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

APPLICATION NOTE NO. 362

December 2005

CUSTOMIZED HAMMING WINDOW WITH FILTER DESIGN APPLICATIONS

1. INTRODUCTION:

In a previous note [1] (AN-319, "Subtle Design Considerations for Hamming Windows," March 1992) we looked at a couple of refinements to the conventional Hamming window. We emphasized that these refinements were not exceedingly important, but we do learn an awful lot by looking at the issues involved, and we end up by being able to say that we have done things right (at least carefully).

In comparison to the conventional procedure of calculating the Hamming window [2,3] which results in two pairs of "lazy zeros" (four zeros close to, but not on the unit circle) a correct procedure gives all zeros on the unit circle. The first corrective measure it to assure that exactly one full cycle of the cosine is sampled (not one full cycle plus one additional point), and this puts two of the lazy zeros on the circle. The second corrective procedure, shifting samples by half a sample spacing, moves the second lazy pair to the circle. Beyond this, it is possible to choose the window parameters so that a pair of zeros is at a particular frequency. An additional interest would be to see if and how the added zeros might be transferred to the filter in the case where windowing is part of the filter design procedure.

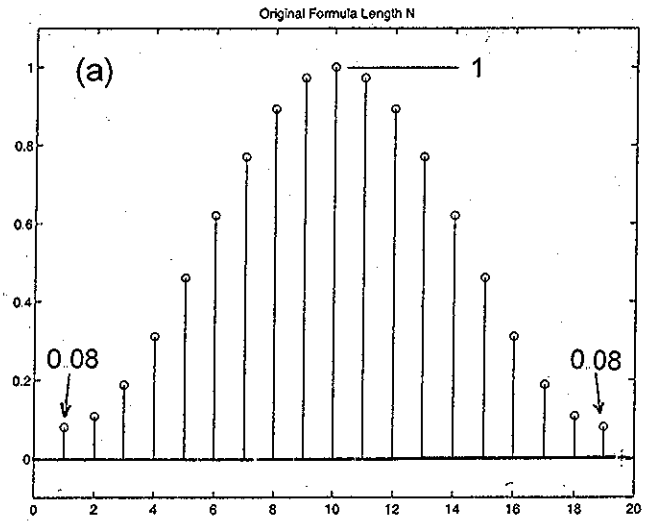
2. GETTING RID OF THE LAZY ZEROS

2a: The Conventional Hamming Window:

The conventional Hamming window definition [2, 3] suggests that the window is 92% raised cosine on a 8% "pedestal" or, as in the Matlab function:

```
w = .54 - .46*cos(2*pi*(0:N-1)'/(N-1));        (1)
```

Here we note that the cosine is evaluated at angular intervals of $2\pi/(N-1)$ from 0 to 2π. The sample at 2π is the periodic repetition of the one at 0. This definition is a logical implementation of a discrete-time Hamming window based on the sampling of a



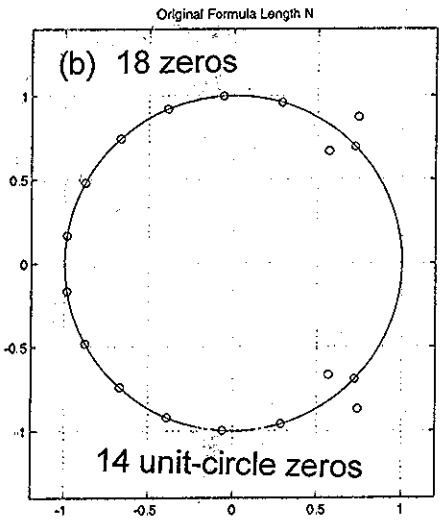
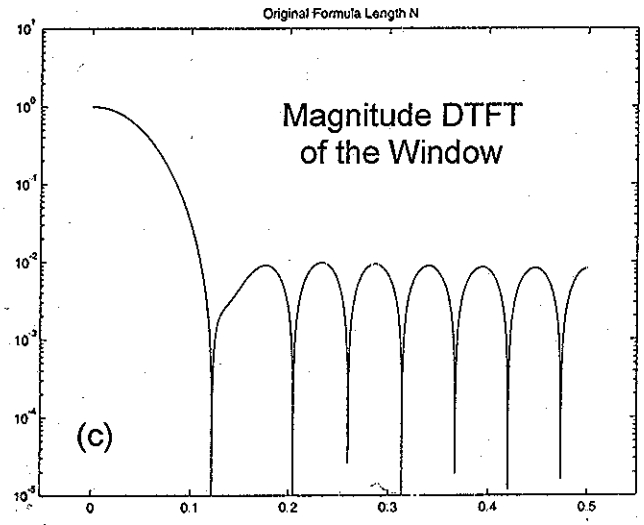


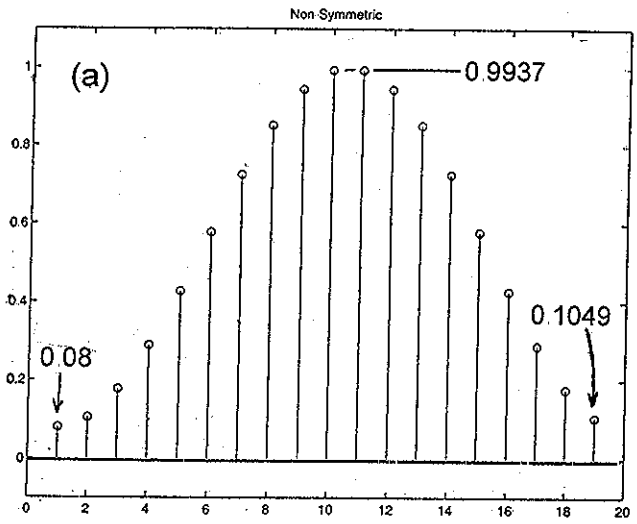
Fig. 1 The length-19 Hamming window (a) by the conventional formula has 18 zeros (b). Four of the zeros are "lazy" (not on the unit circle). These four lazy zeros account for the slight indentation in the first sidelobe of the DTFT of the window (c), but do not contribute well to the achievement of lower sidelobes. The DTFT has seven deep zeros, corresponding to the seven zeros on the top (or bottom) of the unit circle (b). Note that in (c) the first sidelobe (about 0.12 to 0.21) is substantially wider that the others.



continuous-time Hamming window, where the window parameters are set to rounded values (0.54 and 0.46), a logical enough starting compromise.

Fig. 1a shows a window calculated by this formula. Note that the end values are both exactly 0.08 and the middle value, for odd length windows such as this length 19, is exactly 1. The magnitude of the Discrete-Time Fourier Transform (DTFT) for this window is shown in Fig. 1c. We note that the sidelobes of the window peak at the 1% level that is typically quoted for the Hamming window. The sidelobes look quite normal, except for the first one (about 0.12 to 0.21) which has a curious indentation at about 0.14.

The length 19 window has 18 zeros as seen in Fig. 1b. Of these, 14 are on the unit circle while four are in reciprocal and conjugate positions (required by the linear phase even symmetry), not far from the unit circle. It is clear that these "lazy zeros" are responsible for the indentation in the first sidelobe. We consider these zeros "lazy" because they might as well be on the unit circle where they can do more good by reducing sidelobe amplitudes.



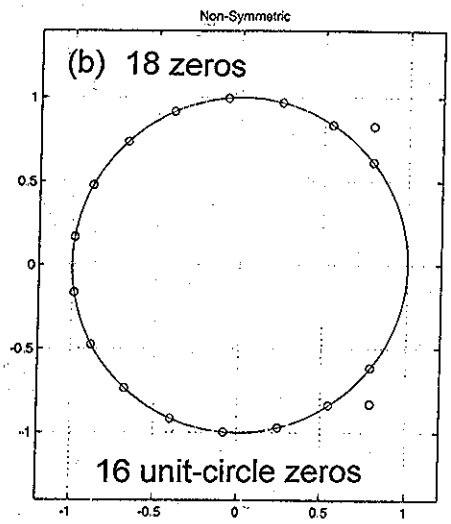
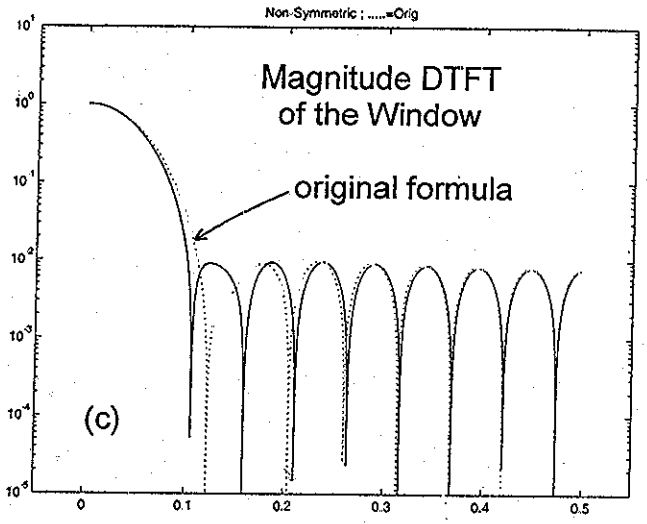


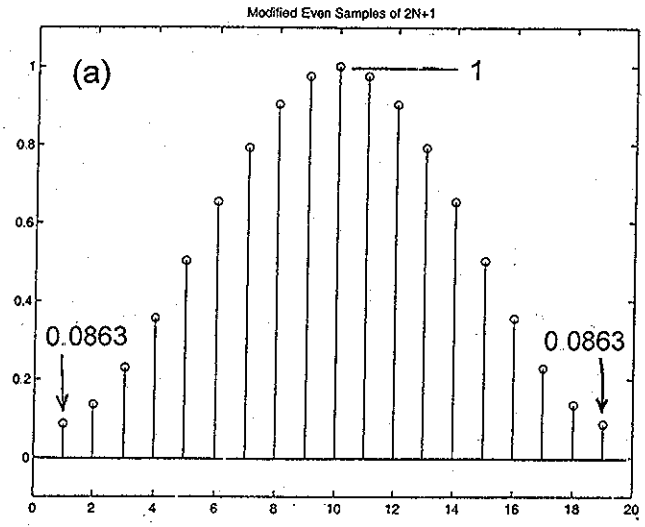
Fig. 2 Here (a) we start with a length 20 window (standard formula) and remove the last point. It now represents one full cycle of the cosines but is no longer symmetric. In (b) we see an additional pair of unit circle zeros (16 of 18 now). The DTFT show now eight deep zeros, and a narrower main lobe. Probably a better window, but the broken symmetry may bother us.



2b: Getting the Length Right

We note that one problem with the conventional definition is that we have sampled the same point at both ends. One simple cure for this fault (which is an option in some current versions of Matlab) is to design the conventional window for length N+1 and drop the last point. In fact, this does put one pair of lazy zeros to work, as we shall see. A major objection to stopping with this single repair might be that the window is no longer linear phase, and if we apply this window to the linear phase impulse response of a digital filter [4], the filter will not be exactly linear phase anymore.

