ELECTRONOTES
1016 Hanshaw Rd
Ithaca, NY 14850

APPLICATION NOTE NO. 370

March 2007

LIFTING – A THIRD APPROACH TO WAVELETS

1. INTRODUCTION

Here we want to look at the notion of "Lifting" that we regard as a "third approach" to wavelets. The first two approaches are the purely mathematical (dilation equations, etc.) and filter banks (perfect reconstruction filters). Lifting offers an approach originating with a notion of a prediction operation [1].

2. THE LIFTING STRUCTURE

Fig. 1 shows the so-called lifting structure. Here we have a two-channel decomposition of the input waveform composed of a delay (z^{-1}), downsamplers (↓), a Prediction operation (P), and Update operation (U), upsamplers (↑), an advance (z), plus summers (Σ). We note that the delay and downsamplers on the left decompose the input sequence into even (e) and odd (o) samples. On the right, we have recovered versions of (e) and of (o) that are upsampled, offset, and interleaved back to give the original. Accordingly, if we were to remove the center portion involving the P and U elements, and just connect (e) to (e) and (o) to (o) we get perfect recovery. This supposed toy without the P's and U's is sometimes called the "lazy wavelet transform."

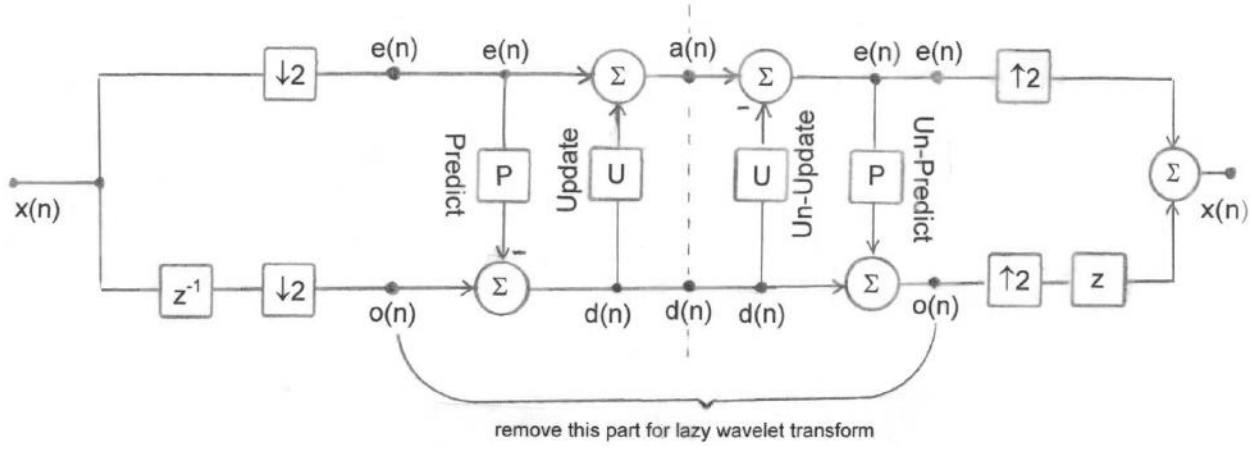


Fig. 1 The basic lifting scheme

Now, what about the P's and the U's. The first thing to note is that the P and U boxes could be anything. By this we mean that they might be constants, systems [P(z) and U(z)] or even non-linear blocks. We also mean that their actual parameters could be anything. For example, P might be a constant of 1077.22 while U squares the sample at its input. All we have done is add and subtract the same things! Clearly this is a "fraud" of some sort. Accordingly perhaps we should say that we are looking to find values of P and U such that we could discard (or represent by fewer bits) one of the intermediate points (a) or (d) as being of lesser significance. If we can do this, it reminds us of the way we might discard one or the other of the channels of a perfect reconstruction filter pair. In fact, the downsampler and upsampler pair further remind us of the perfect reconstruction filters – our next item to consider.

3. THE PERFECT RECONSTRUCTION FILTER PAIR

Fig. 2 shows the usual view of a Perfect Reconstruction Filter (PRF) pair. Note that it, like the lifting scheme, decomposes the input signal into to channels. Unlike the lifting scheme that divides the signal in time into even and odd samples, the PRF divides the signal into two frequency bands, typically low-pass and a high-pass components. Following the filterings, the signals can be downsampled by a factor of two because the bandwidths have been reduced by a factor of two. Strictly speaking, practical filters are not going to divide the frequency band by two, but perhaps surprisingly, even some poor-looking filters, properly chosen, can result in perfect reconstruction [2].

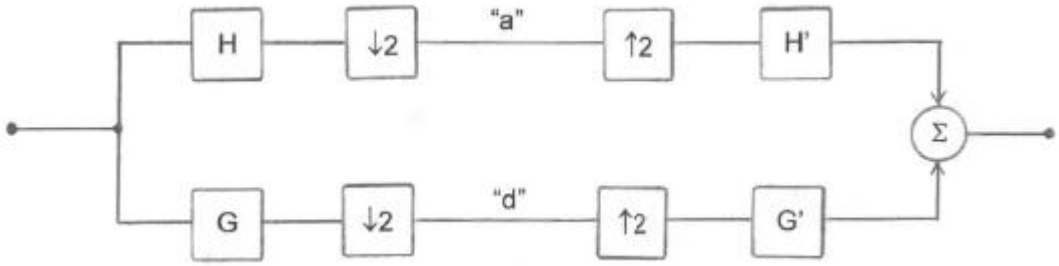


Fig. 2 The traditional Perfect Reconstruction Filter (PRF) view. The paths marked "a" and "d" are a low-pass and high-pass respectively. When we choose H, G, H', and G' correctly, these will become the same as those in the lifting scheme.

4. MAKING THE TWO LOOK ALIKE

Our goal next is to see if we can make the lifting structure and the PRF structure in some way equivalent. That is, given P and U, find H and G, or vice versa. To do this,

we will make use of polyphase decompositions [3]. This will accomplish two things. First, it will introduce a delay following the input into one channel, as in Fig. 1. Secondly, it will make the poly-phase components functions of z^2 rather than just of z, so that we will be able to slide the downsamplers to the left. Fig. 3 shows the polyphase decomposition of the left side (analysis side) of the PRF – a similar approach can be used to decompose the right side (synthesis side). In simplest terms, what we have done for both H(z) and G(z) is to separate their impulse responses into even and odd sequences. Accordingly there are two delays between the taps of $H_e(z^2)$ and $H_o(z^2)$, and similarly for $G_e(z^2)$ and $G_o(z^2)$, which is why we have z^2 instead of z (z^{-2} instead of z^{-1} if you prefer).

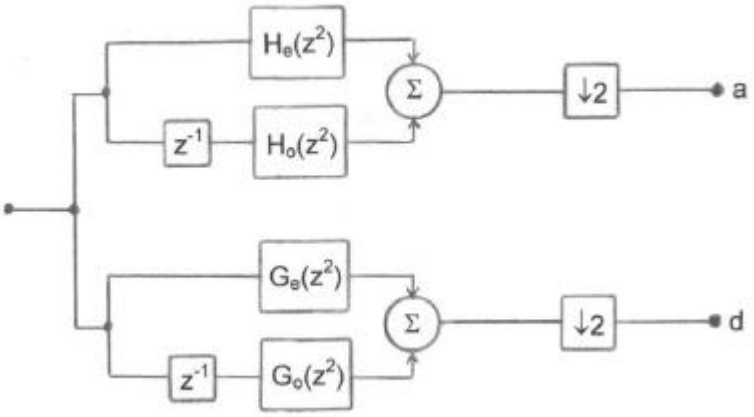
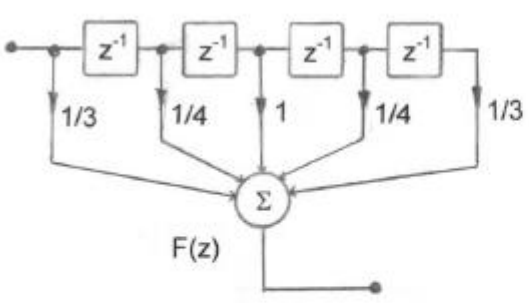


Fig. 3 Polyphase Decomposition of Analysis Side of PRF

In order to make this even more clear, consider a filter F(z) which has an impulse response given by the sequence { 1/3 1/4 1 1/4 1/3 }. (This is just an example, not something we are going to use later in this note.) Fig. 4 shows how this is decomposed into polyphase components.



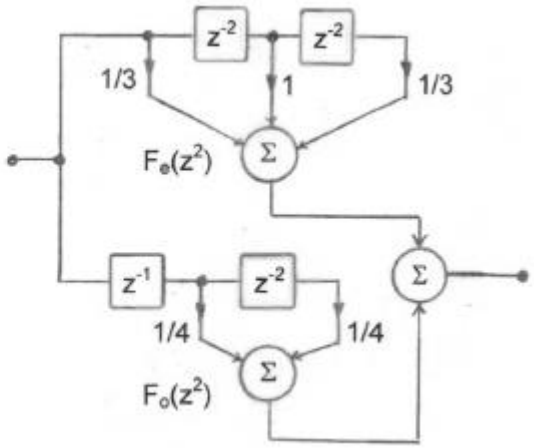


Fig. 4 An example (unrelated to the lifting issue here) to illustrate polyphase decomposition

Fig. 5 shows Fig. 3 with the downsamplers moved to the left (through the filters that are functions of z^2, making them functions of z, but not through the delay z^{-1}). Now we have on the far left the exact same structure (the delay and downsamplers) that we have in the lifting scheme for separating the input into even and odd sequences. This is common to both the H filter and the G filter, and can be combined as in Fig. 6, from which we now find an interconnection of the polyphase filters with input points (e and o) and output points (a and d) that are the same as the lifting scheme of Fig. 1.

