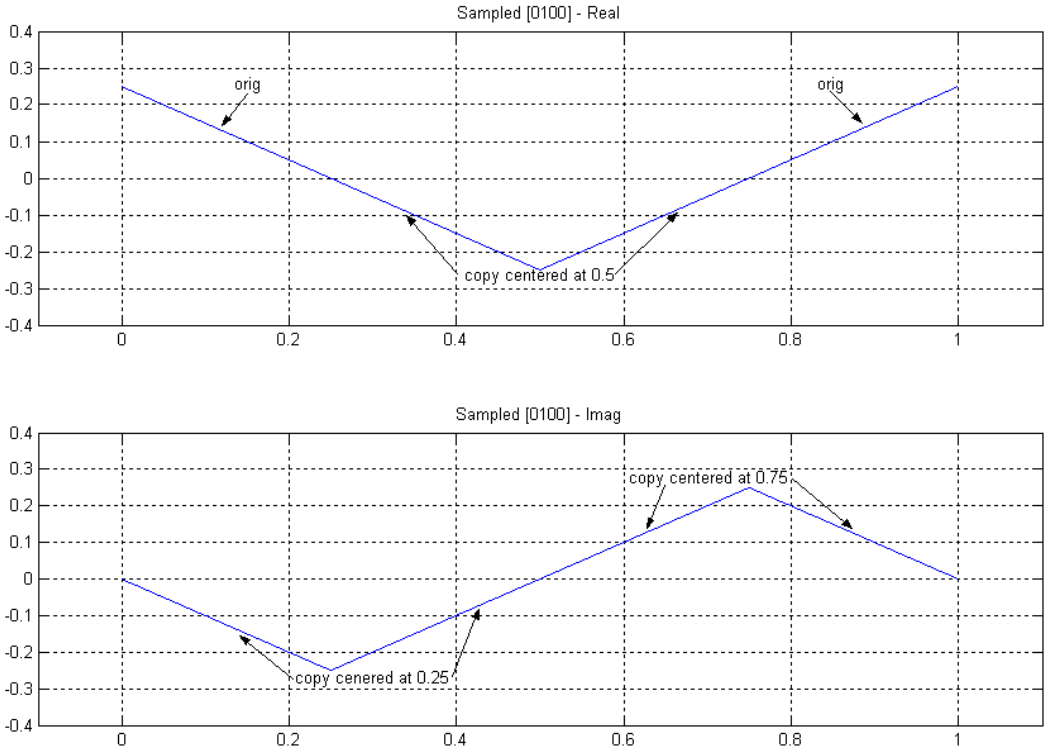
Fig. 9 An original triangular spectrum with a cutoff at 1/4 for a sampling frequency of 1.

Fig. 9 shows the starting spectrum, which we take to be purely real as shown. Note that the DFT of course represents frequencies from 0 to f_s=1, so the portion shown from 0.5 to 1 can also be thought of as the negative frequency side. Programming in the triangular spectrum, it is a simple matter to invert this to the time domain using the inverse DFT, and then we can samples the time domain sequence as we wish, and use the DFT to get the resulting spectra. Figures (10a) and (10b) show the spectra that result from the polyphase [1 0 0 0] and [0 1 0 0] respectively.



<u>Fig. 10a</u> By taking only every fourth samples, n=…-8, -4, 0, 4, 8, 12 …. two things happen to the original triangular spectrum of Fig. 9: we get copies at frequency multiples of 1/4, and the copies are smaller by 1/4. In the case of the triangular shape, note that the four copies add to a constant value of 1/4, purely real, for all frequencies.