Fig. 2a shows a length 19 Hamming window derived from a length 20 window by dropping the last value. Now the last value is 0.1049 instead of 0.08, and there is no center value equal to 1, but rather two value 0f 0.9937. Note that the window is no longer exactly even symmetric. Fig. 2c shows the magnitude DTFT of this new window. It has a somewhat narrower main lobe (probably a good feature) and an extra deep zero between the zeros at 0.11 and 0.21. Indeed, the first sidelobe of the original window (dotted line) seems to have been split in two. The DTFT has eight deep zeros instead of seven. In general, we see the sidelobes are still maxing at about 1%. Fig. 2b shows the zeros of the new window and we see that in fact, one pair of the lazy



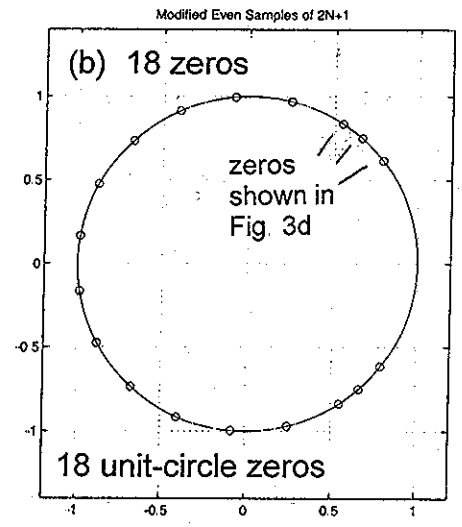
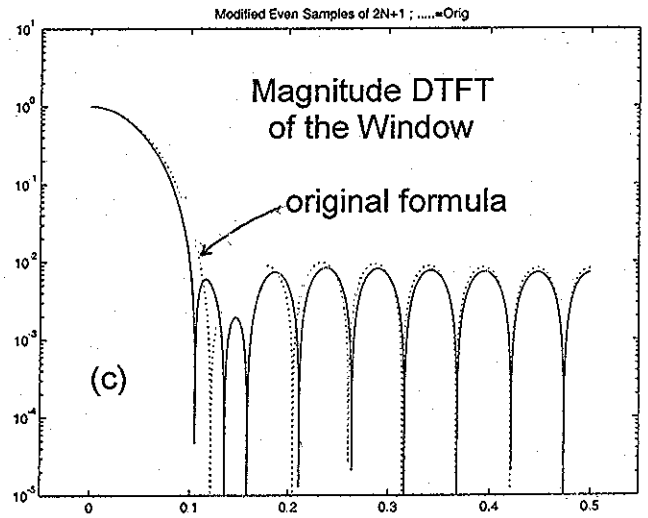
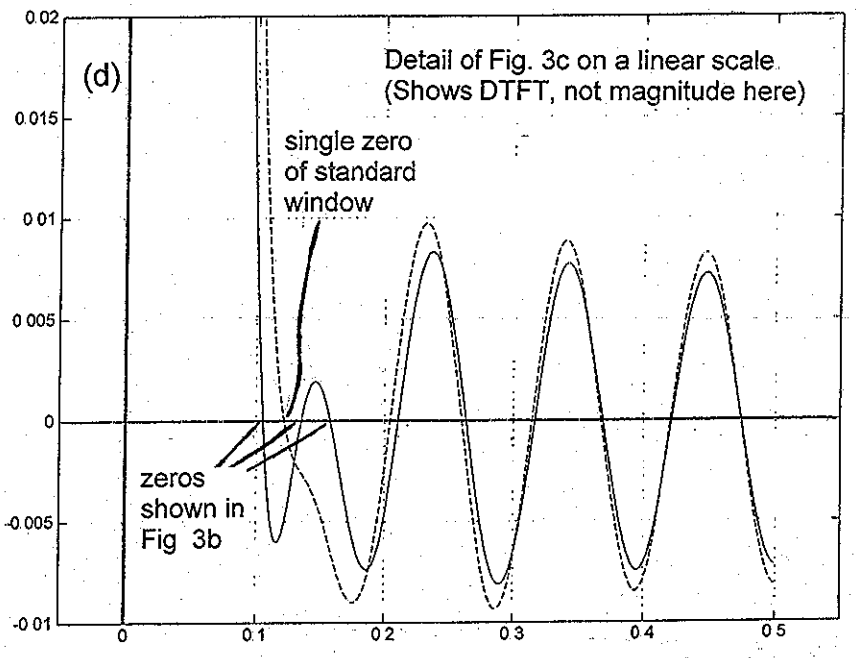


Fig. 3 Here(a) we have a length 19 window that has one full cycle of the cosine and linear-phase symmetry (achieved by shifting the sample points by half their spacing). This results in the remaining two lazy zeros of Fig. 2 moving onto the unit circle (b) and the DTFT (c) now shows nine deep zeros. A somewhat better idea of the improvement of the first sidelobe is seen in (d) which is a zoom-in on a linear scale. Notice the cluster of three zeros in both (b) and (d).





zeros has moved to the unit circle (16 on the unit circles now, with two lazy zeros remaining). The lazy zeros are un-reciprocated, a consequence of the destruction of the even symmetry linear phase of the window.

2c. Shifting by a Half Sample

It is now a simple matter to restore linear phase symmetry by shifting each sample time by half the usual sample spacing. In terms of using the original formula, this is a matter of calculating the window of length 2N+1 and then keeping every other sample, starting with the second one, for a length N result. Such a length 19 window is shown in Fig. 3a. The ends are both 0.0863, and the middle is exactly 1 again. From the magnitude DTFT of Fig. 3c, we see another deep zero has appeared, here about 0.13. In fact, in the region from about 0.1 to about 0.15, the response is below about 0.6%, and the remainder of the sidelobes are 0.8% or below, due to the slightly tighter zero spacing. Fig. 3d gives a more detailed view of the sidelobe region. The main lobe remains in its somewhat narrower form (relative to the original formula). Thus we seem to have a better window. It is not spectacularly better, but it is clearly better.

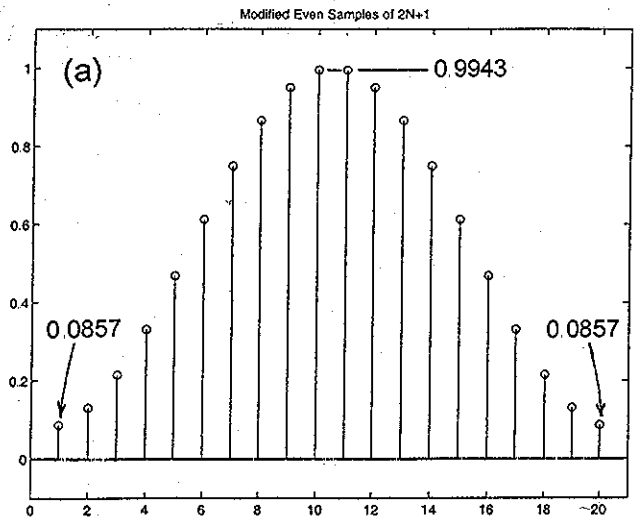
From Fig. 3b we see that now all 18 zeros are on the unit circle, so the second pair of lazy zeros has moved to useful positions, and we understand the decreased sidelobe levels in terms of the denser clustering of zeros in the regions that are about 45 degrees either side of z=1. These better results have all been a consequence of the way samples have been positioned. As yet, we have made no attempt to adjust the window parameters from their original value of 0.54 and 0.46.

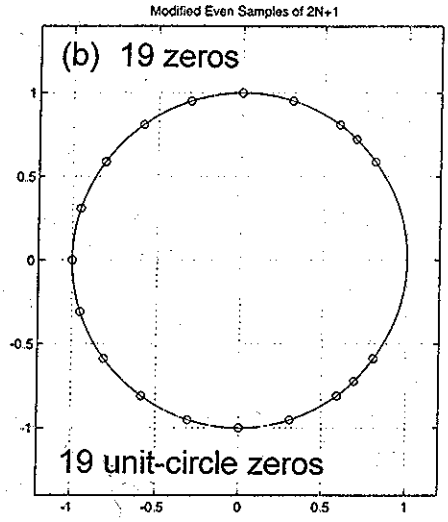
However, before leaving the issue of sample spacing and shifting, we note in Fig. 4 that the results for an even length (length 20 shown) window are essentially the same as for odd length (length 19).

3. ADJUSTING THE CONSTANT RELATIVE TO THE RAISED COSINE

In Section 2 we looked for the best way to sample the raised cosine, and we apparently found it. In consequence, we need to set this part of the problem aside - indeed, there is no actual parameter to set or adjust with regard to this part of the design. The remaining part of the design is to choose an appropriate proportions between the constant and the raised cosine, the numbers set to 0.54 and 0.46 in the original formula, equation (1). By adjusting this ratio, we get to adjust one additional performance feature - the position of one of the zeros. In particular, we choose to place one of the zeros at a specified frequency.

The Hamming window is obtained by taking one full cycle of a raised cosine. Since we are talking about a discrete Hamming window, we think of truncating an infinite duration raised cosine by multiplying it by a length N discrete rectangular window. Accordingly, the DTFT of an infinite duration (discrete) raised cosine is convolved with the DTFT of the discrete rectangular window. Both these DTFT's (all DTFT's) are periodic, so it is sufficient to convolve a single cycle. The DTFT of the raised cosine consists of three discrete values, A, B, and A (Fig. 5, Fig. 7) the A values being the "AC" term and B being the "DC" term. We want the peak of the Hamming window to be 1, so:





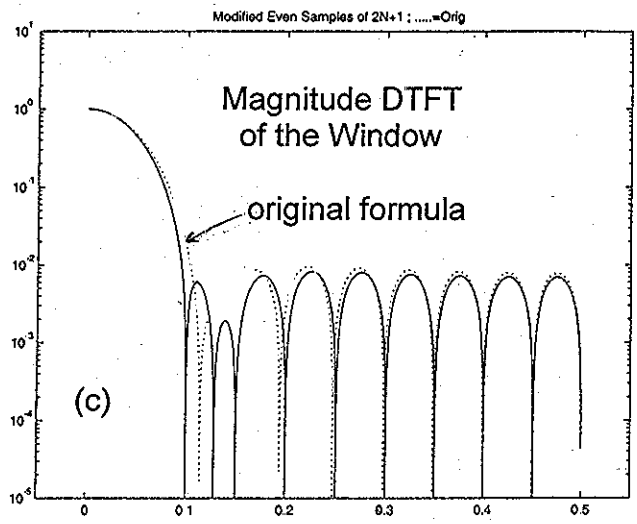


Fig. 4 Here we have a length 20 window that corresponds to the length 19 window of Fig. 3. We just needed to show that it works the same for even length. Note that the center has two values of 0.9943 instead of a single value of 1.