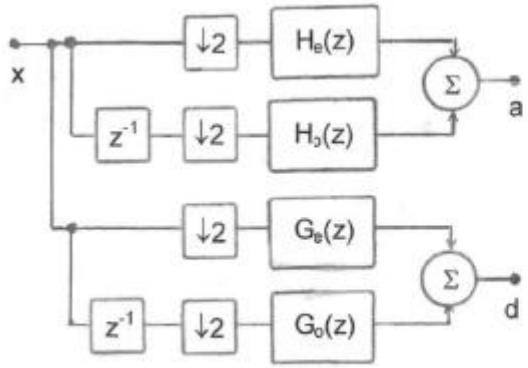


Fig. 5 Downsamplers moved through filters

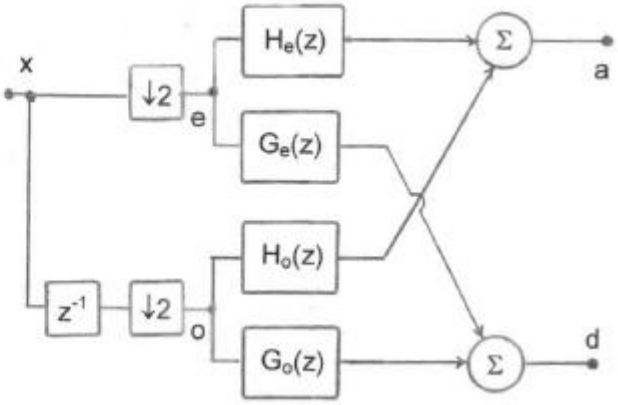


Fig. 6 Simplified Structure of Fig. 5

From Fig. 6, we can write down a matrix relationship (in the z-domain) relating the even and odd sequences e and o to the output points a and d:

$$\begin{bmatrix} A \\ D \end{bmatrix} = \begin{bmatrix} H_e & H_o \\ G_e & G_o \end{bmatrix} \begin{bmatrix} E \\ O \end{bmatrix} \qquad (1)$$

We can write a corresponding matrix for the lifting scheme. Here we will consider P and U to be filters P(z) and U(z) so that this too can be written in the z-domain. From Fig. 1 we have, by inspection (just follow the “flow diagram”):

$$\begin{bmatrix} A \\ D \end{bmatrix} = \begin{bmatrix} 1-P(z)U(z) & U(z) \\ -P(z) & 1 \end{bmatrix} \begin{bmatrix} E \\ O \end{bmatrix} \qquad (2)$$

Thus we have:

$$H_e(z) = 1 - P(z)U(z) \quad (3a)$$

$$H_o(z) = U(z) \quad (3b)$$

$$G_e(z) = -P(z) \quad (3c)$$

$$G_o(z) = 1 \quad (3d)$$

We will need to recognize that these relationships describe filters. For example, P(z)U(z) represents the product in the z-domain, but the convolution in the time domain. P(z) and U(z) may be non-causal, or require time shifts. Fortunately, doing it right is reasonably intuitive.

5. AN EXAMPLE

Here as an example we will choose a predictor P and an update U and find the corresponding filters. We could do it the other way around and choose a PRF pair and compute P and U. One thing we need to keep in mind is that we want to see if we can discard "d" (or perhaps "a") and get reasonable recovery.

Here we will predict that the odd samples are the average of the even samples on either side. (This is of course the same thing as linear interpolation, but the standard terminology with lifting calls this a prediction. However, thinking in terms of interpolation suggests many other choices for a "predictor.") For no really good reason, we will choose U as half of P! recall that we can choose P and U arbitrarily and still get perfect reconstruction, so our goal is ultimately to find choices for P and U that concentrate the information in one channel ("d" or "a").

At this point, we encounter a bit of confusion. The impulse response of the linear interpolator P clearly should have values 1/2 and 1/2 [4]. This is hard to think about in terms of only the even samples or the odd samples. This is because we normally think of interpolation as a two step process of inserting zeros in an input sequence and then calculating, and "inter-leaving" the interpolated values. For example, in terms of prediction, the first odd sample (index 1) is calculated as the average of the first and second even samples (index 1 and index 2). In terms of interpolation, we might prefer to say that the first odd sample (index 1) is the average of the original samples indexed at 0 and 2. Further complicating the situation is the fact that in our Matlab programs we need to index starting at 1. All this means that we need to incorporate two ideas into our study: (1) it will often be useful to keep both even and odd indices in our sequences, recognizing that half are automatically zero, and (2) it is important to look at each step to see if it "looks right" (use your engineering intuition). Bear with us.

Now we are ready to write things down. Thinking in terms of keeping all the original indices, the prediction filter is:

$$P(z) = (1/2)z + (1/2)z^{-1} \quad (4)$$

and

$$U(z) = (1/4)z + (1/4)z^{-1} \quad (5)$$

Thus P(z) is not a linear interpolation filter, but rather the polyphase component of a linear interpolation (by 2) filter. (The linear interpolation filter would be: $(1/2)z + 1 + (1/2)z^{-1}$.) It is also necessary for us to recognize that U(z) as written in equation (5) is offset by one time unit relative to P(z), because the update is on the odd indexed samples rather than the even indexed samples. So, with some care, and an eye for the symmetry we expect, we can run through equations (3a) to (3d)

$$H_e(z) = 1 - P(z)U(z) = -(1/8)z^2 + (3/4) - (1/8)z^{-2} \quad (6a)$$

$$H_o(z) = U(z) = (1/4)z + (1/4)z^{-1} \quad (6b)$$

It looks very promising to just add these two to get H(z), but then again, we note that somehow we were supposed to use a delay: $H(z) = H_e(z) + z^{-1}H_o(z)$. In fact, here we do not need to include the delay because it was already included in our premise that we were predicting the odd samples (shifted by one original position) based on the even samples. Similarly,

$$G_e(z) = -P(z) = -(1/2)z + -(1/2)z^{-1} \quad (6c)$$

$$G_o(z) = 1 = 1 \quad (6d)$$

This gives us our two PRF pairs:

$$H(z) = -(1/8)z^2 + (1/4)z + (3/4) + (1/4)z^{-1} - (1/8)z^{-2} \quad (7a)$$

$$G(z) = -(1/2)z + 1 - (1/2)z^{-1} \quad (7b)$$

We note that these are not the same length (something we <u>have</u> seen with other approaches to PRFs).

Inverting the PRF “analysis” should be a matter of repeating the procedure above for the right half of the lifting scheme. Or it might seem easier to just invert equations. But we need to be careful!

Equation (2) can be inverted, or we can just use the flow-diagram procedure on the right side of Fig. 1:

$$\begin{bmatrix} E \\ O \end{bmatrix} = \begin{bmatrix} 1 & -U(z) \\ P(z) & 1-P(z)U(z) \end{bmatrix} \begin{bmatrix} A \\ D \end{bmatrix} \tag{8a}$$

$$\begin{bmatrix} E \\ O \end{bmatrix} = \begin{bmatrix} 1 & -(1/4)z - (1/4)z^{-1} \\ (1/2)z + (1/2)z^{-1} & -(1/8)z^2 + 3/4 - (1/8)z^{-2} \end{bmatrix} \begin{bmatrix} A \\ D \end{bmatrix} \tag{8b}$$

where we have used the P(z) and U(z) chosen above.

At this point, it may be most useful to jump ahead to the "right answer" and work backward from that. How can we "guess" the right answer? Well we have H(z) and G(z) and we know some things about what to expect in a PRF. We expect H'(z), like H(z), to be basically low-pass while G'(z), like G(z), should be basically high-pass. Further since H(z) is linear-phase, length 5 (delay of 2), and G(z) is linear-phase, length 3 (delay of 1), we expect that the synthesis filter H'(z) should be length 3 while G'(z) should be length 5, such that the overall delay is length 3. Note that our actual filters have a delay (3 in this case) and scale changes and delays are allowable within the usual definitions of a PRF. This is something that we can easily adjust (using advances), and it turns out, this is important in this case, since the lifting scheme as given shows no delay (it has an advance z in the reconstruction).

Accordingly we want to form a low-pass H'(z) from G(z) and a high-pass G'(z) from H(z). All this intuitive reasoning leads to Fig. 7, where we have:

$$H'(z) = (1/2)z + 1 + (1/2)z^{-1} \tag{9a}$$

$$G'(z) = -(1/8)z^2 - (1/4)z + 3/4 - (1/4)z^{-1} - (1/8)z^{-2} \tag{9b}$$

Note that G(z) and G'(z) do have a dc gain of 0 as is appropriate for a high-pass filter. It is not difficult to verify, by simulation, that the structure in Fig. 7 is in fact, perfect reconstruction, with a delay of 3.

We can easily decompose H'(z) and G'(z) into their polyphase components:

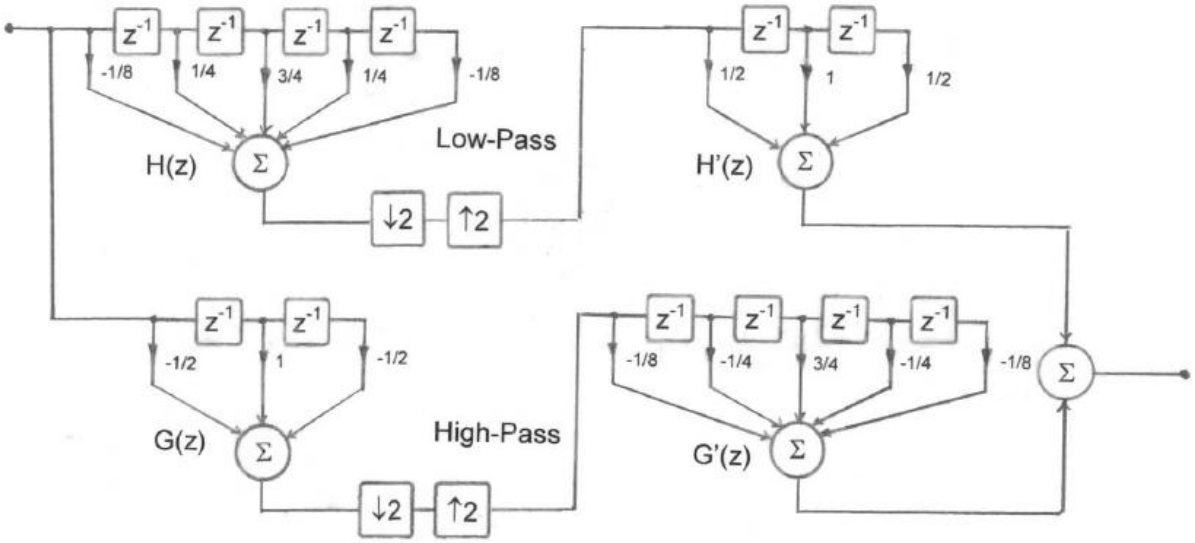


Fig. 7 Perfect Reconstruction Filter Corresponding to Choice of P and U

$$H_e'(z) = (1/2)z + (1/2)z^{-1} \quad (10a)$$

$$H_o'(z) = 1 \quad (10b)$$

$$G_e'(z) = -(1/8)z^{2} + 3/4 - (1/8)z^{-2} \quad (10c)$$

$$G_o'(z) = -(1/4)z - (1/4)z^{-1} \quad (10d)$$

These we can put back into equation (8b), giving:

$$\begin{bmatrix} E \\ O \end{bmatrix} = \begin{bmatrix} H_o'(z) & G_o'(z) \\ H_e'(z) & G_e'(z) \end{bmatrix} \begin{bmatrix} A \\ D \end{bmatrix} \quad (11)$$