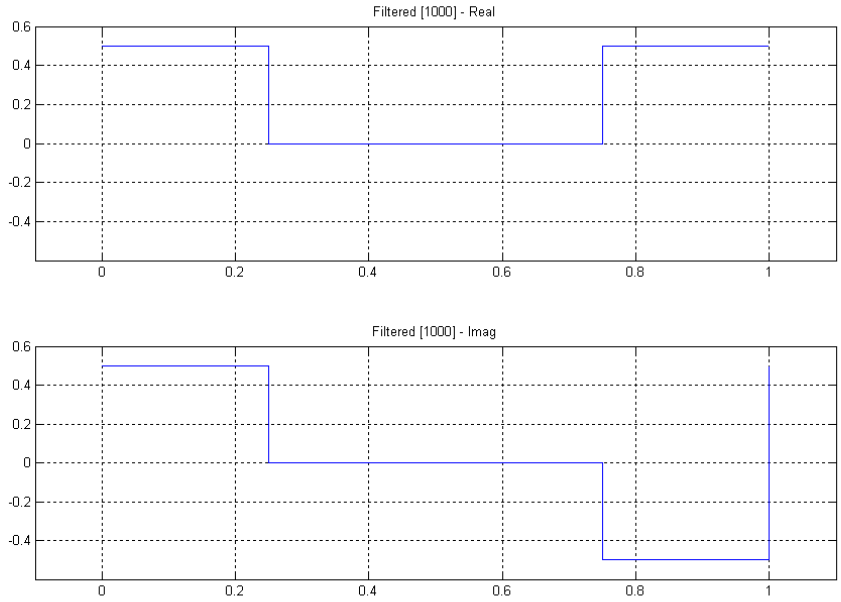
These two figures probably look quite different, so we need to makes some remarks about them. Fig. 10a, corresponding to samples that are integer multiples of 4, is purely real and a constant value of 1/4. This actually results from the superposition of four triangles (shown as dotted lines) in the figure. The spectral replicas are the result of the sampling (re-sampling) by a factor of 4, and the loss of amplitude is the result of the discarded samples (essentially a Parceval result). The result is of course “aliased” since a spectrum of width 1/4 is sampled at a rate of 1/4 (not of 1/2 as would be required by the sampling theorem).



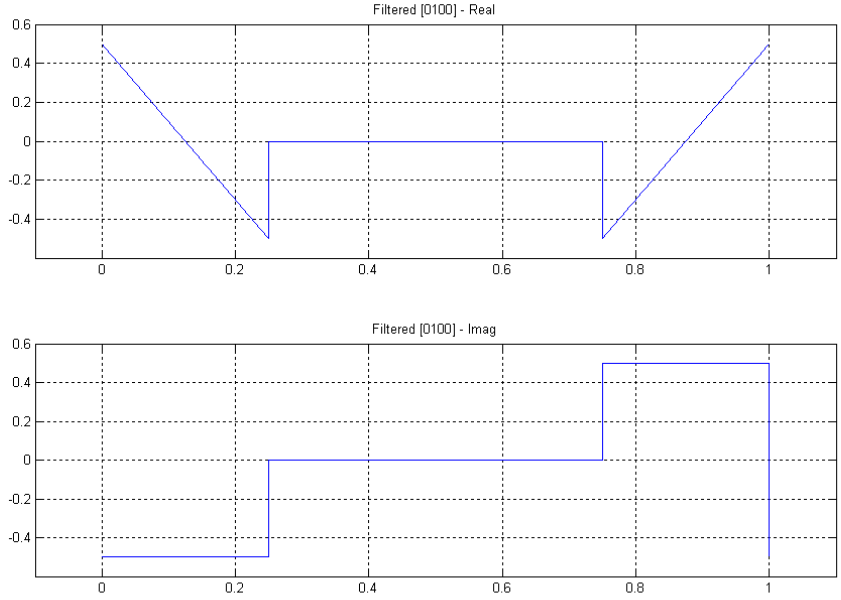
<u>Fig. 10b</u> As in the case of Fig.10a, when we take every fourth sample, this time for n=…-7, -3, 1, 5, 9….. we get four copies offset by frequencies of 1/4, and of magnitude 1/4. But now there us an imaginary part. The real part shows the original copy (from 0 to 1/4, and from 3/4 to 1) and the copy (inverted) centered at 0.5 (from 0.25 to 0.75). The imaginary part shows the copies centered at 0.25 and 0.75.