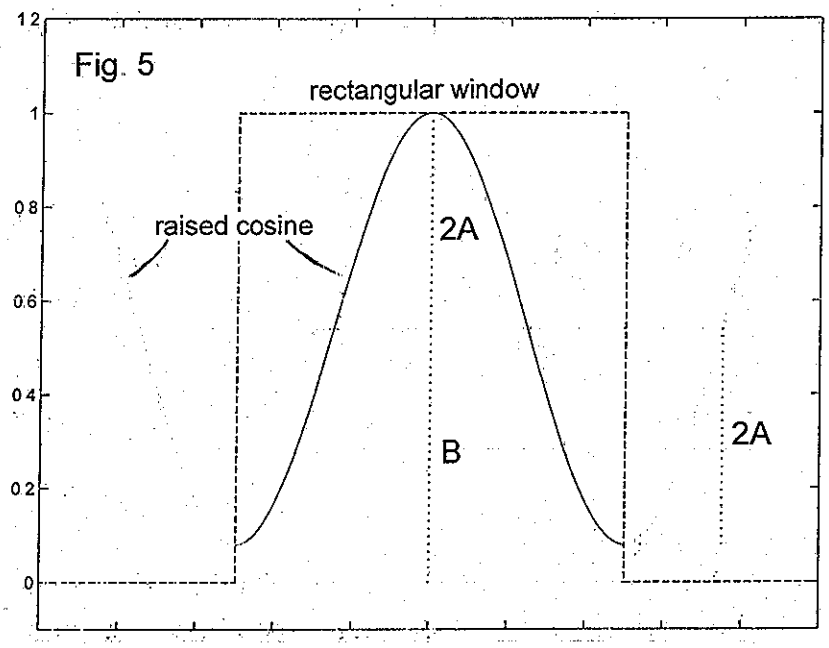


Fig. 5 A single cycle of a "raised cosine" should be viewed as a raised cosine (infinite duration) multiplied by a rectangular window. The resulting "Hamming window" has two parameters, A and B. For the standard Hamming window, 2A=0.46 and B=0.54, usually called 92% raised cosine on an 8% "pedestal." Here we allow A and B to vary, looking for a better window.

$$B + 2A = 1 \tag{2}$$

The DTFT of the rectangular window of length N is the "Periodic sinc" or "Dirichlet" function (Fig. 6):

$$H_R(\omega) = \sin(N\omega/2) / (N \sin(\omega/2)) \tag{3}$$

Fig. 7 shows the convolution of the DTFT's, which corresponds to the multiplication of the time sequences. We see that the DTFT of the Hamming window is the sum of three periodic sincs. The sum (by actual calculation, and by inspection) does not reach zero until we reach a frequency (marked b in Fig. 7) that is twice the distance to the first zero for the same length rectangular window (marked a in Fig. 7). Beyond this, we see that the sum is small, for the case we are plotting (A=0.23, B=0.54). Since the sum is small in these "sidelobes," we might well look to see if we can make a particular point in the sidelobes take on a value of exactly zero, by adjusting A and B, maintaining equation (2).

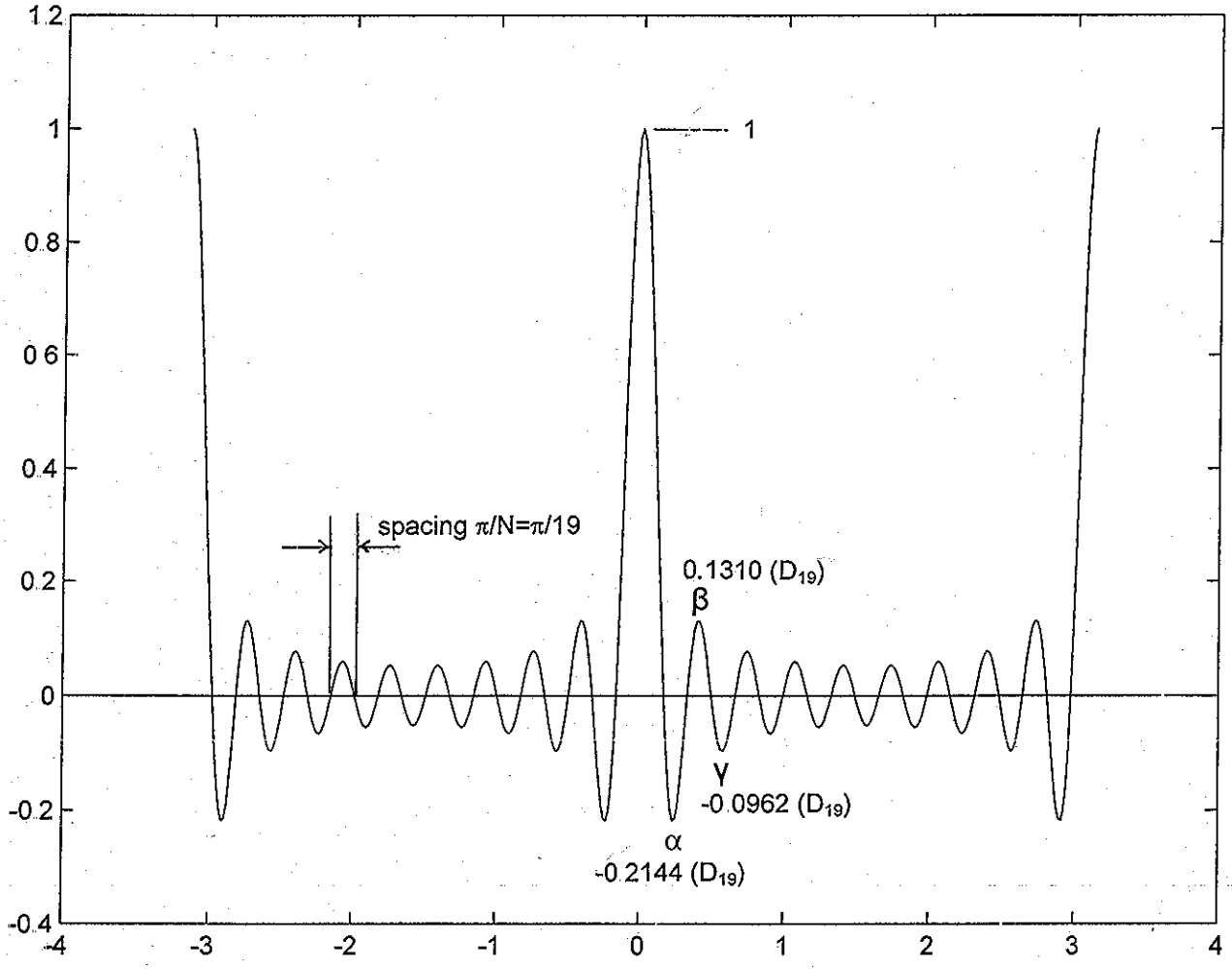


Fig. 6 In Fig. 5 we saw that the raised cosine window was obtained by multiplying a raised cosine by a rectangular window. This multiplication in the time domain of course implies convolution in the frequency domain. Thus we would convolve a sinc function (in frequency) with three "spikes" corresponding to the raised cosine. However, for the discrete Hamming window, we need to use a "periodic sinc" or "Dirichlet" function, and this is different for different lengths. Here it is shown for length 19 (note: 18 zeros in any one full cycle).

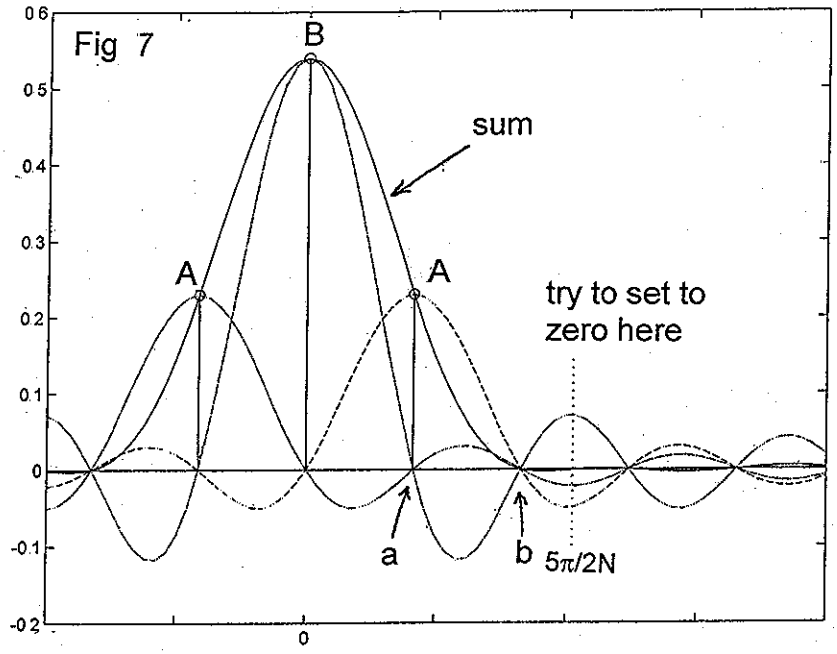


Fig. 7 In the convolution in the frequency domain, we will have three terms, the positive and negative frequencies of the raised cosine (A) and the constant (dc) term (B). The convolution "sum" is twice as wide as the DTFT of the rectangular window, but beyond that, the sum can be small (above) and we can look to manipulate A and B to place a particular zero at a desired frequency.

The most logical frequency to set to zero (to begin with at least) would be the center of the first sidelobe, a frequency of $\omega=5\pi/2N$. Using Fig. 6 and Fig. 7, we note that the contribution of the three convolved Dirichlet functions sums to:

$$B\beta + A\alpha + A\gamma \rightarrow 0 \qquad (4)$$

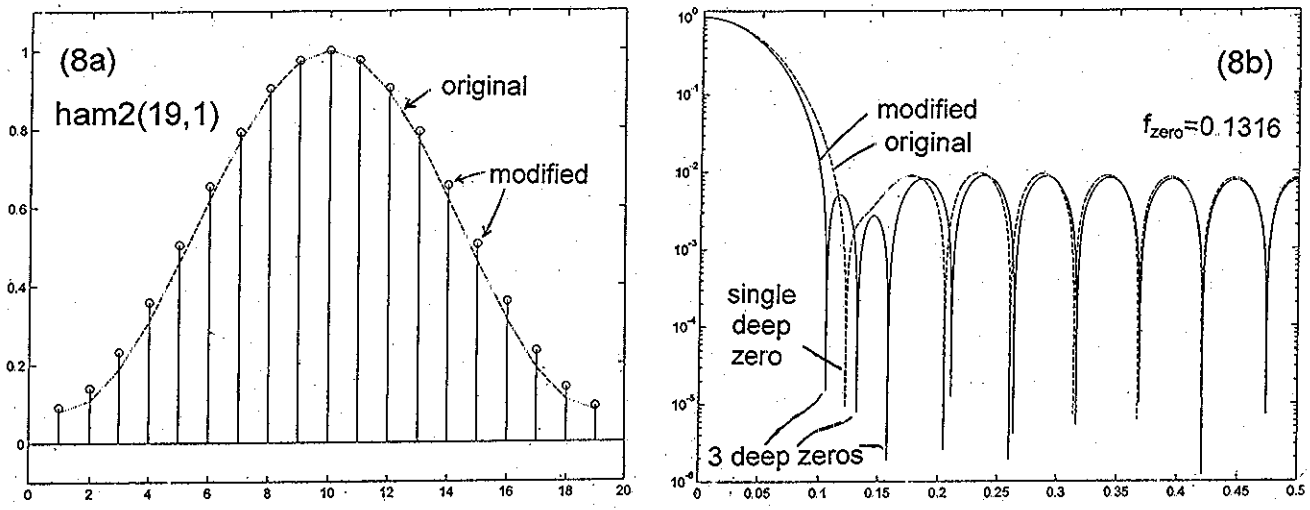
In the case where the length of the rectangular window is sufficiently long (perhaps greater than 10 or 20), the values of α, β, and γ closely approach those of the amplitudes at the centers of the sidelobes of a continuous time sinc.

$$\begin{aligned} \alpha &= -0.2122 \\ \beta &= 0.1273 \\ \gamma &= -0.0909 \end{aligned} \qquad (5)$$

In the case where we want to be exact, we can use the amplitudes at the centers of the sidelobes of the appropriate Dirichlet function [Fig. 6 for length 19, using equation (3)].

$$\begin{aligned} \alpha &= -0.2144 \\ \beta &= 0.1310 \\ \gamma &= -0.0962 \end{aligned} \qquad (6)$$

