The Manifold Joys of Conformal Mapping: Applications to Digital Filtering in the Studio – 2nd Try

James A. Moorer

This is an edited version of a paper that appeared in the Journal of the Audio Engineering Society [1]. Many typos have been corrected, and some supplementary material has been added. It has also been combined with an IEEE article [2] that also involves conformal mapping.

Abstract:
Conformal mapping is a well-known technique for modifying the cut-off or resonant frequencies of filters. Conformal mapping is applied to the problem of digital filters for professional audio production. Methods are examined for computing the coefficients of presence filters, such as parametric equalizers with or without shelving, and of band-pass or band-reject filters. Formulas are given for the coefficients of these filters in terms of the specifications (center frequency, boost/cut, and so on), which are exact and relatively straightforward to compute. A new kind of shelving filter is introduced that is equiripple in the passbands and can have an arbitrarily narrow transition band (with increasing filter order).

Introduction

Every studio mixing desk has a filter section for each and every channel. These go from the simplest bass-treble controls to sophisticated parametric boost/cut filters. Also quite popular as "outboard" equipment are graphic equalizers and high-order band-stop and notch filters. With the coming of digital audio it seems certain that the mixing desks of the future will have to realize all these functions entirely in the digital domain. To realize some of the real advantages of digital control (precision and repeatability), the characteristics of such filters must be very well understood, and their design formulas must be exact and explicit. In this paper, we go over some of the most common studio filters (and some rather uncommon ones) and develop design equations for each of them. These equations will allow the implementer to calculate exactly the filter coefficients for the desired characteristics.

A second running theme in the paper is conformal mapping. We use this mapping as a tool for decomposing the problem of filter design into individual stages of which each one is tractable. Since many of the readers will not be familiar with this technique, it is explored in a tutorial fashion (up to a point).

I will use radian frequency in the digital domain, which is independent of the sampling rate. An angle q in the digital domain corresponds to the frequency $f = f_s \frac{q}{2p}$, where

f_s is the sampling frequency. In Table 1, we use normalized frequency, which differs by a factor of $2p$. Normalized frequency is the same as frequency at a sampling rate of unity.

I will use *z* to represent the unit advance operator in both the transformed and the untransformed filters. In the literature, the transformed domain will often have a new and different variable representing the unit advance. I am avoiding this proliferation of variables here because I am dealing with both the analog domain and the digital domain. All transfer functions in the analog domain will use the variable *s* to denote the differentiation operator, and the domain will be called the *s*-plane. Similarly, the digital - domain will be called the *z*-plane.

About Conformal Maps

The maps that I will discuss are complex functions of a complex variable. They map a simply-connected domain (*i.e.*, the complex plane) onto another simply-connected domain (also the complex plane) [4]. It has been known for some time that conformal mapping has applications in filtering. It has been used to map low-pass filters into high-pass, band-pass, or band-stop filters in the analog domain [5]. It has been used to map analog filters into the digital domain [6]. It has been noted elsewhere [2, 3] that conformal maps can be used in the digital domain to map a prototype low-pass filter into a filter having any number of pass and stop bands. I will try to demonstrate here the use of conformal mapping applied to the problem of digital filter coefficient determination for professional digital audio usage (graphic and parametric equalizers, high-selectivity bandpass and bandstop filters, and others).

A digital filter is represented mathematically as a rational function of negative powers of *z*, the unit advance operator. The presence of z^{-1} indicates a one-sample delay in these formulas. The class of conformal maps that I will deal with in this paper are all rational functions of *z*. In fact, they could be considered filters in their own right, but this identification is not particularly useful. These rational functions are all what we call *all-pass* filters, in that the magnitude of their frequency response is exactly unity. They represent pure phase distortion. All-pass filters have been used as colorless reverberators [7] and as phase equalization for other kinds of filters [8].

In general, a conformal map does not have to be an all-pass filter, but making it so provides one important advantage: the resulting frequency response will have the same as the original frequency response but with the frequency axis displaced, stretched or squashed, and possibly replicated. The all-pass filter when used as a conformal map *maps points on the unit circle onto the unit circle*. That is, each point on the unit circle in the untransformed domain will be mapped into one (or more) points on the unit circle in the transformed domain (if the all-pass is not degenerate). If the all-pass is first order (*i.e.*, the numerator and denominator polynomials contain no greater powers of *z* than 0 or 1), then there will be exactly one image point for every untransformed point. If the all-pass is second order, there will be exactly two image points, and so on. If we imagine a filter with a certain cutoff frequency, applying a first-order conformal map to the filter will produce a new filter with a different cutoff frequency. All properties relating to relative values of the magnitude of the filter, such as passband ripple or stopband ripple, will be unchanged. Properties relating to the frequency of the filter, such as transition width or locations of poles and zeros, will be altered in a predictable manner.

The Littlest Conformal Map

Skipping over the trivial conformal map, which is the identity map, the simplest mapping I will consider here is a bilinear all-pass transform:

$$(1) \qquad z^{-1} = \frac{a + z_1^{-1}}{1 + a z_1^{-1}}$$

This maps the complex variable z_1 onto the complex variable z. Clearly if $a=0$, the map is the identity and nothing is changed. This map is very important to digital filtering, since it maps points on the unit circle onto points on the unit circle, changing only angular position (frequencies). This means that if you have a digital filter with some cutoff or resonant frequency, you can produce a new digital filter with a different cutoff or resonant frequency by substituting the right side of (1) for every occurrence of z in the prototype filter. The new filter will be a rational function of z_1. We can see how this works by substituting $z_1 = e^{jf}$ and $z = e^{jr}$. Thus f corresponds to transformed frequency and r corresponds to the un-transformed frequency. After some manipulation, we obtain the relation between these two as follows:

$$(2a) \qquad \tan r = \frac{(1-a^2)\sin f}{2a + (1+a^2)\cos f}$$

$$(2b) \qquad 0 = a^2 \sin(r+f) + 2a\sin(r) + \sin(r-f)$$

This shows that the cutoff frequency of the original filter r is moved to a new position f. This equation can be solved for either f or r so that a can be calculated given these two data. To invert the formula, for instance, so as to exchange the roles of r and f, we need only replace a with $-a$ in (2a). To solve this in a general fashion for a, given f and r, involves solving the quadratic equation (2b) in a. The two roots will be reciprocal – one inside the unit circle and one outside. The inside root, of course, should be selected.

The significance of this is that we need only figure out the filter we want once for a given frequency, then we can transform it *ad naseum* to any other frequency. With some extra computation, this calculation can be done along with the application of the filter to a set of samples to give time-varying filters with a frequency response with very precise control[1].

To find out how the root locations of the prototype filter have changed, we just substitute for *z* in (1) the specific root as a complex number, then solve for z_1, which will be the image of the root in the transformed filter. Solving for z_1, we get the following relation:

$$z_1^{-1} = \frac{-a + z^{-1}}{1 - az^{-1}} \tag{3a}$$

and similarly,

$$z_1 = \frac{-a + z}{1 - az} \tag{3b}$$

This relation is identical in form to (1) with –*a* substituted for *a*. This should not be entirely surprising. This is just the inverse transform which would map a transformed filter back to its original form. This shows that we may invert (2) by exchanging r and f and replacing *a* with –*a*. I will note without proof that it is straightforward to show [4] that the bilinear transforms form a group[2].

As noted above, this map is interesting because it is easier to design some filters around particular frequencies rather than around an arbitrary frequency. This is clearly demonstrated in the case of the presence filter.

The Digital Presence Filter

This section is taken largely from McNally [9], except for our usage of the conformal map rather than the *z*-transform.

A presence filter, as is commonly found on commercial mixing consoles, is a single second-order section with a pole pair and a zero pair. By adjusting the parameters of this filter, we can obtain a boost or a cut of any desired amount. Similarly, the bandwidth of the boost or cut may be freely adjusted. It is our task here to show how the coefficients of this filter may be computer from the specifications, which are boost/cut amount in dB, bandwidth, and center frequency. Figure 1 shows an example of the frequency response of a presence filter. A presence filter, for our purposes, is defined to have unity response

[1] Conformal mapping provides a solution to one of the issues in time-varying filters, namely the calculation of the filter coefficients. The implementation of time-varying filters involves other problems as well. See [Moorer] for more details on the problem of time-varying filters.

[2] The general bilinear transform is $z^{-1} = \frac{a + bz_1^{-1}}{c + dz_1^{-1}}$. We are only concerned with a special case of this transform. To force the transform to be an all-pass, *a* and *d* must be equal, and *b* and *c* must be equal. Furthermore, to normalize to unit amplitude, *c* must be one.

at 0 and p. In Figure 1a, we see the presence filter centered around the frequency $p/2$. In this case, the response is perfectly symmetric, and the bandedges q_- and q_+ will not be independent, since $q_+ = p - q_-$. In Figure 1b, the center frequency has been moved to w by use of the transformation shown in (1). The response is no longer symmetric, and the new bandedges q_- and q_+ are related to the old bandedges by (2a) where r is either q_- or q_+. The transformation preserves the unit response at 0 and p.

We may note here one important difference between the analog presence filter and the digital presence filter, and, in fact, between analog filters and digital filters of the same order in general. Whereas the response of the analog presence filter is symmetric when plotted on a log-frequency scale, the response of the digital presence filter is not. The reason why is quite simple: the magnitude-squared response of an analog filter is a ratio of polynomials in w, the applied frequency. The magnitude-squared response of a digital filter is a ratio of polynomials in $\cos(w)$. For a given order, these cannot be equal. This is *regardless of the method of calculating the coefficients*. Digital filters designed by conformal mapping, *z*-transform, or any other method will share this property. It is a somewhat fortunate side effect, however, that at low frequencies (lower than about 1/6 of the sampling rate) or high Q (greater than about 4), the responses of analog and digital presence filters are substantially equal. At higher frequencies, there will be differences which may or may not be significant.

The bandwidth of the digital presence filter needs some explanation. If we are boosting by some amount, then the response of the filter reaches a maximum at the resonant frequency and falls off both above and below the resonant frequency. The bandwidth is usually defined as the difference between the frequencies above and below the resonant frequency where the response falls to 3dB below the maximum value. It is analogously defined for a cut as for a boost. The problem comes when we specify a boost of less than 3dB. We will adopt the convention (after McNally [9]) that when the boost or cut is less than 6 dB, the bandwidth will be defined by the frequency points where the gain falls by half the boost or cut amount in dB. For example, if we ask for a 4dB boost, then the bandwidth will be the difference between the frequencies where the gain reaches 2dB.

The frequency $p/2$ is "magic" in many ways. In the case of the presence filter, the response is exactly symmetric about the center frequency, as shown in Figure 1. The pole-zero plot, also shown in Figure 1, is exactly symmetric about the *j*-axis. This yields the transfer function of the filter in a marvelously simple form:

$$(4) \qquad T(z) = \frac{(1+a_n) + z^{-2}(1-a_n)}{(1+a_d) + z^{-1}(1-a_d)}$$

For this to be stable, a_n and a_d should have magnitudes less than one. If $a_n = a_d$, then the response of the filter is identically one. If a_d is greater in magnitude than a_n, then the filter will boost (*i.e.*, show a gain at resonance ($=p/2$) greater than one). Otherwise, the filter will cut.

Setting $z = e^{jq}$, the magnitude-squared response of the filter can be expressed as follows:

(5) $$\left|T(z = e^{jq})\right|^2 = \frac{(1+a_n^2)+(1-a_n^2)\cos(2q)}{(1+a_d^2)+(1-a_d^2)\cos(2q)}$$

From either (4) or (5), we note that at $q = \frac{p}{2}$ this reduces to the following:

(6) $$\left|T(z = e^{jp/2} = j)\right|^2 = \frac{a_n^2}{a_d^2} \equiv A^2$$

I will use *A* to denote the magnitude of the response of the filter at resonance. For any other frequency, I will use *F* to denote the magnitude of the response of the filter[3].

To design a presence filter, we are given *A*, the magnitude at resonance, *F*, the desired response at a given frequency, and q, which is that frequency. By convention, q is taken to be the 3dB point (where $F = A/\sqrt{2}$), or the halfway point when *A* is of magnitude less than 6 dB (where $F = \sqrt{A}$)[4]. Solving for a_n and a_d yields the following:

(7a) $$a_d^2 = \frac{F^2-1}{A^2-F^2}\cot^2(q)$$

(7b) $$a_n^2 = A^2 a_d^2 = A^2\frac{F^2-1}{A^2-F^2}\cot^2(q)$$

Since the response is perfectly symmetric around $p/2$, the bandwidth will be $2\left|q - p/2\right|$. This completes the design of the prototype filter with a resonant frequency of $p/2$ and an arbitrary boost or cut amount, and an arbitrary bandwidth. It remains now only to transform this to the desired frequency.

Replacing every occurrence of z^{-1} in (4) with the right-hand side of (1), we obtain the following, transformed filter:

[3] This follows the notation of McNally[1981].
[4] I am being somewhat careless in switching between amplitude (*A* and *F*) in decibels versus linear scale. When *A* is small, we set *F* to one-half of *A* when expressed in dB. This is equivalent to setting *F* to the square-root of *A*.

(8) $$T(z) = \frac{\left\{(1+a^2) + a_n(1-a^2)\right\} + 4az^{-1} + \left\{(1+a^2) - a_n(1-a^2)\right\} z^{-2}}{\left\{(1+a^2) + a_d(1-a^2)\right\} + 4az^{-1} + \left\{(1+a^2) - a_d(1-a^2)\right\} z^{-2}}$$

To realize this, of course, we must make the first denominator term unity by dividing each coefficient by $\left\{(1+a^2) + a_d(1-a^2)\right\}$.

