Designing Second-order IIR Filters for Cascade-Implementation Digital Audio Applications

RUSTY ALLRED

Texas Instruments, Incorporated, Dallas, Texas USA

It is possible to implement IIR filters of any order by cascading first- and second-order IIR sections. Today, as Digital Audio Processors that offer configurable cascades of second-order sections are becoming commonplace in the market, the designer's challenge is to find a way to design coefficients for these to accomplish the overall filtering goal. This paper brings together the basic information needed to design and implement a wide variety of audio filters, presents it in a manner accessible to the system designer, and includes numerous examples as well as Matlab® implementations.

0 INTRODUCTION

Until recently, audio systems have largely been analog. Now the trend is moving rapidly to digital implementations. While the literature contains most of the information needed to design and implement digital filters, it might not be easily found in the most useful forms. This paper gives the system designer a concise tutorial containing enough information to quickly get started with the design and implementation of digital filters.

Digital filters are much more flexible than analog filters since they do not depend upon the availability of special components to implement the desired mathematical function. Rather, digital second-order filters have 5 degrees of freedom that can be exploited to accomplish the desired filtering task. Unfortunately, it is not always clear how to find the coefficients to fully exploit this flexibility. As a partial answer to that dilemma, this paper surveys many of the common techniques and gives practical implementation tips.

The coefficients designed in this paper are those for the so-called Direct Form digital filters.

Although focusing on second-order filters, this paper also discusses some first-order filters as well, since some applications require them. Furthermore, there is no problem implementing first-order filters even in systems that offer only coefficient-configurable second-order sections. As this paper explains, the first-order filter can be implemented individually in the second-order section, or combined with another first-order filter to form a second-order filter.

It is possible to design filters in the analog domain and convert them for use in digital systems, and these techniques are also discussed in this paper. But it is also possible to design many filters directly in the digital domain. Usually a direct approach, if available, will be favored since it offers the designer more direct control of the filter transfer function and, often, ease of design.

1 NOMENCLATURE

For purposes of this paper the following nomenclature is observed. The transfer function of the second-order IIR filter is as shown in Eq. (1).

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \qquad (1)$$

For simplification, this is also expressed as

$$H(z) = \frac{B(z)}{A(z)} = \frac{B}{A} = \frac{[b_0 \;\; b_1 \;\; b_2]}{[1 \;\; a_1 \;\; a_2]} \qquad (2)$$

Note that the sample-domain equation for Eq. (1) is as follows:

$$\begin{aligned} y(n) &= b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) \\ &\quad - a_1 y(n-1) - a_2 y(n-2) \end{aligned} \qquad (3)$$

where the relationship back to Eq. (1) is understood by converting Eq. (3) back to z-transform notation by observing the following rules:

1. Lower case x's and y's become upper case.
2. n's are substituted with z's.
3. Delays, n-Δ, become $z^{-\Delta}$.
4. $H(z) = \dfrac{Y(z)}{X(z)}$ (4)

Making these substitutions, Eq. (3) becomes:

$$\begin{aligned} Y(z) &= b_0 X(z) + b_1 X(z) z^{-1} + b_2 X(z) z^{-2} \\ &\quad - a_1 Y(z) z^{-1} - a_2 Y(z) z^{-2} \\ Y(z)[1 &+ a_1 Y(z) z^{-1} + a_2 Y(z) z^{-2}] = \\ &\quad X(z)[b_0 + b_1 z^{-1} + b_2 z^{-2}] \end{aligned} \qquad (5)$$

Which becomes Eq. (1) upon application of Eq. (4) and some additional manipulation.

A common audio implementation of this filter is shown in Fig. 1 [1].