Equation (4), along with equation (2), are then solved for A and B. For the continuous time sinc, we have A=0.2283 and B = 0.5434, rounding to the conventional 92% raised cosine on an 8% DC pedestal (see Fig. 5). For the length 19 discrete window, the results are very close:



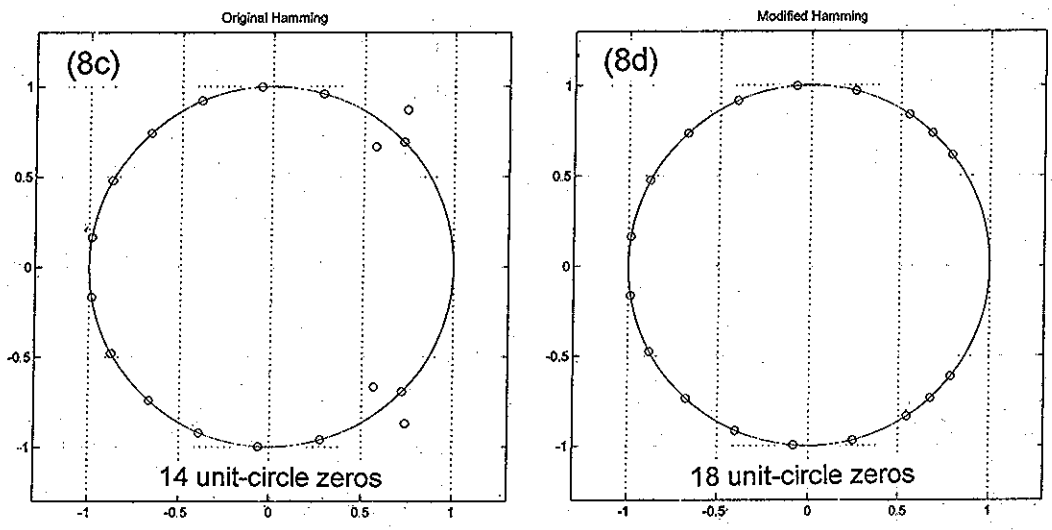


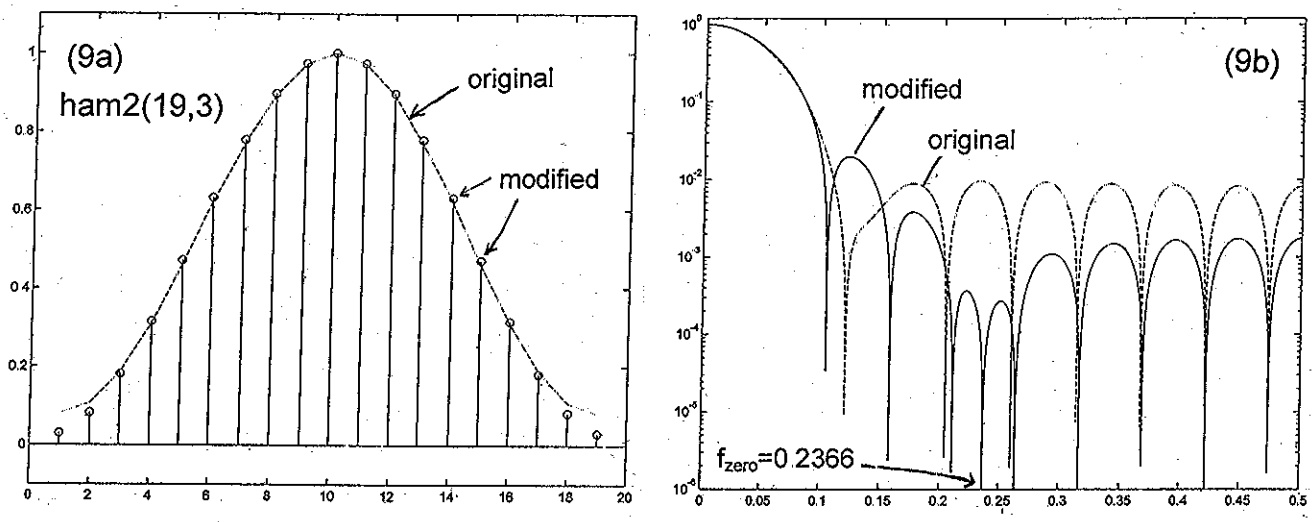
Fig. 8 Modified window with an improved first sidelobe results from forcing zeros to the unit circle Note the cluster of three deep zeros.

A = 0.2288
B = 0.5424 (7)

This gives us our modified Hamming window. To be clear, here we have adjusted the timing and spacing of the samples to obtain all unit circle zeros, as in Section 2, and now we have further optimized the parameters to place the half-shift-induced zero to a specific frequency. Some examples will follow the presentation of some programs.

4. PROGRAM AND EXAMPLES

The Matlab program included here is named ham2.m. [An earlier version, named ham1.m was included in AN-319.] This program designs a modified Hamming window h of length N, with an additional input parameter r. The program first looks at the parameter r, and if r does not exceed 1/2, it assumes that r represents the frequency (on the interval 0 to 0.5, for a sampling frequency of 1) at which the extra zero is to be placed. If r is greater than 1/2, the program assumes that the extra zero is to be



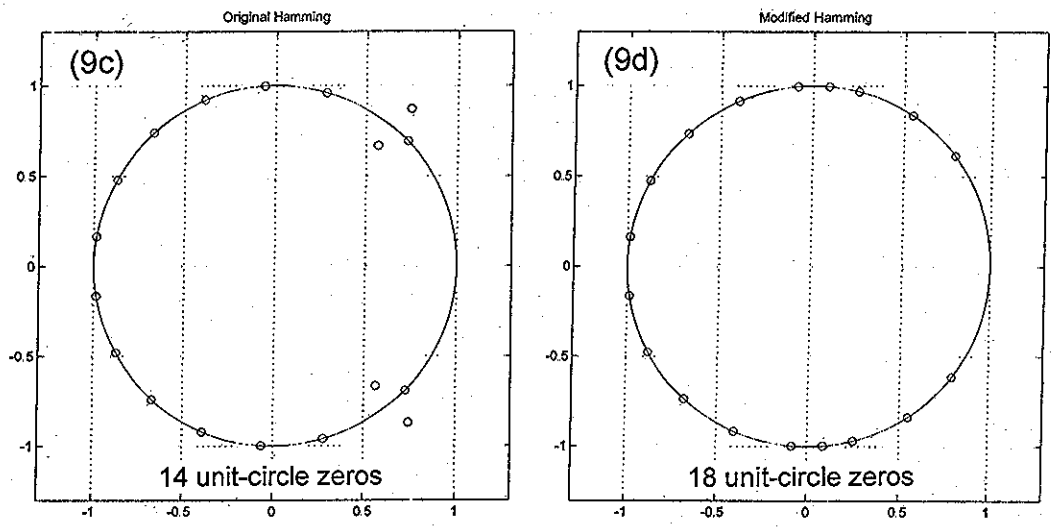


Fig. 9 Here the imposed zero is put in the third side-lobe. Note the deeper rejection in the general region of 0.25.

placed in the center or a sidelobe. For example, we often would make r=1 to indicate that the zero is to be the center of the first sidelobe. (Omitting the input parameter r will default to the center of the first sidelobe.)

Normally, we think of r greater than 1/2 as being a sidelobe with integer value. However, this is not required. Values of r between 0.5 and 1.5 will move the zero down or up relative to the center at 1. The exact frequency is printed out as fzero. Thus specifying the zero in terms of sidelobe is a matter of convenience, but does much the same thing as setting the frequency with r less than 1/2. One case where this sidelobe specification is very useful is when we want to shift the zero slightly off (usually below) center. This we will look at below, where we see that a sidelobe of 0.94 is a useful choice when we want to minimize the energy in the first sidelobe. If we had to have only one choice for r, we could argue that it should be 0.94. Thus h = ham2(N, 0.94) is possibly a better default than ham2(N) or ham2(N,1), the last two being equivalent.

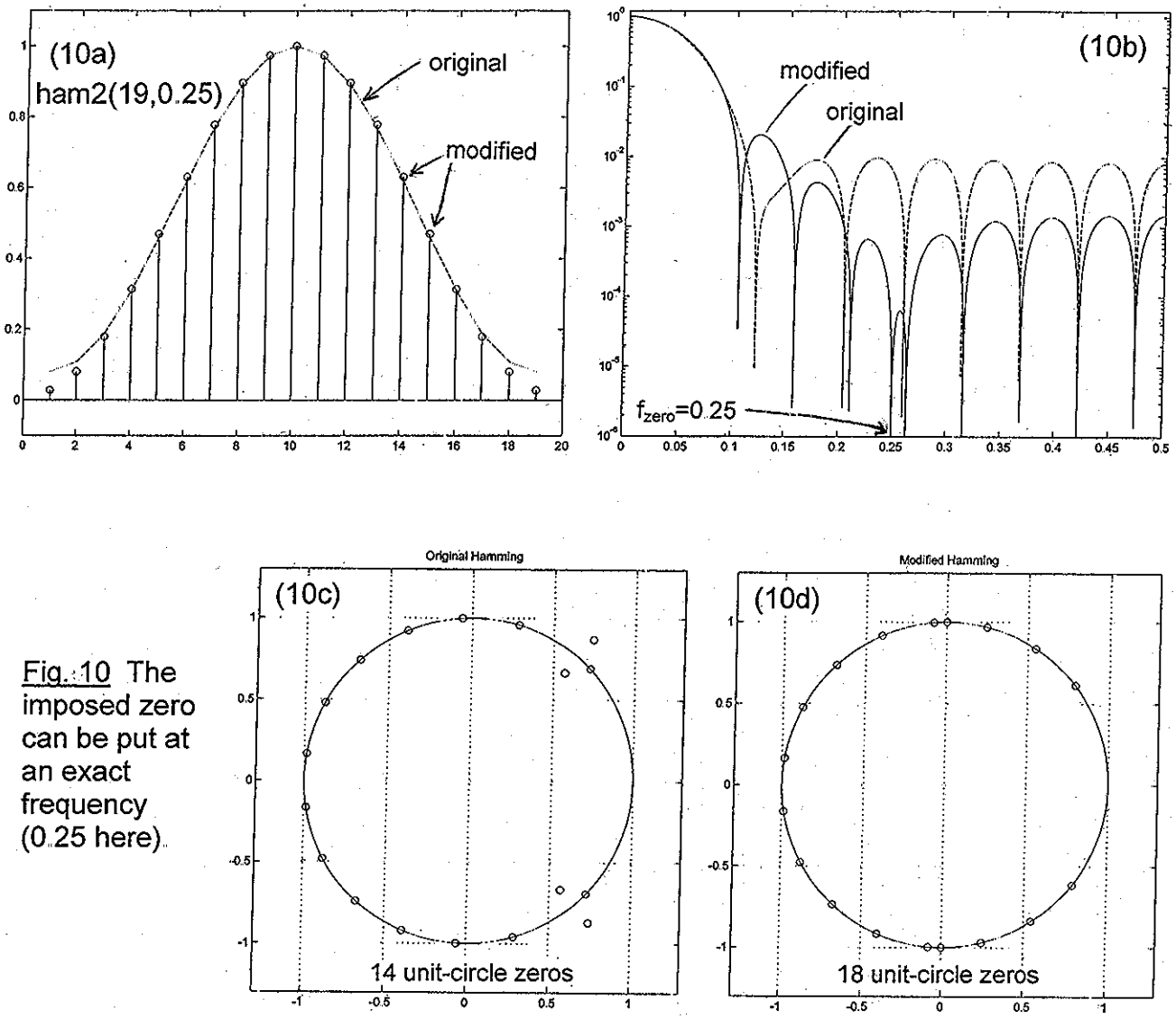


Fig. 10 The imposed zero can be put at an exact frequency (0.25 here).