This looks like it has to be right. But, as we shall see, it needs to be adjusted for delay, and this takes a bit of work to show.

Fig. 8 shows Fig. 7 decomposed into its polyphase components, and this is followed by Fig. 9 where we have moved downsamplers to the left and upsamplers to the right. We note that the left sides here provides a specific example corresponding to Fig. 5 and Fig. 6. We now just want to complete or analysis of the right side.

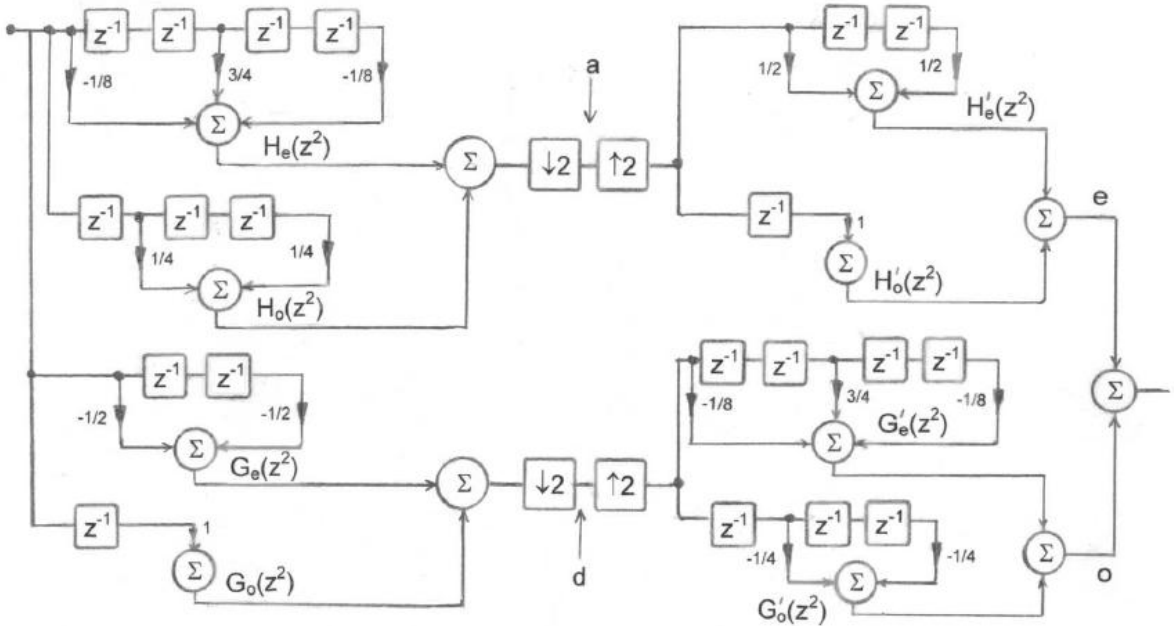


Fig. 8 Polyphase decomposition of Fig. 7

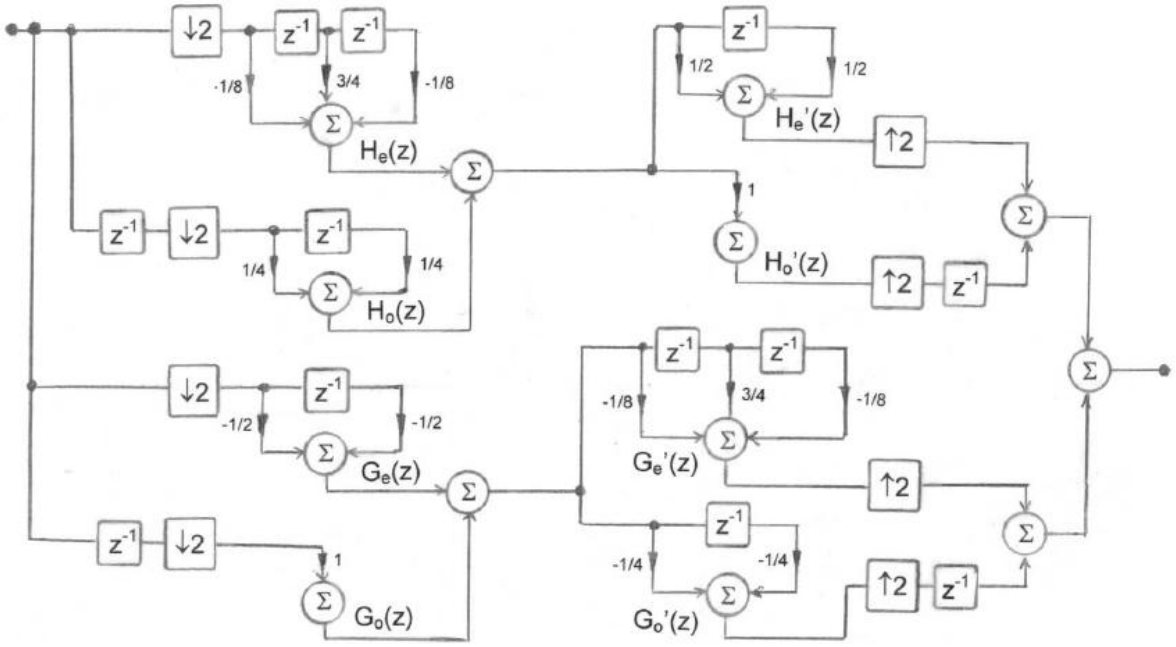


Fig. 9 Moving the upsamplers and downsamplers to the ends.

In a manner similar to what we did on the analysis side, Fig. 10a shows the cross-coupling onto a single set of upsamplers and a delay. From this we can easily write the matrix relationship:

$$\begin{bmatrix} E \\ O \end{bmatrix} = \begin{bmatrix} H_e'(z) & G_e'(z) \\ H_o'(z) & G_o'(z) \end{bmatrix} \begin{bmatrix} A \\ D \end{bmatrix} \qquad (12)$$

We glance back at equation (11) hoping to see this same result, but we don't. But it would be right if we interchanged O and E.

$$\begin{bmatrix} O \\ E \end{bmatrix} = \begin{bmatrix} H_e'(z) & G_e'(z) \\ H_o'(z) & G_o'(z) \end{bmatrix} \begin{bmatrix} A \\ D \end{bmatrix} \qquad (13)$$

The notion of interchanging O and E may seem problematic until we recognize that the choice of O and E is arbitrary anyway. But specifically, here we had a delay of 3 in the filters, and a delay of 3 (or of any odd number) interchanges the notion of odd and even. We can modify Fig. 10a for zero delay by multiplying the output paths by z^3 (Fig. 10b).

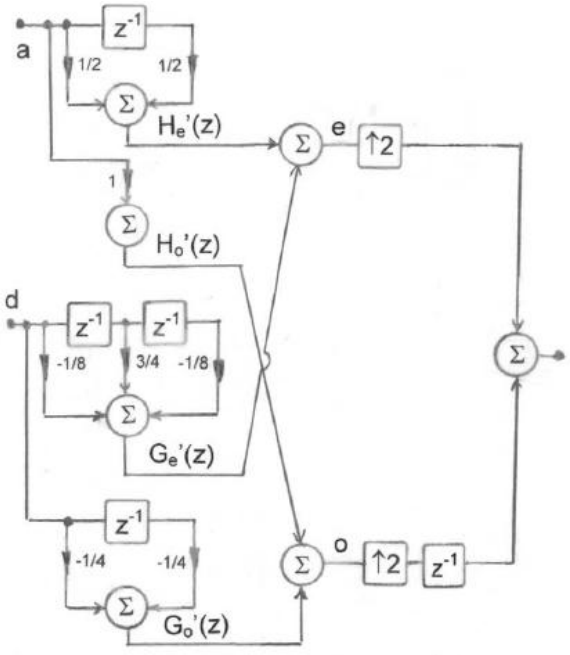


Fig. 10a Reconstruction

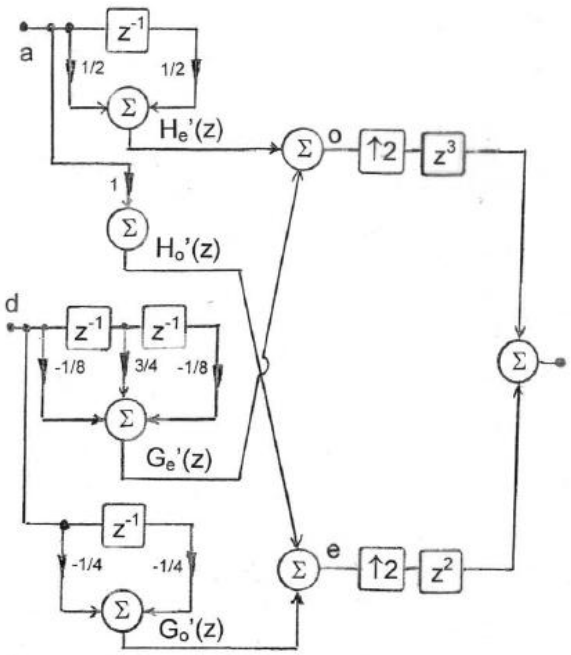


Fig. 10b Reconstruction with shift

6. A TEST PROGRAM – LIFTING

Our first program, onedlift.m, simulates the lifting scheme using the choices of P and U given in Section 5 above. The program code and the examples of the printout are interleaved (just to fit better in the space available) below. The captions to the figures discuss the results, which can be read sequentially.