We must now determine the value of *a* which maps the frequency $p/2$ into any desired center frequency w. This may be done by substituting $f = w$ and $r = p/2$ into the inverse of (2). This yields the following:

(9) $$a = \tan\left\{\frac{1}{2}(w - \frac{p}{2})\right\}$$

For $w = p/2$, the computed value of *a* will be zero, which is what we should expect.

At this point, the resulting bandwidth is unknown. We must find a way to relate the transformed bandwidth with the un-transformed bandwidth. Although the response of the prototype filter is perfectly symmetric about $p/2$, the response of the transformed filter will be asymmetric about w. This may be found, as before, by application of the inverse of (2). Let us define q_+ and q_- as the high and low bandedges of the prototype filter, such that $q_+ - q_-$ is the bandwidth of the untransformed filter. These will be symmetric about $p/2$ so that $q_+ = p - q_-$. Similarly, let us define f_+ and f_- as the transformed bandedges, so that the transformed (*i.e.*, desired) bandwidth is $f_+ - f_-$. We can then compute the transformed bandwidth as follows:

(10) $$\tan(f_+ - f_-) = \frac{(1-a^4)\sin(2q_-)}{2a^2 - (1+a^4)\cos(2q_-)}$$

This formula is derived by using (2) to compute $\tan(q_+)$ and $\tan(q_-)$, then combining them using the angle difference identity for tangents. This formula means that we do not have to know what the exact values of f_+ and f_- will be beforehand: it is sufficient to know only their difference, which will is the desired (transformed) bandwidth of the resulting filter. From *a* and the desired bandwidth, we can calculate q_-, which allows us to compute the desired filter coefficients. We solve for q_- by first defining the angle d and the quantities *T* and *M*:

(11a) $$d \equiv \arctan\left\{\frac{1+a^4}{1-a^4}\tan(f_+ - f_-)\right\}$$

(11b) $$T \equiv \tan(f_+ - f_-)$$

(11c) $M \equiv (1+a^4)^2 T^2 + (1-a^4)^2$

The bandedge frequency may then be computed as follows:

(12) $$q_- = \frac{1}{2}\left\{ \arcsin\left(\frac{2a^2}{M} T \right) - d \right\}$$

Since the arcsine is a many-valued function, we may arbitrarily assign q_- to the first quadrant then choose either $\arcsin()$ or $p - \arcsin()$ in (12) to yield a first-quadrant value of q_-. We may then substitute q_- into (7) to compute a_n and a_d which completes the design of the filter. Appendix I shows a C-subroutine which performs this task.

We note a number of differences between the filter computed by this method and the *z*-transform method described by McNally [9]:

- With conformal mapping, the frequency of the maximum boost/cut and the bandwidth are *exactly* determined, whatever their value. Since the *z*-transform method is an approximation rather than an exact calculation, there is some deviation. Table I shows the deviation in Q and frequency for a range of values.
- For frequencies above about 1/6 of the sampling rate, the magnitude response of the filter designed by conformal mapping becomes noticeably skewed toward the lower frequencies, whereas the *z*-transform produces responses skewed towards higher frequencies. The responses are virtually identical at lower frequencies or higher values of Q. There is no way with second-order filters to achieve perfectly symmetric curves over all values of frequency.
- Our method requires somewhat more compute time to determine the coefficients than the *z*-transform method.

Neither this method nor the *z*-transform method produces ideal behavior when the center frequency of the filter approaches $p/2$, although the conformal map method still produces filters with exact parameters. The requirement that the magnitude of the filter go to zero at $p/2$ is somewhat over-constraining. Orfanidis [10] derives a presence filter that overcomes the problems at high frequencies.

We might note parenthetically here that whereas center frequency and Q are sufficient to describe exactly the response of a second-order analog filter, these are insufficient to describe the digital filter. We might speculate as to what would be a better measure. For instance, rather than the frequency of the maximum boost or cut, we might specify the centroid on a log-frequency plot. This is not a particularly convenient number to calculate, but it expresses more precisely where the energy will be concentrated (or eliminated). Better yet would be a measure that is related more directly to our perception, but this is even less tractable.

Notch Filters

Although the presence filter as defined above can be used to greatly attenuate, the definition of bandwidth does not quite match that used for notch filters. Generally, a notch filter is defined as having unity response at frequencies of zero and $p/2$, and a response of exactly zero at some other frequency. These are often used to eliminate, say, 60 Hz hum from a recording. We may generalize this slightly by allowing the response to be non-zero, rather than exactly zero at the notch frequency. All the design equations above still apply, except that the definition of *F* is changed. With the presence filter, *F* is strictly dependent on the value of *A*. This presents in interesting problem when *A* is set to zero, since *F* also becomes zero. Instead, *F* is set to an absolute value of -3dB, unless, of course, *A* is greater than -6dB, where *F* may be set to $\sqrt{A}$. Note that if we set *A* to zero, a_n also becomes zero. By this definition, the bandwidth refers to the point where the response is reduced from unity to -3dB. This definition of bandwidth produces significantly sharper filters (*i.e.*, higher *Q*) than the definition used for presence filters. The *Q* can be reduced by allowing *A* to be non-zero. To be prudent, *A* should be set to the highest value (*i.e.*, least amount of cut) required to lower the offending signals to an acceptable level. Higher values of *Q* simply increase the response time of the filter without necessarily producing any further effects on the signal being attenuated.

About High-Q Filters:

This seems a good place to insert a comment about the use of high-*Q* filters in general. This comes up often with the use of notch filters to attenuate 60Hz hum. Of course, we all want to preserve the recording as faithfully as possible, so the tendency is to use very high-Q notch filters. There is nothing wrong with this in principle, but one must be aware of the consequences. A 60 Hz notch with a *Q* of 100 has a bandwidth of 0.6 Hz. This means that the transient (impulse) response is on the order of 1.7 seconds long. When a 60 Hz tone is presented to this filter, it will take about 1.2 seconds for it to fall by 6dB, then 1.2 more seconds to fall by 6 more dB, and so on. Moreover, any non-constancy in the 60 Hz tone will cause it to be audible again, since it will take some time for the filter to adapt. For instance, if you edit a recording that has 60 Hz hum, then try to get rid of the hum with a notch filter, you will find that the hum re-appears at *every* edit point. This is because in editing the recording, you produce discontinuities in the hum, which is like forcing the filter to restart. The proper way to apply this kind of high-order filter is to have plenty of room at the beginning of the recording for it to get going (or, more precisely, for the transient response to die out) before the program material starts, and to do all the high-order filtering *before* editing the recording. It is generally not possible to eliminate all hum from a recording without introducing some audible artifacts.

Digital Shelving Filters

Following McNally [9], we may define the transfer function of a shelving filter as follows:

(13) $$T(z) = \frac{(1+2sg_n + g_n^2) - 2(1-g_n^2)z^{-1} + (1-2sg_n + g_n^2)z^{-2}}{(1+2sg_d + g_d^2) - 2(1-g_d^2)z^{-1} + (1-2sg_d + g_d^2)z^{-2}}$$

This formula can be derived by starting in the *s*-plane (“analog” plane) with a pole and a zero that are at the same angle but different radii (g_n and g_d). (13) results from a bilinear transform of the *s*-plane onto the unit circle [6]. In our case, the exact transformation used was $s = (1-z^{-1})/(1+z^{-1})$ which maps the normalized frequency s=j into $p/2$.

From this, we can determine the gain at zero frequency ($z=1$) and at the Nyquist rate ($z=-1$) as follows:

(14a) $$T(z=1) = \frac{g_n^2}{g_d^2}$$

(14b) $$T(z=-1) = 1$$

Thus we see that the filter of (13) is a low-shelving filter, *i.e.*, it has a unit response at high frequencies, and a freely adjustable response at low frequencies. The behavior in between will be determined entirely by s. Curiously, the only value of s that makes sense is $s = \sqrt{2}/2$. This is the fastest transition between the response at low frequencies and the response at high frequencies that still has monotonic response. With this value of s, the filter response will move smoothly between the high-frequency magnitude (14b) and the low-frequency magnitude (14a). Any smaller value of s will produce a noticeable peak in the response. Any larger value widens the transition at mid-frequencies. Taking this value for s, we can compute the magnitude response as follows:

(15) $$\left|T(z=e^{jq})\right| = \frac{1+g_n^4}{1+g_d^4}$$

To design the filter, we will take *A* to be the gain at zero frequency, and *F* to be the gain at $p/2$. As before, *F* will be taken to be the 3dB point of the filter in the case where *A* is greater than 2 (or less than ½). Otherwise $F = \sqrt{A}$. We may now solve (14a) and (15) for g_n and g_d, which yields these design relations:

(16a) $$g_d = \left\{\frac{F^2-1}{A^2-F^2}\right\}^{\frac{1}{4}}$$

(16b) $$g_n = \sqrt{A}\left\{\frac{F^2-1}{A^2-F^2}\right\}^{\frac{1}{4}}$$

By substituting the right side of (1) for each occurrence of *z* in (13), we can then transform the filter to any cutoff frequency w. Again, (9) shows how to calculate *a* from the given bandedge frequency w.

As noted before, (13) is a low-shelving filter. If we wish a high-shelving filter (*i.e.*, the response is unity at zero frequency), then all that is necessary is to substitute –*z* for every occurrence of *z* in (13). The net effect is to change the sign of the z^{-1} term in the numerator and denominator. Note that this substitution should be done *before* applying the transformation of (1) to set the center frequency. This produces high-shelving filters with precise gain and center frequencies. This substitution is another very simple kind of conformal map. In general, substitution of –*z* for *z* will preduce a new filter with the frequency response reversed, *i.e.*, low-pass filters become high-pass filters and *vice-versa*.

We have used $s = \sqrt{2}/2$ in this derivation, since this produces the smoothest response. Smaller values may be used to produce "peaking" effects. The design proceeds as above, but then after g_n and g_d, have been calculated, any desired value of s may be used in (13).

Figure 2 shows the response (untransformed) of a number of digital shelving filters. For all the curves except the 5dB boost and the 5dB cut, the response at $p/2$ differs from the response at zero by 3dB. At the 5dB points, the response differs by 2.5dB. For 20dB boost, the values of *A* and *F* are pointed out by dotted lines.

Appendix II shows a C program to perform the calculations described above.

Higher-Order Filters

So far, I have discussed only second-order filters. Higher-order filters are generally used in professional audio production on an occasional basis. High-order bandpass or bandreject filters are sometimes used to try to separate the desired sound from some annoying background noise.

I will discuss only one general form of high-order filter, and that is one derived from a particular conformal map. This form includes as a special case the elliptic low-pass filter as described in Gold and Rader [6] and implemented in Gray and Markel [11]. It also includes new forms of filters that we will call the elliptic shelving and presence filters. These forms feature equiripple behavior at all frequencies and independent control over the gains of pass and stop bands and transition widths. All these filters may be derived from a single model using conformal mapping.

At this point, I must apologize to the reader. I am no longer able to keep the level to an introductory or tutorial level. To understand fully the remainder of the paper, the reader would have to know something more about conformal mapping and elliptic functions than I can offer here. The references by Neville [12] and Cayley [13] are helpful, although any number of textbooks contain similar information.

Conformal mapping is useful in high-order filters because generating filter coefficients for high-order filters is difficult. With first- and second-order filters, we can often solve the transfer functions explicitly to obtain closed-form expressions for the coefficients, as was done for (4) and (13). For higher-order filters, the design requires the solution of systems of simultaneous nonlinear equations and do not yield simple, closed-form solutions except in certain very special cases. With conformal mapping, we can often give a method, if not a closed-form solution, for determining the filter coefficients with great confidence and accuracy.

Taking this argument to its logical extreme, the approach we will use here is to design, in the *z*-plane, a filter with one zero and one pole that has the required high and low ripple, but unconstrained transition width. Then, by conformal mapping, we will produce a higher-order filter with the same characteristics but increasingly narrow transition width until the desired width is obtained. The resulting filter will have identical pass and stop-band behavior as the prototype first-order filter.

The Elliptic Plane

Let us start with a theoretical infinite-order filter with a regular 2-dimensional array of poles and zeros shown in Figure 3. Since this array extends indefinitely in all directions, either of the two vertical lines shown is an axis of symmetry. In fact, between any pair of columns of poles, or pair of columns of zeros lies an axis of symmetry. It is clear, then, by inspection, that the magnitude of the (infinite-order) rational function represented by this pole-zero array is *equiripple* along these axes of symmetry. It cannot be otherwise. To see this, all that is necessary is to imagine a ripple inspector sitting on one of the roots. The inspector gazes about and sees some ripple. The inspector is then blindfolded, snatched up, and placed on any other root of the same type (pole or zero) and the blindfold is removed. No matter which way the inspector gazes, all that can be seen is the identical array of roots. Consequently, the waves of ripple must appear absolutely identical from the point of view of one peak to the other. Consequently, the ripple must be identical everywhere in all rows and columns off to infinity in every direction. The values along the line between the columns of zeros will be different from the values along the line between the columns of poles, but each will be exactly equiripple. Although this is not a realizable transfer function in its own right, it does posses the equiripple property by inspection. I will call this the *u*-plane.