Notice the negative signs in the figure. These come directly out of Eq. (3). Obviously these can be absorbed into the *a* coefficients, leading to another standard version of Eq. (1), this time with the + signs in the denominator substituted with - signs. For purposes of this paper, the following convention is used: when the coefficients are directly applicable to Eq. (1), they will be written as shown in Eq. (2). When the sign has been absorbed into the a coefficients, the following nomenclature is used:

$$H'(z) = \frac{B(z)}{A'(z)} = \frac{B}{A'} = \frac{[b_0 \;\; b_1 \;\; b_2]}{[1 \;\; -a_1 \;\; -a_2]} \qquad (6)$$

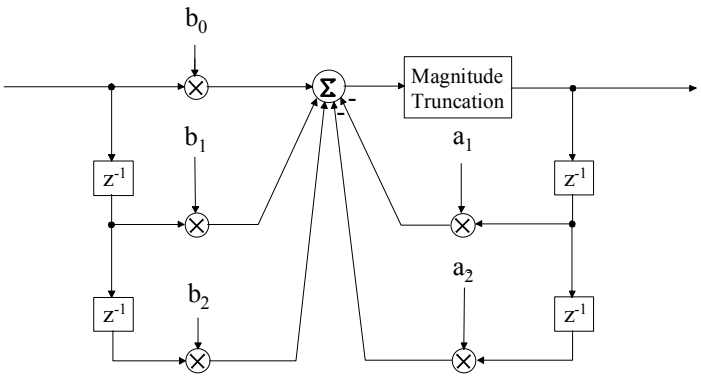

Fig. 1 Direct Form I second-order IIR Filter

2 IMPLEMENTATION TIPS

2.1 Stability

Obviously, stability is a key issue in filter design, and the criterion for digital filters is simple: the filter is stable if its poles lie within the so-called unit circle. Practically speaking it is sufficient to test the coefficients (at implementation precision) to assure that the roots of the denominator polynomial have magnitudes lower than 1. For example, consider the denominator polynomial

A = [1.0 -1.96297931671143 0.96365261077881]

or

$$A(z) = 1.0 - 1.96297931671143 z^{-1} + 0.96365261077881 z^{-2}$$

Its roots, or the poles of the filter, are

0.98148965835571 ± 0.018184095237 18i

which have magnitude 0.98165809260598 < 1, so this filter is stable.

If the filter implementation in use uses A' instead of A, that is, if the signs of the a_1 and a_2 coefficients are reversed such that

A' = [1.0 1.96297931671143 -.96365261077881]

the signs must be re-reversed prior to finding the roots. Due to the implementation, A' results in

the same stable filter as A, but would not be a stable filter, with poles of

-2.36964475064116
0.40666543392974

except in those implementations where the numerator coefficient sign reversal is absorbed in the coefficients, as discussed in Section 2.

2.2 Coefficient Quantization

For fixed point implementations, the filter coefficients must be quantized to the appropriate word size. For most filters these can be either truncated or rounded. However, in the author's experience, magnitude truncation, or truncation toward 0, is the safest method in cases where the available precision is only marginally sufficient. This has not been the object of intense study, but, as a practical matter, the author uses magnitude truncation on coefficients, and tries other options if the transfer function deviates significantly from the desired result.

A fixed-point digital filter coefficient will have some number of integer bits, *i*, and some number of fraction bits, *f*. It is common to call this an *i.f* format, where the point indicates the binary point. For example, a 4.20 coefficient has 4 integer bits and 20 fraction bits, and a 5.23 coefficient has 5 integer bits and 23 fraction bits.

Coefficient quantization can be accomplished in the following way,

$$C_Q = 2^{-f}\,\mathrm{sgn}(C)\left\lfloor \left|2^f C\right| \right\rfloor \qquad (7)$$

which is a fancy way of denoting an up shift of the coefficient, C, by the number of fraction bits, f, truncating toward 0, and then a down shift again by f to produce the quantized coefficient C_Q.

Notice that, since the number of bits available to represent the coefficient is limited, it is also necessary to saturate the coefficient to the maximum and minimum values.

