ELECTRONOTES
1 Pheasant Lane
Ithaca, NY 14850
(607)-273-8030

APPLICATION NOTE NO. 333

October 1995

DESIGN AND APPLICATIONS OF INVERSE CHEBYSHEV FILTERS

It is a popular idea in academic circles to teach a filter design phase of a digital signal processing course by specifying a particular filter to be designed, usually as an approximation to some idealized, perfect filter. A practicing engineer, on the other hand, is unlikely to be handed a set of specs and then be asked to design the filter in such a manner. Instead, the engineer is asked to design a system, and this system may well require one or more filters, but it is up to the engineer to figure out what filters are required, and then to design them. In many cases, the actual specs of the filter are not an issue: the designer simply expends whatever resources are available on a "good filter." However, there are also instances of filter design, as there probably are most everywhere, where less may be more.

One problem with simply deciding to design a "good filter" is that in doing this we are concentrating on the filter itself, and we are thereby likely ignoring the input to the filter. That is, we are preparing our filter to handle all possible inputs, while the expected inputs may fall into more specialized classes. For example, suppose we know that the signal to be filtered consists of a (desired) speech signal, some random noise, and a "cross-talk" or "leakage" signal represented by a 15 kHz sinewave. Let's further assume that the speech signal is large, that the random noise is small, and that the leakage is large.

It is clear that some sort of ideal low-pass would do the job we want. We would arrange for it to have a cutoff of about 4 kHz, which would pass the expected speech bandwidth. This would also pass the random noise that is within its passband, but the random noise outside the passband would be blocked. Further, the 15 kHz leakage would be completely blocked.

If we next consider a "good" (but not ideal) low-pass, we expect that the response in the stopband will be small, but generally not zero, except at isolated frequencies. The fact that we are often trying to approximate a "stopband" response that would be zero in the ideal case means that the approximation passes exactly through zero at certain frequencies, usually several or many times. [In the z-plane, we see zeros on the unit circle at these frequencies.] Generally, these zeros end up in whatever positions are needed so that the stopband is reasonably characterized as being "optimal" in some overall sense (such as least squared error or least maximum absolute error). That is, while we expect these zeros to occur in the passband we have paid no attention to exactly where they occur.

In the type of application postulated above, where we had a strong but narrow-banded component (a sine wave in fact) in the input which we wanted to block, it becomes more useful to consider ways of placing zeros on top of such components, blocking them completely. That is, we choose to expend our resources in a different way. A small interfering component at the input, whose frequency is in the filter's stopband range, which is then subject to substantial rejection by the filter, becomes tiny at the output. On the other hand, a large interfering component subject to substantial rejection may become only small, not tiny. Thus we may well look for total rejection of the worse known offenders as far more desirable than substantial rejection of all possible offenders.

One filter characteristic that fits this bill quite nicely is the so-called inverse Chebyshev. This filter has a flat passband, and ripples in the stopband (Fig. 1). It thus has zeros at finite frequencies. It can be an excellent choice when we would just as soon have a flat passband, where we have one major stopband component to reject, and where the cutoff frequency is not particularly critical.

Here we will find it convenient to use some design equations from Jackson [1]. These equations will allow us to have a desired zero position as an input parameter to our design. Following Jackson, without elaboration, we start with a design parameter γ based on the stopband ripple δ_2.

$$\gamma = [\,(1 + (1 - \delta_2^2)^{1/2})\,/\delta_2]^{1/N} \tag{1}$$

from which we develop:

$$\sinh(\phi) = (\gamma - 1/\gamma)/2 \tag{2a}$$

$$\cosh(\phi) = (\gamma + 1/\gamma)/2 \tag{2b}$$

and using Butterworth pole angles for k = 0,1,....(N-1):

$$\mu_k = (2k-1)\pi/2N \tag{3}$$

we get:

$$\sigma_k = -(\sinh(\phi)\sin(\mu_k))\omega_c \tag{4a}$$

$$\omega_k = (\cosh(\phi)\cos(\mu_k))\omega_c \tag{4b}$$

with the N poles at $\alpha_k + j\beta_k$ given by:

$$\alpha_k = \omega_c\omega_r\sigma_k/(\sigma_k^2+\omega_k^2) \tag{5a}$$

$$\beta_k = -\omega_c\omega_r\omega_k/(\sigma_k^2+\omega_k^2) \tag{5b}$$

PROGRAM 1: INVCHEB.M, ANALOG INVERSE CHEBYSHEV

```
function [an,ad,az,ap]=invcheb(N,d2,wc,wn,NZ)
%
%           INVERSE CHEBYSHEV ANALOG DESIGN
%
%     function [an,ad,az,ap]=invcheb(N,d2,wc,wn,NZ)
%
%     an=numerator, ad=denominator, az=zeros, ap=poles
%
%          N=order
%          d2 = stopband ripple (as decimal fraction)
%          wc = omega cutoff (cutoff frequency)
%          wn = omega notch
%          NZ = zero to place on notch
%
%  example:  [an,ad,az,ap]=invcheb(12,0.1,1,2,3)
%      places third notch on frequency of 2.0
%
%  Design equations from L. Jackson, Digital Filters and
%     Signal Processing, 2nd Edition, KAP

%  B. Hutchins                                Fall 1995

k=1:N;
mu=(2*k-1)*pi/(2*N);       % angles, equation (3)

wr=wn*cos(mu(NZ));         % compute wr using desired notch, eqn (6)

gamma=( ( 1+sqrt(1-d2^2) ) / d2)^(1/N);     % equation (1)

sinhf=(gamma - 1/gamma)/2;                  % equation (2a)
coshf=(gamma + 1/gamma)/2;                  % equation (2b)

sigmak = -sinhf*wc*sin(mu);                 % equation (4a)
omegak =  coshf*wc*cos(mu);                 % equation (4b)

for k=1:N
     alpha(k)=wc*wr*sigmak(k)/(sigmak(k)^2 + omegak(k)^2);  % eqn (5a)
     beta(k)=-wc*wr*omegak(k)/(sigmak(k)^2 + omegak(k)^2);  % eqn (5b)
end

az=j*wr./cos(mu);          % analog zeros, equation (6)

ap=alpha+j*beta;           % analog poles

an=poly(az);               % analog numerator from zeros
ad=poly(ap);               % analog denominator from poles

w=0:.002:5;
H=freqs(an,ad,w);
H=H/H(1);
figure(1)
plot(w,abs(H))
axis([0 5 0 1.2])
figure(1)
pause
figure(2)
plot(alpha,beta,'x')
zz=zeros(1:N);
hold on
plot(zz,imag(az),'o')
axis([-3 1 -4 4])
grid
figure(2)
hold off
```

and N zeros at:

$$s_k = j\omega_r/\cos(\mu_k) \tag{6}$$

In our MATLAB™ [2] program invcheb.m on the previous page, we will concentrate on a design procedure where we choose a stopband frequency to be rejected completely (notched) as one of the input parameters. The program comments probably make the program self-explanatory, and the program also references equations (1- 6) above.

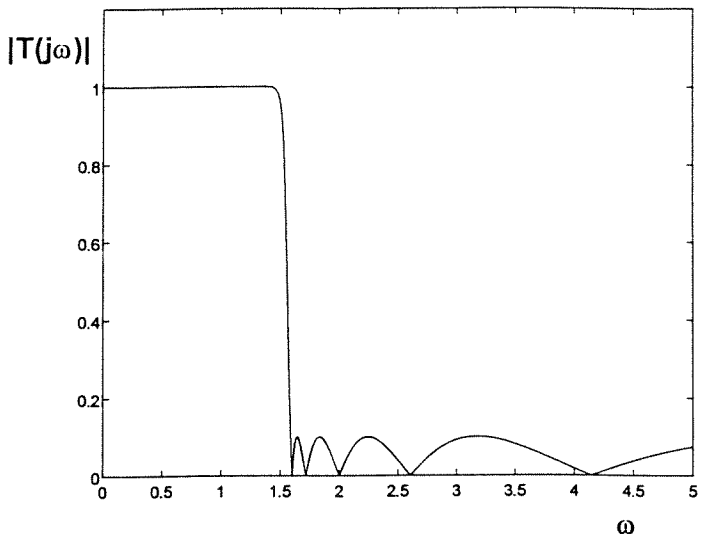

Fig. 1 Frequency response of 12th-order inverse Chebyshev with a third notch at 2 (analog filter)

