ELECTRONOTES
1016 Hanshaw Rd
Ithaca, NY 14850
(607)-266-8492

APPLICATION NOTE NO. 358

June 2004

EXAMPLE PERFECT RECONSTRUCTION FILTERS

BASIC IDEAS

The design of perfect reconstruction filters (PRF) is well-studied [1-3] but is perhaps one of these areas where examples serve us extremely well relative to theory and analysis. We saw this [4] in a simple (the simplest) perfect-reconstruction case where the filters were sum and difference types - very crude low-pass and high-pass filters. In this case, we saw that it seemed very unlikely that the filters, based on a frequency-domain point of view, would work. On the other hand, in the time-domain view, the sum and difference devices made it abundantly clear that the perfect reconstruction worked. Accordingly, we were dissuaded of the notion that the filters had to be ideal (or even a fair approximation to ideal), based on this actual example. By using a somewhat more complicated example, we can go through the correct design procedure, and then vary some of the requirements to see which of them cause the reconstruction to fail.

The example we will give first here will actually be a four-tap filter, and we will be using an approach known as Smith-Barnwell in the literature and on the internet [5]. We have a fair amount of leeway as to where we can start the design (starting with a low-pass "prototype" usually). In fact, almost any standard low-pass linear-phase FIR design method can be tried, or simple low-pass ideas such as averaging, interpolation, or sinc-like impulse responses can be developed. Our first example is based on the equiripple method: Matlab's *remez*.

There are two necessary conditions to get perfect reconstruction prototypes. (There are additional considerations, of course, if we want useful PRF's.) The first condition is that the prototype should be "half-band." That is, the cutoff should be 1/4 of the sampling frequency. (Since signals only up to 1/2 the sampling frequency are allowed, 1/4 is half the available band.) This is equivalent to saying that all even taps of the prototype, except for the zero tap, must be zero. This is entirely equivalent to thinking about the Fourier series of a square wave as being composed of odd harmonics plus dc (See Fig. A). (The Discrete-time Fourier Transform which determines the frequency response of a discrete-time filter is a "dual" of the Fourier series.) The second condition is that the frequency response (non-causal) should be non-negative so that it can be considered a squared magnitude function (a magnitude must be non-negative).

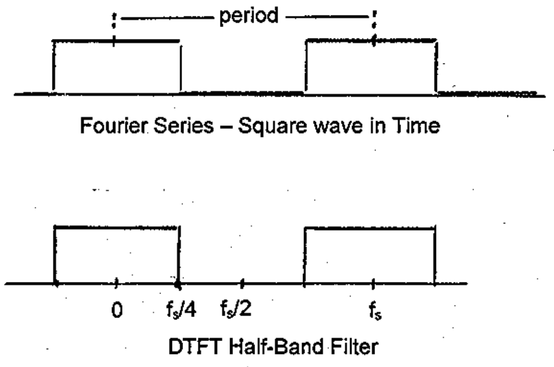

Fig. A DTFT and Fourier Series

The prototype linear-phase filter must have a length that is one less than twice the length of the desired perfect reconstruction filters. This is because we want to "factor" the prototype into two shorter filters. Here, for length 4 filters, the prototype should be length 7. Two length- 4 filters, convolved, are length 7. A length-7 filter (6 zeros) is divided into two filters of 3 zeros each.

A PRF BASED ON REMEZ

Already overdue for an example, let's start with a *remez* based design as follows:

h = remez(6,2*[0 0.225 0.275 0.5],[1 1 0 0])

$$= [\,-0.1852 \quad 0 \quad 0.3195 \quad 0.4997 \quad 0.3195 \quad 0 \quad -0.1852\,] \tag{1}$$

Here we have made the response have a cutoff centered about 0.25. The impulse response and the corresponding frequency response are shown in Fig. 1a. Note that the impulse response shown here from -3 to +3 is zero for even integer positions, except for 0. (That is - it is zero at +2 and -2.) Thus we meet the half-band criterion. What is not yet true is that the frequency response is non-negative. In fact, it is the equiripple (*remez*) filter's "claim-to-fame" that a stopband ripples about zero, hence it must go negative. So how do we make it non-negative? We need to add some constant.

The impulse response of equation (1) does not give the specific time indices of the seven samples. In computing the frequency response, we need to choose something. Our choice just determines the overall delay, and does not affect the magnitude response. If we use Matlab's *freqz*, the sequence h is assumed to be causal, and the symmetry of h about sample 4 means we have a linear phase. In this case, something like H=freqz(h,1,5000) gives us 5000 samples of the frequency response, which are generally complex. However, it is generally no problem to determining the constant we need to add to the frequency response based on abs(H). We simply search for the maximum magnitude in the stopband and this is the amount we add to the center tap of h.

It may be useful at this point to review how the frequency response is computed without Matlab. We simply compute the DTFT as:

$$H(f) = -0.1852 + 0.3195e^{-j4\pi f} + 0.4997e^{-j6\pi f} + 0.3195e^{-j8\pi f} - 0.1852e^{-j12\pi f}$$

$$= e^{-j3(2\pi)f}\,[\ 0.4997 + 2(0.3195)\cos(2\pi f) + 2(-0.1852)\cos(6\pi f)\ ] \tag{2}$$

The term in [ ] is a real frequency response (note similarity to Fourier series):

$$A(f) = [\ 0.4997 + 2(0.3195)\cos(2\pi f) + 2(-0.1852)\cos(6\pi f)\ ] \tag{3}$$

with a delay (linear phase) of 3 units. Accordingly, A(f) is the same as |H(f)| in Fig. 1a except that the portion that goes negative (dashed line) replaces the positive portion. Generally we may not need A(f) since we infer the sign changes from the abrupt changes of magnitude at the zeros. Another way is to remove the linear phase from *freqz* (see Appendix).

Having now determined how far negative A(f) goes (call this amount d), it is very clear how to stop it from going negative – you simply add d to the center tap value [see equation (3)]. It could be argued that you could add any number greater than d, and this is true. In general, we don't want to do this because the filters becomes less frequency selective (imagine adding 100 to Fig. 1a and still calling the result low-pass). On the other hand, perhaps adding a very tiny amount extra won't hurt, and may help avoid some computational peculiarities. Let's assume we have made the response non-negative, and have reached the situation of Fig. 1b. We now have a half-band prototype linear phase that could be a square magnitude function. It is more or less "mechanical" from here.