The polyphase that is offset by one sample, shown in Fig. 10b, is the same as Fig. 10a except for a different phase. It too consists of four copies, offset by 1/4 in frequency and attenuated to 1/4. The difference is that the copies rotate by -j so that the copy at 0 is real, but the one at frequency 1/4 is imaginary and negative, the one at 1/2 is negative, and the one at 3/4 is imaginary and positive. The magnitude of the spectrum is of course still 1/4 for all frequencies. An attempt at a two-dimensional representation appears in [2]. The spectrum of the superimposed polyphases is the sum of Fig. 10a and Fig. 10b of course, but we are not combining them here. Instead, each will be filtered by the appropriate filters.

Here is where we have a clue where the phase of $\pi/4$ might come from. We saw that the offset between polyphases resulted in a 90 degree ($\pi/2$) phase shift (the multiplication by –j). By using the two interpolation filters, one is rotated by $-\pi/4$ while the other is rotated by $+\pi/4$. Done properly, we can guess that this might line them up again. While we might hope this will happen, we can continue our computation to see if it did happen.



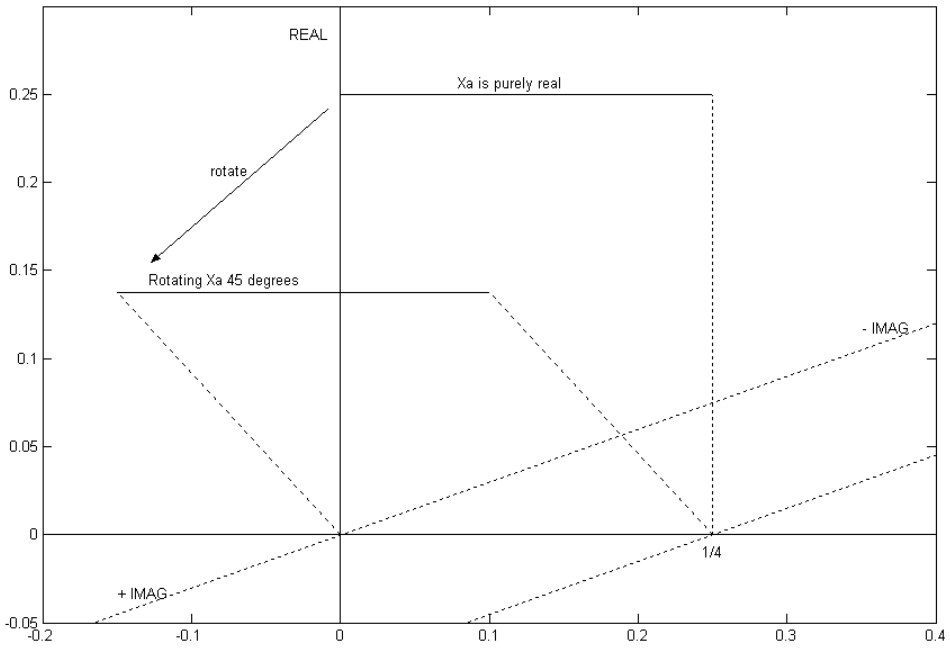
<u>Fig. 11a</u> The filtering of the spectrum of Fig. 10a, by the H_a filter, consists of removing the portion from 0.25 to 0.75, leaving a rectangular section. This piece is then multiplied by $2\sqrt{2}$ and rotated in the positive imaginary direction for positive frequencies (0 to 0.25), and in the negative imaginary direction for negative frequencies (0.75 to 1).



<u>Fig 11b</u> This figure shows the filtering of the spectrum of Fig. 10b by the H_b filter. The procedure is basically the same as that of Fig. 11a, but is more difficult to visualize. Fig. 12 shows some details of the rotations here.

Figures 11a and 11b show the results of the two filterings. Note that both filters were half band, so all the spectral content between 0.25 and 0.75 is removed in both cases. When we add these two together, we obtain Fig. 9 back. We note immediately that the imaginary parts cancel. Neither is it hard to see that the real parts add to the original triangle. So it works.