Since coefficients must, in general, take on negative or positive values, the 4.20 or 5.23 coefficients of the example above will be two's complement numbers. Therefore, the maximum positive number will be 2^{i-1}-2^{-f}, while the maximum negative coefficient will be -2^{i-1}. This saturation is accomplished in the following manner:

$$C_{QS} = \max\left(-2^{i-1}, \min\left(2^{i-1} - 2^{-f}, C_Q\right)\right) \qquad (8)$$

For example, Table 1 shows some quantized and saturated 4.20 coefficients, and Table 2 shows the same coefficients for the 5.23 case:

Full Precision	**4.20**
25	7.99999904632568
9	7.99999904632568
5.3	5.29999923706055
2^{-20}	$9.5367431640625 \times 10^{-7}$
2^{-23}	0
-3.98	-3.97999954223633
-9	-8
-25	-8

Table 1. 4.20 Quantization

Full Precision	**5.23**
25	15.99999988079071
9	9
5.3	5.29999995231628
2^{-20}	$9.5367431640625 \times 10^{-7}$
2^{-23}	$1.192092895507813 \times 10^{-7}$
-3.98	-3.9799998998642
-9	-9
-25	-16

Table 2. 5.23 Quantization

2.3 Conversion to Hexadecimal

While conversion of decimal numbers to hexadecimal is commonplace, and there exist tools for accomplishing this task, converting *i.f*, twos-complement numbers is a slightly different wrinkle, that sometimes causes confusion.

The easiest way to accomplish this, and the way that works for the standard filtering devices in the marketplace, is as shown below:

$$C_H = \begin{cases} \text{dec2hex}\left(2^f C_Q + 2^{i+f}\right) & C_Q < 0 \\ \text{dec2hex}\left(2^f C_Q\right) & \text{Otherwise} \end{cases} \quad (9)$$

This equation makes each coefficient the hexadecimal equivalent non-negative integer that is required for most standard dec2hex commands. In the absence of a standard dec2hex command, the non-negative integer arising from Eq. (8) can be converted to hexadecimal by first converting to binary, and then grouping the bits into groups of 4 and making a hexadecimal conversion.

By way of simple example, consider the value -1.25 to be represented in 2.2 hexadecimal. Quantizing according to Eq. (7) does not change the value. Then, applying Eq. (9) results in the following:

$$-1.25 \cdot 2^2 + 2^4 = 11$$

The binary representation of 11 is 1011, which results in the hex code B.

Had this been 1.25, Eq. (9) returns $1.25 \cdot 2^2 = 5$, which is 0101 in binary, and, of course, simply 5 in hexadecimal.

For more realistic examples, consider -3.97999954223633 from Table 1, and 5.29999995231628 from Table 2 above. Using Eq. (8) for the former, a 4.20 value

$$-3.97999954223633 \cdot 2^{20} + 2^{24} = 1.2603884 \times 10^7$$

= 11000000010100011101100 in binary and

C051EC in hexadecimal.

Likewise, 5.29999995231628 for the 5.23 case produces
00101010011001100110011001100110 in binary, and

2A66666 in hexadecimal.

3 FILTER DESIGN

The previous sections discussed various points regarding the implementation of filters. Now, how are those filters designed to begin with? That question is addressed in this section.

There are numerous possible methods to design digital filters. For example, analog filters can be converted to digital using such methods as the bilinear transform or the impulse invariant method [2], [3], [4], [5]. While these techniques are useful and do help to make the world of analog filters readily applicable to digital filters, many digital filters can also be designed directly. In fact, there is more flexibility in this approach. In this section, some of the most common filters for audio applications are discussed.

3.1 Allpass Filters

Allpass filters can be designed directly in the digital domain. Besides their use in phase

compensation, they are also building blocks from which other filters can be derived [6], [7], [8], [9], [10], [11], [12].

3.1.1 First Order

A first-order allpass is designed as follows:

$$\alpha = \frac{\tan\left(\frac{\pi f_c}{F_s}\right) - 1}{\tan\left(\frac{\pi f_c}{F_s}\right) + 1} \tag{10}$$

where f_c is the critical frequency, F_s is the sample rate and α is one of the coefficients of the final filter:

$$b_0 = \alpha$$
$$b_1 = 1$$
$$a_2 = \alpha$$
$$\mathrm{B} = [b_0\ b_1\ 0]$$
$$\mathrm{A} = [1\ a_2\ 0] \tag{11}$$

The a_0 coefficient is always 1, by definition. Since this is a first-order filter, the b_2 and a_2 coefficients are zero if implementing in a second-order section. Later in this paper the method of combining two first-order filters for implementation in a single second-order section is discussed.