The next task is to "factor" the modified (if necessary) length-7 h into two parts, a minimum phase part and a maximum phase part. This factorization is conveniently done by finding the roots of h, which occur in reciprocal pairs because h is linear phase (even symmetric). There are 6 roots for a length-7 h, and we divide these roots into two sets (one of each reciprocal pair) and keep one set to form the low-pass filter, h0, of our perfect reconstruction pair. Typically, the minimum phase set (those within the unit circle, and one of each pair that may be on the unit circle) is chosen, although this is not required (Fig. B).

The roots of h are at:

$$\begin{array}{l} 0.4967 \\ -0.6276 - 0.7785j \\ -0.6276 + 0.7785j \\ -0.6277 - 0.7785j \\ -0.6277 + 0.7785j \\ 2.0131 \end{array} \qquad (4)$$

and the minimum phase choice are the roots

$$\begin{array}{l} 0.4967 \\ -0.6276 - 0.7785j \\ -0.6276 + 0.7785j \end{array} \qquad (5)$$

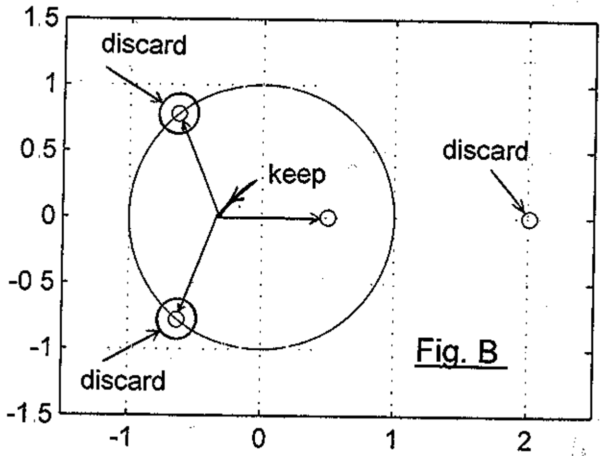

Fig. B

These three roots are combined (Matlab's *poly* for example) to obtain h_0:

$$h0 = \quad 1.0000 \quad 0.7585 \quad 0.3765 \quad -0.4967 \qquad (6)$$

As with any polynomial obtained from the roots, an arbitrary multiplicative factor is possible, so it is convenient here to adjust this impulse response for unit response at dc (divide by the sum of the h_0 terms).

$$h0 = \quad 0.6104 \quad 0.4630 \quad 0.2298 \quad -0.3032 \qquad (7a)$$

Fig. C shows the structure of the full PRF in terms of the four component filters, the low-pass analysis filter H_0, the high-pass analysis filter H_1, and the corresponding synthesis filters F_0 (low-pass) and F_1 (high-pass). The symbol S2 represents sampling by 2, and is equivalent to a downsampler and an upsampler in series. At the output of S2, even order samples are preserved, but odd numbered samples are replaced with zeros (that is, their positions remain as they were, but the samples are given value 0).

Fig. D shows the same structure in the context of this actual length-4 case. In particular, we indicate how to form the impulse response f_0, h_1, and f_1 from h_0. Accordingly:

$$h1 = \quad -0.3032 \quad -0.2298 \quad 0.4630 \quad -0.6104 \tag{7b}$$

$$f0 = \quad -0.3032 \quad 0.2298 \quad 0.4630 \quad 0.6104 \tag{7c}$$

$$f1 = \quad -0.6104 \quad 0.4630 \quad -0.2298 \quad -0.3032 \tag{7d}$$

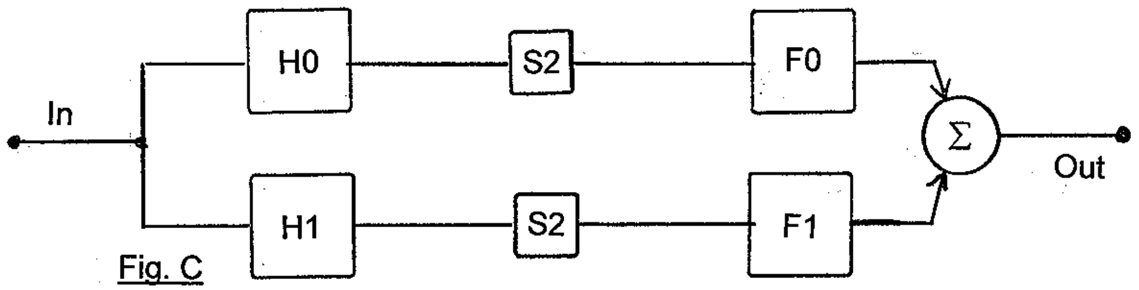

Fig. C

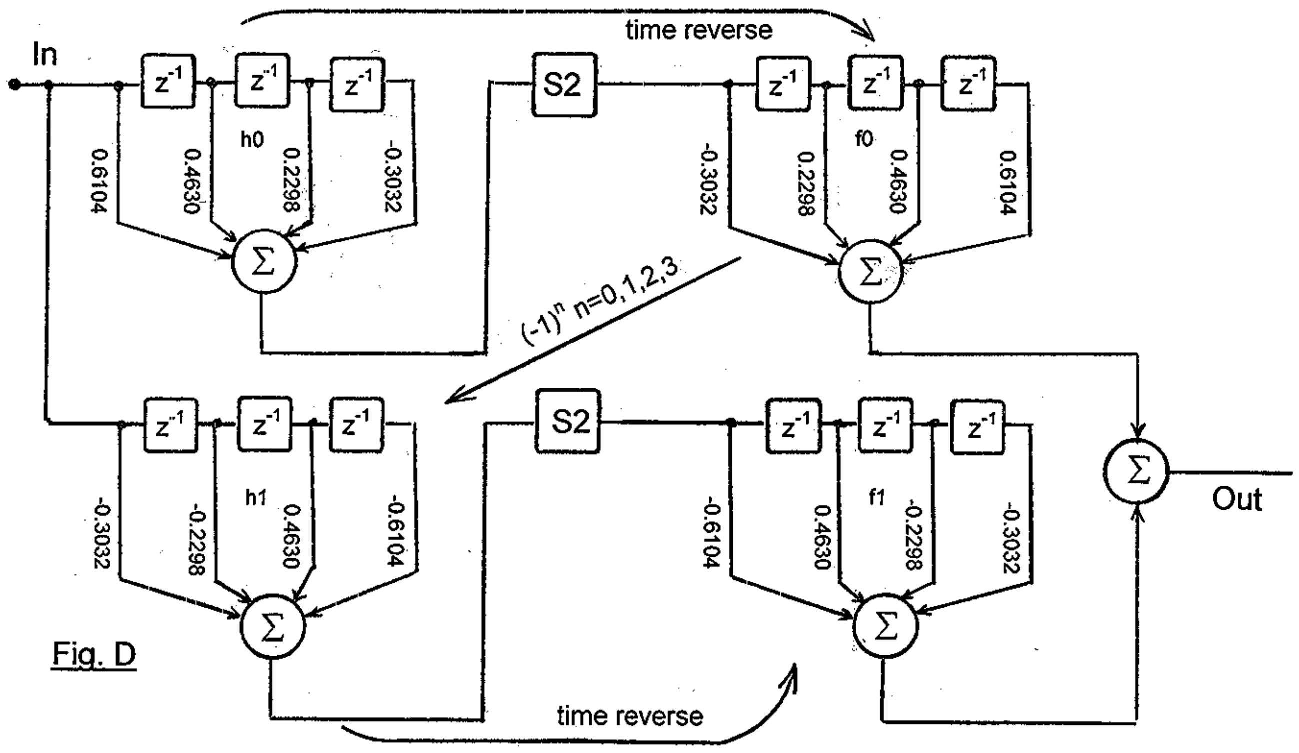

Fig. D

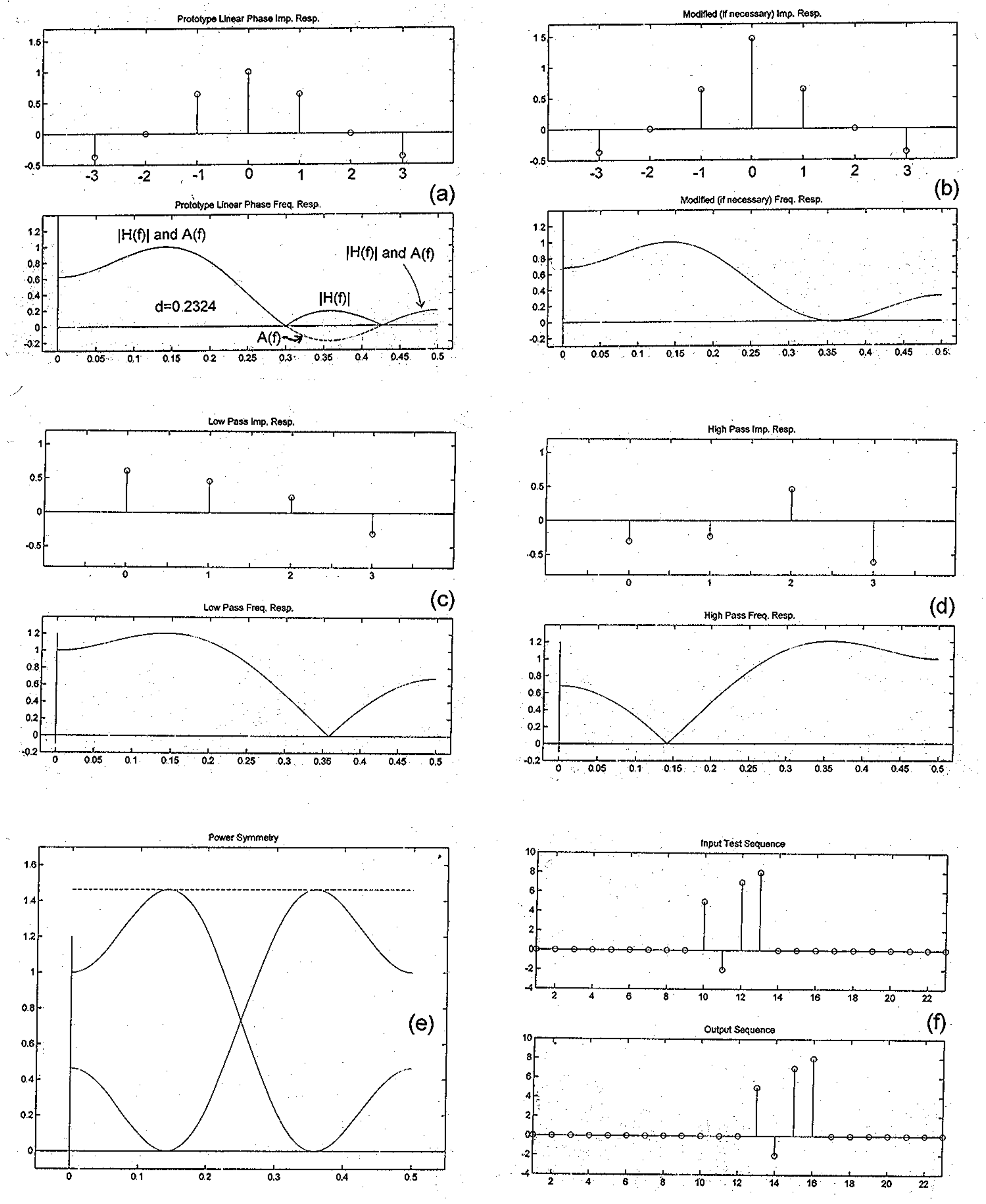

Fig. 1 Prototype equiripple (a) is made non-negative (b), is factored as in (c) and (d), and results in power symmetry (e) and time-domain PRF verification.