The place where conformal mapping comes in is that *any rectangle* in this infinite array (providing its edges fall along axes of symmetry) may be mapped onto the upper half-plane by use of the inverse Jacobian elliptic function $sn^{-1}(u \mid k)$. That is, the *finite* area represented by a rectangle in the *u*-plane may be mapped onto the *infinite* area of the *s*-plane, which in turn may be mapped into the unit circle, thus completing our filter design. This rectangle in the *u*-plane may be scaled to any size or rotated by any multiple of 90^{o}. The important constraint, as noted above, is that the edges of the rectangle must fall along the axes of symmetry of the pole-zero array. This is because the Riemann-Schwartz symmetry principle (See Nehari [4], Chapter V, Section 5) forces this to be so. If we map some other rectangle, the net effect will be that the edges of the rectangle we choose will *automatically* become axes of symmetry. That is, the poles and zeros in the *u*-plane won't be where we think they were. It will realize an equiripple transfer function with a regular array of poles and zeros symmetrically placed around whatever rectangle you specify. These poles and zeros will be the original ones as shown in Figure 3 but reflected and overlapped by the new axes of symmetry[5]. Another way to look at it is to think that the rectangle we map will be extended forever in each dimension by reflection (that is, by "flopping" the rectangle over each edge again and again). Each extended rectangle will contain the exact same pole-zero configuration, or its reflection. A third way of looking at this property is to say that although we map a rectangle from the *u*-plane to the *s*-plane, we cannot know *which* of the infinite array of rectangles will be mapped. Consequently, they must contain the same pole-zero combination, which will be the superposition of *all* the reflected and extended rectangles over the *u* plane.

The mapping we will make is to map a segment of the vertical line between the poles into the low-frequency band, and a segment of the vertical line between the zeros into the high-frequency band. The upper-half rectangle will represent the upper-half *s*-plane. The upper and lower edges of the rectangle must fall exactly between two pole-zero rows (by the symmetry principle) and will represent the transition band between the low-frequency response and the high-frequency response. We may adjust the height of the rectangle to enclose any number of pole-zero pairs. This will have the effect of narrowing the transition band.

Making the Prototype Filter

Note that the pole-zero array of Figure 3 is entirely determined by u_n, u_d, and the vertical spacing of the pole-zero rows. If we can determine these parameters to give the desired values of high and low-frequency ripple, then the design can be solved. My approach, as mentioned above, is to reduce the problem to one that can be solved exactly in closed form, then progressively extended in aq manner that preserves the desired low- and high-frequency ripple but progressively narrows the transition band to the desired range. I do this by making a mapping rectangle that encloses only one pole and one zero. The pole and zero will then lie along the real axis. Figure 4 shows this mapping process. The points u_n and u_d (Figure 4c) will be mapped successively into s_n and s_d (Figure 4b), and then into r_n and r_d (Figure 4a). I then choose r_n and r_d to produce the desired pass- and stop-band ripple. To complete the process, I expand the mapping rectangle to include more pole-zero pairs, thus increasing the order of the resulting filter and reducing the transition width to the desired neighborhood. Since u_n and u_d are held constant during this expansion, the low- and high-frequency ripple amounts will remain constant.

[5] This property appears to be forgotten from time to time. Storer [1957] refers to a paper by Mattai [1952] where the property is routinely ignored. The technique of Mattai does not work, except in some very restricted examples.

To compute r_n and r_d, I first define *A*, *B*, *C*, and *D* to be the magnitude of the response of a 1-pole, 1-zero digital filter at the frequencies 0, $p/2$ (the end of the low-frequency band and the beginning of the transition band), q (an unknown representing the end of the transition band), and p. These are the quantities that the user specifies. As an example, to make a shelving filter, we might specify that $A = 1$, and that A/B represents a .1-dB low band ripple. Similarly, we might specify that $D = 0.1$ giving a 10-dB high-frequency cut and that C/D represents a .1 dB high band ripple. The low and high band ripple do not have to be the same. Since the prototype filter has only one real pole and one real zero, the response must be monotonic. The term "ripple" is abused in the prototype filter, since we don't really have "ripple" until we look at higher-order versions of the filter. In the prototype filter, we are really just talking about the specific quantities *A*, *B*, *C*, and *D*, since these will define the ripple in the resulting filter. The maxima and minima in the high and low bands of the resulting filter will exactly achieve the values *A*, *B*, *C*, and *D*. Since the response of the prototype filter is monotonic, we require that $A > B > C > D$ or $A < B < C < D$ for a real-valued solution.

If we set $D = 0$, we get the classical elliptic low-pass filter as discussed in Gold and Rader [6], Storer [14], Grey and Markel [11], and others. Any nonzero value of D produces a kind of shelving filter.

We start by representing the transfer function of the prototype filter as follows:

(17) $$T(z) = G\frac{1 - r_n z^{-1}}{1 - r_d z^{-1}}$$

We can form four simultaneous (nonlinear) equations by equating (17) to the given values *A*, *B*, *C*, and *D* as follows:

(18a) $$T(z = 1) = G\frac{1 - r_n}{1 - r_d} \equiv A$$

(18b) $$\left| T(z = e^{j\frac{p}{2}} = j) \right|^2 = G^2 \frac{1 - r_n^2}{1 - r_d^2} \equiv B^2$$

(18c) $$\left| T(z = e^{jq}) \right|^2 = G^2 \frac{1 + r_n^2 - 2r_n \cos(q)}{1 + r_d^2 - 2r_d \cos(q)} \equiv C^2$$

(18d) $$T(z = -1) = G\frac{1 + r_n}{1 + r_d} \equiv D$$

G is included as an arbitrary scale factor so that we have four equations in four unknowns. Noting that (18a), (18b), and (18d) do not involve q, we may solve them for

r_d first, then r_n, G, and q. Solving for r_d yields the following quadratic equation and solution:

(19a) $$0 = 1 - \frac{2(A^2 - D^2)}{A^2 + D^2 - 2B^2} r_d + r_d^2$$

(19b) $$r_d = \frac{A^2 - D^2 \pm \sqrt{(A^2 - B^2)(B^2 - D^2)}}{A^2 + D^2 - 2B^2}$$

This solution yields two values that will be reciprocals: one of magnitude greater than unity and the other with magnitude less than unity. We will, of course, choose the one inside the unit circle. From this value of r_d, we may calculate the other unknowns as follows:

(20a) $$r_n = \frac{-(A-D) + (A+D) r_d}{(A+D) - (A-D) r_d}$$

(20b) $$G = A \frac{1 - r_d}{1 - r_n}$$

(20c) $$q = \arccos \left\{ \frac{C^2 (1 + r_c^2) - G^2 (1 + r_n^2)}{2(C^2 r_d - G^2 r_n)} \right\}$$

We must be careful to choose *arccos()* such that the resulting value of q is in the second quadrant.

This completes the design of the prototype filter in the *z*-plane. We next map it successively into the *s*-plane and then the *u*-plane. The mapping to the *s*-plane is done by a bilinear transform [6]. This yields the real-axis images of the pole and zero as follows:

(21a) $$s = \frac{1 - z^{-1}}{1 + z^{-1}}$$

(21b) $$s_n = \frac{r_n - 1}{r_n + 1}$$

(21c) $$s_d = \frac{r_d - 1}{r_d + 1}$$

Similarly, the bandedge frequency q is mapped onto the imaginary axis on the *s*-plane as $w = \tan(q/2)$.

To then map this to the *u*-plane, we must first find the *parameter* of the elliptic function, denoted by k. This determines the aspect ratio of the rectangle in the *u*-plane.

The bilinear transform of (21a) is patterned after that of Gold and Rader [6]. This will map the frequency $p/2$ in the *z*-plane into $w = 1$ in the *s*-plane. Note that this is different from the blinear transform of (1) in that there is no parameter, and it is not an all-pass filter. This transform will map the unit circle onto the jw-axis (and *vice-versa*). The relation between frequencies on the circle and the axis will be the tangent function mentioned above.

I should note here that the difference between this development and that of the standard elliptic low-pass filter is that we have allowed the positions of the zeros to be defined by a parameter, u_n. The elliptic low-pass filter forces r_n to -1, which causes the two adjacent columns of zeros in Figure 3 to be coincident, so that all zeros of the resulting filter lie on the unit circle. This filter is widely respected because it has the narrowest transition width for the specified amount of ripple and number of roots[6].

The Jacoby Elliptic Functions

The Jacobi elliptic functions are, in some sense, a generalization of the trigonometric functions and can be derived (see Cayley [13]) by purely geometric arguments. The trigonometric functions are periodic when the argument is real, but are monotonic when the argument is imaginary. The elliptic functions are doubly periodic in that they are periodic both for pure real and pure imaginary arguments. The period along the real axis is denoted *K(k)*. The period along the imaginary axis is denoted by *K'(k)*. *K* and *K'* are called *complete elliptic integrals of the first kind*. *K* is called the *parameter* of the integral. *K* and *K'* are related in that $K'(k) = K(\sqrt{1-k^2})$. The quantity $\sqrt{1-k^2}$ is denoted by *k'* so that $K'(k) = K(k')$. When the parameter *k* is evident by context, its explicit mention is dropped. The function $sn(u\,|\,k)$ maps the real line into a rectangle in the following manner:

(22a) $sn(0, k) = 0$

(22b) $sn(K(k)\,|\,k) = 1$

(22c) $sn(K(k) + jK'(k)\,|\,k) = \dfrac{1}{k}$

(22d) $sn(jK'(k)\,|\,k) = \infty$

[6] It is not necessarily the minimum transition width when the number of explicit poles and zeros are allowed to differ, or when equiripple behavior is not required.

The function $sn(u\,|\,k)$ is an *odd* function, so $sn(-u\,|\,k) = -sn(u\,|\,k)$. We have given in equations (22a, b, c, and d) the four corners of the rectangle. The elliptic function is such that values in-between those mentioned fall on straight lines between the points mentioned in the four equations. Consequently, the straight line from 0 to *K(k)* is mapped into the straight line from 0 to 1. Similarly, the straight line from *K(k)* to *K(k)+jK'(k)* maps into the straight line from 1 to $1/k$. The line from *K(k)+jK'(k)* to *jK'(k)* maps into the line from $1/k$ to $+\infty$. The fourth side, the line from *jK'(k)* to *0* maps into the positive imaginary axis from $+j\infty$ to *0*. Thus we see that the inverse elliptic function, $sn^{-1}(s\,|\,k)$ maps the real axis onto a rectangle with the imaginary axis as the "crosspiece" of the rectangle. This is the mapping we want.

<u>Prototype to Final Form</u>

The mapping I will use may be represented like this:

(23) $$u = j\,sn^{-1}(-js\,|\,k)$$

I included the factors *j* and *–j* to effect a rotation of the rectangle by 90º, thus mapping the imaginary axis in the *s*-plane. This will map *j* in the *s*-plane to *jK* in the *u*-plane, and *w* (the *s*-plane image of the bandedge frequency q) into *jK-K'*. This means that we can calculate the parameter of the elliptic function directly from (22c) as follows:

(24) $$k = \cot(\frac{q}{2})$$

It now remains only to calculate the root images u_n and u_d, which lie along the negative real axis in the *u*-plane. Using the following identity:

(25) $$sn(ju\,|\,k) = j\frac{sn(u\,|\,k')}{cn(u\,|\,k')} \equiv sc(u\,|\,k')$$

We can then calculate u_n and u_d as follows:

(26a) $$u_n = sc^{-1}(s_n\,|\,k')$$

(26b) $$u_d = sc^{-1}(s_d\,|\,k')$$

The inverse elliptic function $sc^{-1}(s\,|\,k)$ can be computed by use of the arithmetic-geometric mean Abramowitz and Stegun, [15], equations 17.4.41, 17.6.1, 17.6.8, and 17.6.9.

Once we have calculated the *u*-plane images, we have all the information about the infinite pole-zero array. The pole columns are placed at the coordinates

$j(iK'(k) \pm u_d)$ and the zero columns are placed at the coordinates $j(iK'(k) \pm u_n)$ for all integer values of *i*. The vertical spacing of the pole-zero rows will be *K(k).*

To find a new filter of higher order, all we need to do is find the parameter *k* of a new rectangle that has the aspect ration *NK/K'* rather than *K/K'*. This rectangle will then enclose *N* pole-zero pairs. The sides of the rectangle will, again, be the vertical lines between the pole columns and between the zero columns, but the top and bottom of the rectangle will now enclose more roots.

We can calculate this new rectangle in two ways. If we are given the desired transition width, this determines the new rectangle automatically, and we need only determine *N*, the number of pole-zero pairs that fit in the new rectangle. This is the approach taken in Gold and Rader [6]. We can also go the other direction and from a given value of *N*, calculate the parameter *k* that gives the appropriate aspect ratio and thus determine the transition ratio that value of *N* yields. This is the approach taken in Gray and Markel [11]. For completeness, we will show the calculations involved in both methods.

To compute *N* from a given value of $\hat{q}$ (in the *z*-plane), we first transform to the *s*-plane by the formula $\hat{w} = \tan(\hat{q}/2)$. The new parameter, $\hat{k}$, from (24), will then be the following:

$$(27) \qquad \hat{k} = \frac{1}{\hat{w}} = \cot(\frac{\hat{q}}{2})$$

We then choose *N* to be the smallest integer greater than the new aspect ratio:

$$(28) \qquad N = \left\lceil \frac{K(\hat{k})K'(k)}{K'(\hat{k})K(k)} \right\rceil$$

We must now take this value of *N* and recomputed $\hat{k}$, which is the second method mentioned above, *i.e.*, calculating $\hat{k}$ given a value of *N*. This will give a transition ratio equal to or less than the desired value. If the exact low- or high-frequency ripple is not critical, then the above value of $\hat{k}$ and *N* can be used directly, which yields a transition ratio that is exact. We can have one or the other of transition ratio or ripple specified exactly, but not both.

