*'It's a naïve domestic Burgundy without any breeding, but I think you'll be amused by its presumption.'*

**EE 311** **April 3, 2019**
**Lecture 21**

## 6.7 Applications of IIR Filters

We have considered the resonator and notch filter in section 6.6 which represents second order IIR filter applications. In this section we consider further IIR applications such as the all pass filter, the moving average IIR filter, the comb filter, and the inverse filter.

**All Pass Filters**

As the name implies, an all pass filter passes all frequencies with a gain of unity. The usefulness of an all pass filter lies in what it does to the phase of the incoming signal. All pass filters are often used in conjunction with an IIR filter which has nonlinear phase. The all pass filter can be used to alleviate some of the problems caused by the nonlinear phase characteristics.

If an all pass filter is given by

$$H_A(z) = K \cdot \frac{N_A(z)}{D_A(z)}$$

then

$$\left| K \cdot \frac{N_A(e^{j\omega T})}{D_A(e^{j\omega T})} \right| = 1 \quad \text{or,} \quad K\left|N_A(e^{j\omega T})\right| = \left|D_A(e^{j\omega T})\right|$$

One function which satisfies this criterion is

$$H_A(z) = K \cdot \frac{z - 1/r}{z - r}$$

since

$$|z - 1/r| = \{[\cos(\omega T) - (1/r)]^2 + \sin^2(\omega T)\}^{1/2} = (1/r)[r^2 - 2r\cos(\omega T) + 1]^{1/2}$$

and

$$|z - r| = \{[\cos(\omega T) - r]^2 + \sin^2(\omega T)\}^{1/2} = [r^2 - 2r\cos(\omega T) + 1]^{1/2}$$

so that

$$\left| \frac{z - 1/r}{z - r} \right| = 1/r$$

If we choose $K = r$ then the magnitude of $H_A(z) = 1$ for all frequencies.

In general, an all pass filter may be constructed with a transfer function given by

$$H_A(z) = K \cdot \prod_{i=1}^{L} \frac{z - (1/r_i)}{z - r_i} \tag{6.47}$$

In this equation $r_i$ may be real or complex. If $r_i$ is complex then its conjugate must also be included in the filter in order to have real coefficients for implementation. The gain constant, $K$, is chosen to give the filter a gain of unity across the band. Figure 6.37 shows how the phase shift varies for several values of $r_i$.

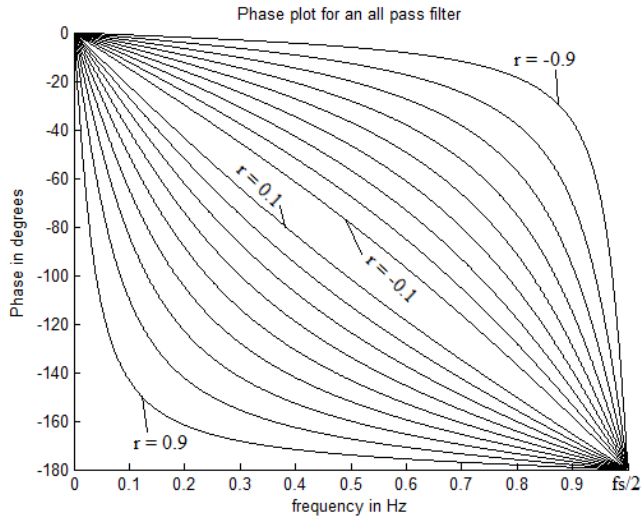



**Figure 6.37**
Phase shift for an all pass filter which has one real pole and one real zero. The pole moves from -.9 to +.9 in steps of .1.

While an all pass filter does not provide a definitive solution to correcting nonlinear phase in an IIR filter it does provide a mechanism to make phase corrections over a portion of the frequency band.

**Example 6.20**
A low pass Chebyschev filter is to be designed to meet the following specifications:
Sample frequency: 44.1 KHz
Passband: $0 \rightarrow 8$ KHz with ripple less than 0.001
Stopband: 15KHz $\rightarrow f_s/2$ with ripple less than 0.0001
Show that this filter's phase curve can be linearized in the passband by the application of a first order all pass filter stage. Take the maximum deviation between a straight line and the phase curve as a measure of the linearity. Iteratively determine the best value of *r* for the all pass filter with a transfer function similar to that of (6.38).