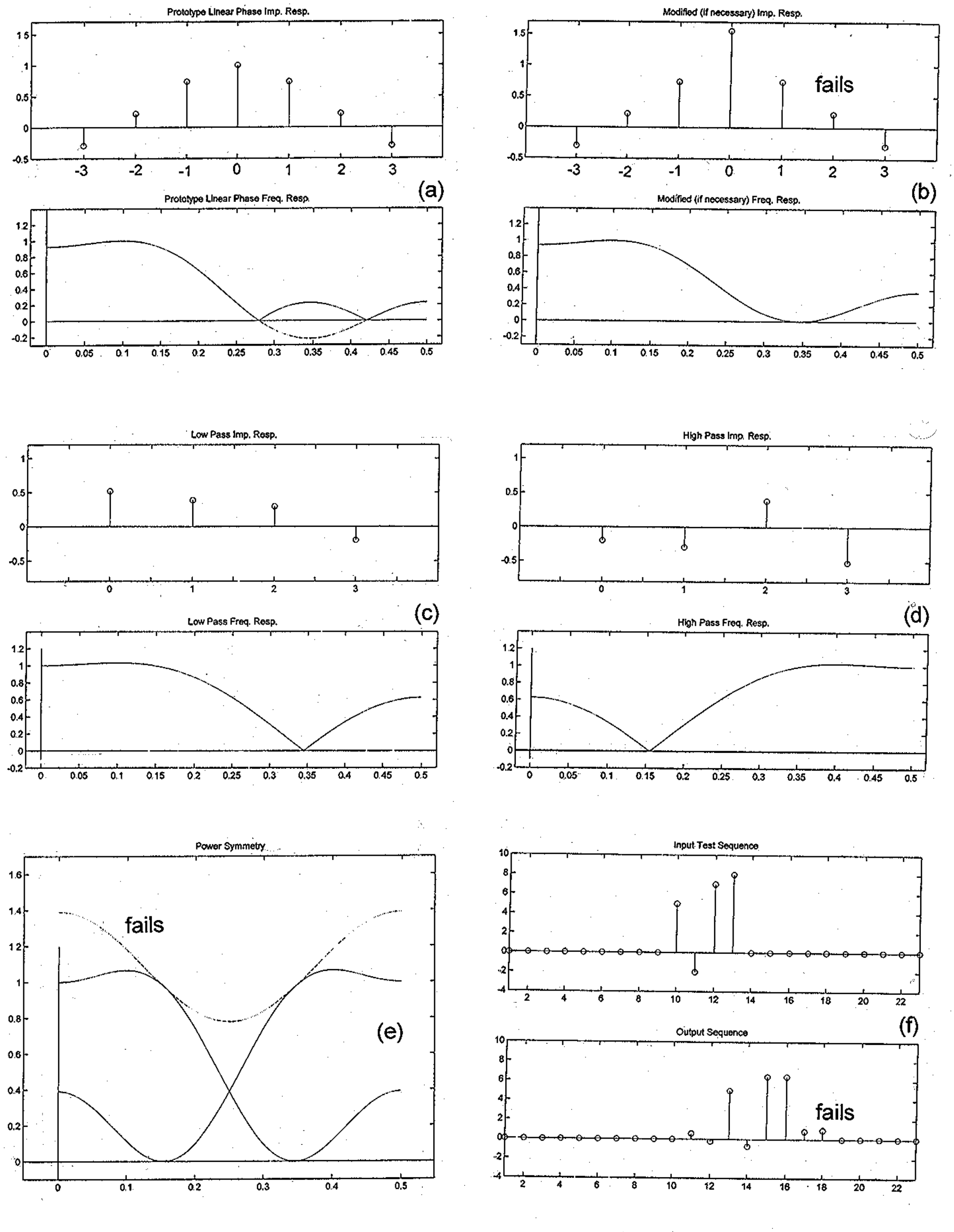

Fig. 2 When the cutoff is not set at 0.25 (0.23 used here) the design fails as is seen in (b), (e) and (f)

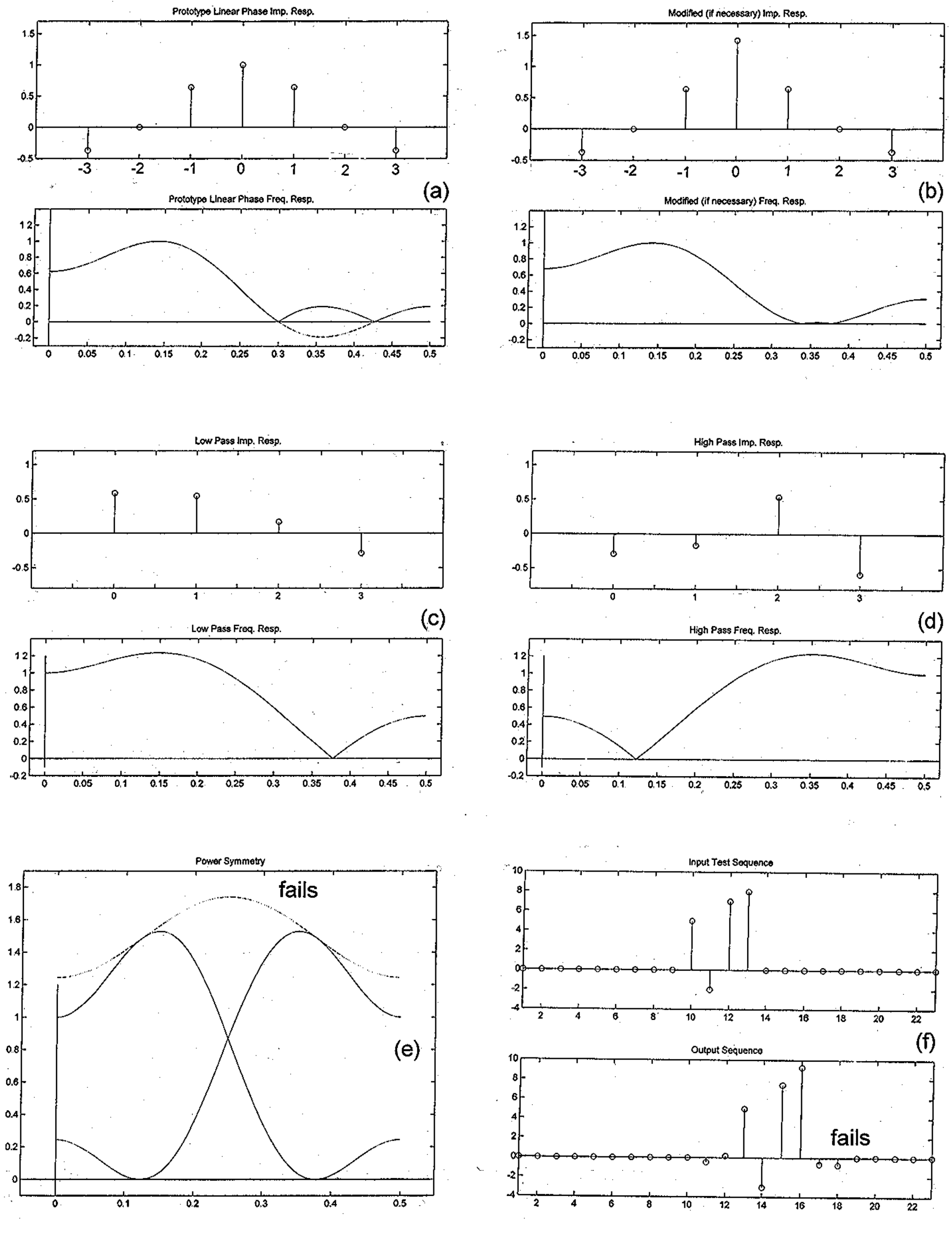

Fig. 3 If we do not add the proper constant (only 90% of the correct constant is added here) the results are seen to fail in (e) and (f)

Having already discussed Fig. 1a and Fig. 1b, we are now ready to discuss the remainder of Fig. 1, and quite promptly, the remainder of the figures. In order to keep the figures uncluttered, the axes are not labeled, but all time-domain plots are shown as "lollypops" (Matlab's *stem*) with time indexes indicated, and all frequency response plots are shown as continuous plots, with frequencies on 0 to 0.5 (half the sampling frequency). The same conventions and plot placement applies to all of Figures 1-6.