Given *N*, we first compute what is called the "nome", *q*, which is defined as $e^{pK'/K}$. From this, there are expansions for *K* and *kK* wholely in terms of the nome. These are equations 16.38.5 and 16.38.7 in Abramowitz and Stegun [15]. We use this detour through the nome because it is one of the few quantities in the theory of Jacobi elliptic functions that has a series expansion that converges relatively rapidly.

Defining $\hat{q}$ as follows:

(29) $$\hat{q} \equiv e^{-p\frac{K'(k)}{NK(k)}}$$

Then we have these two simultaneous equations in $K(\hat{k})$ and $\hat{k}K(\hat{k})$:

(30a) $$\left\{\frac{2K(\hat{k})}{p}\right\}^{\frac{1}{2}} = 1 + 2\sum_{n=1}^{\infty}\hat{q}^{n^2}$$

(30b) $$\left\{\frac{2\hat{k}K(\hat{k})}{p}\right\}^{\frac{1}{2}} = 2q^{\frac{1}{4}}\sum_{n=0}^{\infty}\hat{q}^{n(n+1)}$$

These may be solved for $\hat{k}$ and $K(\hat{k})$.The new column positions will be the following:

(31a) $$\hat{u}_n = u_n \frac{K(\hat{k}')}{K(k')}$$

(31b) $$\hat{u}_d = u_d \frac{K(\hat{k}')}{K(k')}$$

All that is left is to map these into the *s*-plane, then into the *z*-plane. The mapping to the upper *s*-plane may be done by substitution into (23). Defining $n = \lceil N/2 \rceil$ we have the following (for *N* odd):

(32a) $$s_i = j\ sn\left\{ j\hat{u} - 2\frac{K(\hat{k})}{N}i \middle| \hat{k} \right\}$$

for *i=0,1,…,N-1*. Similarly, for *N* even, we have the following:

(32b) $$s_i = j\ sn\left\{ j\hat{u} - 2\frac{K(\hat{k})}{N}(2i+1) \middle| \hat{k} \right\}$$

$\hat{u}$ should be set to $\hat{u}_d$ to find the pole positions and $\hat{u}_n$ for the zero positions. These formulas give the upper half-plane roots only. The lower half-plane roots are, of course, simply the complex conjugates of the upper half-plane roots.

Figure 5 shows the poll-zero array of Figure 3 with the prototype rectangle delineated in heavy lines. The new rectangle, which is *N* times higher, is marked in lighter lines. The dotted lines show the extension by the reflection principle to an infinite array of such rectangles, each containing the same pole-zero configuration. The set of all extensions forms a tessellation of the plane.

These roots are them mapped into the *z*-plane by the bilinear transform as follows:

$$(33) \qquad z_i = \frac{1+s_i}{1-s_i}$$

One complication occurs when $\hat{u} = -K(\hat{k}')$. In this case, the value of s will be infinite. In this case, (32a) should be bypassed, since we know that the value of *z* will be -1. Note that a similar complication occurs in (21), and should be resolved similarly by substitution of $u = -K(k')$, thus bypassing the explicit calculation of s.

To calculate the elliptic function of a complex argument shown in (32), we first expand the function in terms of purely real arguments as follows[7]:

$$(34) \qquad sn(x+jy|k) = \frac{sn(x|k)dn(y|k') + j\,cn(x|k)\,dn(x|k)\,sn(y|k')\,cn(y|k')}{cn^2(y|k') + k^2 sn^2(y|k)\,sn^2(y|k')}$$

This is equation 16.21.2 in Abramowitz and Stegun [15]. The individual elliptic functions of real arguments may be computed by use of the arithmetic-geometric mean (see sections 16.4 and 17.6 of Abramowitz and Stegun [15]).

The resulting filter must be normalized so that the low-band gain is actually equal to *A*. For a filter of odd order, this can be done simply by computing the gain at zero frequency and normalizing appropriately. For even order, the maximum will occur beside a pole in the *u*-plane, so we need only compute the frequency in the *z*-plane where it occurs. This may be done by (32b) with $\hat{u} = 0$ and *i=0*, followed by (33). This will give a value of *z* on the unit circle where the first maximum will occur.

Figure 6 shows an example of this kind of filter. Here, we have taken the ratios *A/B* and *C/D* to be 0.1 dB, which is a very severe constraint on the ripple in these bands. We see that at first order (denoted by *n=1* in the figure), the transition band is so wide as to cover almost the entire high band. At higher order, however, we see that the transition band narrows significantly.

To illustrate the effect of the high and low band ripple, Figure 7 shows a third-order filter with highly exaggerated scale. We see that in each band, the extremes are touched exactly *N+1* times. Note that this means that the elliptic shelving filter, even for *N=2*, never

[7] I apologize to the long-suffering reader about including this formula, especially since it uses the "other" elliptic function, *dn()*, which hasn't been mentioned at all up to this point. Alas, I must refer you to the references for further edification on these points.

looks exactly like the digital shelving filter of (13) with $s = \sqrt{2}/2$, since the response is no longer monotonic. There is some value of s, however, for which these will generate the identical curve for *N=2*.

After the transforms and normalization, we may use (1) to map the filter to the desired frequency. Note that there is one difference in the development in this section which is that the beginning of the transition band was set at $p/2$, whereas the earlier development used $p/2$ as the middle of the transition band. By use of (9), we can calculate the value of *a* that will map the beginning of the transition band into any desired frequency w.

Other Transformations:

The tessellation of the plane by rectangles as noted above is not the only tessellation that produces interesting filters. For example, the well-known Chebychev low-pass filters (Storer [14]) are derived by dividing the complex plane into strips that extend left and right (parallel to the real axis) off to infinity. These are transformed to the *s*-plane by the hyberbolic trigonometric functions.

Other tessellations have not been shown to be useful. For instance, one could contemplate a covering of the plane based on triangles. Recall that each triangle must contain the same number of poles and zeros as every other triangle. Because of the reflection principle, wherever the poles and zeros are placed in the triangle, they must match exactly the locations in every other triangle as you flop (reflect) each triangle about each of its sides. There is no placement of poles and zeros in infinite rows and columns that will satisfy these requirements and produce any interesting “standard” filter, such as a low-pass or band-pass filter. Similarly, more complex repeating figures, such as hexagons, produce even more problems since the reflection principle gets more complicated as the number of edges is increased. To date, no one (that I have heard of) has made much practical use out of any other tessellation than those mentioned here.

High-Order Transformations

So far, we have only discussed transformations that map the unit circle onto something else (*e.g.* the unit circle, a rectangle, the jw axis) on a one-to-one basis. In this section, I will discuss a class of transformations that map the unit circle onto itself more than once. Although this might seem a bizarre thing to do, I will show that it is a way that we can generate band-pass and band-stop filters from prototype low-pass filters. In this manner, we need only design a few low-pass filters for reference, and we may then use those reference designs to produce band-pass or band-stop filters will continuously-variable low and high cutoff points.

The Second-Order Map

A second-order conformal map from the unit circle onto the unit circle may be described as follows:

(35) $$p^{-1} = \frac{a_0 + a_1 z^{-1} + z^{-2}}{1 + a_1 z^{-1} + a_0 z^{-2}}$$

This will convert, for instance, a low-pass filter into either a band-pass or band-stop filter. Constantinides [3] and Moorer [2] show methods of computing the coefficients a_i to produce the desired filter. I will discuss an alternate method that I feel has more intuitive appeal. I will decompose the transformation of (35) into the composition of two first-order bilinear transforms and one trivial second-order transform. This follows Nehari [4], Chapter VI, Section 1. The three transformations in order are described as follows:

(36a) $$z^{-1} = \frac{a + z_1^{-1}}{1 + a z_1^{-1}}$$

(36b) $$z_1^{-1} = z_2^{-2}$$

(36c) $$z_2^{-1} = \frac{b + z_3^{-1}}{1 + b z_3^{-1}}$$

The final filter will be a rational function of z_3. This way of decomposing the transform gives us a step-by-step way of understanding the process. Figure 8 shows this in the context of a prototype elliptic low-pass filter with a cutoff of $p/2$.

As described earlier, the first bilinear transformation will have the effect of changing the cutoff and bandedge of the filter to some different frequencies which may be calculates by (2). The effect of squaring the variable may be seen by observing the mapping of the frequencies involved. A frequency r in the un-transformed domain will be mapped into a frequency f in the transformed domain as follows:

(37) $$f = \frac{r}{2}$$

This transformation maps the unit circle onto itself twice (*i.e.*, two complete revolutions onto one revolution). The net effect is to halve the significant frequencies (*e.g.*, the cutoff frequency, the bandedge, the transition width), and to lower a new copy of the transfer response into the range below the Nyquist rate. For our example of the low-pass filter, the first bilinear transform (36a) moves the cutoff from $p/2$ to, say, r .The squaring (36b) produces the first cutoff at $r/2$, then another cutoff at $p - r/2$. There is, consequently, a stopband centered on $p/2$ which is symmetric and of width $p - r$. This is shown in Figures 8b and 8c. The second bilinear transform (36c) moves the center of the stopband to any desired frequency as shown in Figure 8d.

In general, we are required to work backwards in that we are given the ultimate position of the stopband, and we must then find the coefficients *a* and *b* of the required map. Conveniently enough, the calculation of (12) may be used to determine r from the required bandedges. *b* can be determined beforehand entirely from the desired position of the center frequency by the use of (9). *A* may then be determined as that coefficient that maps $p/2$ to r, also using (9).

This completes the design of a second-order transformation. It remains only to discuss some of the applications. I have shown the use of a prototype low-pass filter to produce bandstop filters with adjustable bandedges. If we substitute –*z* for *z* in the prototype low-pass filter, it will be converted to a high-pass filter. This may then be transformed by (36) into a bandpass filter with adjustable bandedges. Perhaps the most interesting new filter this process yields is the high-order presence filter.

The elliptic presence filter has unity gain (except for some amount of ripple) everywhere but in a certain region f_- to f_+. After a transition band that can be made arbitrarily narrow by increasing the order of the filter, the gain in the noted region may be set to any desired value, with any amount of ripple (or lack of ripple) that is desired.

Figure 9 shows an elliptic presence filter for a 0.1 dB ripple in the bands. Again, for order-2 (the minimum order for a presence filter), the transition band covers almost the entire region. For increasing order, we see that the transition band can be reduced arbitrarily.

This completes the circle. I started this article with the design of second-order presence and shelving filters, then after a lengthy excursion to the elliptic plane, I have concluded with a design method for higher-order shelving and presence filters, with an added bonus of bandpass and bandstop filtering using the identical design technique. All these filters are designed by conformal mapping of relatively simple prototype filters.

Let me mirror my earlier caution about high-*Q* filters for the case of very high-order filters. When the order of, say, an elliptic presence filter is raised, the transient response gets longer and longer. That is, the *Q* of some of the roots of the filter gets larger and larger. Ultimately, the perceptual result is that you may begin to hear a “whistle” at frequencies that correspond roughly to the bandedges. It is near these transition frequencies where the roots will approach the unit circle most closely, and thus can become audible. Care must be used when using high-order filters to keep artifacts down to an acceptable level.

High-Order Notch Filters

Conformal mapping may also be used to create high-order notch filters. We could simply take the same second-order notch described earlier and apply it again and again. This would have the effect of widening the notch. Each application will lower the frequency of the 3 dB point by 3 more dB. After four applications, that same frequency will be down by 12 dB. We could just design a different second-order notch that is even more narrow, so that multiple applications result in an acceptable response. Another way is to start with a prototype low-pass filter and transform it into a notch. This gives us a way to design high-order notch filters of any (even) order.

To start, we select the kind of low-pass filter that has all the zeros at *z=-1*. These filters include the well-known Butterworth and Chebychev forms (Storer [14]). The development would proceed by transformations (36) as shown in Figure 8, except that these prototype filters have only one point where the response goes exactly to zero, in contrast with the elliptic filters which touch zero at *N* points (after the frequency doubling transform of (36b), shown in Figures 8c and 8d). This design procedure is roughly the same as designing bandstop filters.

Figure 10 shows a notch filter that was made by starting with a 5th-order Butterworth low-pass filter with a 3 dB point at 0.45p. The frequency doubling transform of (36b) has moved the zeros to $p/2$. The simple transform of (1) may now be used to move the notch to any desired frequency.

Still Higher-Order Mappings

Moorer [2] showed that any (finite-order) rational all-pass function will transform, say, a prototype low-pass filter into a filter with any number of pass and stop bands. I will denote the critical frequency of the prototype filter by b. Each pass band will have the identical maximum and minimum values (ripple) as the passband of the prototype filter, and likewise for each stop band. The transform can be described as follows:

(38) $$g(z^{-1}) = \frac{\sum_{i=0}^{N} a_i z^{i-N}}{\sum_{i=0}^{N} a_i z^{-i}}$$

We take a_0 to be 1 by convention.

This transform maps the unit circle onto itself a total of *N* times.

To determine the coefficients a_i, we need only equate $g(e^{-jq_k})$ with $e^{\pm jb}$ for each desired bandedge frequency q_k. This yields *N* simultaneous equations of the following form:

(39) $$g(e^{-jq_k}) = e^{\pm jb}, \qquad \textit{for } k=1\ldots N$$

If we represent $\pm b$ by f_k, this reduces to the following (real) set of equations:

(40) $\sum a_i \cos\left\{(i-\frac{N}{2})q_k + \frac{1}{2} f_k\right\} = 0,$ *for k=1…N*

This is *N* simultaneous equations in *N* unknowns (since a_0 is known to be 1) with all coefficients real. We will take the q_k to be order of ascending value. We may then identify the values of the f_k to be b, $2p - b$, $2p + b$, $4p - b$, $4p + b$, and so on. If the band edge frequencies are distinct (that is, b is not zero or p), then (40) has a solution, and the solution is unique.