In addition to computing the modified Hamming window, the program includes various displays, and comparisons to the standard Hamming window. These are not essential to the computation of course, but it is felt that one point about having a modification is to clearly state the original case. It will be noted from the display (figure 1 of the program) that the time domain window is relatively little changed from the standard window, except at the ends. The second display (figure 2 of the program) shows differences in the DTFT's of the two windows. Note that the modified window cuts off faster, and that there is an overall lower sidelobe level in the vicinity of the added zero, and of course, there is the added zero (at the frequency fzero). The final two displays are the zero plots in the z-plane for the two windows, and we note that the "lazy zeros" (figure 3 of the program) move to the unit circle for the modified window (figure 4 of the program). The program ham2.m calls a program pzplot1.m, which is also printed here. (Over various revisions, there have been different "pzplot" programs that one may encounter in Matlab.)

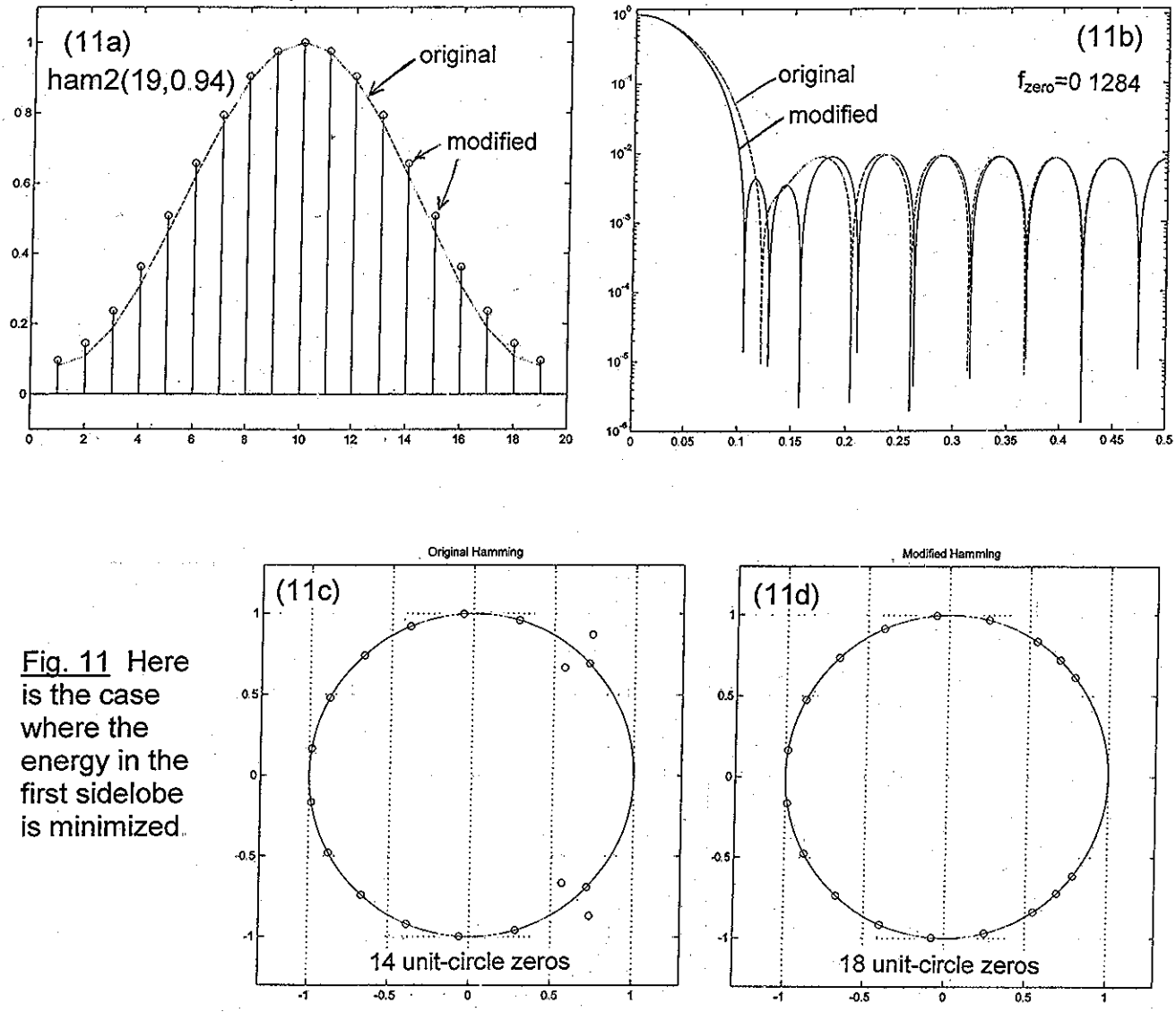


Fig. 11 Here is the case where the energy in the first sidelobe is minimized.

Fig. 8 shows a modified hamming window corresponding to ham2(19,1) , where the added zero is in the first sidelobe, with the window shown in 8a, the DTFT in 8b (f_{zero}=0.1316), and the zero plots in 8c and 8d. Fig. 9 shows a similar set of plots for h=ham2(19,3) where the added zero is in the third sidelobe (f_{zero}=0.2368). Fig. 10 shows plots corresponding to ham2(19,0.25) which places a zero exactly at the frequency 0.25, somewhat higher than the middle of the third sidelobe.

Above, when we considered placing the extra zero in a sidelobe, we thought about the center of that sidelobe (Figs. 8, 9) or at a particular frequency (Fig. 10). This is logical enough, but additional thought may suggest that we might want to choose some other performance criterion. Possibly we might want to choose a position for the extra zero that equalized the split sidelobe. A bit of trial and error shows that this occurs with r approximately 0.92 [try ham2(19,0.92)]. Another rational thing to try might be to minimize the energy in the sidelobe. This is a bit more complicated to investigate, but when done, the value of r of 0.94 seems to be the result (0.95 if the length is more like 12). A modification of ham2 that searches a range of r and sums the energy of the first sidelobe is program ham7 in the appendix. Fig. 11 shows the results of the case of ham2(19,0.94).

5. THE EFFECT ON FILTER DESIGN - THE REAL TEST

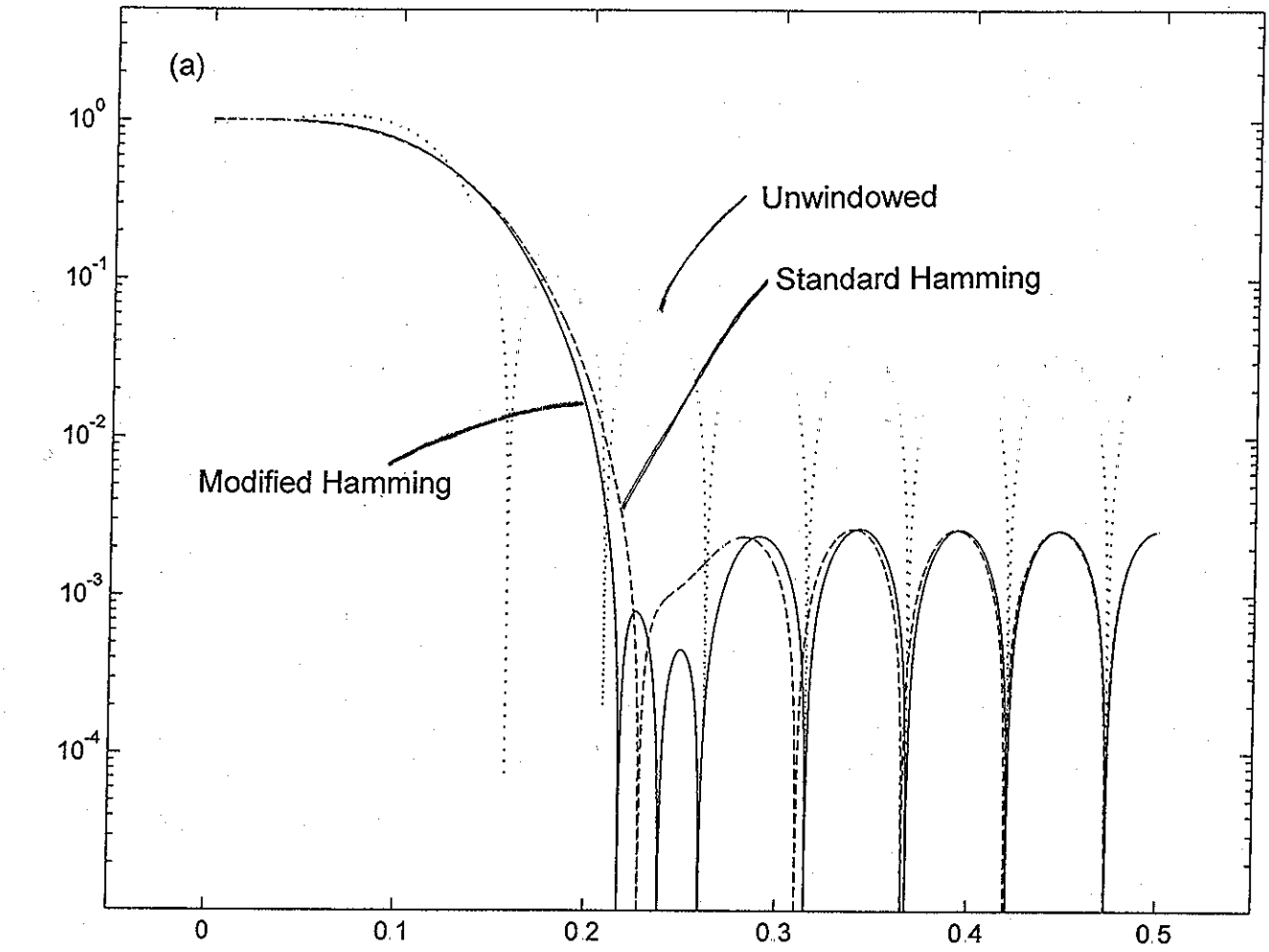
The real test here is not so much how well the modified window performs on its own, but how well it performs in an actual application. We have seen that our modified windows are able to get rid of "lazy zeros" through the use of correct sampling of the continuous-time window, and we are further able to manipulate the window parameters (A and B) to improve sidelobe rejection. Do these improvements carry over once we make an application?

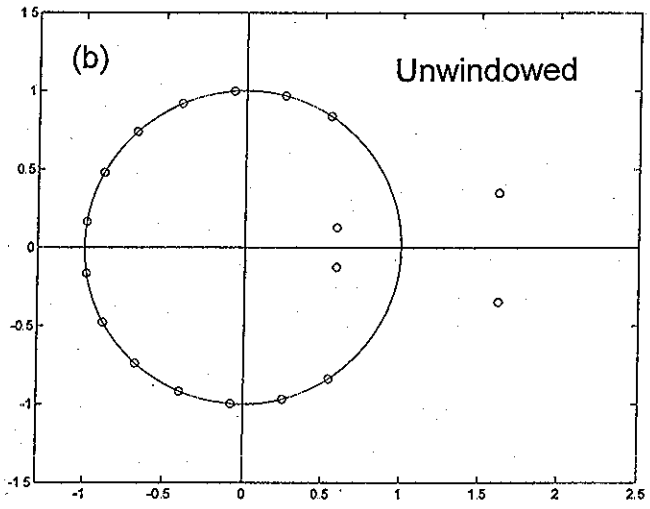
Windowing is often used for data acquisition, and in filter design, and we will look at the latter application here. Typically we make an initial design using inverse DTFT, and then multiply the initial impulse response ("taps"), point-by-point, by a Hamming (or other) window of the same length, to taper the taps at the ends [4]. The main goal of this windowing is to "smoothen" the passband and stopband "ripples." Usually it is the reduction of stopband ripple that is of the most interest, as this corresponds to better rejection in the stopband - more or less the filters main function. At the same time, there is an "engineering trade-off" in that the roll-off rate in the transition band (from passband to stopband) is more gradual, and the filter is less "sharp."