**Solution:**
We begin by using the following MATLAB® code to calculate and plot the frequency response for the filter. The magnitude and phase plots are shown in Figure 6.38. A straight line has been added to the phase curve to show the nonlinearity in the pass band.

```
fs = 44100;                          %Sample frequency
Rp = .001;RpDB = -20*log10(1-Rp);    %Pass band ripple
Rs = .0001;RsDB = -20*log10(Rs);     %Stop band ripple
fpass = 8000;fstop = 15000;
%Get the order for a Chebyshev type 1 filter.
[N Wn] = cheb1ord(fpass/(fs/2),fstop/(fs/2),RpDB,RsDB);
[num den] = cheby1(N,RpDB,Wn);       %Create a filter of that order
[Hc f] = freqz(num,den, 512, fs);    %Find the frequency response
figure(1); clf;
subplot(2,1,1)
plot(f, abs(Hc))                     %Plot the magnitude
```

```
subplot(2,1,2)
HcAngle = unwrap(angle(Hc))*180/pi;   %Put the angle in degrees
plot(f, HcAngle)                 %plot the phase curve
line([0 f(i)],[0 HcAngle(i)]); %Draw a line from 0 to passband edge
```

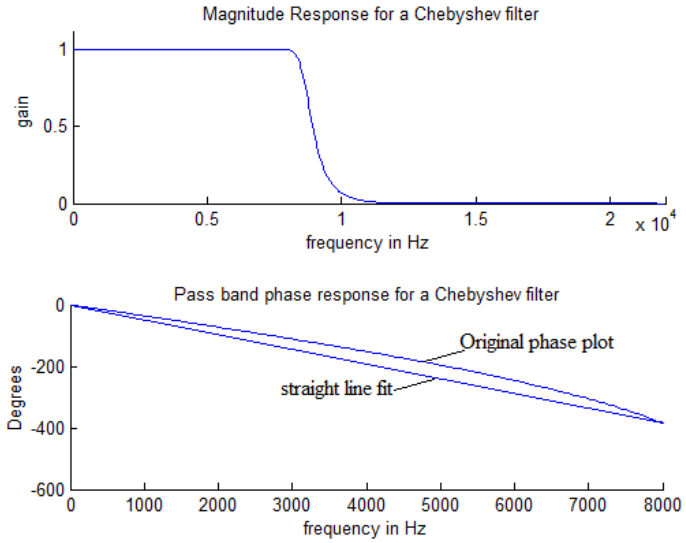



**Figure 6.38**
The magnitude (top) and phase(bottom) plot for a Chebyschev low pass filter of type 1. A straight line has been added to the phase curve to show the nonlinearity.

To linearize the pass band we will apply a single stage all-pass filter with a transfer function given by

$$H_{All}(z) = r\frac{z-1/r}{z-r}$$

We want the phase curve to be linear in the pass band so we seek a value of *r* which will minimize the maximum absolute value of the difference between the phase curve and a straight line at all frequencies through the pass band. While we cannot determine a closed form solution, we can find a "best" solution by iterating successive value of *r* using MATLAB®. From the comparison of the original phase curve in Figure 6.38 to a straight line we see from Figure 6.37 that we need a positive value of *r* to make a correction. In MATLAB® we can write a loop that iterates the value of r beginning at *r* = 0.1 in increments of 0.001. For each iteration, we find the new transfer function for the all-pass filter; multiply it times the transfer function for the original Chebyshev filter, and find the frequency response of the result. Compare the phase response to a straight line and continue iterating the value of *r* until the absolute value of the maximum difference between the phase response and the straight line is no longer decreasing. A MATLAB® program to do this produces the result of *r* = 0.7990 with the maximum difference between the phase curve of the corrected filter and a straight line being about 18.2 degrees. Figure 6.39 shows the result.

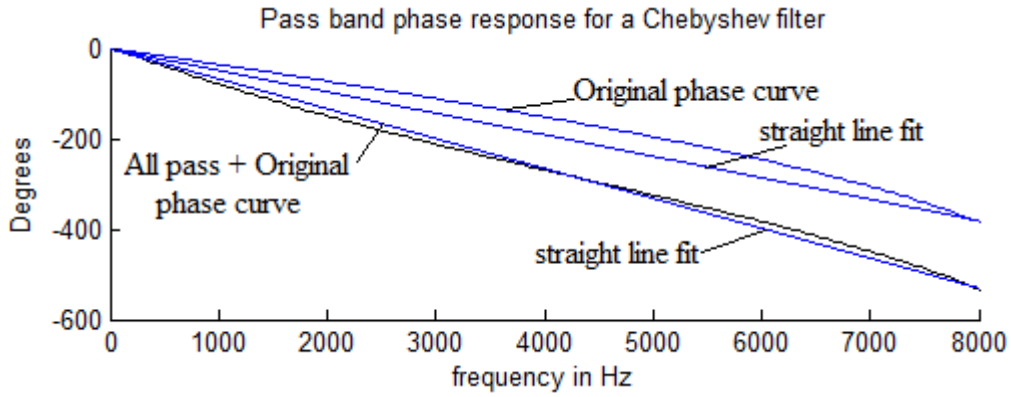



**Figure 6.39**
This graph shows the phase curve for the pass band for the original Chebyschev filter and for the combination Chebyschev filter and allpass filter stage. In both cases a straight line is drawn along to the pass band edge to illustrate linearity.

