Part One

Efficient Digital Filters

Chapter 1

Lost Knowledge Refound: Sharpened FIR Filters

Matthew Donadio
Night Kitchen Interactive

What would you do in the following situation? Let's say you are diagnosing a DSP system problem in the field. You have your trusty laptop with your development system and an emulator. You figure out that there was a problem with the system specifications and a symmetric FIR filter in the software won't do the job; it needs reduced passband ripple or, maybe, more stopband attenuation. You then realize you don't have any filter design software on the laptop, and the customer is getting angry. The answer is easy: you can take the existing filter and *sharpen* it. Simply stated, filter sharpening is a technique for creating a new filter from an old one [1]–[3]. While the technique is almost 30 years old, it is not generally known by DSP engineers nor is it mentioned in most DSP textbooks.

1.1 IMPROVING A DIGITAL FILTER

Before we look at filter sharpening, let's consider the first solution that comes to mind, filtering the data twice with the existing filter. If the original filter's transfer function is $H(z)$, then the new transfer function (of the $H(z)$ filter cascaded with itself) is $H(z)^2$. For example, let's assume the original lowpass N-tap FIR filter, designed using the Parks-McClellan algorithm [4], has the following characteristics:

Number of coefficients: $N = 17$

Sample rate: $F_s = 1$

Passband width: $f_{\text{pass}} = 0.2$

Passband deviation: $\delta_{\text{pass}} = 0.05$ (0.42 dB peak ripple)

Streamlining Digital Signal Processing: A Tricks of the Trade Guidebook, Second Edition. Edited by Richard G. Lyons.

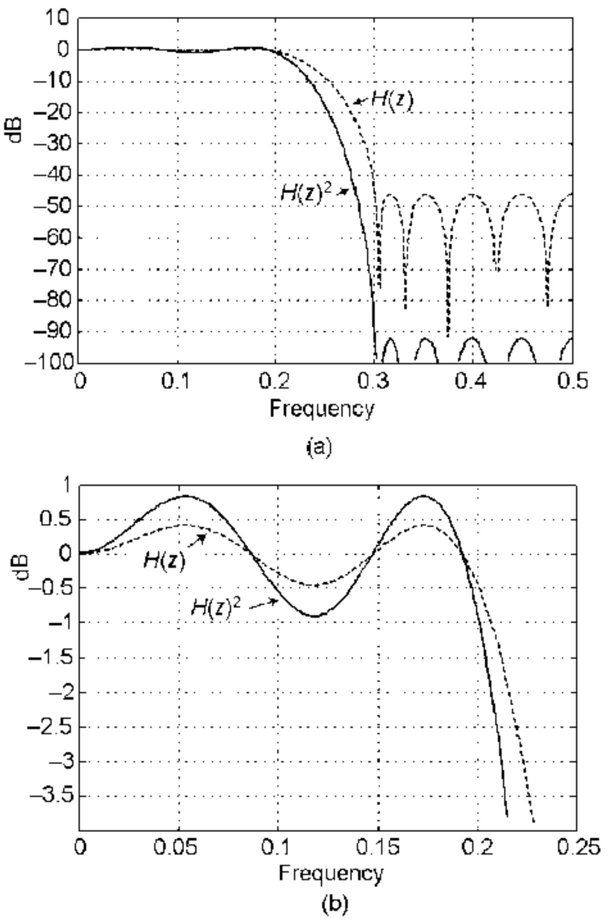

Figure 1–1 *H*(*z*) and *H*(*z*)^2 performance: (a) full frequency response; (b) passband response.

Stopband frequency: $f_{\text{stop}} = 0.3$

Stopband deviation: $\delta_{\text{stop}} = 0.005$ (–46 dB attenuation)

Figure 1–1(a) shows the performance of the *H*(*z*) and cascaded *H*(*z*)^2 filters. Everything looks okay. The new filter has the same band edges, and the stopband attenuation is increased. But what about the passband? Let's zoom in and take a look at Figure 1–1(b). The squared filter, *H*(*z*)^2, has larger deviations in the passband than the original filter. In general, the squaring process will:

1. Approximately double the error (response ripple) in the passband.
2. Square the errors in the stopband (i.e., double the attenuation in dB in the stopband).
3. Leave the passband and stopband edges unchanged.