This transform was rediscovered by Franchitti [16]. He added an explicit proof of uniqueness, and a "fast" method of solving the simultaneous equations (40). He also showed that the above ordering of the bandedge frequencies is necessary and sufficient for a stable solution to the equations (*i.e.*, that the poles of (38) are inside the unit circle).

This is an interesting transformation from a theoretical point of view. It represents the entire class of rational mappings from the unit circle onto itself. I have yet to find any direct practical application for transformations above second order, but I am still looking.

Some reasons can be identified for the lack of interest in this transformation. In general, when we need multi-band filters, we need different features in each of the various bands, and not just the repetition of a single set of features. For instance, one band might need a different gain, or a different ripple than another band[8].

Example: a 3rd-Order Mapping

I will start with a prototype low-pass filter that is a fourth-order elliptic filter with a cutoff frequency (ß) of $p/2$. I will transform this into a filter with two pass bands and two stop bands. One pass band will extend from 0 to 0.3p and the second passband from $p/2$ to 0.6p. The transform polynomial, a_i, is found to be {1.0, -0.1583844, 0.8042261, -0.1583844). The argument of the function $g(e^{-jq})$ is shown in Figure 11. As q goes from 0 to p, the argument moves from 0 to 3p, showing that three complete revolutions around the unit circle are accomplished. The pole-zero plot of the prototype filter and the transformed filter are shown in Figure 12. The frequency responses are shown in Figure 13.

[8] The original version of this paper said that there was no easy way of determining the root positions of the resulting filter (after the substitution of (38) into the prototype filter). I no longer sure that this is the case. One would have to compute the inverse of (38), then map the roots using that inverse into its various values. (38) cannot be inverted directly, but it is a relatively simple numerical procedure to locate the root images using Newton's method. Since we know roughly where the roots will be, Newton's method can probably be made to converge to the correct solution in all but the most pathological cases (it is not *guaranteed* to converge). This seems to me to be simpler than factoring the resulting polynomials, since the order of the function (38) is *N*, which is surely less than the order of the filter polynomials.

I should point out again that in general, these transformations increase the Q of each of the poles. These transforms that increase the order are especially notorious for bringing the roots closer and closer to the unit circle. This makes sense. The more roots you try to cram into the same unit circle, the closer they will have to be to the unit circle.

Inverse Conformal Maps

One interesting way to view these transforms is to plot the inverse transform. Any transform above first order will be multi-valued, of course, so there is some question of how to plot these. I have plotted a number of 3rd and 4th-order transforms by displaying the images of the unit circle with radial, dotted lines. The prototype image has 16 radial lines, equally spaced around the circle. The dots on the radial lines are equally spaced. After the inverse transform, of course, they are no longer equally spaced. In fact, generally they are no longer straight lines. For instance, the inverse transform of the 3rd-order example above is shown in Figure 14. There are three images of the unit circle shown – one on the real axis, one above and one below. The map is symmetric around the real axis, since all the coefficients of the transform are real. Note how the upper and lower images have the radial lines all crammed together, emphasizing how the roots are all clumped up near the unit circle.

Figure 15 shows a progression of 3rd-order mappings with gently changing parameters. Figure 16 shows some 4th-order mappings. I find these quite pretty.

FIR Filters? Linear Phase?

All the filters shown in this paper are IIR. Can conformal mapping be applied to FIR filters? Well, yes they can, but after the transformation, they will no longer FIR. FIR filters have “implied” poles at zero frequency. These implicit poles become explicit after any conformal map. By “explicit,” I mean that they are no longer at zero frequency, thus the filter is no longer FIR. Even the simplest transform of (1) will cause the origin of the complex plane to be displaced left or right along the real axis, so all the “implicit” poles at zero become non-zero (and all equal). Higher-order transforms produce multiple images of the origin which are not necessarily on the real axis.

Having dismissed the idea of transforming FIR filters, we might look at why anyone would want to do that in the first place. The attraction of FIR filters is the linear phase property. Unfortunately, the linear phase property is not preserved under any but the most trivial conformal maps. This becomes evident as soon as we look at the mapping between frequencies in the prototype filter and frequencies in the resulting filter. In general, this correspondence is a curve and not a straight line. Linear phase requires that this correspondence be exactly linear. Only a trivial transform, such as the squaring in (36b), preserves linearity of the frequency axis.

Note that this means that any attempt to map an IIR approximation to a linear phase filter by conformal mapping will produce a filter that is no longer anything like linear phase. This means that analog filters with approximately linear phase, such as Lerner filters

(Lerner [17]) or Bessel filters (Storch [18]), have no particular reason to be used as prototypes for digital filters, since they will lose their distinguishing property. They can be digitized by impulse-invariance (Gold and Rader [6]) if the cutoff frequency is low enough so that no significant aliasing occurs. This will preserve the property of approximate phase-linearity.

Summary

I have shown an approach to the generation of coefficients for digital filters for professional audio applications (among others). Beginning with simple second-order presence and shelving filters that mimic the operations found commonly in mixing-desk equalizers, I showed how the notion of conformal mapping may be used to generalize the notion of shelving and presence filters of higher order by the use of elliptic functions.

All of the filters described here (except for the high-order notch that I just thought of) have been available in commercial digital audio systems for more than a decade at the time of this writing.

References:

[1] James A. Moorer "The Manifold Joys of Conformal Mapping: Applications to Digital Filtering in the Studio," Journal of the Audio Engineering Society, Volume 31, Number 11, November, 1983, pp826-841

[2] J.A. Moorer "General Spectral Transformations for Digital Filters," IEEE Transactions on Acoustics, Speech and Signal Processing, Volume ASSP-29, Number 5, pp1092-1094, October 1981

[3] A.G. Constantinides "Spectral Transformations for Digital Filters," Proceedings of the IEEE, Volume 117, Number 8, August 1970, pp1585-1590

[4] Zeev Nehari, "Confo rmal Mapping," McGraw-Hill, New York, 1952

[5] E.A. Guilliman, "Synthesis of Passive Networks," John Wiley & Sons, New York, 1957

[6] B. Gold and C. Rader "Digital Processing of Signals," McGraw-Hill, New York, 1969

[7] M.R. Schroeder, B.F. Logan, "Colo rless Artificial Reverberation," Journal of the Audio Engineering Society, Volume 9, Number 3, p192, July 1961

[8] Andrew G. Deczky "Synthesis of Recursive Digital Filters Using the Minimum *p*-Error Criterion," IEEE Transactions on Audio and Electroacoustics, Vol. AU-20, No. 4, October 1972, pp257-263

[9] G.W. McNally "Digital Audio: Recursive Digital Filtering for High-Quality Audio Signals," BBC Research Department Report 1981/10, 1981

[10] Sophocles J. Orphanidis "Digital Parametric Equalizer Design with Prescribed Nyquist-Frequency Gain," Presented at the 101st Convention of the Audio Engineering Society, November 8-11, 1996, Los Angeles, California, preprint 4361.

[11] A.H. Gray, Jr., and J.D. Markel "A Computer Program for Designing Digital Elliptic Filters," IEEE Transactions on Acoustics, Speech, and Signal Processing, Volume ASSP-24, Number 6, pp529-537, December 1976

[12] E.A. Neville, "Jacobian Elliptic Functions, 2nd Edition," Oxford University Press, London, England 1951

[13] Cayley "An Elementary Tretise on Elliptic Functions," Dover Publications, Inc., New York, 1956

[14] James E. Storer "Passive Network Synthesis," McGraw-Hill, New York, 1957

[15] M. Abramowitz and I.A. Stegun "Handbook of Mathematical Functions," Dover Publications, Inc., New York, 1965

[16] Jean-Claude Franchitti "All-Pass Filter Interpolation and Frequency Transformation Problems," Master of Science Thesis, Department of Electrical Engineering, University of Colorado, 1985

[17] R.M. Lerner "Band-Pass Filters with Linear Phase," Proceedings of the IEEE, Volume 52, pp249-268, 1964

[18] Leo Storch "Synthesis of Constant Time-Delay Ladder Networks Using Bessel Polynomials," Proceedings of the IRE, November, 1954, pp1666-1675

[19] G.L. Mattai "A General Method for Synthesis of Filter Transfer Functions," Electronics Research Lab., Technical Report 39, Stanford University, August 1952

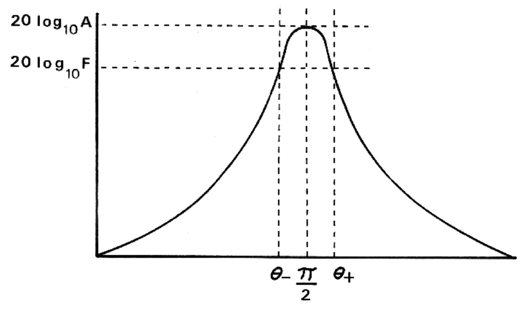
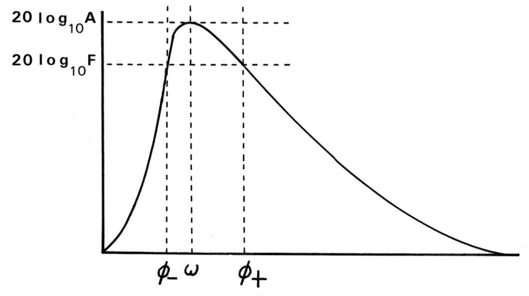
	
1a	1b

Figure 1 – Response of the digital second-order presence filter of (4). 1a shows such a filter centered at $p/2$, which makes the response perfectly symmetric. The bandedges q_- and q_+ are then related by $q_- = p - q_+$. 1b shows the presence filter after transformation by (1) to a new center frequency w. The response is no longer symmetric. The new bandedges f_- and f_+ are related to the old ones by (2a) with *rho* taking on the values q_- and q_+.

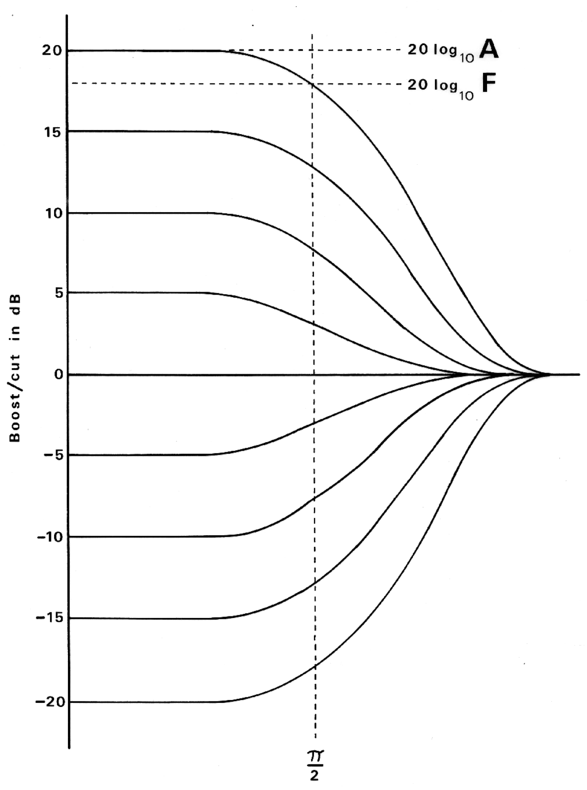


Figure 2 – Response of the digital second-order shelving filter of (13). The zero-frequency response, *A*, is varied in 5dB steps from +20 (boost) to -20 (cut). The bandedge value, *F*, is set to differ from *A* by 3 dB at the frequency $p/2$, except when the boost/cut amount falls below 6 dB, in which case it is set to half the boost/cut amount (in dB).

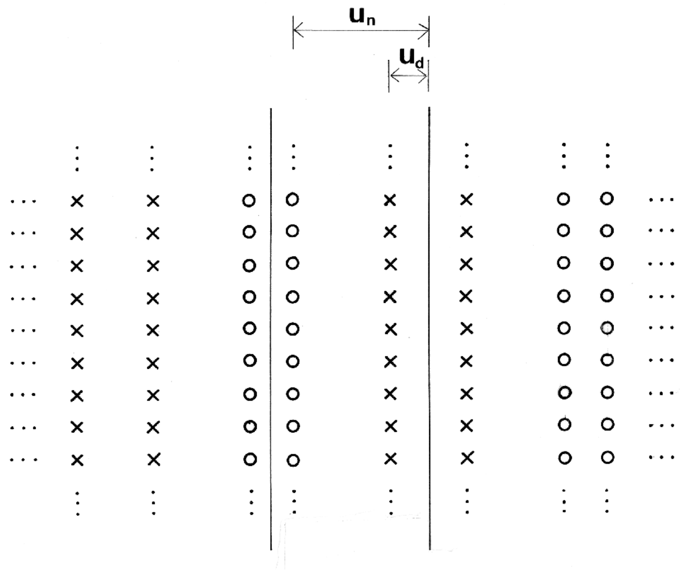


Figure 3 – Regular infinite 2-dimensional array of poles and zeros. The two vertical lines between pole columns and between zero columns are axes of symmetry. Every row is also an axis of symmetry, as well as a line between any two rows. The response along any axis of symmetry is equiripple by inspection.