---

To get more correction from the all pass filter we can add successive sections as indicated in (6.38). If the value of r is complex, a second order all pass filter has the form

$H_{2All} = \frac{z - 1/r}{z - r} \cdot \frac{z - (1/r)^*}{z - r^*}$ where * indicates the conjugate. If I is located at $R\angle\theta$ = Re + jIm and we can write

$$H_{2All} = \frac{K_2 z^2 - K_1 z + 1}{z^2 - K_1 z + K_2} = \frac{R^2 z^2 - 2RzCos(\theta) + 1}{z^2 - 2RzCos(\theta) + R^2} \qquad (6.48)$$

where $K_1$ = 2Re and $K_2$ = $Re^2$ + $Im^2$

Using the polar form we have two variables $R$ and $\theta$. Since the all-pass filter must be stable we have $0 < R < 1$ and since the pole and its conjugate are both included we can say $0 \le \theta < 180^o$. We can iterate $R$ and $\theta$ over the top half of the unit circle to find the best values which minimize the absolute value of the maximum difference between the amended phase curve and a straight line in the pass band. Applying a second order all pass filter to the Chebyshev low pass filter of Example 6.20 and iterating over $R$ and $\theta$ to find the best solution gives $R = 0.703$ and $\theta = 23.60^o$ with a maximum error of just $7.98^o$.

**IIR Moving Average Filters**

To calculate the moving average of length $L$ over sequence $x[k]$ we can write the following difference equation

$$A_M[k] = \frac{1}{L} \sum_{i=k-L+1}^{k} x[i] \qquad (6.49)$$

where $A_M[k]$ is the moving average at time $k$, $L$ is the length, and $x$ is the input sequence. The $z$-transform shows that this equation naturally becomes an FIR filter equation.

$$H_M(z) = \frac{z^{L-1} + z^{L-2} + \cdots + 1}{Lz^{L-1}} \qquad (6.50)$$

Figure 6.40 shows the magnitude, phase, and $z$-plane plots for an FIR moving average filter of length 8. Note, in particular, that the magnitude plot is the same as that of a rectangular window.

This would be expected since the impulse response of such a filter will be rectangle in the time domain.

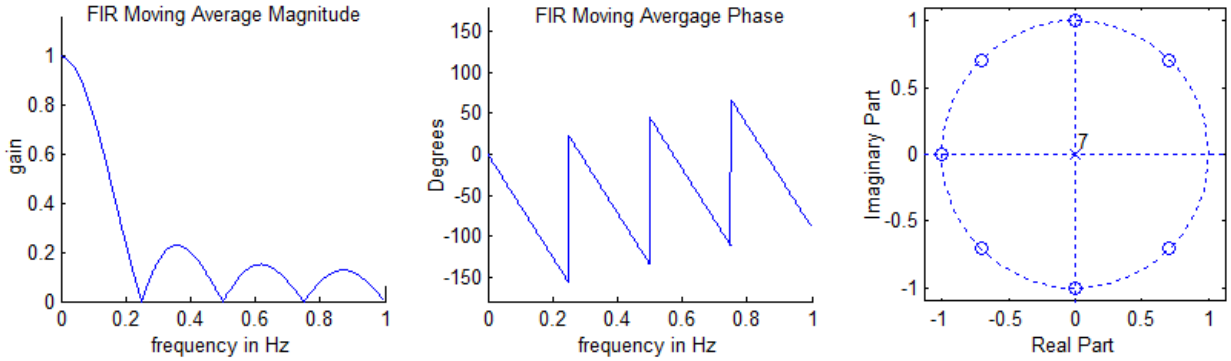



**Figure 6.40**
FIR Moving average filter of order 7 (length 8). Left is the magnitude plot, middle is the phase plot, and right is the pole/zero plot. The discontinuities in the phase curve are caused by zeros on the unit circle.

With some manipulation, (6.50) can be rewritten as an equation for an IIR filter. Expanding the equation gives

$$A_M[k] = \frac{1}{L}\left[x[k] + x[k-1] + x[k-2] + \cdots + x[k-L+2] + x[k-L+1]\right] \qquad (6.51)$$

Likewise,

$$A_M[k-1] = \frac{1}{L}\left[x[k-1] + x[k-2] + x[k-3] + \cdots + x[k-L+1] + x[k-L]\right] \qquad (6.52)$$

Subtracting (6.52) from (6.51) gives

$$A_M[k] = A_M[k-1] + \frac{1}{L}\left[x[k] - x[k-L]\right] \qquad (6.53)$$

Equation (6.53) is a recursive relationship derived from a nonrecursive equation. The transfer equation corresponding to equation (6.53) is

$$H_M(z) = \frac{1}{L}\frac{z^L - 1}{z^{L-1}(z-1)} \qquad (6.54)$$

Equation (6.54) can be seen to be identical to (6.41) except that a ($z$ – 1) term has been added to both the numerator and the denominator – effectively, a pole and a zero have been added at the $z$ = +1 point.