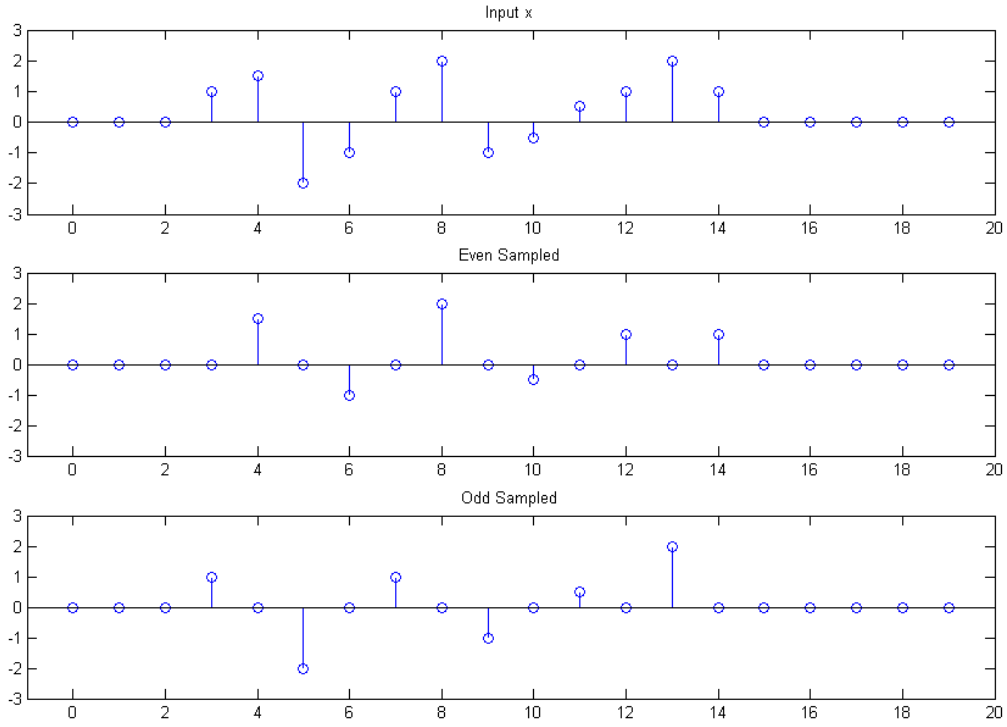


Fig A1 Example using onedlift with x= [0 0 0 1 1.5 -2 -1 1 2 -1 -0.5 0.5 1 2 1 0 0 0 0 0] with dmult=1, pg=1/2, and ug=1/4. The input sequence x is at the top, the even samples are in the middle, and the odd samples are at the bottom. In this display, for the even/odd plots, zeros remain in the positions for the other sequence in order to better show the displacements. See also Fig. A2 below.

********** **onedlift.m CODE BEGINS**********

```
function onedlift(x,dmult,pg,ug)
% 1 Dimensional Lifting
% x is a length 20 sequence
% dmult turns on(1)/off(0) throughput of d
% pg = prediction gain (typically 1/2)
% ug = update gain (typically 1/4)
% B. Hutchins    Spring 2007
```

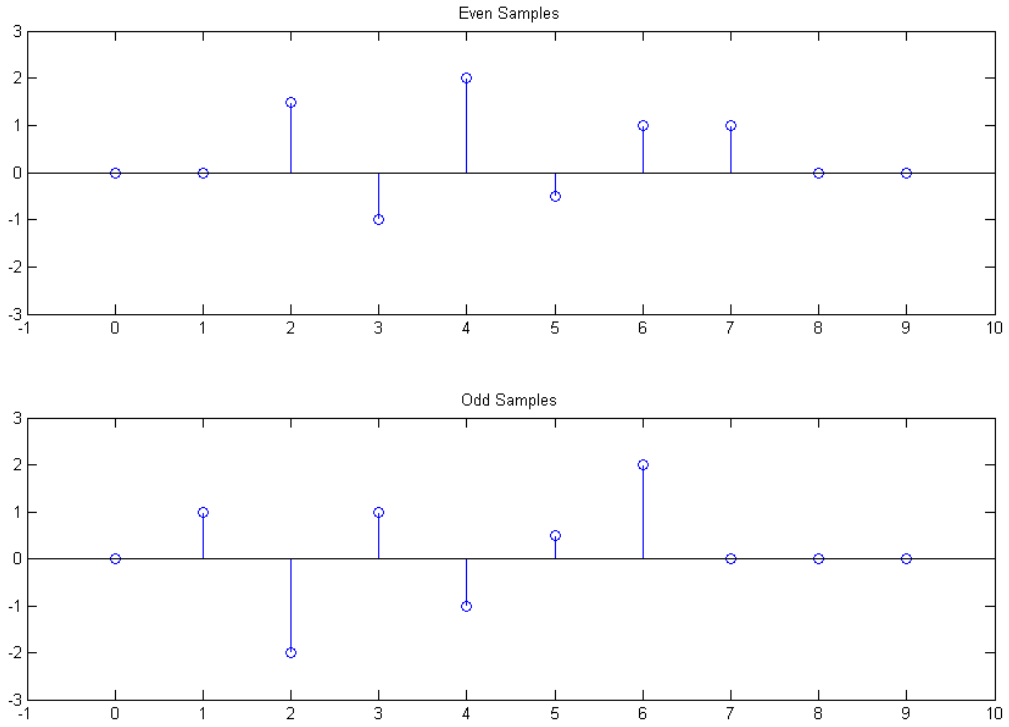


Fig. A2 Here we show only the even samples (top) and only the odd samples (bottom) – exactly the same samples as in Fig. A1. This display better emphasizes the fact that these two sequence are only half the original length (each being 10 rather than 20). However, it is likely less clear that there is an offset between them. Compare carefully to Fig. A1 above.

********** **onedlift.m CODE CONTINUES** **********

```
se=[1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0];
so=[0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1];
e=x.*se;
o=x.*so;
figure(1)
subplot(311)
stem([0:19],x)
axis([-1 20 -3 3])
title('Input x')
subplot(312)
stem([0:19],e)
axis([-1 20 -3 3])
title('Even Sampled')
subplot(313)
stem([0:19],o)
axis([-1 20 -3 3])
title('Odd Sampled')
```

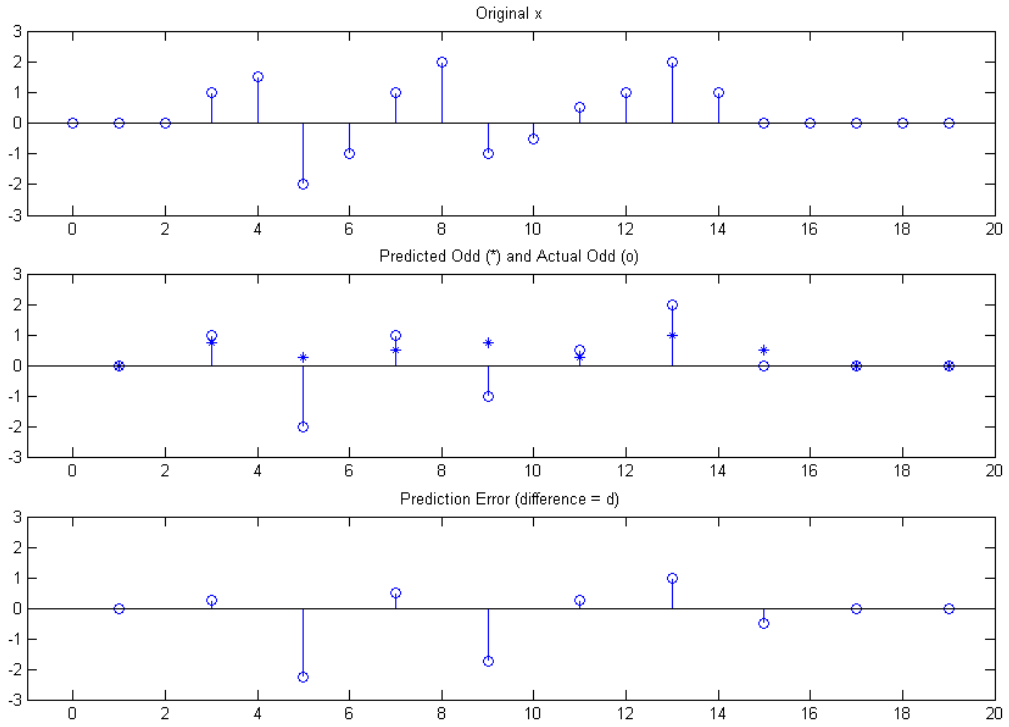


Fig. A3 In the first step in lifting, the odd samples are predicted from the even samples. Here the prediction is that the odd samples will be the average (linear interpolation) of the even samples on each side. For example, the actual odd sample at 7 is a 1(shown as o at 7). The predicted value of the odd sample at 7 is (-1 + 2)/2 = 0.5 (shown as * at 7). That is, half of the even sample at 6 (which is -1), plus half the even sample at 8 (which is 2). The prediction errors are shown at the bottom, and we call this d.