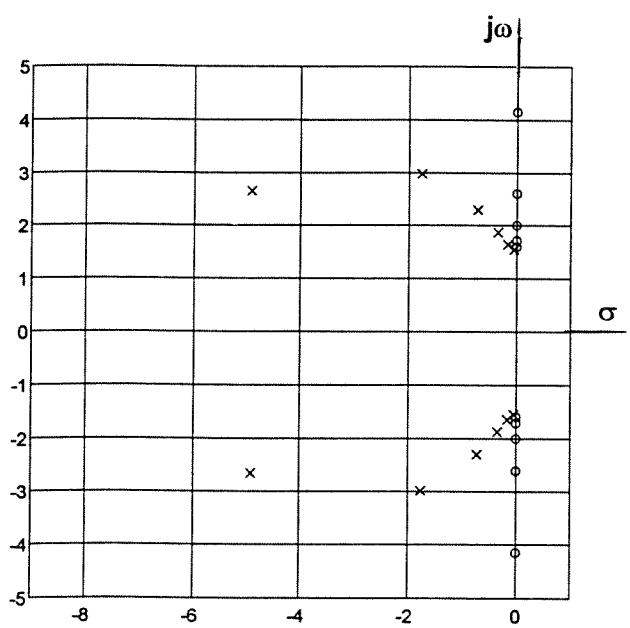

Fig. 2 Pole/zero plot in the s-plane of 12th-order inverse Chebyshev showing finite jω-axis zeros

Fig. 1 and Fig. 2 show the results of running the example indicated in the header to the program. Here, in Fig. 1, we see a nice flat passband and a stopband that rejects, coming up no higher than 0.1, our choice of δ_2. We also see five of the six notches, the third of which is precisely at 2 as we specified (one off scale at 12.1562). The pole/zero plot of Fig. 2 shows the $j\omega$-axis zeros (two at $\pm 12.1562j$ not shown) and note that the poles here swing out in a wider arc than we usually see for low-pass, essentially supporting the stopband in this case.

So far we have been designing analog filters, but we can also easily design digital filters based on an inverse Chebyshev prototype using Bilinear z-Transform. The filter design procedure is very similar to that for other types of analog prototypes such as Butterworth and Chebyshev. There are a couple of interesting wrinkles, however.

One of these wrinkles is that in the case of inverse Chebyshev, we have finite analog zeros (one infinite zero for odd order). In the case of Butterworth and Chebyshev, the zeros at analog infinity all mapped into digital z=-1, and we tend to treat this as just an automatic step - we just add these (often ad hoc) after we map the poles. For inverse Chebyshev, we will need to transform the finite analog zeros, but this is no problem and is done in <u>exactly</u> the same way that we transform the poles.

The second wrinkle is that, against the particular application scenario postulated above (strong stopband component), we are interested in having a particular zero fall at a particular frequency. With Bilinear-z, we get to have one and only one frequency mapped exactly from the analog domain to the digital domain, and for usual low-pass designs, we almost always choose this one frequency to be some well-defined "cutoff" frequency. Here we will want to make the frequency match at our desired stopband notch frequency. In consequence of our concentration on this notch placement, the low-pass cutoff frequency, already not too well defined, may become even less well-defined due to the warping. This simply means that we may need to go back and modify the design if we are not happy with the overall results. In general, the first modification to consider would involve placing the frequency to be notched at a different zero. Placing the frequency to be notched on a lower zero will of course effectively raise the low-pass cutoff, and conversely.