The IIR version of the moving average filter (6.54) offers no particular computational advantage over the FIR version since we still have to store $L$-1 terms. There are fewer additions in (6.54) but additions are not computationally expensive. However, we can use the IIR version of the moving average filter to approximate a moving average and thereby gain some efficiency. Equation (6.53) has the term $x[k\text{-}L]$ which can be approximated by the average value $A_M[k\text{-}1]$. The difference equation then simplifies to

$$A_M[k] = \frac{L-1}{L}A_M[k-1] + \frac{1}{L}x[k] \qquad (6.55)$$

The corresponding transfer function is

$$H_M(z) = \frac{1}{L} \cdot \frac{z}{z-(L-1)/L} \tag{6.56}$$

**Example 6.21**

The following MATLAB® code creates 1000 random integers whose average value ranges from 0 to 5 for the first 500 numbers and holds the average at 5 for the next 500 numbers. Use the MATLAB® filter function to filter this sequence using an FIR moving average filter of length 20 and compare the results to the same sequence when filtered with the approximate moving average sequence of (6.47) where $L$ = 20.

```
randSeq = (1:500)/500;
a = 0; b = 10;
randSeq = randSeq.*(a + (b-a).*rand(1, 500));
randSeq = [randSeq (a + (b-a).*rand(1, 500))];
```

**Solution**

The MATLAB® sequence below uses the MATLAB® filter function to filter a pseudorandom sequence with a length 20 FIR filter and a length 20 filter based on (6.47). The results are shown side by side in Figure 6.41.

```
L = 20;                          %Length
nT = [1:1000];                   %x-axis vector
numFIR = (1/L)*ones(1, L);       %Create an FIR filter of length L
denFIR = 1;
yFIR = filter(numFIR, denFIR, randSeq); %Filter the sequence
figure(1);clf;
plot(f, yFIR);                   %Plot the result
num = (1/L)*[1 0];               %Approximate filter numerator
den = [1 -(L-1)/L];              %  and denominator
y = filter(num, den, randSeq); %filter the sequence
figure(2);clf;
plot(f, y, 'k');                 %Plot the result
```

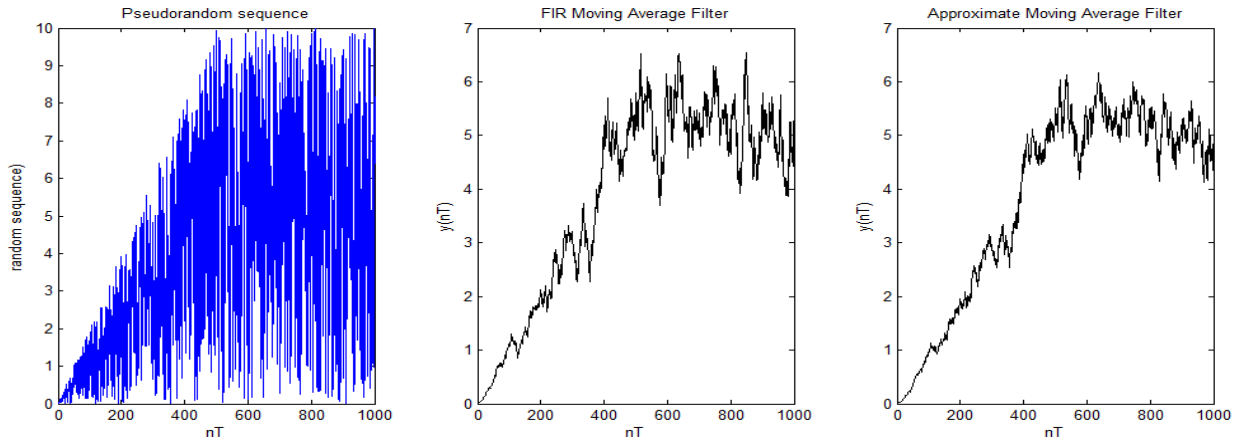



**Figure 6.41**

The plot on the left is that of the original pseudorandom sequence. The middle figure shows the sequence filtered by a length 20 FIR moving average filter. On the right, the pseudorandom sequence is filtered by an approximate moving average filter whose transfer function is in Equation (6.47).

**IIR Comb Filters**

The comb filter derives its name from the fact that its magnitude response resembles a comb with the teeth pointing up. The comb filter is used to separate data which occurs at periodic intervals. Examples of applications where comb filters may be used include: 1) Applications which look at cycles of the sun, moon, the tides, and other naturally occurring periodic phenomena. 2) Radar or sonar images where a moving target is being tracked and the background appears in each image at a new location. 3) Filtering of periodic noise such as power line noise and its harmonics.

As with the case of the moving average filter, the comb filter can be implemented as either a FIR filter or an IIR filter. A FIR comb filter can be created from a moving average filter by adding in a single zero at z = +1. The transfer function for such a FIR filter is

$$H_C(z) = K\frac{z^N - 1}{z^N} \tag{6.57}$$

A typical FIR comb filter of order 10 is shown in Figure 6.42.