Figs. 2 and 3 show respectively the phase responses and group delays for 44.1 kHs sample rate implementations of four first-order allpass filters with critical frequencies of 0.1, 0.4, 1.6, and 6.4 kHz.

Appendix A gives Matlab® code for the implementation of this filter, and the other discussed within this paper. To make this code generically useful as pseudo code, for any reader not familiar with Matlab®, only general mathematical commands, which can be found in most any math library or computation tool, have been used rather than relying upon Matlab's extensive libraries of filtering tools.

Note that the response curves in this paper are for non-quantized filter coefficients. For some filters the responses might deviate from those shown due to coefficient quantization effects. In particular, for higher sample rates, lower cut frequencies, and for lower coefficients this will be more pronounced. For a thorough discussion of quantizaton effects and response distortions, please see [13] and [14].

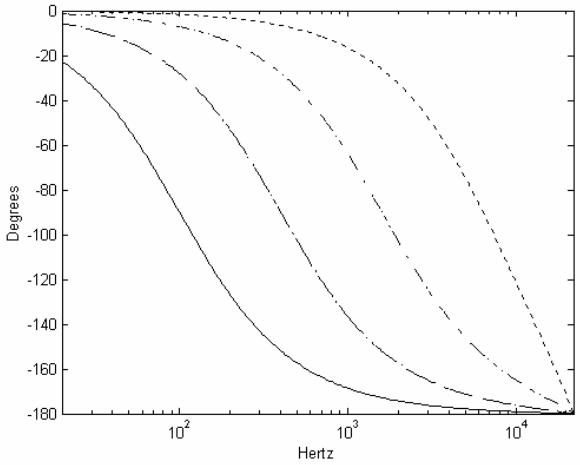

Fig. 2 Phase Responses for First-order Allpass Filters

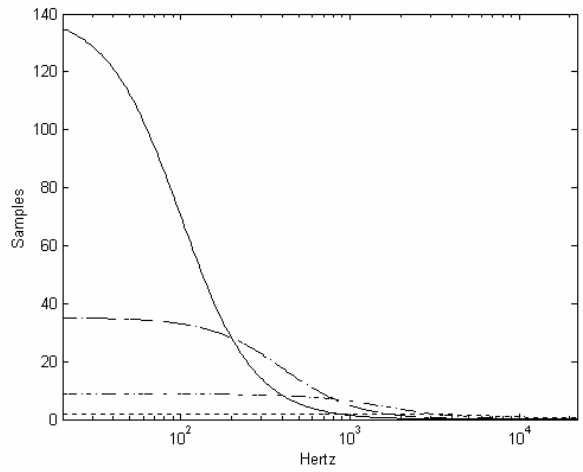

Fig. 3 Group Delays for First-order Allpass Filters

Table 3 shows the coefficients for the first of these four filters first in unquantized decimal, then in 4.20 hexadecimal and in 5.23 hexadecimal with reversed-sign denominator coefficients.

f_c = 400 Hz	*Fs* = 44.1 kHz
B Decimal	-0.94457402736173 1.0 0.0
A Decimal	(1.0) -0.94457402736173 0.0
B 4.20 Hex	F0E307 100000 000000
A 4.20 Hex	F0E307 000000
B 5.23 Hex	F871833 0800000 0000000
A' 5.23 Hex	078E7CD 0000000

Table 3. Coefficients for First-order Allpass Filters

3.1.2 Second Order

A second-order allpass filter is designed as follows:

$$\alpha = \frac{\tan\left(\frac{\pi b}{F_s}\right) - 1}{\tan\left(\frac{\pi b}{F_s}\right) + 1} \tag{12}$$

where b is the desired bandwidth.