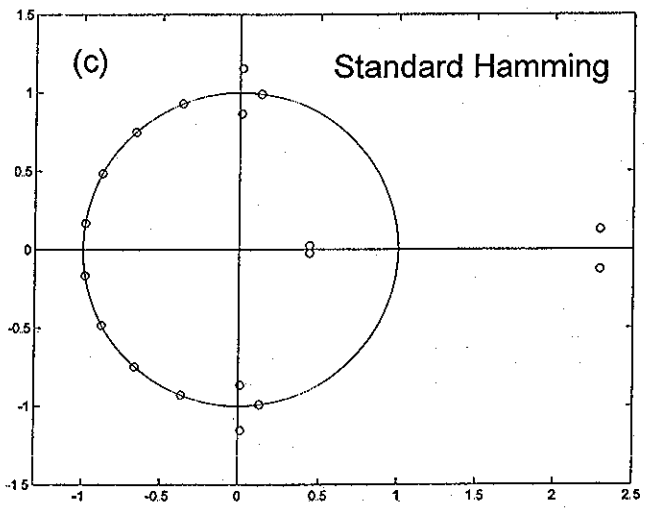
With regard to stopbands, unit-circle zeros are of particular interest because they represent frequencies where the response is exactly zero, and because a dense clustering of zeros should correspond to superior rejection. In particular, we would want to see if any "lazy zeros" are coaxed into helping out more by moving to the unit circle. Does the modified window help?

The answer is clearly "yes" since we almost always get what we can argue is a better filter when we use a modified window relative to use of the standard window. We may not get additional unit-circle zeros in all cases. In our example filter designs, our starting point will be a length N FIR low-pass with cutoff set to 1/8 the sampling frequency, and inverse DTFT will be used. Fig. 12 shows our first example, length 19 (N=19), and we show the original unwindowed filter, and designs using the standard and modified windows. The test program is hamtest.m in the appendix.

Fig. 12(a) shows the frequency response (magnitude DTFT) of the three filters. First note the unwindowed response (dotted line). This is not a very good filter. It does have a cutoff that might be 0.125 (1/8) but its first sidelobe (at about 0.18) is only down to about 10%. But the contest of interest here is between the standard Hamming window (dashed line) and the modified Hamming window, using ham2(19,1) (solid line). Note that both these have a less sharp cutoff (observe for example, the rolloff between 10^{-1} and 10^{-2}) relative to the unwindowed case. Both windowed cases do end up below 0.003. Which of the two is better? Well, except for a small region around frequency 0.228, the modified window case is clearly as good or better than the standard window. This is certainly because of the extra deep zeros (see region from about 0.22 to 0.26).







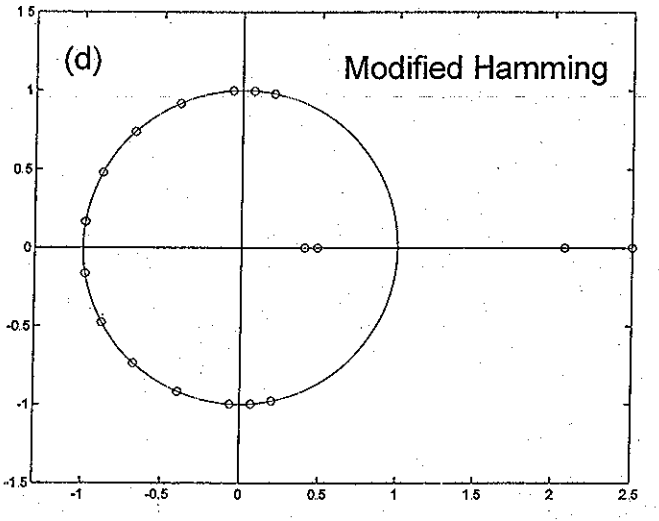


Fig. 12 The real test is to see if the extra unit-circle zeros in the window will also appear in a filter designed using the improved window. Here we see the extra zeros forming a slightly better first sidelobe in the stopband.

Fig. 12(b) shows the zero plot of the original unwindowed filter. Here we see the four reciprocal zeros typical of a passband, and 14 unit-circle zeros typical of a stopband. With windowing (c, d), the zeros rearrange, basically pulling back from the cutoff region (which is at a ±45 degree angle with respect to the positive real axis). The difference is that in (d), two pair of lazy zeros move to the unit circle. This accounts for the improved rejection in the general region of 0.25. Not all the cases give us this nice extra zero.

In Fig. 13, we have a case corresponding to Fig. 12, but the length was increased to 21 (from 19). We don't get extra zeros, but from Fig. 13(a) we see, none the less, the response in the first sidelobe is improved. We see this by comparing Fig. 13(c) and Fig. 13(d), and we see that use of the modified window [ham2(21,1)] the lazy zeros have moved toward the unit circle, although they have not reached it.

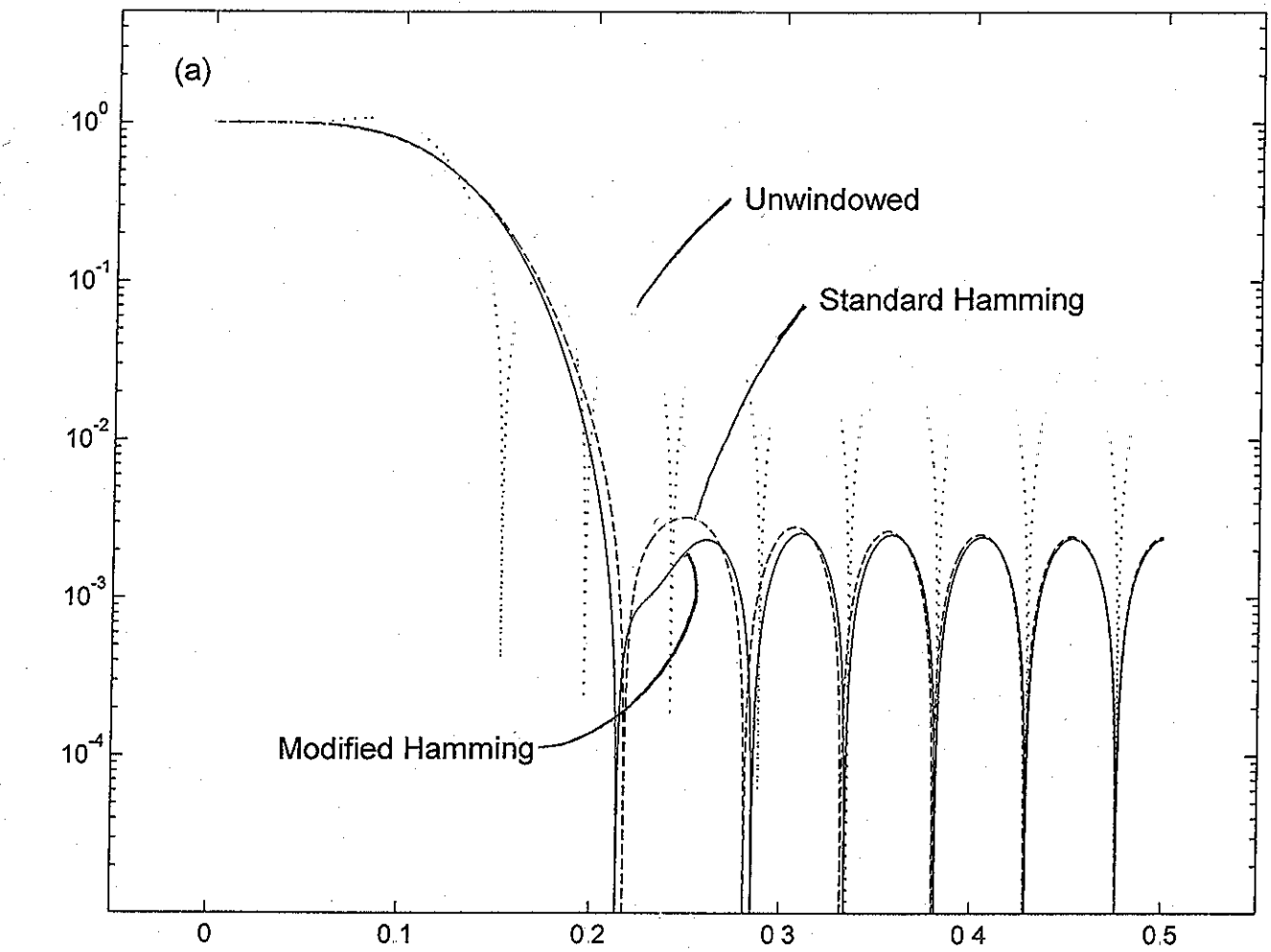
Fig. 14 shows an additional case where we move the zero of the window from the first sidelobe to the second. We use ham2(21,2). This results in a filter that is surprisingly good. Note that, except for a region right around the frequency of 0.22, it is better than the standard Hamming window case, and ends up with better than twice the stopband rejection. It would be easy to argue that this is a better result than Fig. 13 as well.

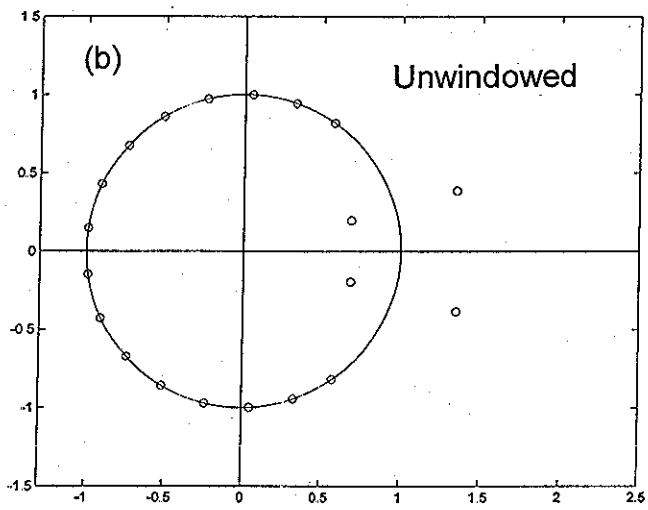
One might wonder if getting a favorable result, extra stopband zeros, is not just a matter of choosing a length and sideband lobe, but rather, we may also need to manipulate the parameter r. In particular, the procedure worked great for length 19 in that we got two extra pairs of zeros, but did not work the same for length 21. In fact, if we study a range of filter length, we find success occurs in clusters:

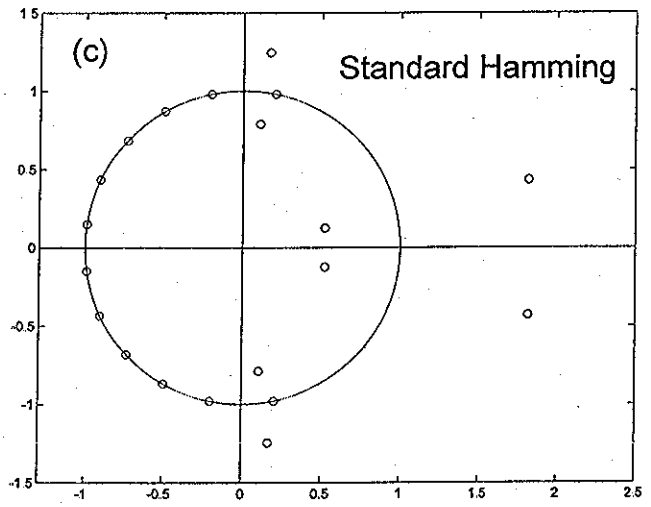
Extra Unit Circle Zeros	No Extra Unit-Circle Zeros
N=7-11	N=12-16
N=17-19	N=20-24
N=25-27	N=28-32
N=33-35	N=36-41
N= 42	N=43.....