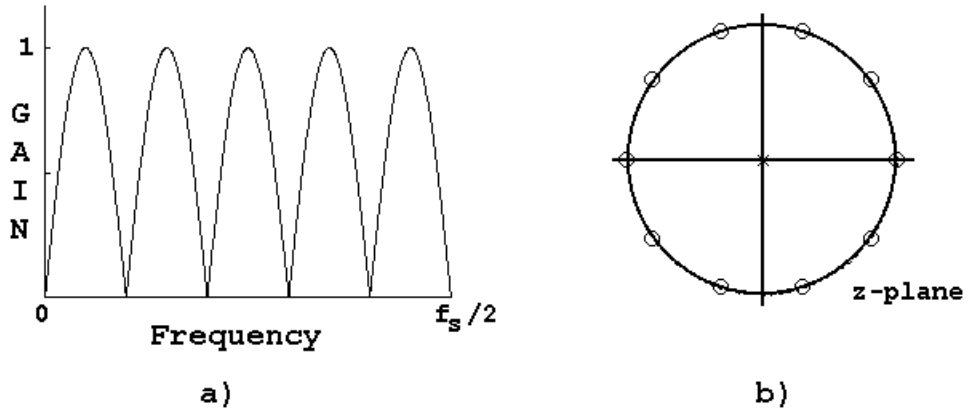



**Figure 6.42**

A FIR comb filter created by adding a zero to a moving average filter. The order is 10. If the order is odd there is no zero at $z = -1$ and the gain at $f_s/2$ is unity.

We can sharpen the lobes of the comb filter by moving the poles at the origin radially out to the vicinity of the unit circle and spaced between the zeros. This changes the filter to an IIR filter and increases the computational load. This type of filter is easily designed using pole/zero placement. The transfer function for a "sharpened" filter is given by

$$H_{CS}(z) = K\frac{z^N - 1}{z^N + r^N} \tag{6.58}$$

where r is the pole magnitude. Figure 6.43 illustrates the pole/zero plot and the magnitude plot for a sharpened comb filter of order 10.

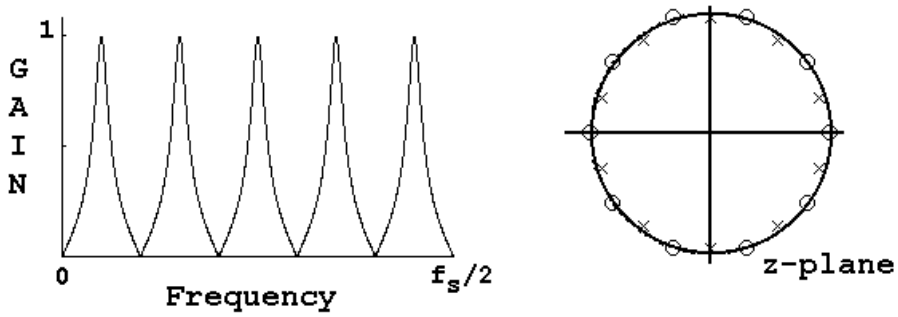
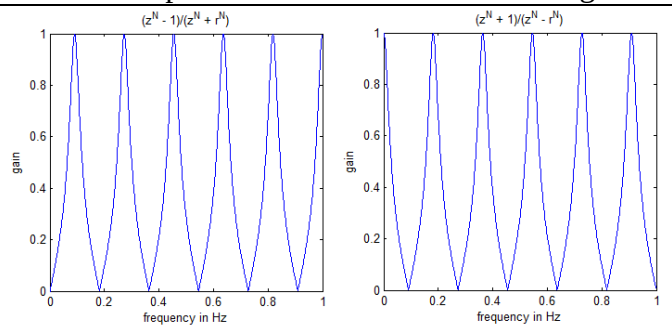
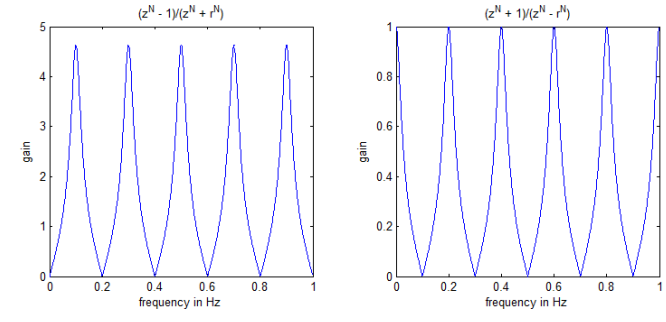
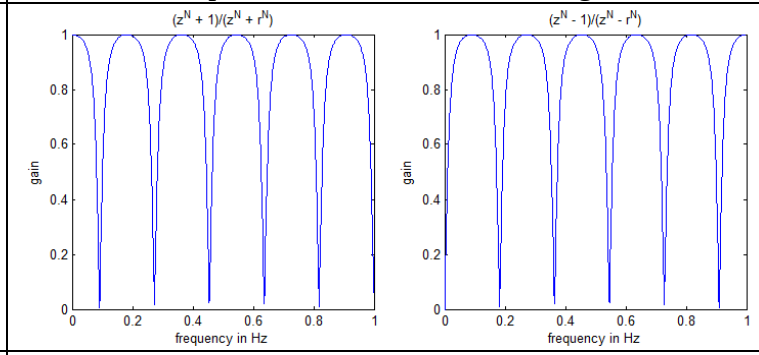
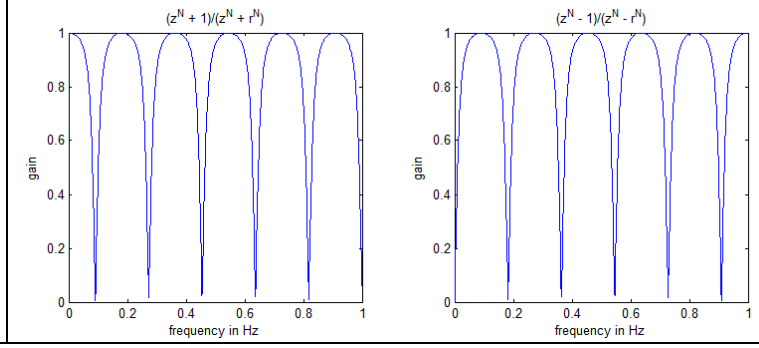