One final perspective is afforded by Fig. 12, which attempts to depict a three-dimensional view of the spectral region between frequencies 0 and 0.5. Fig. 12a shows the contribution of the $x_a(n)$ component. This is composed of overlapping triangles, forming a purely real rectangle, which is then rotated 45° by H_a producing both real and imaginary rectangles (Fig. 11a from 0 to 0.25, not including the $2\sqrt{2}$ gain here). This is simple. Fig. 12b shows he corresponding case for the $x_b(n)$ component, which is more difficult to envision, because there are both real and imaginary components, and both are triangles. The 45° rotation by the filtering of H_b is in the opposite direction as that of H_a. (Here it is strongly recommended that the reader cut out some cardboard models to play with to better envision the rotations.) The important thing about Fig. 12b is that the sum of the two rotated components has a negative imaginary part that is rectangular, and a real part that is triangular, positive at frequency 0, and negative at frequency 1/4 (Fig. 11b). Following filtering and summation (Fig. 1), the imaginary rectangles of X_a and X_b cancel, while the real rectangle due to X_a lifts the bipolar real triangle of X_b to a purely real triangle, the original spectrum.



<u>Fig. 12a</u> The spectral contribution of $x_a(n)$, showing filtering

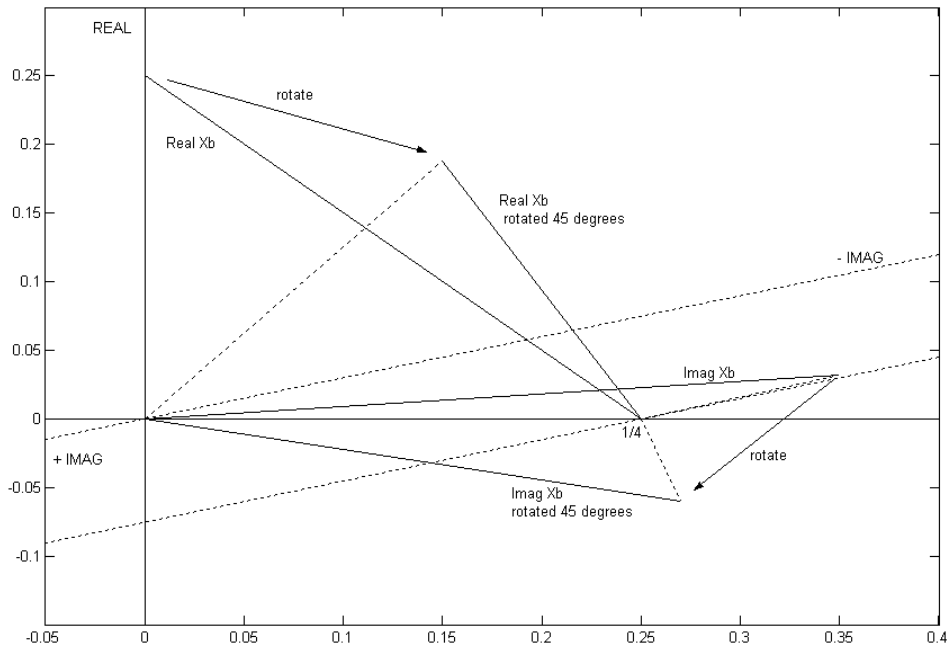


Fig. 12b The spectral contribution of $x_b(n)$, showing filtering

REFERENCES

[1] B. Hutchins, "Sampling and Recovery Based on an Average Sampling Rate," Electronotes, Vol. 19, No. 187, August 1996, pp 3-24

[2] B. Hutchins, "Some Additional Sampling Examples," Electronotes, Vol. 21, No. 201, Feb 2002

[3] B. Hutchins, "Spectral Recovery for a Class of Non-Uniform ("Bunched") Sampling" Electronotes Application Note No. 356, Oct. 2003

[4] R. N. Bracewell, The Fourier Transform and its Applications, (2nd Ed.) McGraw-Hill (1978) pp 201-202. See also our own presentation of Bracewell's interpolating functions in Electronotes , Vol. 21, No. 200, Dec. 2001 ("Sampling Element"), pp 33-35

[5] D. A. Linden, "A Discussion of Sampling Theorems," Proc. IRE, Vo. 47, pp 1219-1226, July 1959

[6] A. Kohlenberg, "Exact Interpolation of Band-Limited Functions, " J. Applied Physics, Vol. 24, No. 12, pp 1432-1436, Dec. 1953

[7] B. Hutchins, "Sampling element," Electronotes , Vol. 21, No. 200, Dec. 2001