The reasons for these results are not explained here. It should be noted however that in the cases where we do get extra unit-circle zeros, it is not necessarily true that we get a great filter.

We might well wonder if we choose a length where we do not get good extra zeros "automatically" if we can force them to occur by a better choice of r? It appears not, but here we are at the extremes of our investigation, so our conclusions are perhaps tentative. Here is what happens. Fig. 15 shows the case of N=21 where we did not get automatic unit-circle zeros. True enough, if we manipulate r (which is the same as the frequency for the zero, since r is less than 0.5) we can force the zeros onto the unit circle. This clearly corresponds to a downward loop in the frequency response reaching and crossing zero. The point is that the frequency response is poor in these cases,







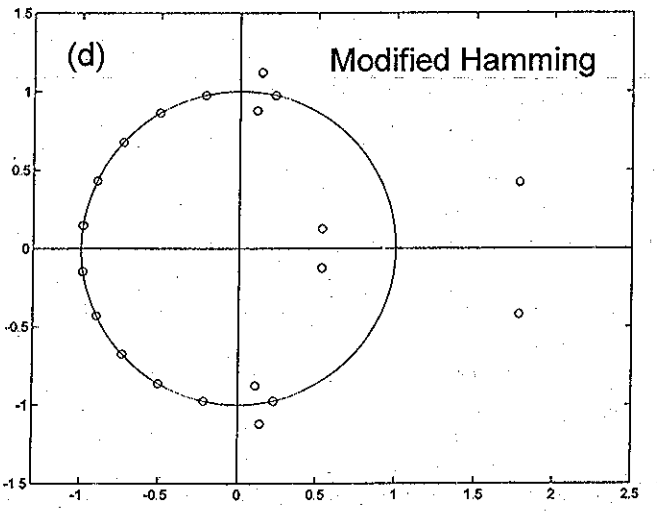
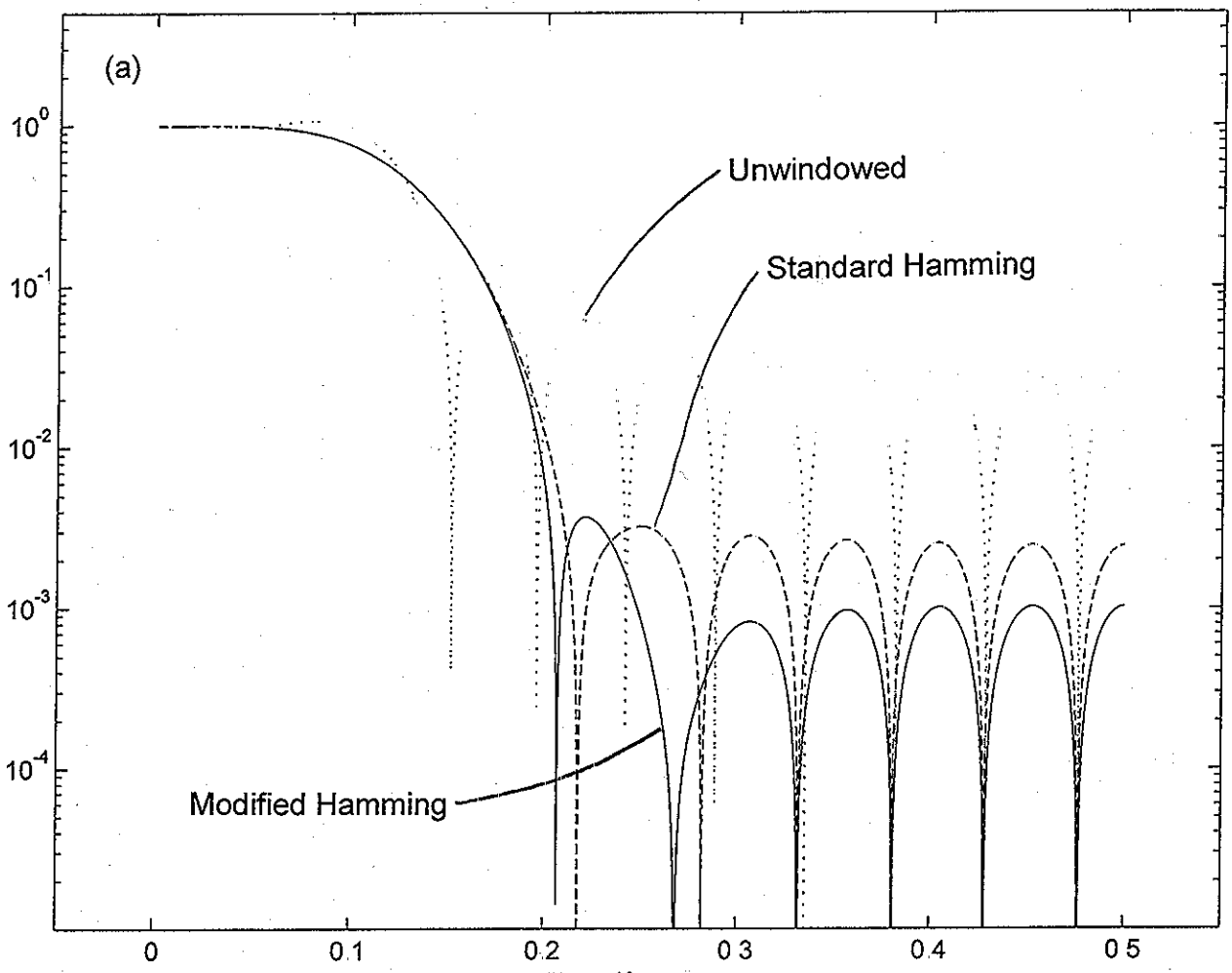
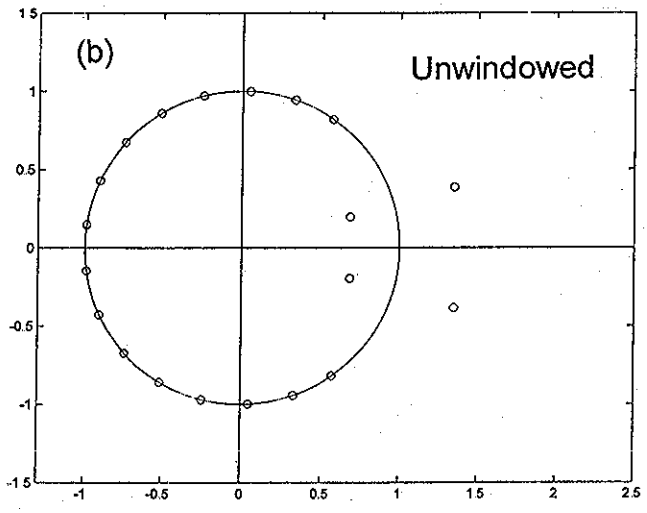
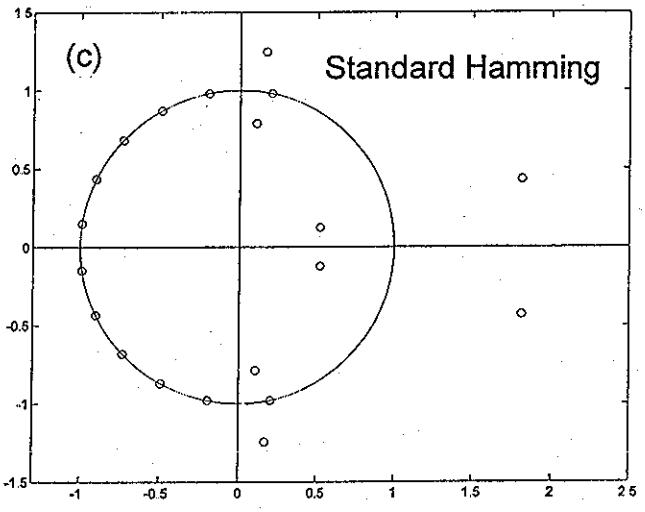


Fig. 13 In this case where we have just changed the length to 21 from the length 19 of Fig. 12, we do not see any additional unit-circle zeros. None the less, the first sidelobe of the stopband is improved.







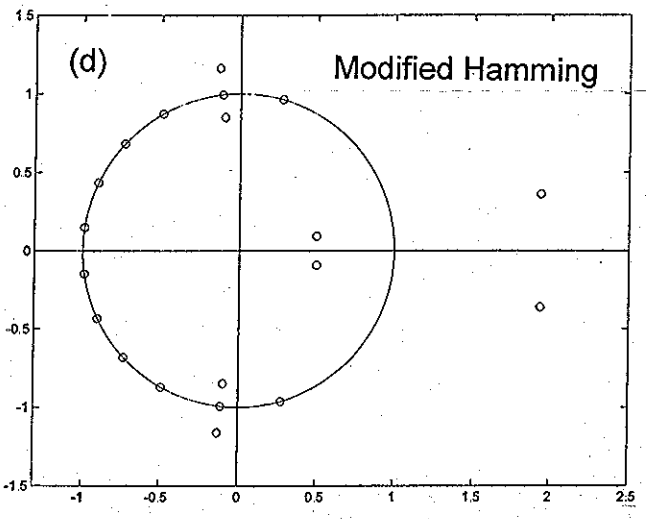


Fig. 14 Here for length 21 we have placed the added zeros of the window in the second sidelobe. This appears to be quite a nice improvement overall.