********** **onedlift.m CODE CONTINUES** **********

```
eonly=e(1:2:19);
oonly=o(2:2:20);
figure(2)
subplot(211)
stem([0:9],eonly)
axis([-1 10 -3 3])
title('Even Samples')
subplot(212)
stem([0:9],oonly)
axis([-1 10 -3 3])
title('Odd Samples')
```

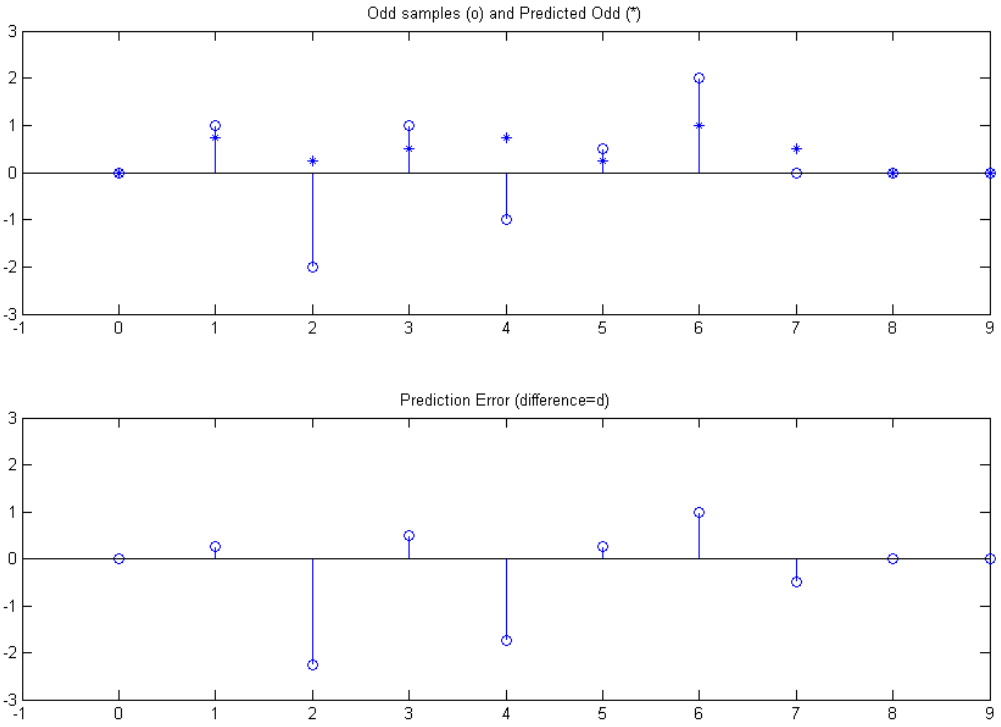


Fig. A4 This is the same data as appear in Fig. A4, except we show only the odd values, which are re-indexed. In consequence, the example singled out in Fig. A3 that was at 7 is seen here at index 3.

********* onedlift.m CODE CONTINUES**************************

```
eP=filter([0 pg 0 pg],1,e)
eP=[eP(3:20),0,0]
d=o-eP
figure(3)
subplot(311)
stem([0:19],x)
axis([-1 20 -3 3])
title('Original x')
subplot(312)
plot([1:2:19],eP(2:2:20),'*')
hold on
stem([1:2:20],o(2:2:20))
hold off
axis([-1 20 -3 3])
title('Predicted Odd (*) and Actual Odd (o)')
subplot(313)
stem([1:2:19],d(2:2:20))
axis([-1 20 -3 3])
title('Prediction Error (difference = d)')
```

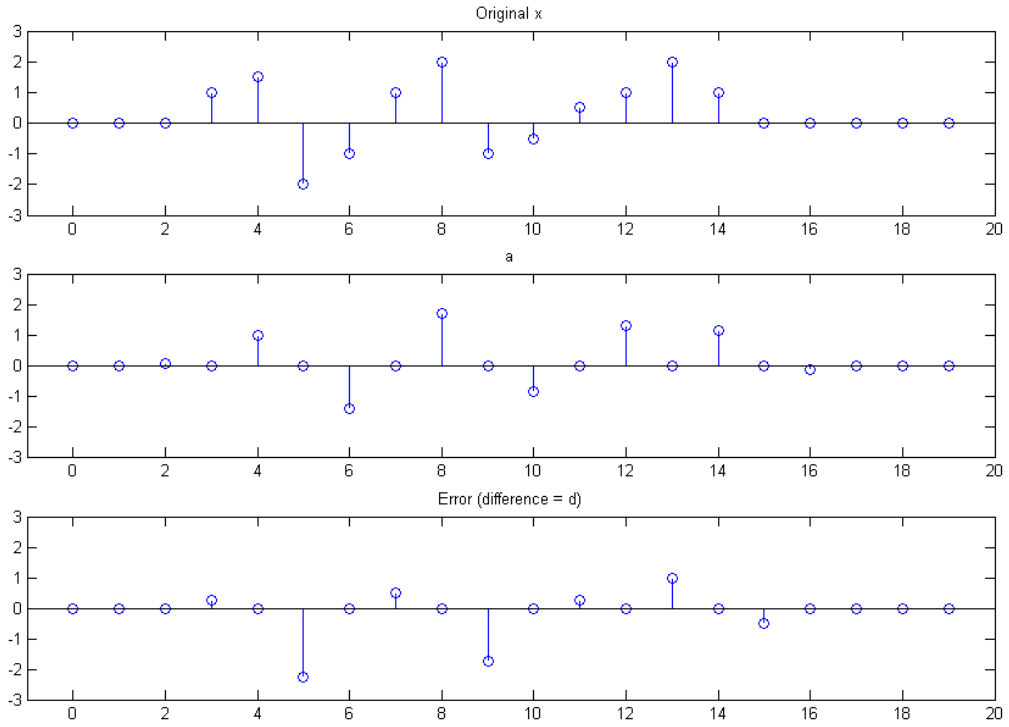


Fig. A5 At this point, we have completed the lifting process. This was done by taking the difference d, updating it, and adding the result to the even sequence, giving us a sequence "a". We end up now with two sequences "a" and "d" which replace the sequences "e" and "o". Here we have not removed the zeros (as we did in going from Fig. A3 to Fig. A4), and at this point, it will become our practice not to do this. However it is still true that "a" is only non-zero on the even positions while "d" is only non-zero on the odd positions. Note that our hope is that if "d" is small enough (it certainly isn't here), we can discard it from here on.

********** **onedlift.m CODE CONTINUES** **********

```
figure(4)
subplot(211)
stem([0:9],oonly)
hold on
plot([0:9],eP(2:2:20),'*')
hold off
axis([-1 9 -3 3])
title('Odd samples (o) and Predicted Odd (*)')
subplot(212)
stem([0:9],d(2:2:20))
axis([-1 9 -3 3])
title('Prediction Error (difference=d)')
```

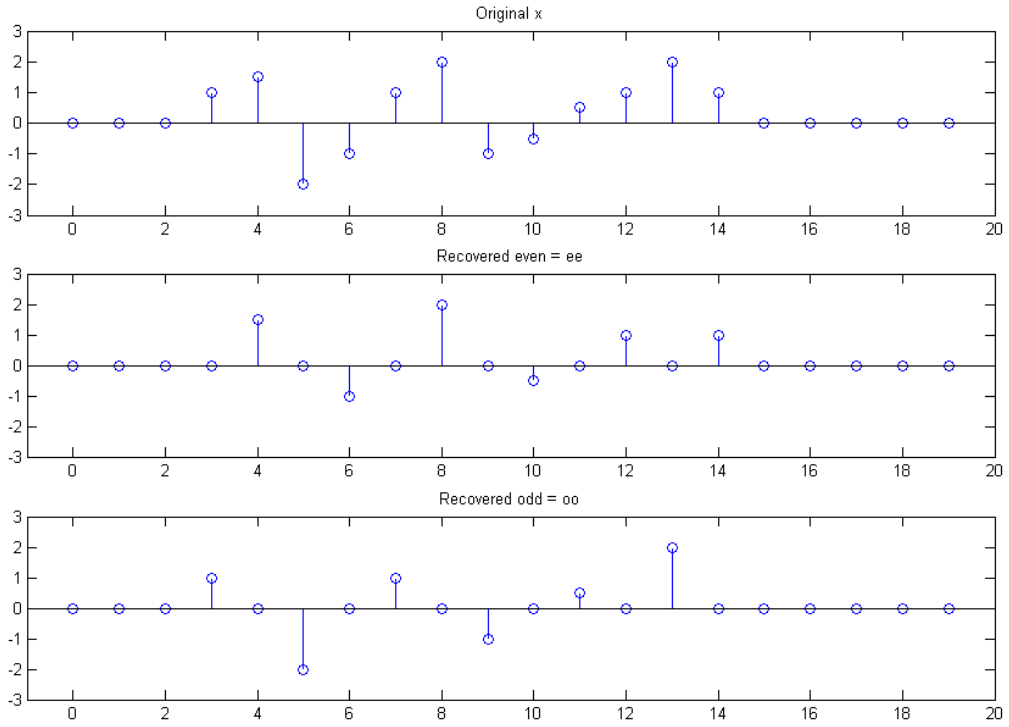


Fig. A6 Here we are reversing the lifting process (completing the apparent fraud!) At this point, we are keeping "d". (We will look at the case where we discard "d" in a moment.) That is, we take "d", update it exactly as we did to form "a", but now subtract it, giving back "e" (see Fig. 1). This we can call un-update. Then from the recovered "e", we again predict the odd samples, and add this prediction to "d" (un-predict) to recover "o".