**Figure 6.43**
A sharpened comb filter. The poles lie on a circle of radius .95 and are equally spaced between the zeros.

In equation 6.58 we can changes the sign of the pole and zero and we can change the order from even to odd to alter the shape of the final comb filter. If the pole and zero have different signs, the poles will fall between the zeros. If the pole and zero have the same sign, the poles and zeros will be at the same angle and the filter will acquire a more classic comb shape. Changing the order from odd to even determines whether or not a pole or a zero is at z = +1 or z = -1. Figure 6.43 illustrates these differences.

| | poles and zeros have different signs |
|---|---|
| Even order |  |
| Odd order |  |

| | poles and zeros have same sign |
|---|---|
| Even order |   |
| Odd order |   |

A second method of designing a comb filter involves a mapping function. That is, if we take a moving average filter which has a single main lobe at 0 Hz and substitute $z^R \rightarrow z$ we will get a new filter in which there are $R$ duplicates of the main lobe over the interval $-f_s/2 \le f \le f_s/2$. To understand why this is true, consider that the frequency response of a digital filter is obtained by substituting $e^{j\omega T} \rightarrow z$. To scale the frequency variable we multiply $\omega$ by a scaling constant, $R$. Since $e^{j\omega RT}$ corresponds to $z^R$ it is clear that raising $z$ to a power is equivalent to frequency scaling.

To illustrate this method consider an IIR fifth order moving average filter given by

$$H_M(z) = \frac{1}{5}\frac{z^5 - 1}{z^4(z-1)}$$

To make five duplicates of this filter we make the substitution $z^5 \rightarrow z$ to get

$$H_C(z) = K\frac{z^{25} - 1}{z^{20}(z^5 - 1)} = K\frac{z^{25} - 1}{z^{25} - z^{20}}$$

where $K$ is chosen to make the maximum gain unity. The magnitude plot and the pole/zero plot is shown in Figure 6.44.

This technique of creating a comb filter can be applied to any narrow band low pass filter. Example 6.21 illustrates the design of a comb filter from a third order low pass filter.

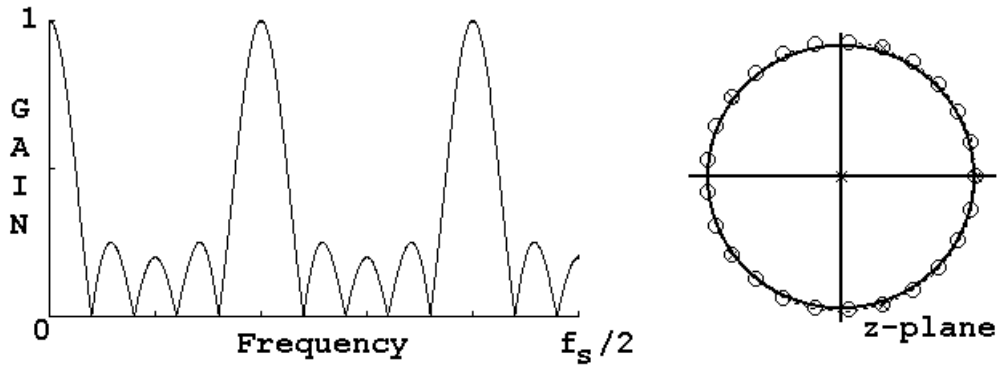



**Figure 6.44**
A comb filter created by duplicating a moving average filter in frequency space.

**Example 6.21**
Create an IIR comb filter with a repetition factor of 7 from the third order elliptic low pass filter given by

$$H_E(z) = .021714\frac{z^3 - .848137z^2 - .848137z + 1}{z^3 - 2.7117z^2 + 2.4905z - .77213}$$

This filter has a pass band edge at 30Hz and a stop band edge at 60Hz. The pass and stop band ripple are .05. The sample frequency is 1000Hz.