Fig. 1c shows the low-pass analysis filter, h_0, while Fig. 1d shows the high-pass analysis filter, h_1. Note that these are certainly not great filters for their types. The corresponding synthesis filters have the same frequency responses, and are not shown. One "proof" that the design is correct is Fig. 1e, which shows the "power symmetry." To get this, we square the frequency responses from Fig. 1c and Fig. 1d (shown solid in Fig. 1e) and add them (dashed line). We see that the result is a constant. Probably the most convincing proof that the design is correct is afforded by showing the results of passing a specific test sequence [5 -2 7 8] through the system. We see that it is reproduced, delayed by 3. The exact way in which this is done (and normalized) is indicated in the Matlab program that generated these plots (see Appendix).

MORE REMEZ EXAMPLES – FAILURES

Having specified the conditions and procedures, and the indications of success, we have found that a *remez* approach works for PRF. We should also be able to show that if we violate the conditions, the PRF attempt will fail. Specifically, Fig. 2 shows the corresponding results for the case where we violate the half-band condition (the cutoff is set at 0.23 rather than at 0.25). Here we immediately see a problem in that the original h is not zero at -2 and at +2. Also, the filters are not power symmetric (Fig. 2e), and we do not achieve perfect reconstruction (Fig. 2f).

The other failure is shown in Fig. 3, where the prototype frequency response is allowed to go negative. This was done by calculating the amount the center tap must be increased, but only using 90% of this value. Here the failure is again apparent in the lack of power symmetry, and in the failure of time-domain reconstruction (Fig. 3f). It is worth noting that in general this failure comes about as a result of relatively minor changes in the design parameters. In particular, the analysis filters are not all that much different for the three cases of Fig. 1 through Fig. 3.

A PRF BASED ON INTFILT – ON TO WAVELETS

We happened to choose *remez* as a method of prototype design. This is not required, and in fact, a polynomial interpolation method may well be nicer overall. Fig. 4 shows the case where we choose h as:

```
h = intfilt(2,3,'lagrange')
```

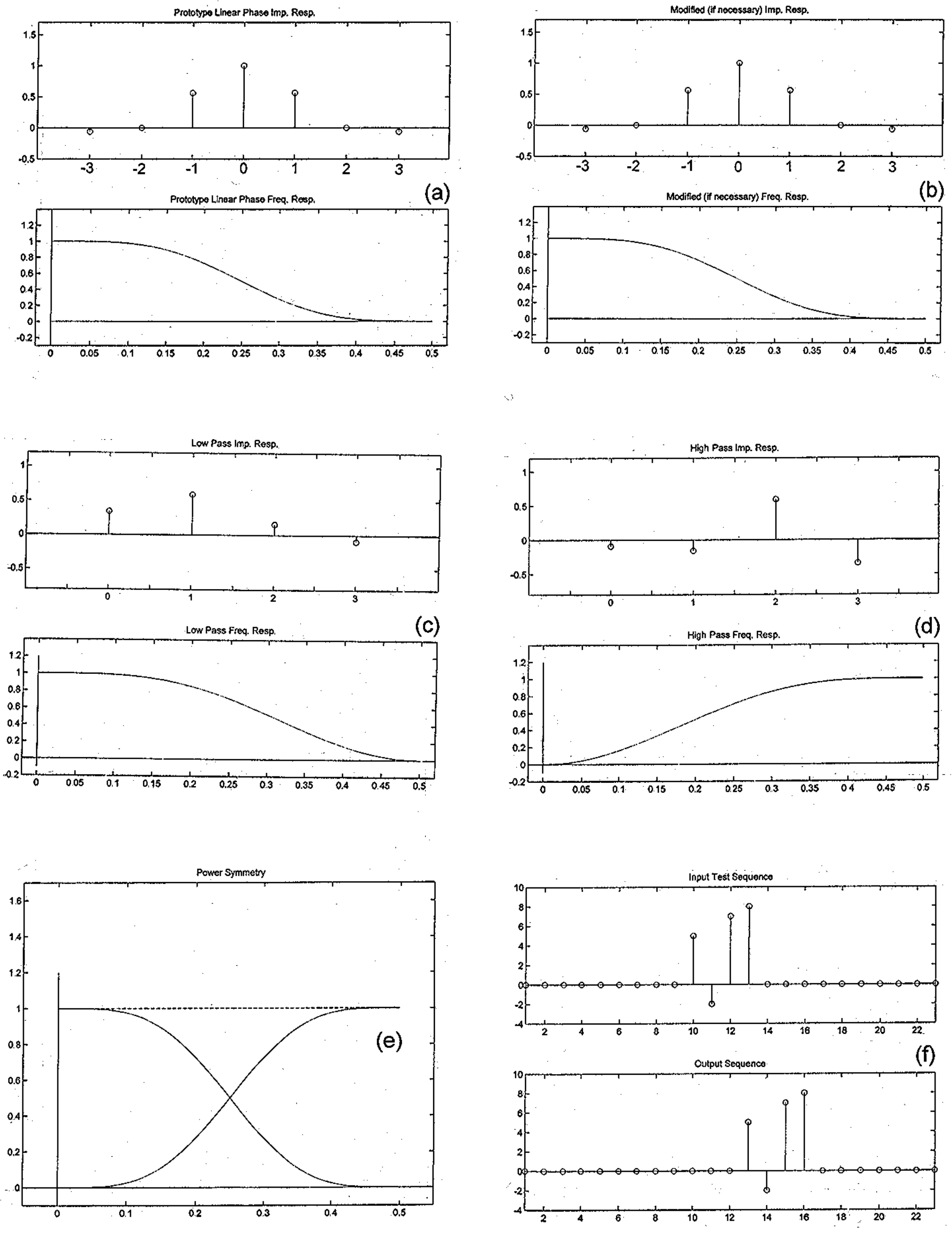

Fig. 4 This PRF design using a polynomial interpolation filter (cubic) is successful and is quite neat looking.