$$\beta = -\cos\left(\frac{2\pi f_c}{F_s}\right) \tag{13}$$

$$b_0 = -\alpha$$
$$b_1 = \beta(1-\alpha)$$
$$b_2 = 1$$
$$a_1 = \beta(1-\alpha)$$
$$a_2 = -\alpha$$

$$B = [b_0\ b_1\ b_2]$$
$$A = [1\ a_1\ a_2] \tag{14}$$

Figs. 4 and 5 respectively show phase responses and group delays for 44.1 kHz sample rate implementations of second order allpass filters with the following parameters:

f_c	*b*
100 Hz	200 Hz
400 Hz	200 Hz
1600 Hz	400 Hz
6400 Hz	800 Hz

Table 4. Parameters used for Second-order Allpass Filter Examples

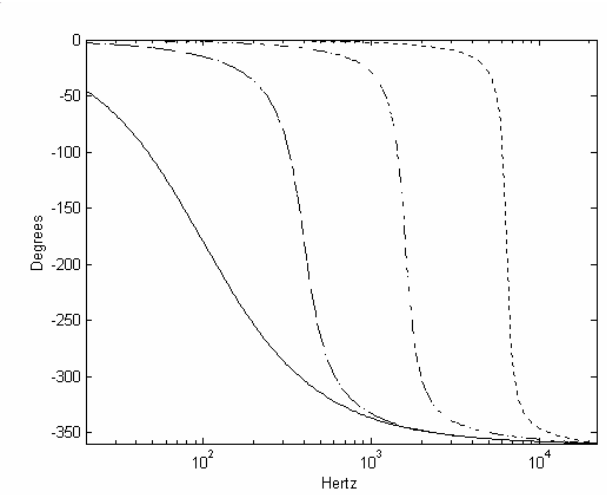

Fig. 4. Phase Responses for Second-order Allpass Filters

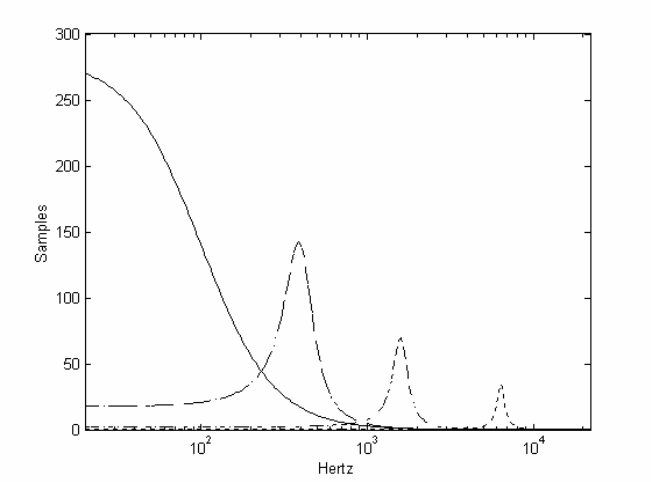

Fig. 5. Group Delays for Second-order Allpass Filters

f_c = 6400 Hz	*b* = 800 Hz	*Fs* = 44.1 kHz

B Decimal	0.89205429 -1.158481541 1.0
A Decimal	(1.0) -1.1584815 0.8920543
B 4.20 Hex	0E45DA ED76DD 100000
A 4.20 Hex	ED76DD 0E45DA
B 5.23 Hex	0722ED5 F6BB6E1 0800000
A' 5.23 Hex	094491F F8DD12B

Table 5. Coefficients for Second-order Allpass Filters

Table 5 shows the coefficients for the last of these four filters first in unquantized decimal, then in 4.20 hexadecimal and in 5.23 hexadecimal with reversed-sign denominator coefficients.

3.2 Parametric Equalization Filters

One of the most commonly encountered second-order IIR filters is the bell-shaped parametric equalization filter. Orfanidis offers a unique approach with some beneficial features, along with Matlab® code to implement it [15]. Other authors, [6], [7], [12], [16], [17], [18] offer other approaches, such as the one presented here, which is based upon an allpass filter.

$$\alpha = \begin{cases} \dfrac{\tan\left(\dfrac{\pi b}{F_s}\right) - g}{\tan\left(\dfrac{\pi b}{F_s}\right) + g} & g < 1 \\ \dfrac{\tan\left(\dfrac{\pi b}{F_s}\right) - 1}{\tan\left(\dfrac{\pi b}{F_s}\right) + 1} & \text{Otherwise} \end{cases} \tag{15}$$

where b is the desired bandwidth, and g is the desired (linear) gain.