**Solution:**
The magnitude plot for the original low pass elliptic filter is given in Figure E7.21-1. For a repetition factor of $R = 7$ we make the following substitution in the transfer function: $z^7 \rightarrow z$.

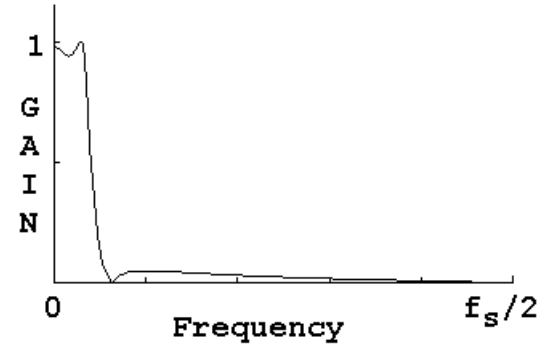



**Figure 6.45**
A low pass third order elliptic filter.

The transfer function for the comb filter is given by:

$$H_C(z) = .021714\frac{z^{21} - .848137z^{14} - .848137z^7 + 1}{z^{21} - 2.7117z^{14} + 2.4905z^7 - .77213}$$

Figure 6.46 shows the magnitude and pole/zero plot for the comb filter.

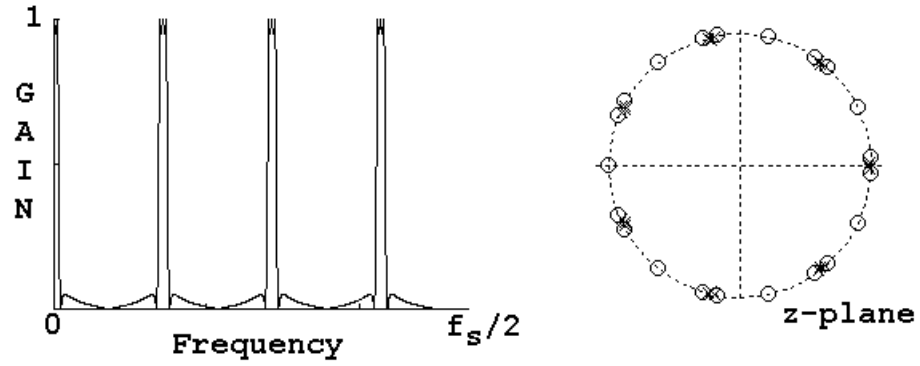



**Figure 6.46**
The magnitude plot and pole/zero plot for a comb filter created from a third order low pass elliptic filter. The repetition factor is seven.

---

**Inverse Filters**
Consider an application in which data is sent over a transmission line which alters the frequency characteristics of the data in an undesirable manner. It is possible in many cases to correct the undesirable changes made by the transmission line by processing the data with a filter which has the inverse frequency characteristics of the transmission line. Such a filter is referred to as an *inverse filter.*

If a system has a transfer function given by $H(z) = N(z) / D(z)$, then the inverse filter for this system is given by $H_I(z) = D(z) / N(z)$ so that $H(z) \cdot H_I(z) = 1$. The obvious problem with creating the inverse filter is that the zeros of the original system will become the poles of the inverse filter so that a system which has zeros located on or outside the unit circle will produce an unstable inverse filter.

**Example 6.22**
Consider a transmission line which has improperly matched impedance on both the source and the load ends. In this case a signal transmitted is reflected at both ends and tends to reverberate in the channel. If the impulse response of such a channel is given by
$h(nT) = \{1, 1/4, 1/16, 1/64, \cdots\}$
find an appropriate inverse filter.

**Solution:**
In this case we can write the impulse response in closed form as
$h(nT) = (1/4)^n$
The corresponding *z* transform is
$$(1/4)^n \Leftrightarrow \frac{z}{z - 1/4} = H(z)$$
The inverse filter is given by

$$H_I(z) = \frac{z - 1/4}{z}$$

The inverse filter is a FIR filter whose impulse response is given by

$h_I(nT) = \{1, -1/4, 0, 0, \cdots\}$

If the inverse filter is appended to the reverberating transmission line, the transfer function for the corrected system is

$$H_C(z) = \frac{z}{z - 1/4} \cdot \frac{z - 1/4}{z} = 1$$

or, in the time domain, the impulse response of the corrected system is given by the convolution of the impulse response of the transmission line and the impulse response of the inverse filter.

$$h_C(nT) = h(nT) * h_I(nT) = \sum_{k=0}^{\infty} h(kT) \cdot h_I([k-n]T) = \delta(t)$$

---

In some applications it is not feasible to write the system function, $H(z)$, in closed form. It may possible to solve these systems in the time domain and arrive at the impulse response function for the inverse filter. By definition of the inverse filter we can write

$H(z) \cdot H_I(z) = 1$

This equation in the time domain becomes a convolution summation.

$$\sum_{k=0}^{\infty} h_I(kT) \cdot h([k-n]T) = \delta(nT) \tag{6.59}$$

In (6.59), if we know the impulse response terms for the original system we can find the impulse response terms for the inverse system by solving for $h_I(nT)$. Equation (6.59) can be solved by constructing a convolution table, or we can use the MATLAB® function *deconv* (deconvolution) to solve the equation.