This is a length 7 filter, like the *remez* case. The function *intfilt* ('lagrange') gives the cubic interpolation filter for interpolation by 2 [6]. The suitability of this to the PRF problem is clear from its more-or-less automatically giving a correct prototype, and in the smooth results. Because an interpolation (by 2) filter is expected to return the original samples with no contribution except that of the original samples, the impulse response must be non-zero (one is nice) at n=0, and 0 at all even integers – exactly what we need. When we look at the frequency response, we see that it is already non-negative and needs no correction (Fig. 4a and Fig. 4b are the same). We see ripple-less results for the frequency responses of the filters, the power symmetry is agreeably obtained, and reconstruction is perfect. So this is a nice result, and it can further be seen that this case relates to familiar wavelets. Specifically the Daubechies wavelets and scaling functions (D4) can be obtained from h_0 and h_1. These come from repeated upsampling of the impulse responses and convolving with the originals. Fig. E shows some results, and the program is given in the appendix.

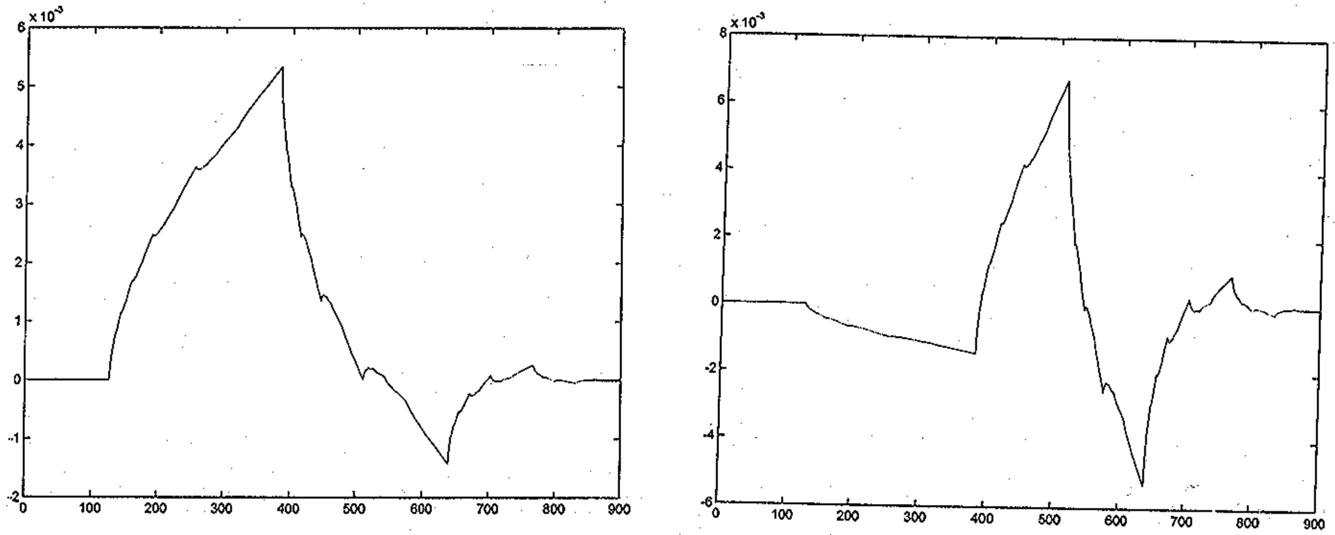

Fig. E Daubechies Scaling Function (left) and Wavelet (right)

WHY NOT LONGER FILTERS?

While we have restricted our examples so far to length 4 PRF's, we can easily modify the programs for longer (even) lengths. Fig. 5 shows a case where we start with a length 23 *remez* prototype, find the 22 zeros, factor to 11 zeros, and achieve length 12 PRF's, all with good results. In fact, for the first time, we start to see filters that begin to look like respectable half band filters. The original filter here came from:

```
h=remez(22,2*[0 0.215 0.285 0.5],[1 1 0 0])
```