********** **onedlift.m CODE CONTINUES** **********

```
dU=filter([0 ug 0 ug],1,d)
dU=[dU(3:20),0,0]
a=e+dU
figure(5)
subplot(311)
stem([0:19],x)
axis([-1 20 -3 3])
title('Original x')
subplot(312)
stem([0:19],a)
axis([-1 20 -3 3])
title('a')
subplot(313)
stem([0:19],d)
axis([-1 20 -3 3])
title('Error (difference = d)')
```

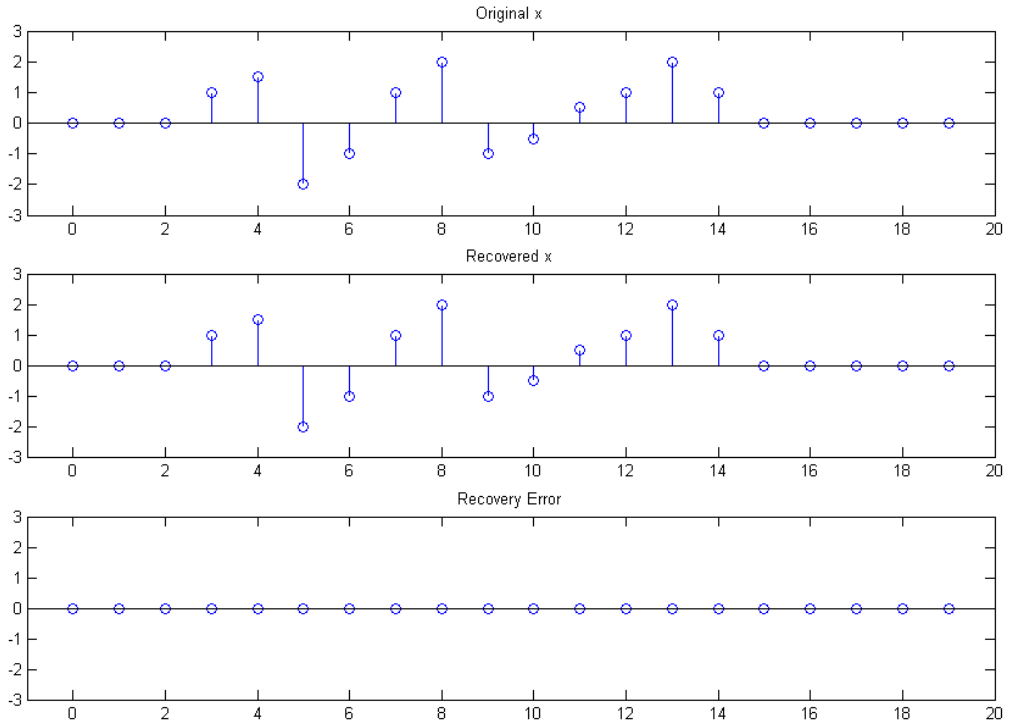


Fig. A7 Here we have completed the recovery by adding the recovered even and odd samples from Fig. A6. The recovery error is of course zero. This is the end of the fraud.

********** onedlift.m CODE CONTINUES **********

```
d=d*dmult

dUs=filter([0 ug 0 ug],1,d)
dUs=[dUs(3:20),0,0]
ee=a-dUs
eeP=filter([0 pg 0 pg],1,ee)
eeP=[eeP(3:20),0,0]
oo=eeP+d
figure(6)
subplot(311)
stem([0:19],x)
axis([-1 20 -3 3])
title('Original x')
subplot(312)
stem([0:19],ee)
axis([-1 20 -3 3])
title('Recovered even = ee')
subplot(313)
stem([0:19],oo)
axis([-1 20 -3 3])
title('Recovered odd = oo')
```

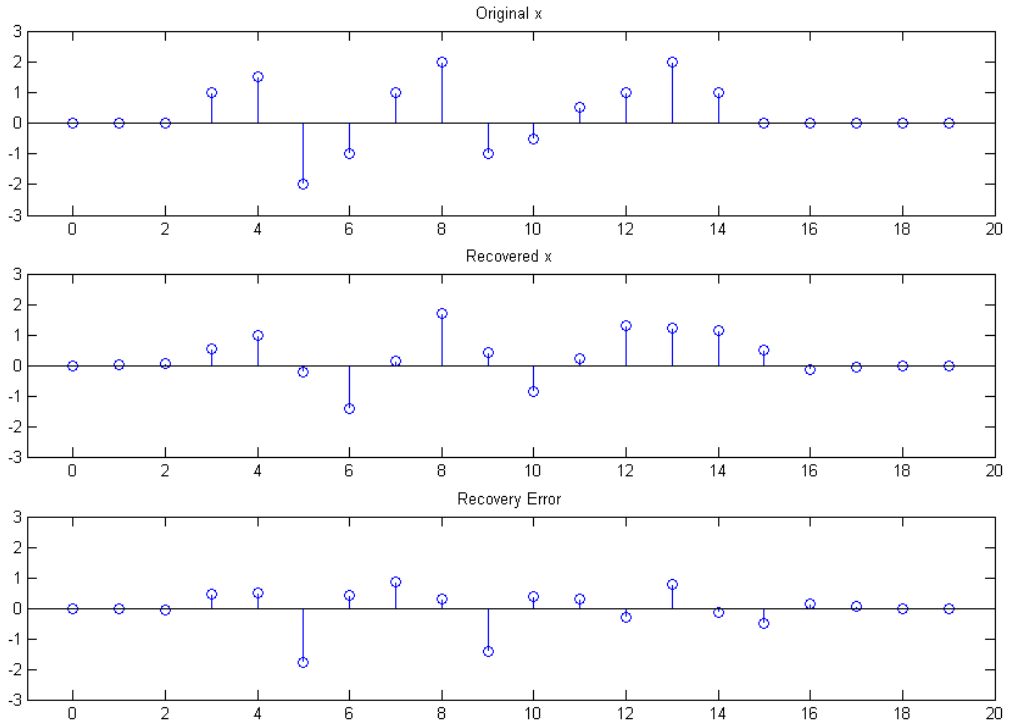


Fig. B7 The lifting and recovery worked perfectly in the A example because we did not discard "d". We kept all the information, and we lost nothing. Now we want to discard "d", and note that we do this after the first update. Thus "a" is updated by "d", but instead of fully recovering the even sequence "e", the sequence "a" must stand in for "e" during recovery. Accordingly, the full recovery of the odd sequence "o" is not achieved, and only the prediction from "a" is available. This produces considerable recovery error.

********************* onedlift.m CODE CONTINUES *****************

```
xr=ee+oo
figure(7)
subplot(311)
stem([0:19],x)
axis([-1 20 -3 3])
title('Original x')
subplot(312)
stem([0:19],xr)
axis([-1 20 -3 3])
title('Recovered x')
subplot(313)
stem([0:19],x-xr)

axis([-1 20 -3 3])
title('Recovery Error')
re=x-xr
```

*****END onedlift.m**

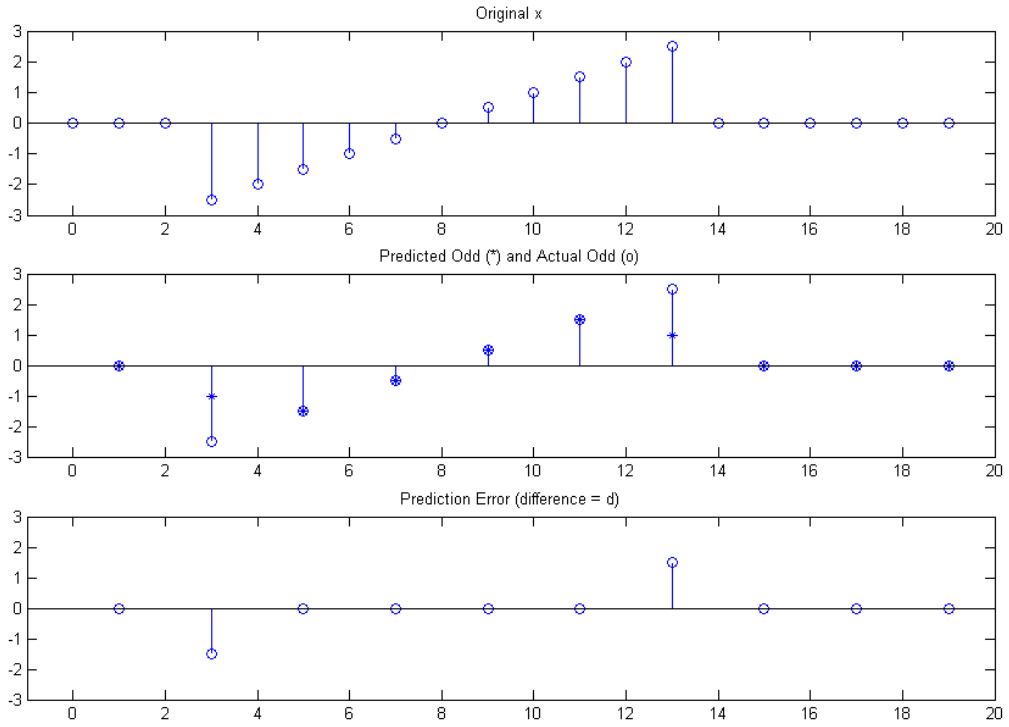


Fig. C3 Here instead of the sequence used in Example A and Example B, we use the more structured "ramp" x=[0 0 0 -2.5 -2 -1.5 -1 -0.5 0 .5 1 1.5 2 2.5 0 0 0 0 0 0]. Thus, at least in the middle of the non-zero samples, we are predicting samples, using linear interpolation, that are actually on a straight line. We see that this gives us exactly zero error for index 5, 7, 9, and 11. Errors ("end transients") remain at index 3 and 13. However, here we have a better excuse for discarding "d".