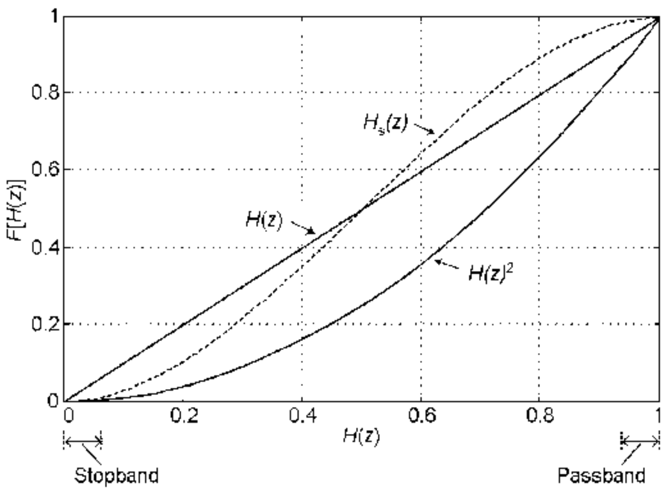

Figure 1–2 Various $F[H(z)]$ functions operating on $H(z)$.

4. Approximately double the impulse response length of the original filter.
5. Maintain filter phase linearity.

It is fairly easy to examine this operation to see the observed behavior if we view the relationship between $H(z)$ and $H(z)^2$ in a slightly unconventional way. We can think of filter squaring as a function $F[H(z)]$ operating on the $H(z)$ transfer function. We can then plot the output amplitude of this function, $H(z)^2$, versus the amplitude of the input $H(z)$ to visualize the amplitude change function.

The plot for $F[H(z)] = H(z)$ is simple; the output is the input, so the result is the straight line as shown in Figure 1–2. The function $F[H(z)] = H(z)^2$ is a quadratic curve. When the $H(z)$ input amplitude is near zero, the $H(z)^2$ output amplitude is closer to zero, which means the stopband attenuation is increased with $H(z)^2$. When the $H(z)$ input amplitude is near one, the $H(z)^2$ output band is approximately twice as far away from one, which means the passband ripple is increased.

The squaring process improved the stopband, but degraded the passband. The improvement was a result of the amplitude change function being horizontal at zero. So to improve $H(z)$ in both the passband and stopband, we want the $F[H(z)]$ amplitude function to be horizontal at both $H(z) = 0$ and $H(z) = 1$ (in other words, have a first derivative of zero at these points). This results in the output amplitude changing slower than the input amplitude as we move away from zero and one, which lowers the ripple in these areas. The simplest function that meets this will be a cubic of the form

$$F(x) = c_0 + c_1x + c_2x^2 + c_3x^3. \tag{1–1}$$

Its derivative (with respect to x) is

$$F'(x) = c_1 + 2c_2x + 3c_3x^2. \tag{1–2}$$

Specifying $F(x)$ and $F'(x)$ for the two values of $x = 0$ and $x = 1$ allows us to solve (1–1) and (1–2) for the c_n coefficients as

$$F(x)\big|_{x=0} = 0 \Rightarrow c_0 = 0 \tag{1–3}$$

$$F'(x)\big|_{x=0} = 0 \Rightarrow c_1 = 0 \tag{1–4}$$

$$F(x)\big|_{x=1} = 1 \Rightarrow c_2 + c_3 = 1 \tag{1–5}$$

$$F'(x)\big|_{x=1} = 0 \Rightarrow 2c_2 + 3c_3 = 0. \tag{1–6}$$

Solving (1–5) and (1–6) simultaneously yields $c_2 = 3$ and $c_3 = -2$, giving us the function

$$F(x) = 3x^2 - 2x^3 = (3 - 2x)x^2. \tag{1–7}$$

Stating this function as the sharpened filter $H_s(z)$ in terms of $H(z)$, we have

$$H_s(z) = 3H(z)^2 - 2H(z)^3 = [3 - 2H(z)]H(z)^2. \tag{1–8}$$

The function $H_s(z)$ is the dotted curve in Figure 1–2.

1.2 FIR FILTER SHARPENING

$H_s(z)$ is called the "sharpened" version of $H(z)$. If we have a function whose z-transform is $H(z)$, then we can outline the filter sharpening procedure, with the aid of Figure 1–3, as the following:

1. Filter the input signal, $x(n)$, once with $H(z)$.
2. Double the filter output sequence to obtain $w(n)$.
3. Subtract $w(n)$ from $3x(n)$ to obtain $u(n)$.
4. Filter $u(n)$ twice by $H(z)$ to obtain the output $y(n)$.