and an interpolation filter:

```
h=intfilt(2,11,'lagrange')
```

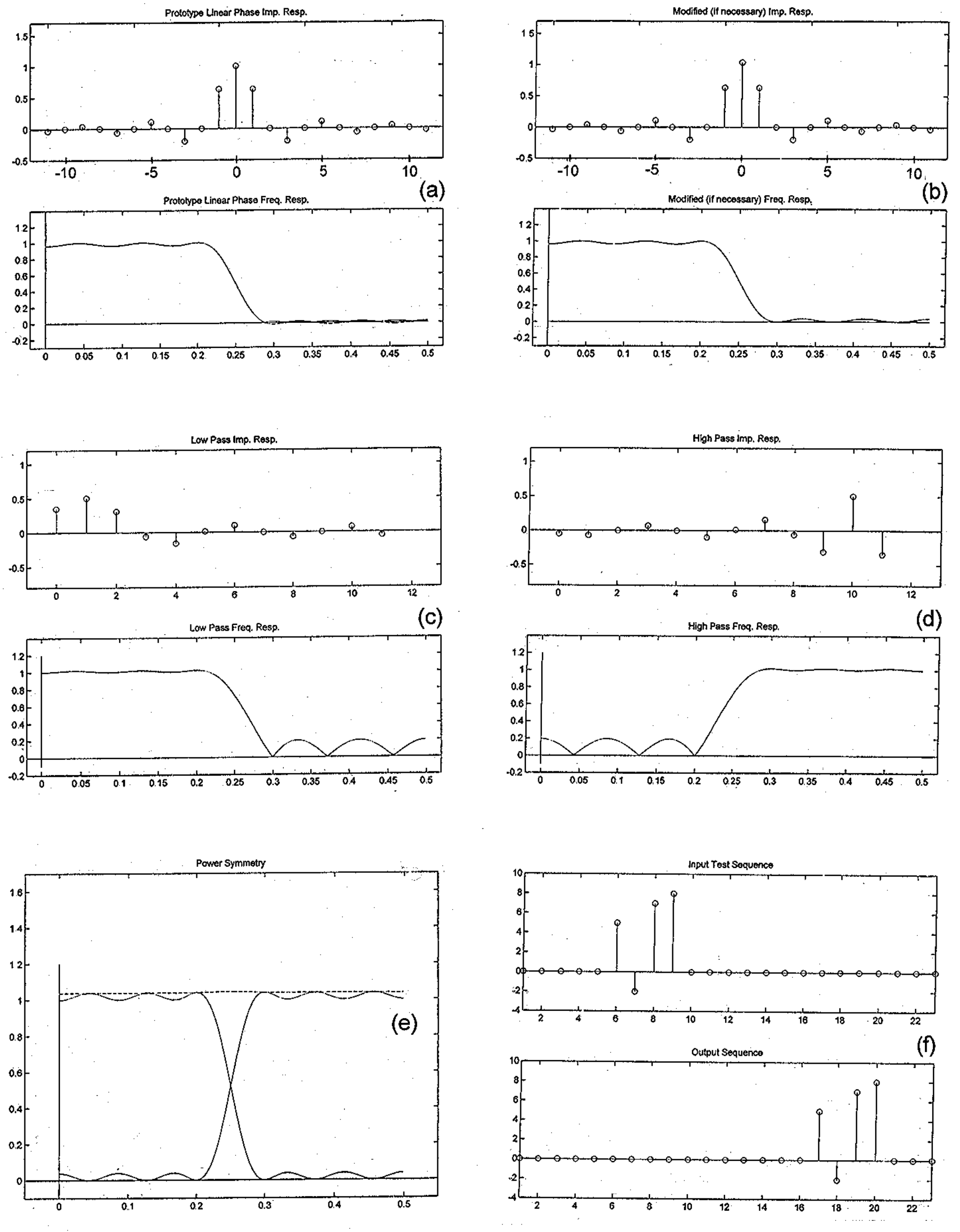

Fig. 5 A length 12 example starting with *remez*. A successful design and the filters (c) and (d) begin to look line respectable low-pass and high-pass types.

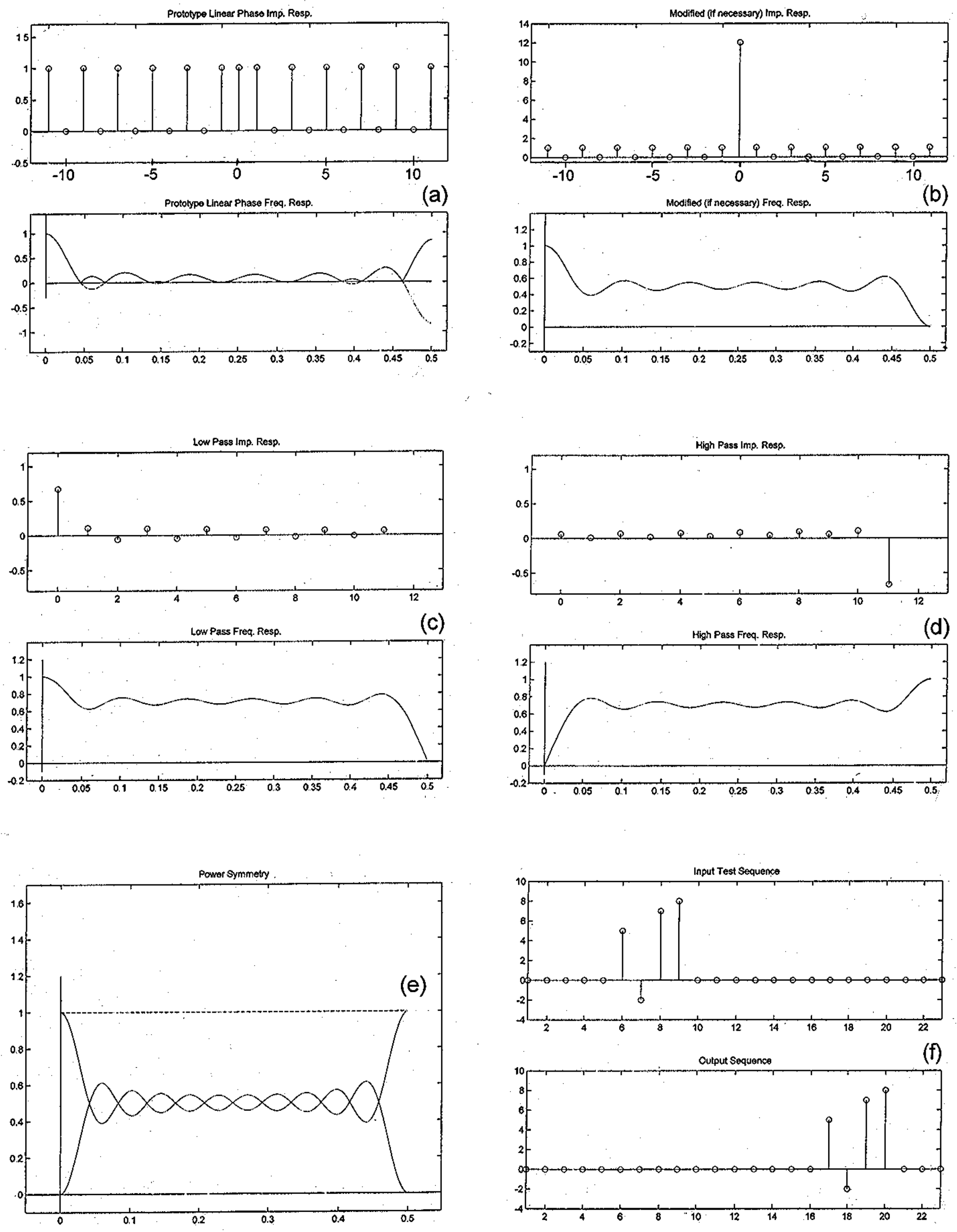

Fig. 6 A “strange” but “legal” beginning results in a successful PRF, but not one that is likely to be very useful.

would be another example. In this case, we are fitting an eleventh order polynomial to given points to do a 2:1 interpolation.

More or less just to show that it can be attempted, we will just make up a simple example prototype:

h = [1 0 1 0 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1 0 1]

for our length 23 starting point. This result does not resemble our more reasonable choices. Note from Fig. 6a that the response goes sharply negative at frequency 1/2, and that we must add on a very large amount (11) to the center term to get it non-negative. As a result, the analysis low-pass and high pass filters do not differ by much at all, except at 0 and at 1/2, and are very poor examples of their types. Nonetheless, we have not violated the conditions, and we see that power symmetry and time-domain reconstruction are exactly achieved. We get a PRF – but probably one we don't have any use for.

From this example we also get an idea why it is not a good idea to just add a very large amount (much more than necessary) to the center tap to assure the response is non-negative. In such a case, we expect the two filters to be very similar, and we have no prospect of being able to treat the intermediate signals in a different manner.