In our digital filter example, let's suppose that we want a low-pass digital filter that clocks at a sampling frequency of 10000 Hz, which has a passband of something like 1000 Hz. In addition, we want to have a zero at exactly 2000 Hz. If we were designing for an ordinary low-pass application, we would know the desired digital filter cutoff frequency and would use a Bilinear-z warping to find the corresponding cutoff of the analog prototype filter. Here we have a desired digital filter stopband notch (2000 Hz) and we must warp this for the corresponding analog prototype notch. This we find, using the usual Bilinear-z tangential warping between an analog frequency f_A and digital frequency f_D:

$$f_A = (f_s/\pi)\tan(\pi f_D/f_s) = (10000/\pi)\tan(2000\pi/10000) = 2312.6567 \text{ Hz} \quad (7)$$

This means that we need to design our analog prototype to have a zero at 2312.6567 Hz. This is easily done. For example, we can use the program invcheb as:

```
[an,ad,az,ap]=invcheb(12, 0.1, 1, 2.3127, 3)
```

which results in a response very similar to Fig. 1, except the third notch is at 2.3127 instead of at 2. Here we are entering frequencies in units of kHz for convenience and for plotting consideration. We will just have to remember to enter the sampling frequency in kHz as well later.

The conversion of the analog prototype to a digital filter is accomplished with the Bilinear-z transform and is done by transforming each analog pole and each analog zero separately. The mapping is simply derived from the usual substitution:

$$s \longleftarrow (2/T)(z-1)/(z+1) \quad (8)$$

which is solved for z to give the inverse:

$$z = (2/T + s) / (2/T - s) = (2f_s + s) / (2f_s - s) \quad (9)$$

where $f_s = 1/T$. It is important to note that here s must be in rad/sec while T is in seconds, making f_s have units of Hz. The two are <u>not in the same units, and attempts to enter them in the same units will give wrong answers</u>. You must check to see what units you are working in, and convert if necessary (see example).

The MATLAB program bz.m given below is simply an implementation of equation (9). For our example, we want to convert the analog poles/zeros generated by invcheb to digital poles/zeros. Here we use:

```
[dn,dd,dz,dp] = bz(10, 2*pi*az, 2*pi*ap)
```
(10)

Note that the sampling frequency is entered as 10 since we have agreed to use units of kHz. The analog poles were similarly in units of kHz, so we have multiplied them by 2*pi in order to get them in kiloradians/sec. Glancing at equation (9) convinces us that multiplying all frequencies by 1000 (going from kHz to Hz) would make no difference, canceling out of the ratio. We could also have divided the sampling frequency in equation (10) by 2*pi (instead of multiplying s by 2*pi) and gotten the correct results, but this is confusing and should be avoided.

PROGRAM 2: BZ.M, BILINEAR Z-TRANSFORM

```
function [dn,dd,dz,dp]=bz(fs,az,ap)
%function [dn,dd,dz,dp]=bz(fs,az,ap)
% convert analog poles/zeros to digital poles/zeros
% az=analog zeros          ap=analog poles
% dz=digital zeros         dp=digital poles
% dn=digital numerator     dd=digital denominator
%
%  Uses  s <- (2/T)(z-1)/(z+1) the usual Bilinear-z substitution
%    transformed to the inverse:
%
%            z = (2/T + s)/(2/T - s)
%
%  Note that 2/T is the sampling frequency fs in Hz while s is
%    thought of in terms of rad/sec.  The two are supposed to
%    be in different units.  If your poles/zeros to be
%    transformed are in Hz, multiply by 2*pi.

%  B. Hutchins                                    Fall 1995

dp=(2*fs + ap)./(2*fs - ap);     % digital poles
dz=(2*fs + az)./(2*fs - az);     % digital zeros
dn=poly(dz);                     % digital numerator
dd=poly(dp);                     % digital denominator
```

The resulting digital filter frequency response is shown in Fig. 3. Note that the third notch is exactly at 0.2 times the sampling frequency (thus at 2 kHz as desired). Fig. 4 shows the corresponding pole/zero plot for the digital filter. This filter has a low-pass cutoff of about 1.6 kHz. This may or may not be close to what we want. (Keep in mind that we are controlling the third notch, not the cutoff.) So suppose that the cutoff is too high. In such a case, we choose to put a different zero on the 2 kHz frequency. Fig. 5 shows such a case where the fifth rather than the third notch is used. In this case, the low-pass cutoff is more like 800 Hz.