**Example 6.23**

A sensor has an impulse response function given by

$$h(t) = \left| e^{-t/.01} \sin(200\pi t) \right|$$

Use (6.50) to find the impulse response of the inverse filter.

**Solution:**

The following lines in MATLAB® create a sampled version of this response.

```
Tau = .01;f = 100;
t = linspace(0,.8,64);
for i = 1:512
  h(i) = abs(exp(-t(i)/Tau)*sin(2*pi*f*t(i)));
end
```

This value of $h(nT)$ can be applied to (6.59) which can be solved for the impulse response of the inverse filter $h_I(nT)$. To do this we must solve the convolution equation backwards. In MATLAB® the `deconv` function can be used to do inverse convolution. Thus we want to deconvolve the delta function out of the impulse response. If we think of the coefficients of the impulse response as coefficients of a polynomial, then the convolution of two impulse response

functions is equivalent to polynomial multiplication. Deconvolution is equivalent to polynomial division. In MATLAB® the `deconv` function takes the form of
`[q r] = deconv(a,b).`
If a and b are thought of as vectors representing polynomial coefficients then a/b = q with a remainder of r. In our case we need to create a vector to represent the delta function that is sufficiently long to accommodate the division. In MATLAB® this can be done with the following statements:

```
delta = zeros(1, 127);
delta(1) = 1;
```

The inverse filter impulse response terms can be created using the `deconv` function.

```
[hinv r] = deconv(delta,h);
```

Figure 6.47 shows the results of using `deconv` to create the inverse filter impulse response.

A transfer function could be written for the filter as an FIR filter or, by using Prony's method, as an IIR filter. In both cases, the impulse response must be truncated so that the result is an approximation. For the IIR filter, reasonable results can be achieved with a 10th order filter.

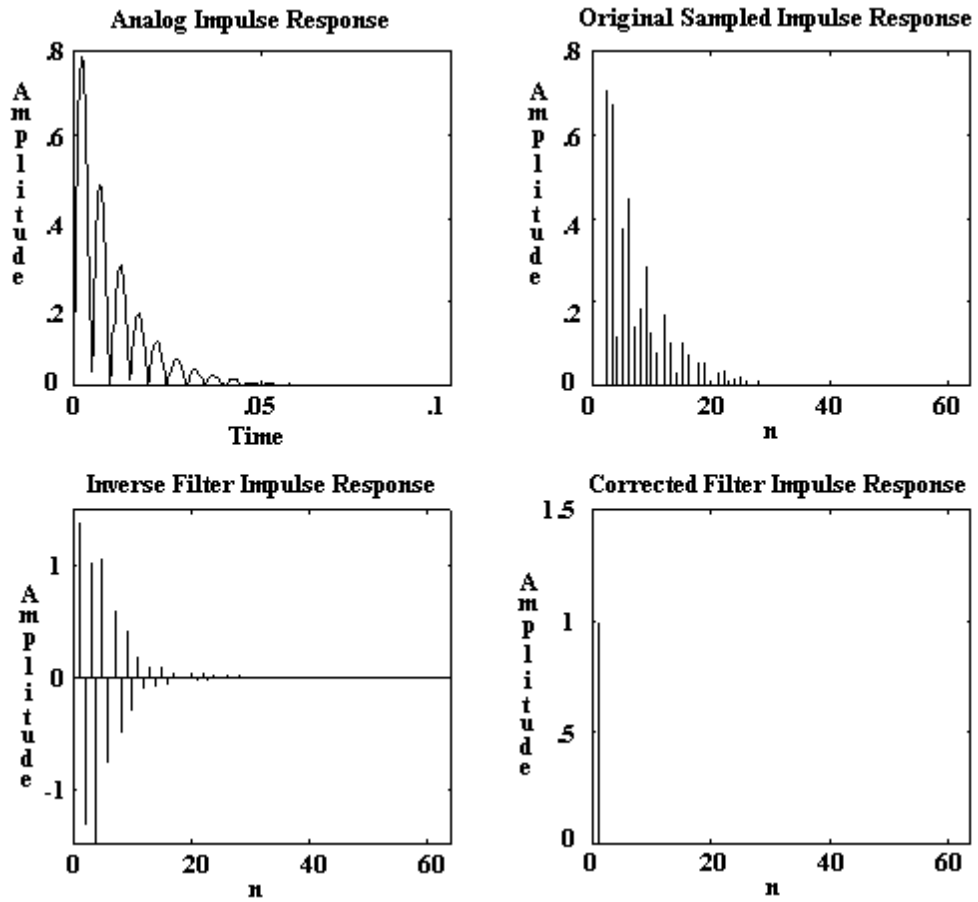



**Figure 6.47**
This figure shows the creation the impulse response terms for an inverse filter using the *deconv* function in MatLab. The corrected impulse response is the impulse response of the cascaded original filter and its inverse.