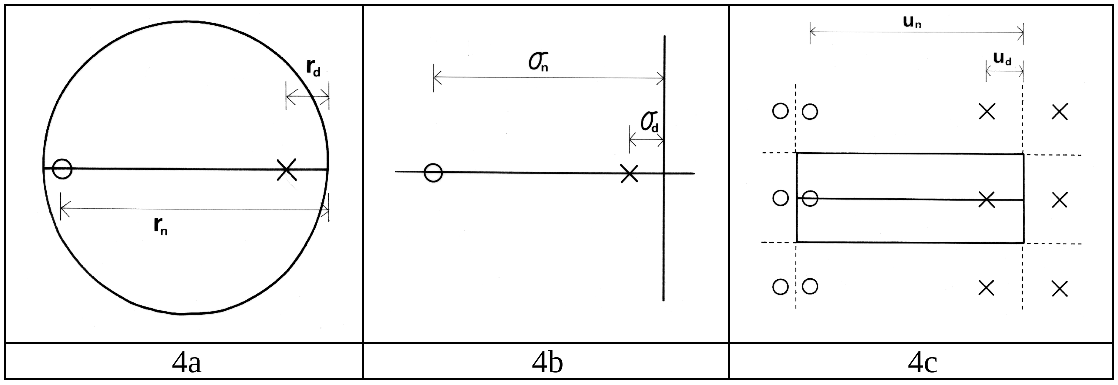


Figure 4 – Mapping of a single pole and zero from the *z*-plane in 4a to the *s*-plane in 4b to the *u*-plane in 4c. The mapping from the unit circle to the *s*-plane is accomplished by a bilinear transform. The mapping of the imaginary axis to a rectangle in the *u*-plane is accomplished by the Jacobi elliptic function $sn(u \mid k)$.

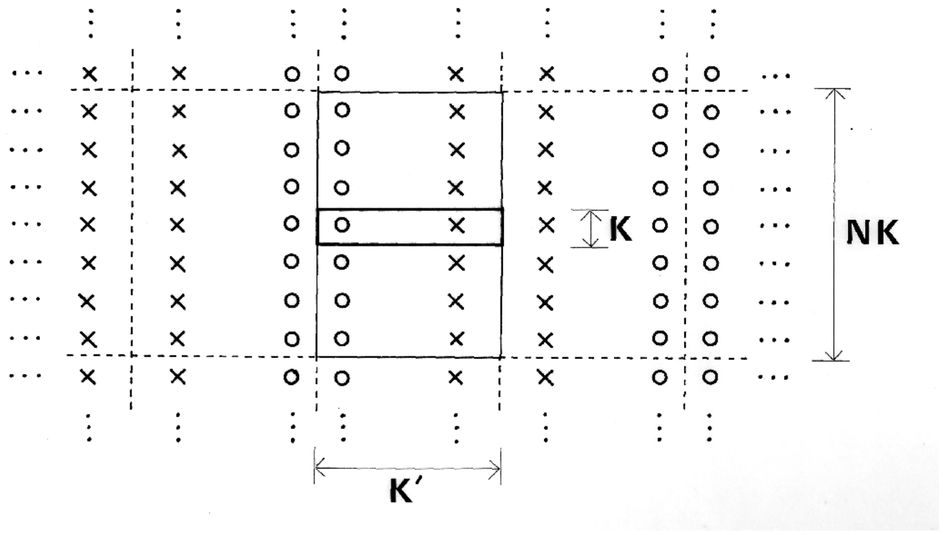


Figure 5 – The regular array of poles and zeros of Figure 3 with the prototype first-order pattern of Figure 4c shown in the rectangle in heavy lines. The height and width of this rectangle will be *K(k)* and *K'(k)*. To reduce the transition width, we choose a new rectangle *N* times higher to enclose *N* zeros and poles. This will have the same value of ripple since the geometry of the pole-zero array is unchanged. Since we are performing the mapping on axes of symmetry, the reflection principle is respected. The transition width is reduced as we enclose more and more roots. This is because the transition region is the top and bottom of the rectangle. The top and bottom become proportionately smaller in relation to the perimeter of the rectangle as the height of the rectangle is increased to enclose more roots.

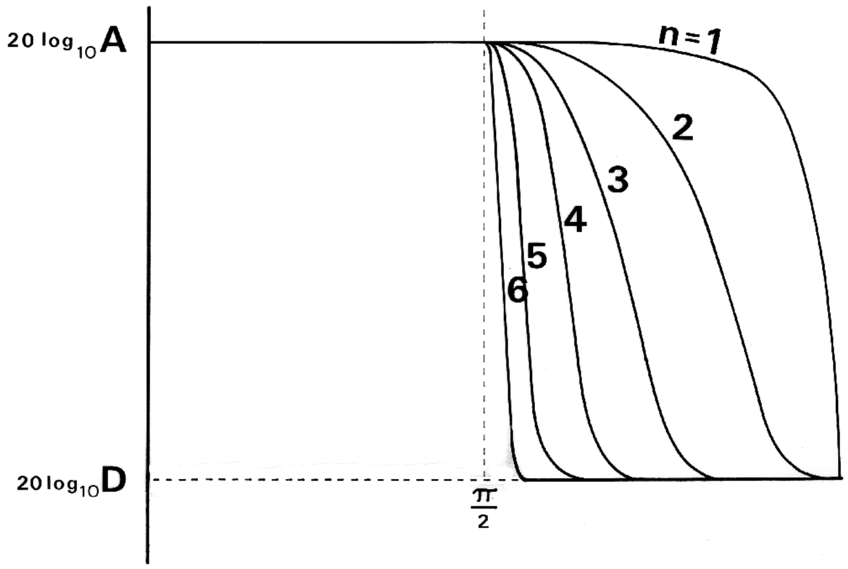


Figure 6 – The elliptic shelving filter for varying orders. The low and high bands have a ripple of 0.1 dB, which is so restrictive that the first order filter has a transition band that covers nearly the entire frequency axis. As the order is increased, the transition band is progressively narrowed.

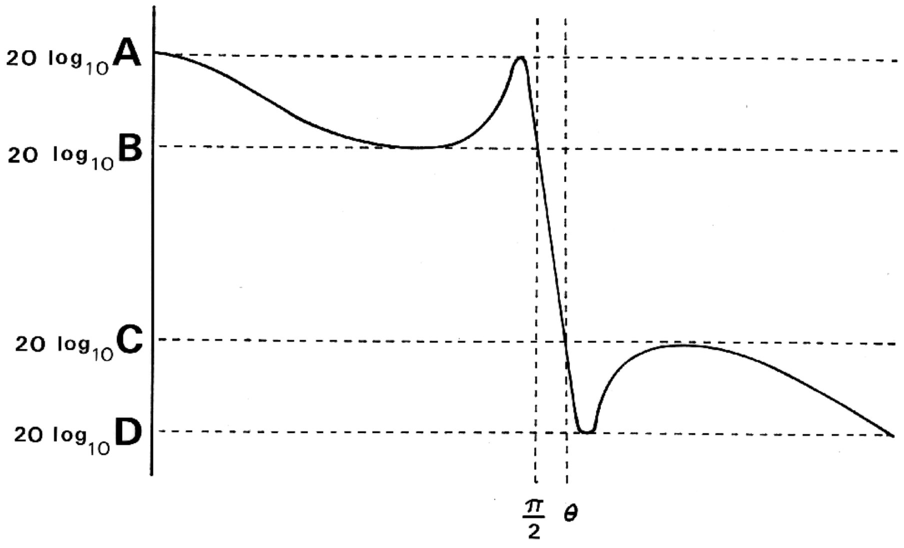


Figure 7 – A 3rd-order elliptic shelving filter with the vertical greatly exaggerated to show the nature of the ripple. We see that the bounds (*A* and *B* for the low-frequency band, *C* and *D* for the high-frequency band) are touched exactly *N+1* times. q represents the high edge of the transition band. The amount of ripple can be set to any desired value, with the consequence that as the ripple is reduced (as *N*, the filter order is kept constant), the transition width will increase.

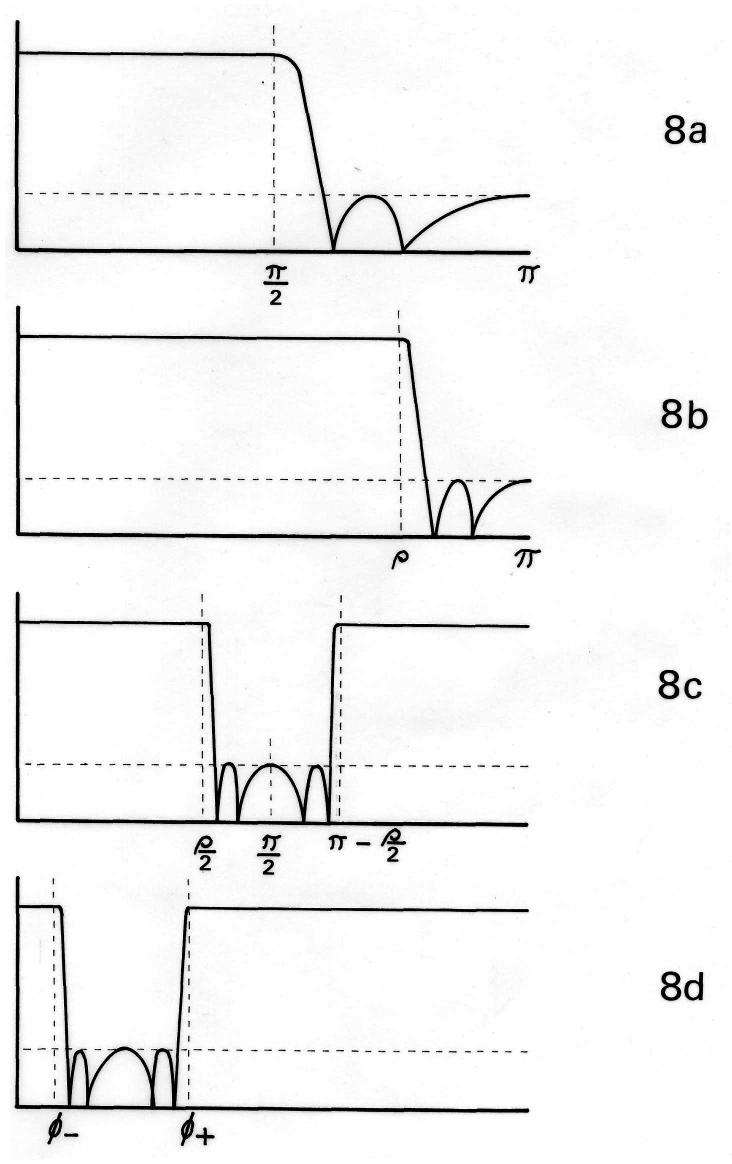


Figure 8 – Second-order mapping of an elliptic low-pass filter in Figure 8a by the transformations of (36). In Figure 8b, the bandedge has been moved by (36a) to the new cutoff frequency r . The substitution (36b) halves all the frequencies and produces a new bandedge, as is shown in Figure 8c. The response is now symmetric about $p/2$. The final substitution (36c) moves the center frequency from $p/2$ to any desired value w, producing new bandedges f_- and f_+, as shown in Figure 8d. Note that filters produced by using the substitution (36b) will always have *even* order.

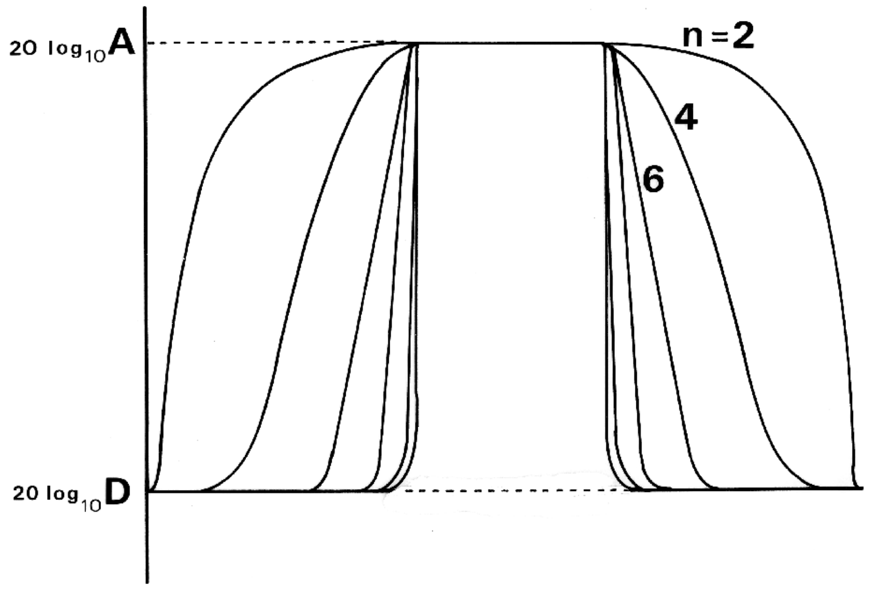


Figure 9 – The filter shown in Figure 6 after being transformed by (36) to a digital elliptic presence filter. The substitution of (36b) doubles the order of the filter. Again, the 0.1 dB value of ripple forces a very wide transition band at the lowest order (*n=2*). The transition band is reduced with increasing filter order.