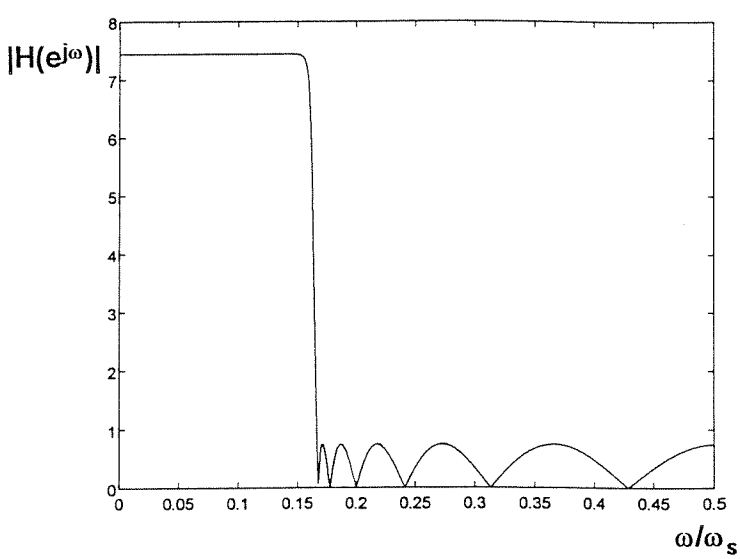

Fig. 3 Frequency response of digital filter showing flat passband, and third notch at $0.2f_s$. The analog prototype for this digital filter had its third notch at 2312.66 Hz.

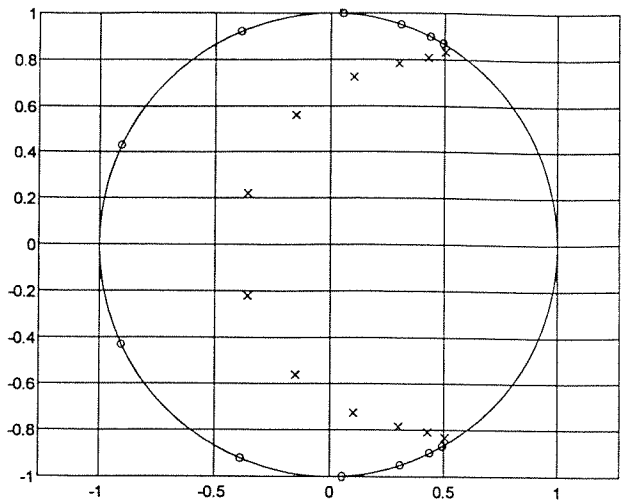

Fig. 4 Pole/zero plot for digital filter of Fig. 3

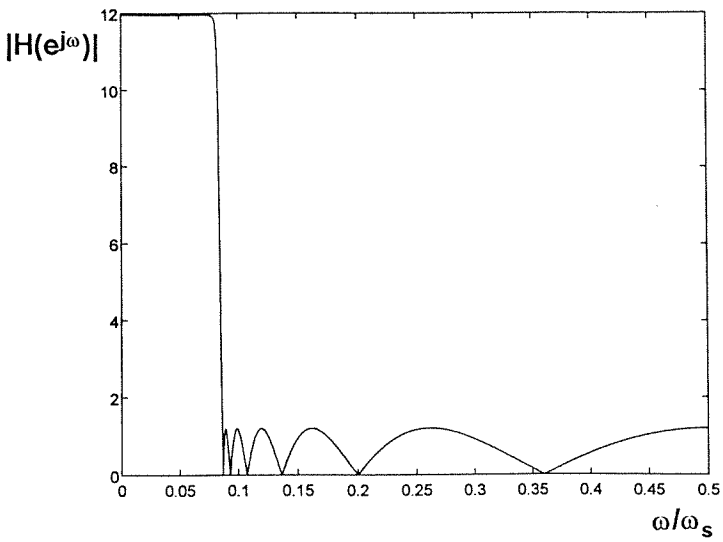

Fig. 5 Here we redo the design so that the 5th notch rather than the third is on $0.2f_s$. Note the lowering of the low-pass cutoff that results from this change.

References:

[1] L. B. Jackson, Digital Filters and Signal Processing, (Second Edition) Kluwer Academic Publishers (1989) pp 195-197

[2] Anon, The Student Edition of MATLAB, Prentice-Hall (1992)