REFERENCES

[1] N.L. Fliege, Multirate Digital Signal Processing, Wiley (1994)

[2] M. Vetterli & J. Kovacevic, Wavelets and Subband Coding, Prentice-Hall (1995)

[3] P.P. Vaidyanathan, Multirate Systems and FilterBanks, Prentice-Hall (1993)

[4] B. Hutchins, "A Simple View of a Perfect Reconstruction Filter," Electronotes Application Note AN-347, April 1998

[5] M.J.T. Smith & T.P. Barnwell, "Exact Reconstruction Techniques for Tree-Structured Subband Coders," IEEE Trans. Acoustics, Speech, and Signal Processing, Vol. ASSP-34, No. 3, June 1986, pp 434-441

[6] G. Oetken, T.W. Parks, & H.W. Schussler, "New Results in the Design of Digital Interpolators," IEEE Trans. Acoustics, Speech, and Signal Processing, Vol. ASSP-23, No. 3, June 1975, pp 301-309. B. Hutchins, "Time Domain Filter Design Methods, Interpolator Based Filter Design," Electronotes, Vol. 20, No. 198, June 2001, pp 44-50

APPENDIX MATLAB PROGRAMS

SMITH BARNWELL – LENGTH-4 REMEZ – TYPICAL OF ALL PROGRAMS

```
% sb4ex1.m

h=remez(6,2*[0 0.225 0.275 .5],[1 1 0 0])
% h=intfilt(2,3,'lagrange')
h=h/max(abs(h));
figure(1)
subplot(211)
stem([0:6],h)
hold on
plot([-1 7],[0 0])
hold off
axis([-1 7 -.5 1.7])
title('Prototype Linear Phase Imp. Resp.')
subplot(212)
H=freqz(h,1,5000);
HH=real(ncfreqz(h,1,5000));
HH=HH/max(HH);
plot([0:.0001:.4999],HH,'--')
hold on
plot([0 .5],[0 0])
plot([0 0], [-0.3 1.6])
plot([0:.0001:.4999],abs(HH))
axis([-0.02 .52 -0.3 1.4])
title('Prototype Linear Phase Freq. Resp.')
hold off

% Add to center tap if necessary
d=0;
if min(HH)<-.000001
d=max(abs(H(3000:5000)));
end
d
h1=h;
h1(4)=h1(4)+d;
figure(2)
subplot(211)
stem([0:6],h1)
hold on
plot([-1 7],[0 0])
hold off
axis([-1 7 -.5 1.7])
title('Modified (if necessary) Imp. Resp.')
subplot(212)
H1=freqz(h1,1,5000);
H1=abs(H1)/max(H1);
plot([0 .5],[0 0])
hold on
plot([0 0], [-0.3 1.6])
plot([0:.0001:.4999],abs(H1))
axis([-0.02 .52 -0.3 1.4])
title('Modified (if necessary) Freq. Resp.')
hold off

% Now Factor
z1=roots(h1)
z1=sort(z1)
z1in=z1(1:3)

% analysis LP
h0=poly(z1in);
h0=real(h0)
h0=h0/sum(h0)
% analysis HP
h1=[1 -1 1 -1].*h0(4:-1:1)
% synthesis LP
f0=h0(4:-1:1)
% synthesis HP
f1=h1(4:-1:1)

figure(3)
subplot(211)
stem([0:3],h0)
hold on
plot([-1 4],[0 0])
hold off
axis([-1 4 -0.8 1.2]);
title('Low Pass Imp. Resp.')
subplot(212)
plot([0:.001:.499],abs(freqz(h0,1,500)));
hold on
plot([-.05 .555],[0 0])
plot([0 0],[-.1 1.2])
hold off
axis([-.02 .52 -.2 1.4])
title('Low Pass Freq. Resp.')
```

```
figure(4)
subplot(211)
stem([0:3],h1)
hold on
plot([-1 4],[0 0])
hold off
axis([-1 4 -0.8 1.2]);
title('High Pass Imp. Resp.')
subplot(212)
plot([0:.001:.499],abs(freqz(h1,1,500)));
hold on
plot([-.05 .555],[0 0])
plot([0 0],[-.1 1.2])
hold off
axis([-.02 .52 -.2 1.4])
title('High Pass Freq. Resp.')


% Power Symmetry
figure(5)
HLOWS=(abs(freqz(h0,1,500))).^2;
HHIS=(abs(freqz(h1,1,500))).^2;
plot([0:.001:.499],HLOWS)
hold on
plot([0:.001:.499],HHIS)
plot([0:.001:.499],(HLOWS+HHIS),'--')
plot([-0.05 0.55],[0 0])
plot([0 0],[-.1 1.2])
hold off
axis([-0.05 0.55 -0.1 1.7])
title('Power Symmetry')


% Time Domain Test
x=[0 0 0 0 0 0 0 0 0 5 -2 7 8 0 0 0 0 0 0 0 0 0 0 0];
wlow=conv(x,h0);
whi=conv(x,h1);
s=[1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0];
vlow=s.*wlow;
vhi=s.*whi;
ylow=conv(vlow,f0);
yhi=conv(vhi,f1);
z=(ylow+yhi);
z=5*z/z(13);
figure(6)
subplot(211)
stem(x)
axis([1 23 -4 10])
title('Input Test Sequence')
subplot(212)
stem(z)
title('Output Sequence')
axis([1 23 -4 10])
```

NON-CAUSAL FREQZ

```
function H=ncfreqz(b,a,N)
%  H=ncfreqz(b,a,N)
%   This function does a non-causal version of freqz.
%   It should be used principally for FIR filters.
%   In particular, even symmetric sequences will have
%      a purely real result while odd symmetric
%      sequences will have a purely imaginary result.
%  B. Hutchins
L=length(b);
omega=0:pi/N:(N-1)*pi/N;
H=freqz(b,a,N);
H=H.*exp(j*((L-1)/2)*omega).';
```

DISPLAY WAVELET

```
% ffff.m   Show Wavelets

ff=f0
fff=[];

for k=1:length(ff)
   fff(2*k-1)=ff(k);
end
fff=[fff 0]
ff=fff
ff=conv(ff,f0)

[iterate 6 lines above six more times]

ff=fliplr(ff);
figure(7)
plot(ff)

ff=f1
fff=[];


for k=1:length(ff)
   fff(2*k-1)=ff(k);
end
fff=[fff 0]
ff=fff
ff=conv(ff,f0)

[iterate 6 lines above six more times]


ff=fliplr(ff);
figure(8)
plot(ff)
```