The purpose of the sequence of figures starting with the letter A was to show how the lifting process can reconstruct perfectly. It does not reconstruct properly when we discard "d" prior to reconstruction (Fig. B7). Fig. B7 corresponds to Fig. A7. {In our examples, we are presenting figures by keeping the letter corresponding to the A example, followed by the figure number in the Matlab program. We do not show the full set of figures in B and C, but only the ones we need to illustrate a point.}

For case C we show Figures C3 and C7. The difference in case C is that we have chosen a test signal that has more structure – in fact it is a ramp-like shape. We note that the linear ramp itself should be representable exactly by our linear interpolation

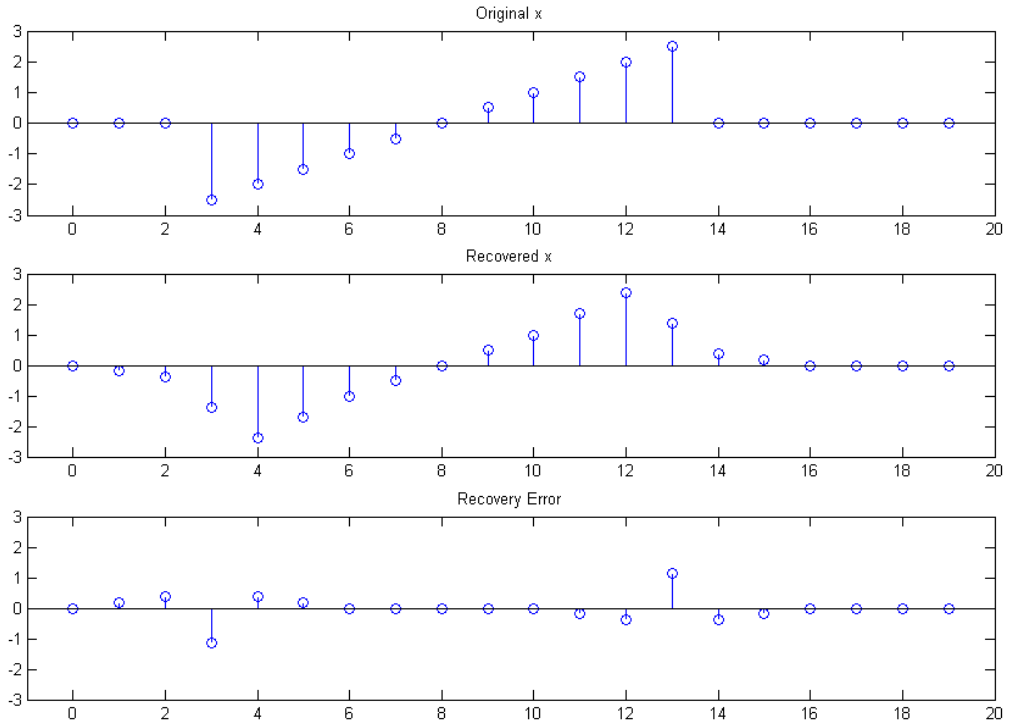


Fig. C7 Here we see the partially successful recovery of "x" following the discarding of "d". (The recovery, like Fig. A7, would have been perfect if we had kept "d".) We see that the center portion of the ramp is recovered perfectly.

prediction model. At least we expect that the middle of it should work well, although we do expect, and find, "transients" on the ends. Fig. C3 (corresponding to A3) illustrates the point that the error sequence "d" is zero in the middle of the ramp, but substantial at the ends. [The lifting always recovers perfectly when we keep the sequence "d" (a result not shown here).] However in Fig. C7 (corresponding to A7 and B7) we show the case where "d" is set to zero after it is used for the update, but prior to its use in the un-update. What this means is that no updated value of "d" is subtracted during reconstruction, so "e" is now just "a." The recovered odd sequence "o" is now just a predicted version of the "a." Another way to look at it is that the signal has been compressed into a single sequence "a," with the influence of the odd samples being carried across (somehow) only by the updated error, "d."

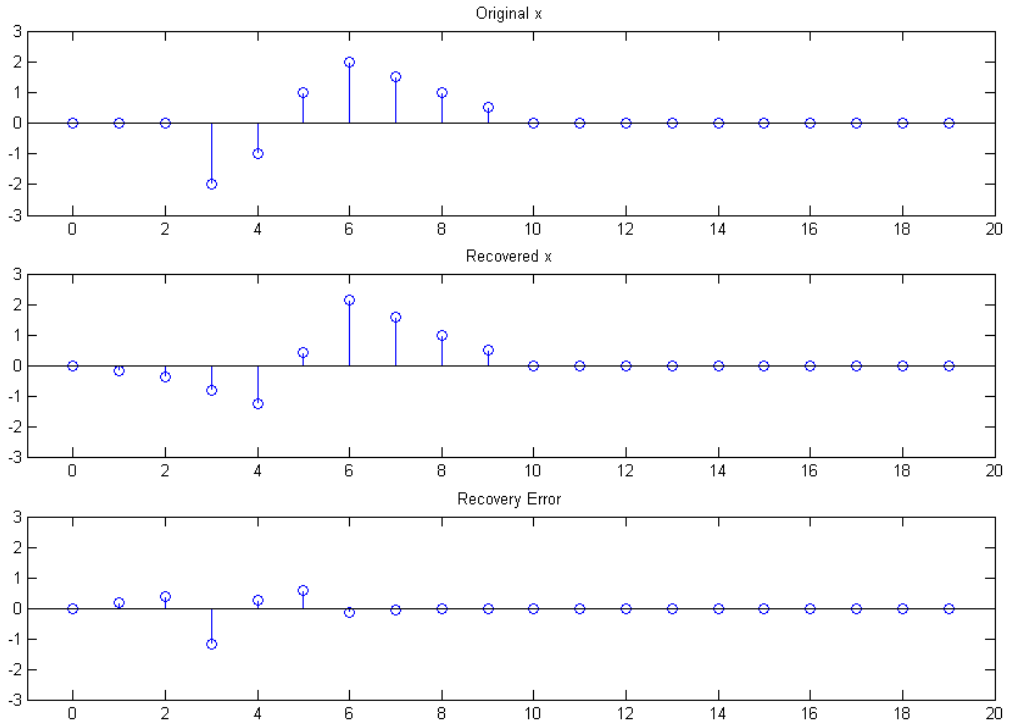


Fig. D7 Using the lifting scheme, a test signal
[0 0 0 -2 -1 1 2 1.5 1 0.5 0 0 0 0 0 0 0 0 0 0 0]
recovers (setting 'd' to 0) as
[0 -0.1875 -0.3750 -0.8125 -1.2500 0.4375 2.1250 1.5625 1.0000
0.5000 0 0 0 0 0 0 0 0 0 0 0]

In Example A we showed complete recovery even though there was no real structure in x, because we kept "d" in the recovery. In Example B, discarding "d", we did not get very good, let alone perfect, recovery. In Example C, x had a ramp-like structure, and except for transients, the recovery was very good. Now in Example D, we choose a low-pass-like event. In fact, it does begin rather suddenly, but goes away slowly (ramp like), and accordingly, we see a starting transient, but not an ending transient.

In any successful compression scheme, including a lifting scheme, we would need to see that the "d" sequence, (or the "d" sequences in the case of a tree-structured decomposition), are small. We do not expect zero values for "d" unless the P choice matches the input signal, as our choice of P here (linear interpolation) matches the ramp portions of the test signal.

7 A SECOND TEST PROGRAM - THE EQUIVALENT PRF

In this second test program, liftfilt.m, we want to show the perfect recovery that we can achieve with the filter bank where the filters are determined by the P and U lifting operations. Another way to describe this is that we are going to simulate Fig. 7. Example E here will use the same input as Example D of the lifting scheme. Fig. E1 shows the filtering by the two filters H (low-pass) and G (high-pass). As we did above, the program will be distributed around the figures at the bottom of the page.

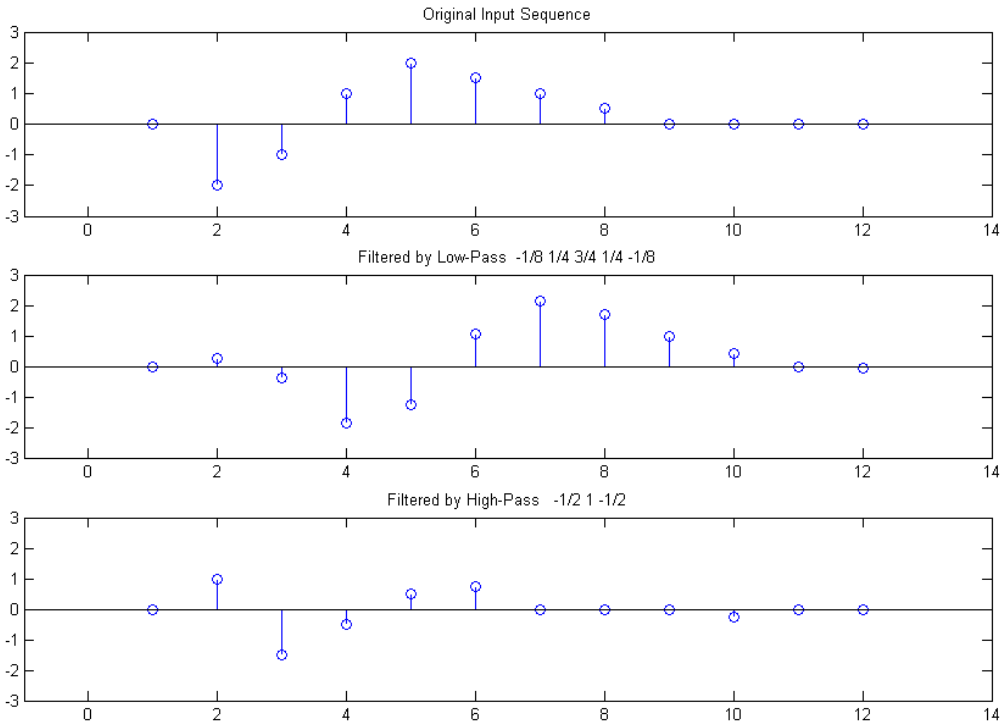


Fig. E1 Original sequence [0 -2 -1 1 2 1.5 1 .5 0 0 0 0] is filtered by low-pass and high-pass perfect recovery filters corresponding to lifting, pg=1/2, ug=1/4.