Using the sharpening process results in the improved $H_s(z)$ filter performance shown in Figure 1–4, where we see the increased stopband attenuation and reduced passband ripple beyond that afforded by the original $H(z)$ filter.

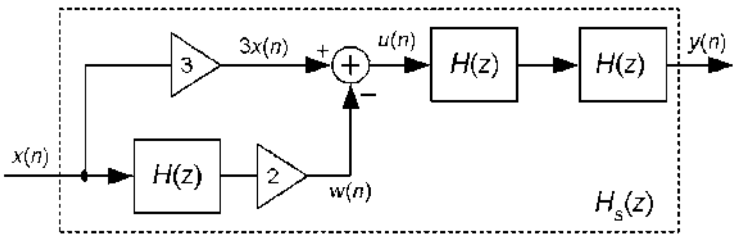

Figure 1–3 Filter sharpening process.

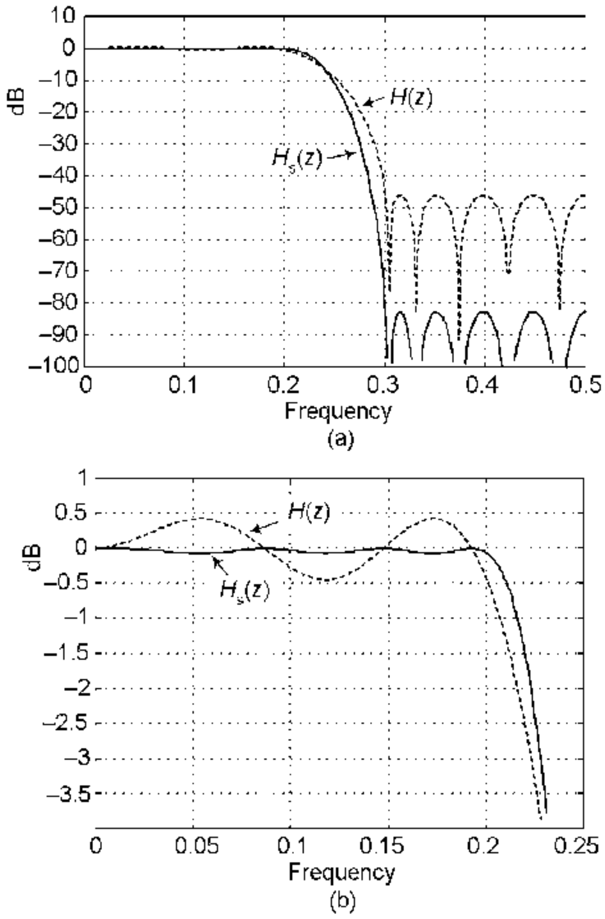

Figure 1–4 $H(z)$ and $H_s(z)$ performance: (a) full frequency response; (b) passband response.

It's interesting to notice that $H_s(z)$ has the same half-amplitude frequency (–6 dB point) as $H(z)$. This condition is not peculiar to the specific filter sharpening example used here—it's true for all $H_s(z)$s implemented as in Figure 1–3. This characteristic, useful if we're sharpening a halfband FIR filter, makes sense if we substitute 0.5 for $H(z)$ in (1–8), yielding $H_s(z) = 0.5$.

1.3 IMPLEMENTATION ISSUES

The filter sharpening procedure is very easy to perform, and is applicable to a broad class of FIR filters; including lowpass, bandpass, and highpass FIR filters having symmetrical coefficients and even-order (an odd number of taps). Even multi-passband FIR filters, under the restriction that all passband gains are equal, can be sharpened.

From an implementation standpoint, to correctly implement the sharpening process in Figure 1–3 we must delay the $3x(n)$ sequence by the group delay, $(N - 1)/2$ samples, inherent in $H(z)$. In other words, we must time-align $3x(n)$ and $w(n)$. This is analogous to the need to delay the real path in a practical Hilbert transformer. Because of this time-alignment constraint, filter sharpening is not applicable to filters having nonconstant group delay, such as minimum phase FIR filters or infinite impulse response (IIR) filters. In addition, filter sharpening is inappropriate for Hilbert transformer, differentiating FIR filters, and filters with shaped bands such as sinc compensated filters and raised cosine filters, because cascading such filters corrupts their fundamental properties.