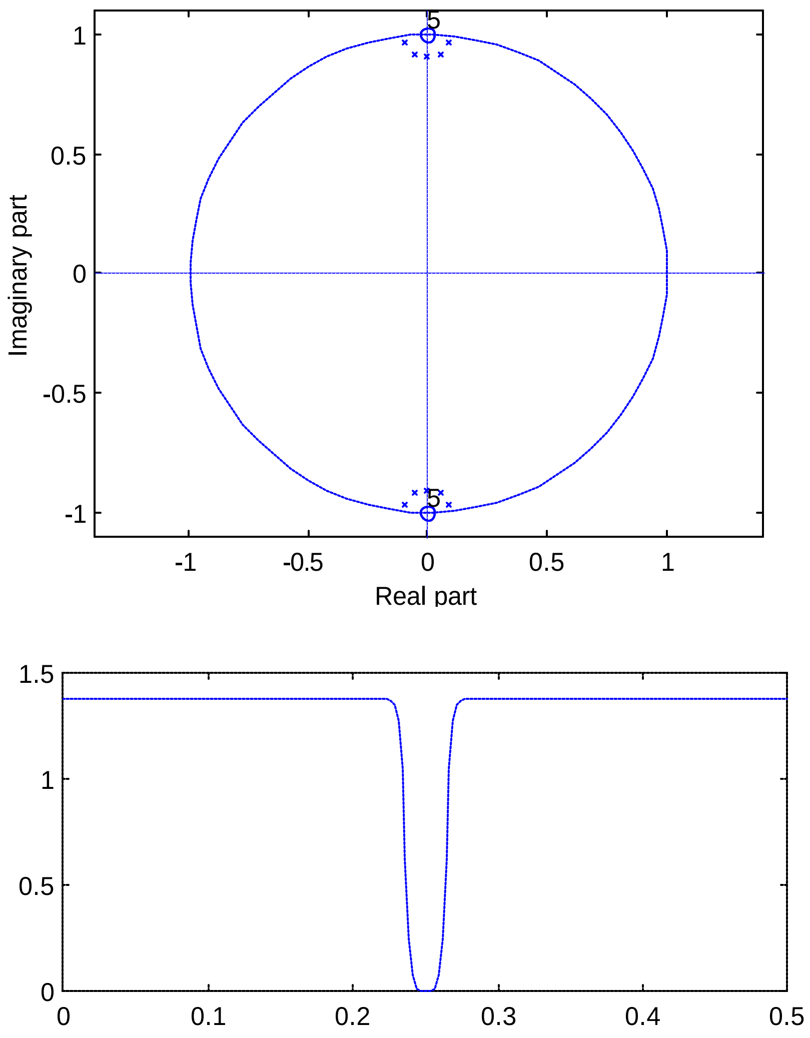


Figure 10: Pole-zero plot and magnitude plot of 10th-order notch filter formed by making prototype Butterworth low-pass filter, then using the transformation of (36b) to bring the stopband to frequency $p/2$. The transform of (1) may then be used to move the notch frequency to any desired value. The width of the notch has been exaggerated. A practical notch would be much more narrow.

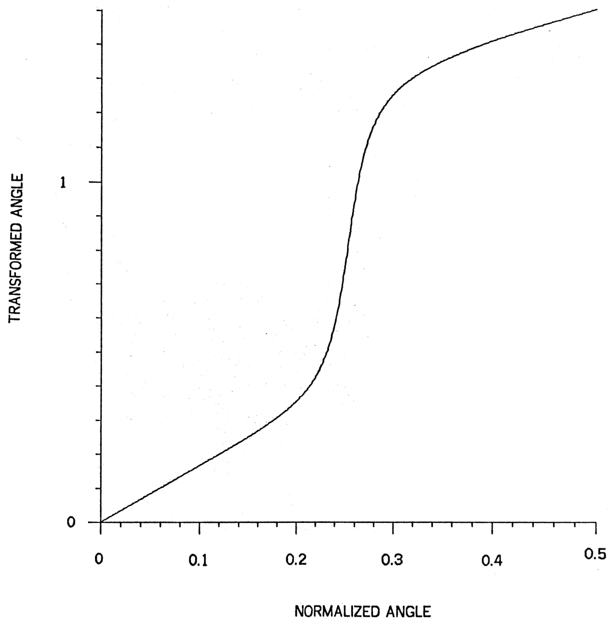


Figure 11:

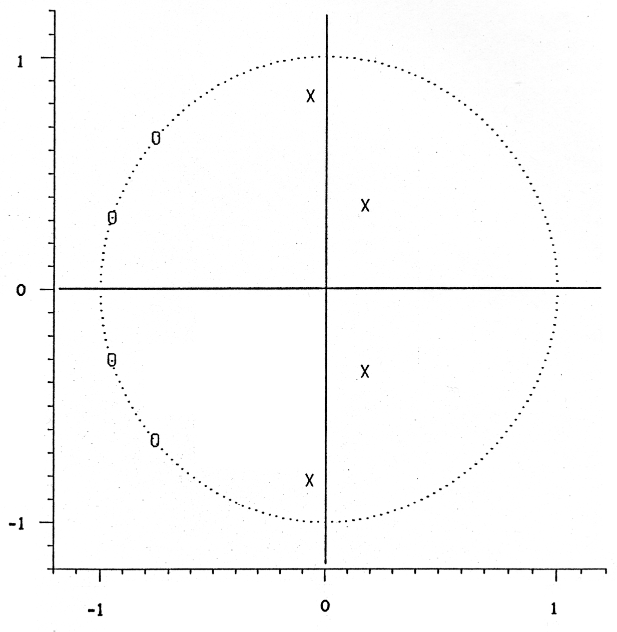
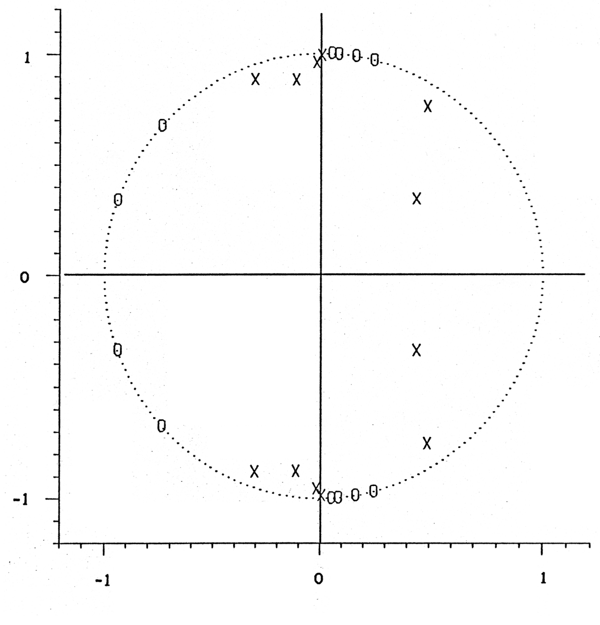
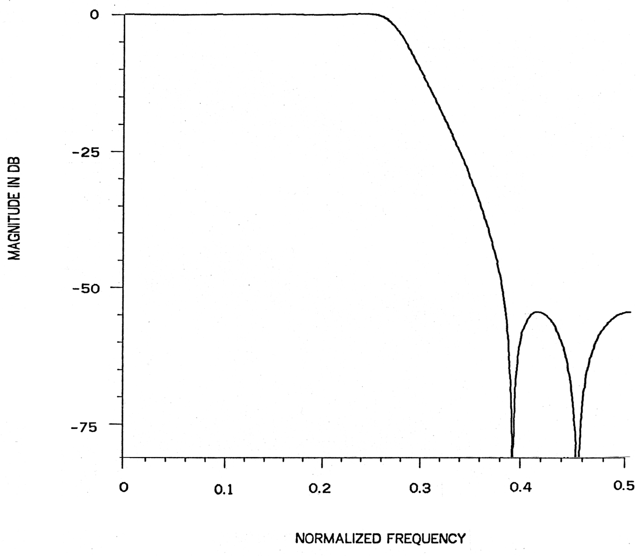
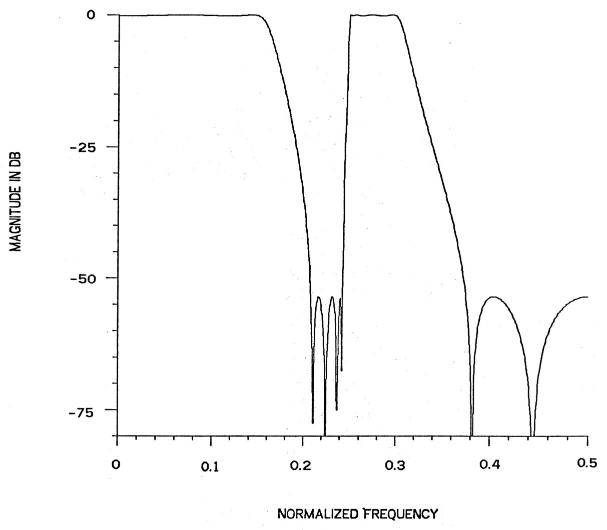
	12a
	12b

Figure 12:



13a



13b

Figure 13:

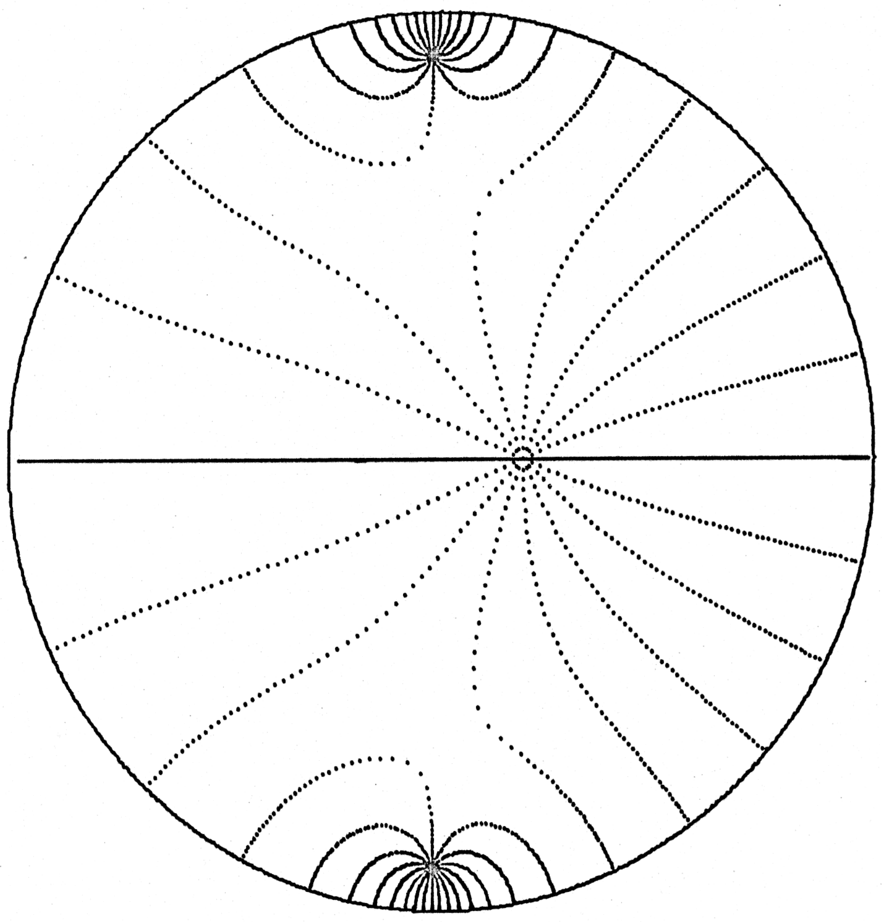


Figure 14: 3rd-order inverse conformal map from the example in the text. The prototype unit circle has 16 dotted radial lines. The lines were at equal angles around the circle, and the dots were uniformly spaced along the radial lines. After the transformation, there are three images of the circle. The origin of one is along the real line, just to the right of center. There are two other images above and below the real line. Note the bunching of the lines in the upper and lower images, showing the increase in the *Q* of the roots that can happen under conformal filter transformations.