********** **liftfilt.m CODE BEGINS**********

```
% liftfilt.m
function z=liftfilt(x,mult)

h=[ -1/8 1/4 3/4 1/4 -1/8]      % H of text
hp=[1/2 1  1/2]                 % H' of text

g=[-1/2 1 -1/2]                 % G of text
gp=[ -1/8 -1/4 3/4 -1/4 -1/8]   % G' of text
```

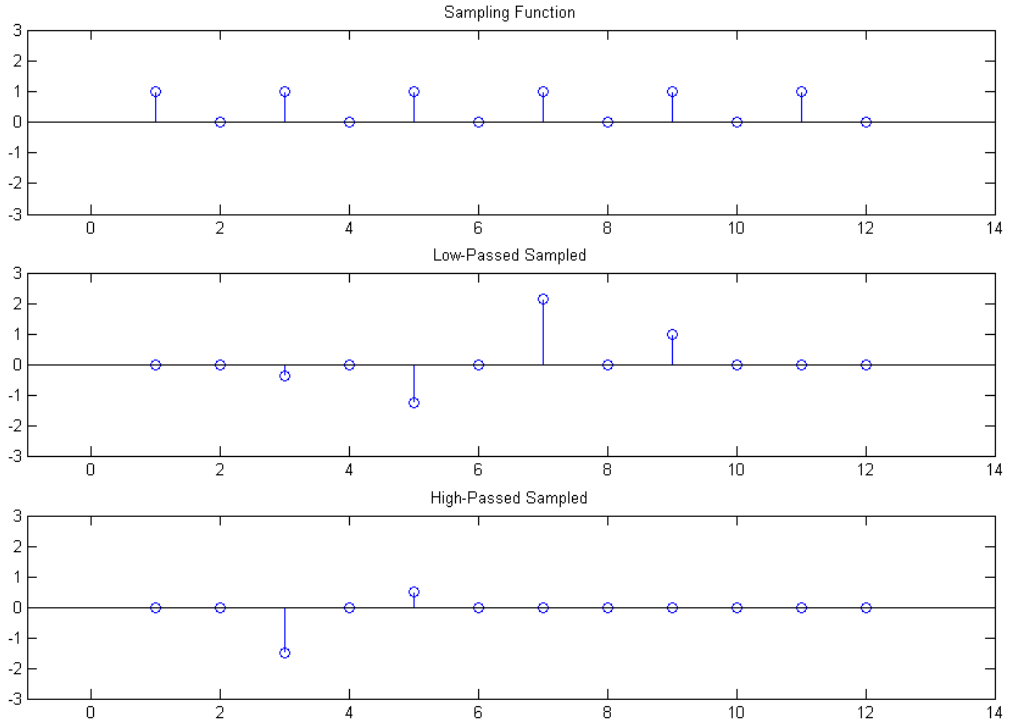


Fig. E2 The low-pass and high-pass outputs are sampled.

********** **liftfilt.m CODE CONTINUES**********

```
figure(1)
subplot(311)
stem(x)
axis([-1 14 -3 3]);
title('Original Input Sequence')
xlow=filter(h,1,x)
xhi=filter(g,1,x)
subplot(312)
stem(xlow)
axis([-1 14 -3 3]);
title('Filtered by Low-Pass  -1/8 1/4 3/4 1/4 -1/8')
subplot(313)
stem(xhi)
axis([-1 14 -3 3]);
title('Filtered by High-Pass   -1/2 1 -1/2')
s=[1 0 1 0 1 0 1 0 1 0 1 0]
```

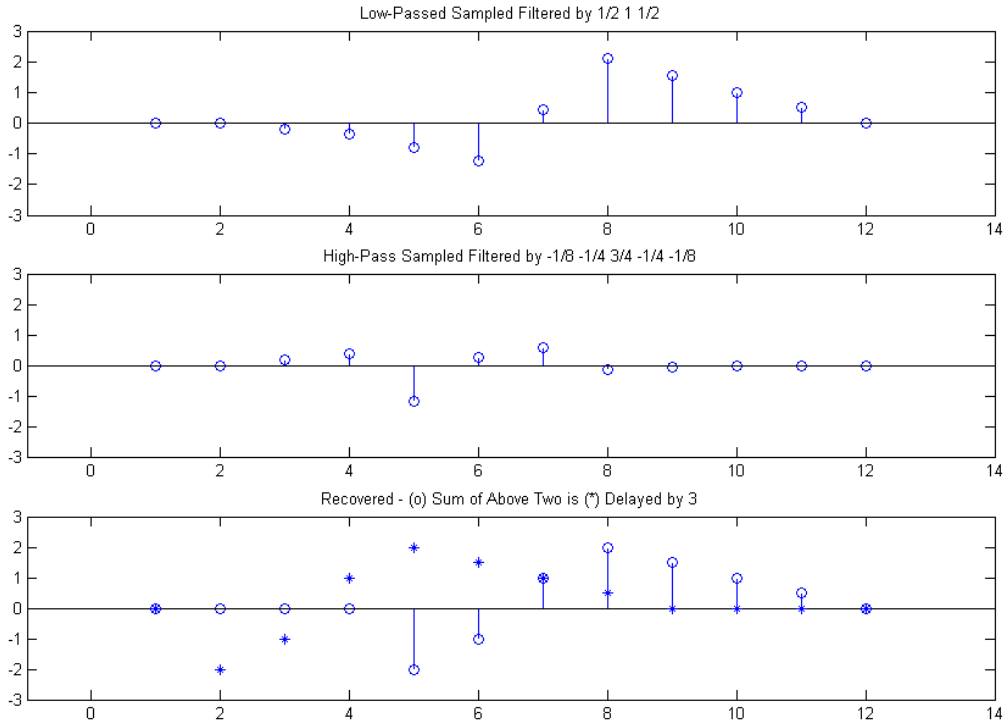


Fig. E3 Here the sampled signals are filtered by the output (synthesis) filters of the perfect reconstruction filter pairs (H' and G'), and combined, giving the original sequence, delayed by 3.

********** **liftfilt.m CODE CONTINUES**********

```
xlows=xlow.*s
xhis=xhi.*s
figure(2)
subplot(311)
stem(s)
axis([-1 14 -3 3]);
title('Sampling Function')
subplot(312)
stem(xlows)
axis([-1 14 -3 3]);
title('Low-Passed Sampled')
subplot(313)
stem(xhis)
axis([-1 14 -3 3]);
title('High-Passed Sampled')

ylow=filter(hp,1,xlows)
yhi=mult*filter(gp,1,xhis)
z=ylow + yhi
figure(3)
```

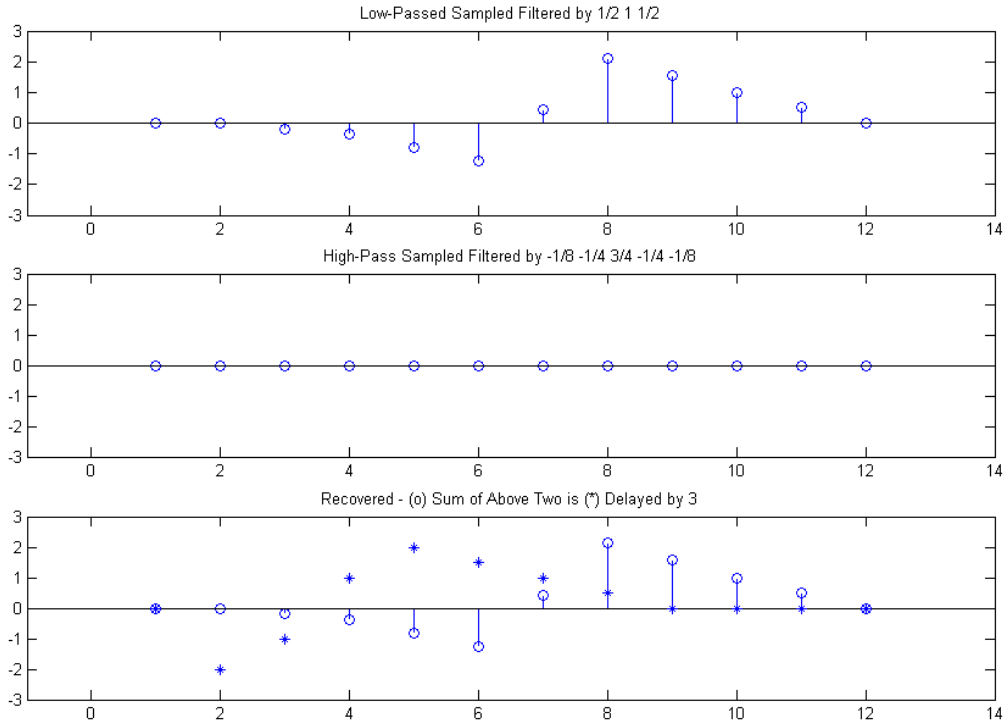


Fig F3 Here we have set the high-pass channel of the perfect reconstruction filter to zero before reconstruction The recovered sequence is: [0 0 -0.1875 -0.3750 -0.8125 -1.2500 0.4375 2.1250 1.5625 1.0000 0.5000 0] . This is the same result as the lifting result of Fig. D7.

********** **liftfilt.m CODE CONTINUES**********

```
subplot(311)
stem(ylow)
axis([-1 14 -3 3]);
title('Low-Passed Sampled Filtered by 1/2 1 1/2')
subplot(312)
stem(yhi)
axis([-1 14 -3 3]);
title('High-Pass Sampled Filtered by -1/8 -1/4 3/4 -1/4 -1/8')
subplot(313)
stem(z)
hold on
plot(x,'*')
hold off
axis([-1 14 -3 3]);
title('Recovered - (o) Sum of Above Two is (*) Delayed by 3')
figure(2)
```

*******END liftfilt.m CODE**

REFERENCES:

[1] J. Kovacevic & W. Sweldens, “Wavelet Families of Increasing Order in Arbitrary Dimensions,” IEEE Trans Image Processing, Vol. 9, No. 3, March 2000, pp 480-496. This is a central paper from which one can move forward or backward in time, with the help of a web search using terms: Lifting, Sweldens, Daubchies, etc.

[2] B. Hutchins, “Example Perfect Reconstruction Filters,” Electronotes Application Note AN-358, June 2004

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

[4] Reference [3] pp 6-8