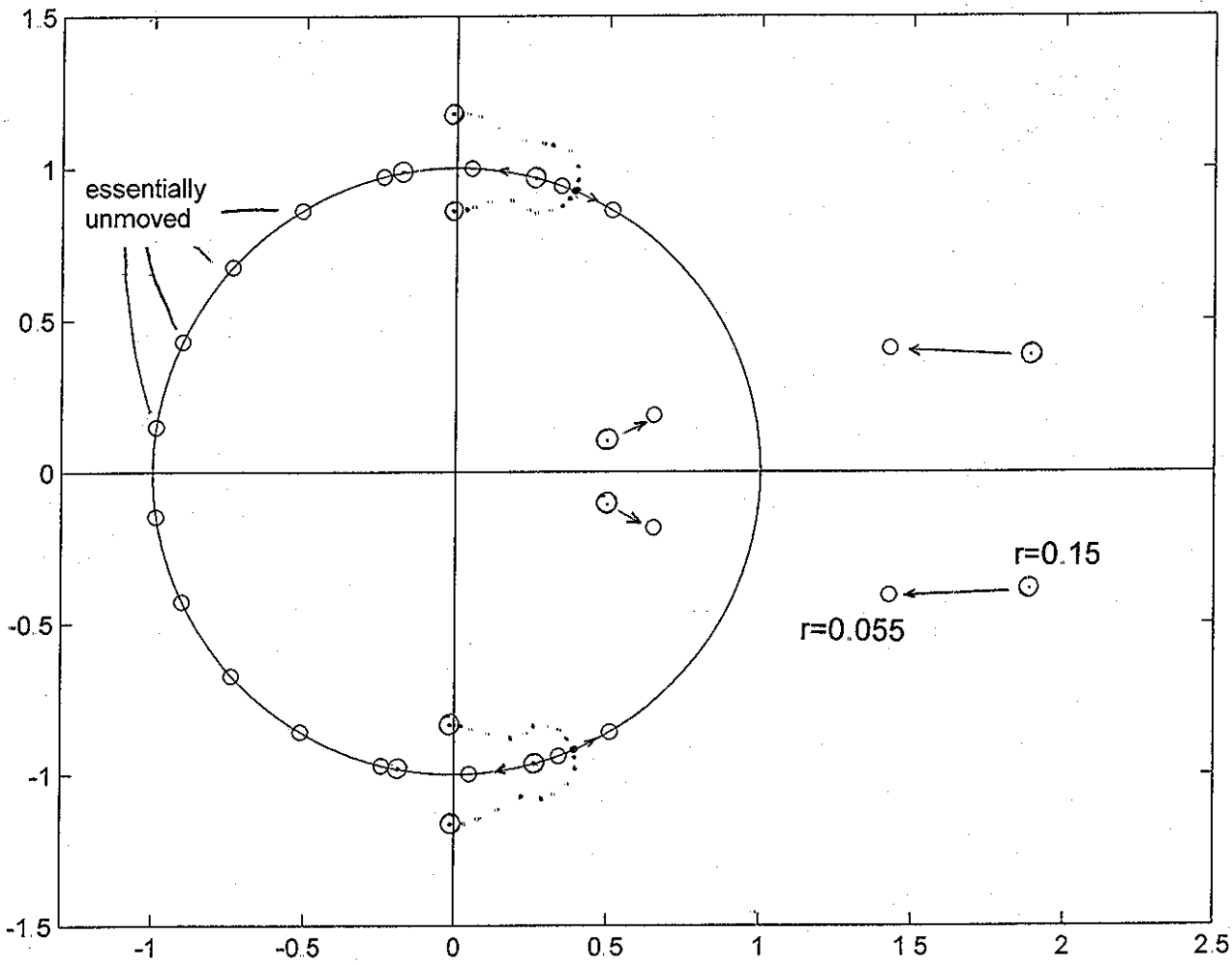


Fig. 15 If we attempt to force zeros onto the unit circle by manipulating the r parameter (the frequency here), we can do this. Here we show the lazy zeros for the case of r=0.15 (larger circles with dots in center) and as we decrease r, the lazy zeros do move to the unitcircle (at r=0.0635) and then spread along the circle. This clearly corresponds to a downward lobe of the frequency response dipping to and below zero. The sequence ends with the smaller circles, and corresponds to r=0.055. Other zeros move as shown, and the ones on the left barely move at all. The interesting point is that while we get unit-circle zeros, the resulting filter is quite poor, so it is quite pointless.

being far far worse than the standard Hamming window, and not much better than the unwindowed case. But it is always fun to look at this sort of "root locus" result.

REFERENCES

[1] "Subtle Design Considerations for Hamming Windows," Electronotes Application Note 319, March 1992.
[2] Oppenheim, A.V. & R.W. Schafer, J.R. Buck, Discrete-Time Signal Processing, Prentice-Hall (1999), pg 468
[3] Matlab hamming.m function (or as built-in)
[4] B. Hutchins, Filter Element Electronotes, Vol. 20, No. 197 pp 19-23

APPENDIX

ham2.asm Main Modified Hamming Window Program

```
function h=ham2(n,r)
%************************************************************************
% function h = ham2(n,r)
%    Refined Hamming Window - Shift by 1/2 Sample + Customized Zero
%       n = length of Hamming Window
%       r = window parameter
%              if r>0.5 the added zero is placed in the r-th sidelobe
%              if r<0.5 then the added zeros is at r where r is a fraction
%                     of the sampling rate.
%
% B. Hutchins                                                  Fall 1999
%************************************************************************

if exist('r')==1
   rr=r;
    if rr>0.5
      r=2*r-2;
      fzero=r/(2*n) + 2.5/n;
    end
    if rr<=0.5
       fzero=rr;
       r = (2*n)*(r - 2.5/n);
    end
else r=0;
end
fzero                  % fzero = frequency (on 0 to 1/2)
                       %   for position of extra zero
                       % r is now offset relative to
                       %   center of first sidelobe

% Compute Dirichlet frequencies and functions
a3=(3+r)*pi/(2*n);
a5=(5+r)*pi/(2*n);
a7=(7+r)*pi/(2*n);
s3=sin(n*a3)/(n*sin(a3));
s5=sin(n*a5)/(n*sin(a5));
s7=sin(n*a7)/(n*sin(a7));

% Compute dc and cosine parameters
a=-s5/(s7+s3-2*s5);
b=1-2*a;
dc=b-2*a;
ac=2*a;

% Compute window, symmetric, one full cosine cycle,
%   offset by 1/2 sample to put all zeros on unit circle
k=0:(n-1);
h=dc+ac*(1+cos((1+2*k-n)*(pi/n)));
```

```
% Display and comparison to standard Hamming window below
figure(1)
stem(h)
hold on
plot(hamming(n),'--')
axis([0 n+1 -.1 1.1]);
title('Modified=o, Orig. Ham. = ----')
hold off

figure(2)
H=freqz(h,1,10000);
H=abs(H)/abs(H(1));
H0=freqz(hamming(n),1,10000);
H0=abs(H0)/abs(H0(1));
semilogy([0:.00005:.49995],H);
hold on
semilogy([0:.00005:.49995],H0,'--');
title('Modified=solid, Orig. Ham. = ----')
hold off
axis([0 .5 0.000001 1.2]);

figure(3)
[p0,z0]=pzplot1(hamming(n),1);
title('Original Hamming')
axis([-1.3 1.3 -1.3 1.3]);

figure(4)
[p,z]=pzplot1(h,1);
title('Modified Hamming')
axis([-1.3 1.3 -1.3 1.3]);
figure(4)
```

ham7.m Modified ham2.m to search out minimum energy sidelobe

```
function h=ham7(n,r)
%************************************************************************
% function h = ham2(n,r)
%    Refined Hamming Window - Shift by 1/2 Sample + Customized Zero
%       n = length of Hamming Window
%       r = window parameter
%              if r>0.5 the added zero is placed in the r-th sidelobe
%              if r<0.5 then the added zeros is at r where r is a fraction
%                     of the sampling rate.
%
% B. Hutchins     Modification of ham2.m              Fall 2003
%************************************************************************
```

```
m=0;
for r=0.75:0.01:1.25  % look for min energy over a range about
                      %   center of first sidelobe
m=m+1
if exist('r')==1
   rr=r;
    if rr>0.5
      r=2*r-2;
      f=r/(2*n) + 2.5/n;
    end
    if rr<0.5
       r = (2*n)*(r - 2.5/n);
    end
else r=0;
end
r;
a3=(3+r)*pi/(2*n);
a5=(5+r)*pi/(2*n);
a7=(7+r)*pi/(2*n);
s3=sin(n*a3)/(n*sin(a3));
s5=sin(n*a5)/(n*sin(a5));
s7=sin(n*a7)/(n*sin(a7));
a=-s5/(s7+s3-2*s5);
b=1-2*a;
dc=b-2*a;
ac=2*a;

k=0:(n-1);
h=dc+ac*(1+cos((1+2*k-n)*(pi/n)));
H=freqz(h,1,10000);
H=abs(H)/abs(H(1));


z=roots(h);          % find the zeros
ang=angle(z);        % find their angles

% keep the positive angles
a=[];
for k=1:n-1
  if ang(k)>0
    a=[a ang(k)];
  end
end

% find first sidelobe - range beetween first and third zero
ang=a;
ang=sort(ang);
ang=ang/(2*pi);
k1=round(ang(1)*20000);
k2=round(ang(3)*20000);
```

```
% add up all the energy in first sidelobe
s=0;
for k=k1:k2
  s=s+H(k)^2;
end
ss(m)=s;

end

% find index for smallest result
[mn,in]=sort(ss)

r=0.74+in(1)*.01

% now compute result for best sidelobe value
h=ham2(n,r)
```

hamtest.m Program to Apply Window to Test Filter

```
%  hamtest1.m
function hamtest1(N,r)

w1=hamming(N)'
w2=ham2(N,r)

hfirls=firls(N-1,[0 .25 .25 1],[1 1 0 0]);

h0=hfirls
h1=hfirls.*w1
h2=hfirls.*w2

H0=abs(freqz(h0,1,5000));
H1=abs(freqz(h1,1,5000));
H2=abs(freqz(h2,1,5000));

figure(5)
subplot(311)
plot([0:.0001:.4999], H0)
title('Orig. firls')
subplot(312)
plot([0:.0001:.4999], H1)
title('Orig. Hamming')
subplot(313)
plot([0:.0001:.4999], H2)
title('Modified Hammming')
```

```
figure(6)
semilogy([0:.0001:.4999], H0,'r:')
hold on
semilogy([0:.0001:.4999], H1,'--g')
semilogy([0:.0001:.4999], H2,'c')
axis([-.05 0.55 0.00001 5])
title('r=firls g=Hamming c=Mod Hamming')
hold off

figure(7)
[z,p]=pzplot1(h0,1)
grid off
figure(7)
hold on
plot([-5 5],[0 0])
plot([0 0],[-5 5])
hold off
title('Original Filter')
axis([-1.3 2.5 -1.5 1.5]);
figure(8)
[z,p]=pzplot1(h1,1)
grid off
figure(8)
hold on
plot([-5 5],[0 0])
plot([0 0],[-5 5])
hold off
title('Original Hamming Filter')
axis([-1.3 2.5 -1.5 1.5]);
figure(9)
[z,p]=pzplot1(h2,1)
grid off
figure(9)
hold on
plot([-5 5],[0 0])
plot([0 0],[-5 5])
hold off
title('Modified Hamming Filter')
axis([-1.3 2.5 -1.5 1.5]);
```

pzplot1.m Plot Poles/Zeros in a-Plane

```
function [z,p]=pzplot1(b,a)
%  function [z,p]=pzplot1(b,a)
%  Plot the poles and zeros of a transfer function in z-plane
%    z are zeros of numerator polynomial b
%    p are poles of denominator polynomial a
%    z (plotted as o) and p (plotted as x) in the z-plane
%    multiple-order singularities are only indicated as single-order
%  B. Hutchins, EE425, Cornell Univ. Fall 1993

%  find roots
z=roots(b);
p=roots(a);

% find max for plotting
pmax=max([abs(real(z)); abs(real(p)); abs(imag(z)); abs(imag(p)) ]);
sc=ceil(pmax);

% begin plot
% prepare circle
n=0:500;
r=exp(j*2*pi*n/500);
axis([-sc, sc, -sc, sc]);
plot(r,'g')
grid
hold on
% plot real and imag, not just z itself or else the real part
% may be plotted as verticle if singularities are not complex
plot(real(z),imag(z),'o')
plot(real(p),imag(p),'x')

hold off
axis('equal')
axis([-sc sc -sc sc])
```