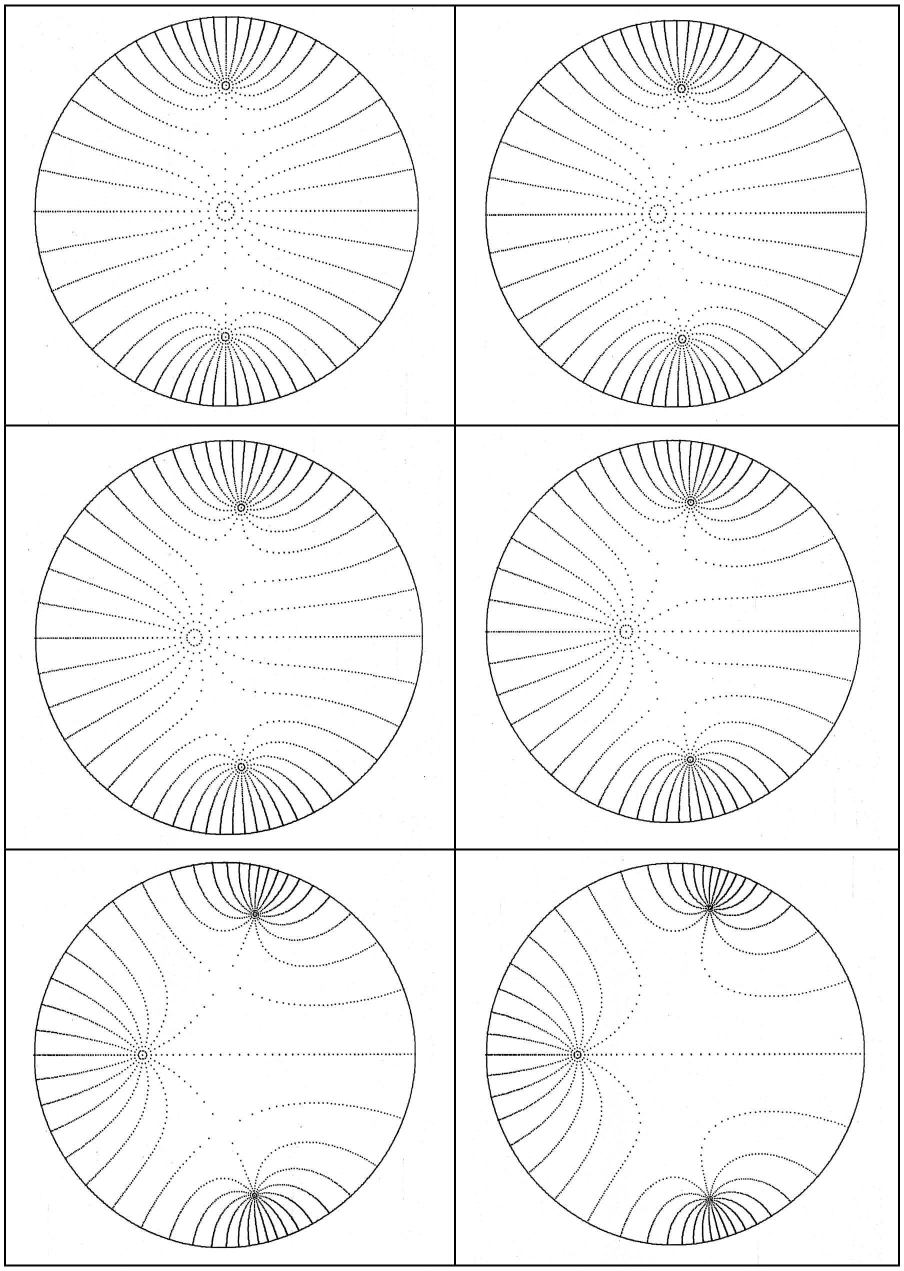


Figure 15: A sequence of 3rd-order inverse conformal maps. This represents a multi-band filter as described in the text as an example. The location of the critical frequencies is being lowered from one picture to the next.

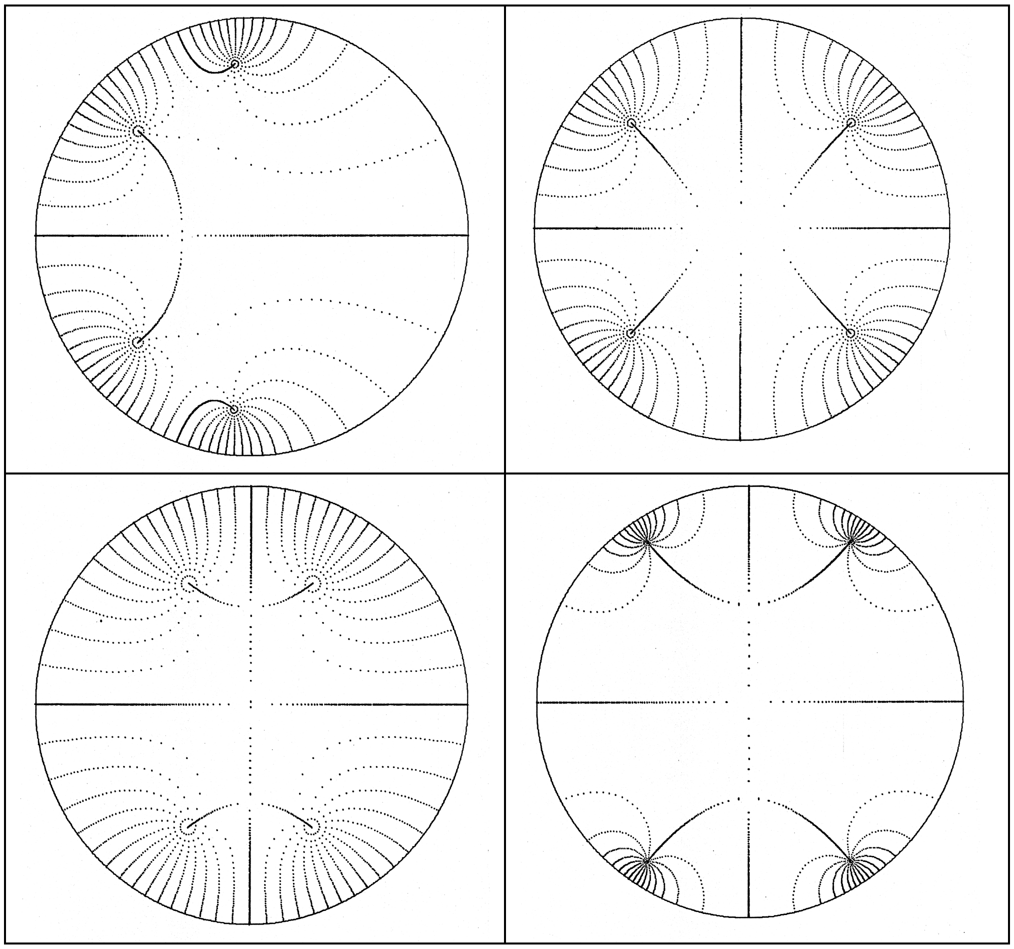


Figure 16: 4th-order inverse conformal maps.

Appendix I:

Here is a C program to compute the coefficients of second-order presence and shelving filters as described in the text:

```
#include <stdio.h>
#include <math.h>

#define PI          3.141592653589793238462643
#define SQRT2       1.4142135623731
#define SPN         1.65436e-24   /* Semi-arbitrary Small Positive Number */
                                  /* Used to test for pathological limiting */
                                  /* cases. */

double flx_flatmp;
#define flabs(arg) (((flx_flatmp = (arg)) >= 0.0) ? flx_flatmp : -flx_flatmp)

/* ----------------------------------------------------------------------
   db2lin - Convert between db and linear
   = 10^(x/20)
---------------------------------------------------------------------- */
double db2lin(double x)
{
      return exp(2.30258509299404568401 * (x * 0.05));
}

/* ------------------------------------------------------------------------
   bw2Angle - Given bilinear transform parameter (from above routine) and
desired bandwidth (as normalized frequency), computes bandedge, e, of filter
as if it were centered at the frequency .25 (or PI/2, or srate/4). The
bandwidth would then 2*(.25-e). e is guaranteed to be between 0 and .25.

To state it differently, given a filter centered on .25 with low bandedge
e and high bandedge .5-e, the bilinear transform by a
will produce a new filter with bandwidth (i.e., difference between the
high bandedge frequency and low bandedge frequency) of bw.
------------------------------------------------------------------------ */
double bw2Angle(double a, double bw)
{   double snT, csT, d, sn, cs, mag, delta, theta, tmp, a2, a4, asnd;

    snT = sin(2.0 * PI*bw);
    csT = cos(2.0 * PI*bw);
    a2 = a*a;
    a4 = a2*a2;
    d = 2.0*a2*snT;
    sn = (1.0+a4)*snT;
    cs = (1.0-a4)*csT;
    mag = sqrt(sn*sn + cs*cs);
    d /= mag;
    delta = atan2(sn, cs);

    asnd = asin(d);
    theta = 0.5 * (PI - asnd - delta);        /* Bandedge for prototype */
    tmp = 0.5 * (asnd - delta);               /* Take principal branch */
    if (tmp > 0.0 && tmp < theta) theta = tmp;
    return (theta/(2.0 * PI));         /* Return normalized frequency */
}
```

```
/* ----------------------------------------------------------------------
   presence - Design straightforward 2nd-order presence filter. Standard
storage method not used - just returns coefficients directly.
Must be given (normalized) center frequency, bandwidth, and boost/cut in dB.

Returns filter of form:

                -1       -2
       A0 + A1 Z   + A2 Z
T(z) = ----------------------
               -1       -2
       1 + B1 Z   + B2 Z

--------------------------------------------------------------------- */
void presence(double cf, doublebw, doubleboost,
             double *a0, double *a1, double *a2, double *b1, double *b2)
{   double a, A, F, xfmbw, C, tmp, alphan, alphad, b0, recipb0, asq,
       F2, oneplusa2, oneminusa2;

    a = tan( PI*(cf - 0.25) );          /* Warp factor */
    asq = a*a;
    A = db2lin(boost);
   /*  A = pow(10.0, boost/20.0); */           /* Cvt dB to factor */
    if (boost < 6.0 && boost > -6.0) F = sqrt(A);
    else if (A > 1.0) F = A/SQRT2;             /* F = A/sqrt(2.0); */
    else F = A * SQRT2;                        /* F = A * sqrt(2.0); */
       /* If |boost/cut| < 6dB, then doesn't make sense to use 3dB point.
          Use of root makes bandedge at half the boost/cut amount.
       */
    xfmbw = bw2Angle(a, bw);

    C = 1.0 / tan(2.0 * PI * xfmbw);    /* co-tangent of angle */
    F2 = F*F;
    tmp = A*A - F2;
       /* Limit case – compare with semi-arbitrary small number */
       /* Substitute small-value approximation if necessary to */
       /* prevent possible divide overflow */
    if (flabs(tmp) <= SPN) alphad = C;
    else alphad = sqrt( C*C * (F2 - 1.0) / tmp );
    alphan = A*alphad;

    oneplusa2 = 1.0 + asq;
    oneminusa2 = 1.0 - asq;
    *a0 = oneplusa2 + alphan*oneminusa2;
    *a1 = 4.0*a;
    *a2 = oneplusa2 - alphan*oneminusa2;

    b0 = oneplusa2 + alphad*oneminusa2;
    *b2 = oneplusa2 - alphad*oneminusa2;

    recipb0 = 1.0/b0;
    *a0 *= recipb0;
    *a1 *= recipb0;
    *a2 *= recipb0;
    *b1 = *a1;
    *b2 *= recipb0;
}
```

Appendix II:

Here is a C program to compute the coefficients of second-order shelving filter as described in the text:

```
/* ----------------------------------------------------------------------------
   shelve - Design straightforward 2nd-order shelving filter. Standard
storage method not used - just returns coefficients directly.
Must be given (normalized) center frequency, and boost/cut in dB.
This is a LOW shelving filter, in that the response at z = -1 will
be 1.0.

Returns filter of form:

                -1       -2
       A0 + A1 Z   + A2 Z
T(z) = ----------------------
               -1       -2
       1 + B1 Z   + B2 Z

--------------------------------------------------------------------------- */
void shelve(double cf, double boost,
             double *a0, double *a1, double *a2, double *b1, double *b2)
{   double a, A, F, tmp, b0, recipb0, asq,
       F2, gamma2, twosigmagamma, oneplusgamma2;
    double gamman, gammad, ta0, ta1, ta2, tb0, tb1, tb2, aa1, ab1;

    a = tan( PI*(cf - 0.25) );          /* Warp factor */
    asq = a*a;
    A = db2lin(boost);
   /*  A = pow(10.0, boost/20.0); */           /* Cvt dB to factor */
    if (boost < 6.0 && boost > -6.0) F = sqrt(A);
    else if (A > 1.0) F = A/SQRT2;      /* F = A/sqrt(2.0); */
    else F = A * SQRT2;                 /* F = A * sqrt(2.0); */
       /* If |boost/cut| < 6dB, then doesn't make sense to use 3dB point.
          Use of root makes bandedge at half the boost/cut amount.
       */

    F2 = F*F;
    tmp = A*A - F2;
       /* Limit case – compare with semi-arbitrary small number */
       /* Substitute small-value approximation if necessary to */
       /* prevent possible divide overflow */
    if (flabs(tmp) <= SPN) gammad = 1.0;
    else gammad = pow( (F2 - 1.0)/tmp, 0.25);  /* Fourth root */
    gamman = sqrt(A)*gammad;

/* Once for the numerator */

    gamma2 = gamman*gamman;
    oneplusgamma2 = 1.0 + gamma2;
    twosigmagamma = SQRT2*gamman;       /* 2*ROOT2OVER2*gamman; */

    ta0 = oneplusgamma2 + twosigmagamma;
    ta1 = -2.0*(1.0 - gamma2);
    ta2 = oneplusgamma2 - twosigmagamma;
/* And again for the denominator */

    gamma2 = gammad*gammad;
    oneplusgamma2 = 1.0 + gamma2;
    twosigmagamma = SQRT2*gammad;       /* 2*ROOT2OVER2*gammad; */

    tb0 = oneplusgamma2 + twosigmagamma;
    tb1 = -2.0*(1.0 - gamma2);
    tb2 = oneplusgamma2 - twosigmagamma;

/* Now bilinear transform to proper center frequency */

    aa1 = a*ta1;
    *a0 = ta0 + aa1 + asq*ta2;
    *a1 = 2.0*a*(ta0+ta2)+(1.0+asq)*ta1;
    *a2 = asq*ta0 + aa1 + ta2;

    ab1 = a*tb1;
    b0 = tb0 + ab1 + asq*tb2;
    *b1 = 2.0*a*(tb0+tb2)+(1.0+asq)*tb1;
    *b2 = asq*tb0 + ab1 + tb2;

/* Normalize b0 to 1.0 for realizability */

    recipb0 = 1.0/b0;
    *a0 *= recipb0;
    *a1 *= recipb0;
    *a2 *= recipb0;
    *b1 *= recipb0;
    *b2 *= recipb0;
}
```