If the original $H(z)$ FIR filter has a nonunity passband gain, the derivation of (1–8) can be modified to account for a passband gain G, leading to a "sharpening" polynomial of

$$H_{\mathrm{s,gain>1}}(z) = \frac{3H(z)^2}{G} - \frac{2H(z)^3}{G^2} = \left[\frac{3}{G} - \frac{2H(z)}{G^2}\right]H(z)^2. \tag{1–9}$$

Notice when $G = 1$, $H_{\mathrm{s,gain>1}}(z)$ in (1–9) is equal to our $H_{\mathrm{s}}(z)$ in (1–8).

1.4 CONCLUSIONS

We've presented a simple method for transforming a FIR filter into one with better passband and stopband characteristics, while maintaining phase linearity. While filter sharpening may not be used often, it does have its place in an engineer's toolbox. An optimal (Parks-McClellan-designed) filter will have a shorter impulse response than a sharpened filter with the same passband and stopband ripple, and thus be more computationally efficient. However, filter sharpening can be used whenever a given filter response cannot be modified, such as software code that makes use of an unchangeable filter subroutine. The scenario we described was hypothetical, but all practicing engineers have been in situations in the field where a problem needs to be solved without the full arsenal of normal design tools. Filter sharpening could be used when improved filtering is needed but insufficient ROM space is available to store more filter coefficients, or as a way to reduce ROM requirements. In addition, in some hardware filter applications using *application-specific integrated circuits* (ASICs), it may be easier to add additional chips to a filter design than it is to design a new ASIC.

1.5 REFERENCES

[1] J. Kaiser and R. Hamming, "Sharpening the Response of a Symmetric Nonrecursive Filter by Multiple Use of the Same Filter," *IEEE Trans. Acoustics, Speech, Signal Proc.*, vol. ASSP–25, no. 5, 1977, pp. 415–422.

[2] R. Hamming, *Digital Filters*, Prentice Hall, Englewood Cliffs, NJ, 1977, pp. 112–117.

[3] R. Hamming, *Digital Filters*, 3rd ed., Dover, Mineola, NY, 1998, pp. 140–145.

[4] T. Parks and J. McClellan, "A Program for the Design of Linear Phase Finite Impulse Response Digital Filters," *IEEE Trans. Audio Electroacoust.*, vol. AU–20, August 1972, pp. 195–199.

EDITOR COMMENTS

When $H(z)$ is a unity-gain filter we can eliminate the multipliers shown in Figure 1–3. The multiply-by-two operation can be implemented with an arithmetic left-shift by one binary bit. The multiply-by-three operation can be implemented by adding a binary signal sample to a shifted-left-by-one-bit version of itself.

To further explain the significance of (1–9), the derivation of (1–8) was based on the assumption that the original $H(z)$ filter to be sharpened had a passband gain of one. If the original filter has a nonunity passband gain of G, then (1–8) will not provide proper sharpening; in that case (1–9) must be used as shown in Figure 1–5. In that figure we've included a *Delay* element, whose length in samples is equal to the group delay of $H(z)$, needed for real-time signal synchronization.

It is important to realize that the $3/G$ and $2/G^2$ scaling factors in Figure 1–5 provide optimum filter sharpening. However, those scaling factors can be modified to some extent if doing so simplifies the filter implementation. For example, if $2/G^2 = 1.8$, for ease of implementation, the practitioner should try using a scaling factor of 2 in place of 1.8 because multiplication by 2 can be implemented by a simple binary left-shift by one bit. Using a scaling factor of 2 will not be optimum but it may well be acceptable, depending on the characteristics of the filter to be sharpened. Software modeling will resolve this issue.

As a historical aside, *filter sharpening* is a process refined and expanded by the accomplished R. Hamming (of Hamming window fame) based on an idea originally proposed by the great American mathematician John Tukey, the inventor of the radix-2 fast Fourier transform (FFT).

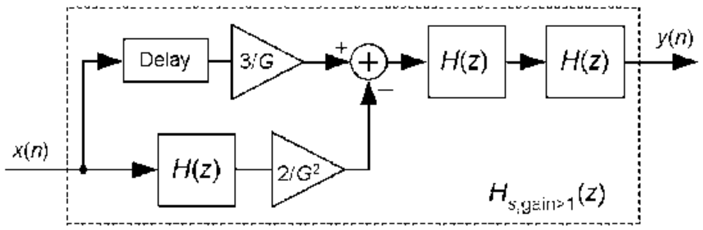

Figure 1–5 Nonunity gain filter sharpening.