$$\beta = -\cos\left(\frac{2\pi f_c}{F_s}\right) \tag{16}$$

$$\begin{aligned} H &= g - 1 \\ b_0 &= 1 + (1+\alpha)\tfrac{H}{2} \\ b_1 &= \beta(1-\alpha) \\ b_2 &= -\alpha - (1+\alpha)\tfrac{H}{2} \\ a_1 &= b_1 \\ a_2 &= -\alpha \end{aligned}$$

$$\mathrm{B} = [b_0\ b_1\ b_2] \tag{17}$$

$$\mathrm{A} = [1\ a_1\ a_2]$$

g	***b***	f_c
2 (6 dB)	200 Hz	100 Hz
0.5 (-6 dB)	200 Hz	400 Hz
2.82 (9 dB)	400 Hz	1600 Hz
0.25 (-12 dB)	800 Hz	6400 Hz

Table 6. Parameters used for Second-order Equalization Filter Examples (Note: $Q = f_c/b$)

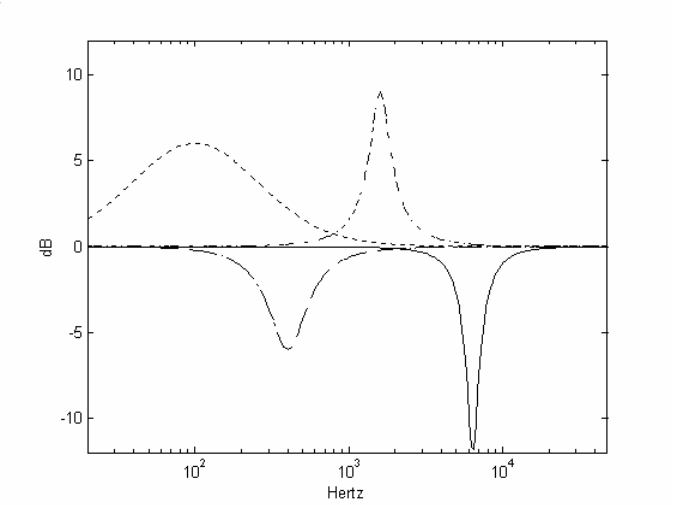

Fig. 6. Magnitude Responses for Second-order Equalization Filters

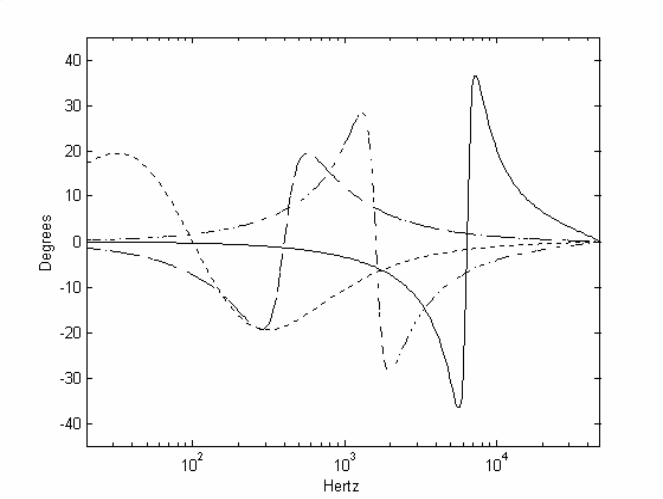

Fig. 7. Phase Responses for Second-order Equalization Filters

***g* = 2**	***b* = 200 Hz**	**f_c = 100 Hz**	***Fs* = 96 kHz**
B Decimal	`1.006503 -1.986952 0.980492`		
A Decimal	`(1.0) -1.986952 0.986995`		
B 4.20 Hex	`101AA2 E03572 0FB018`		
A 4.20 Hex	`E03572 0FCABB`		
B 5.23 Hex	`080D513 F01AB8C 07D80C6`		
A' 5.23 Hex	`0FE5474 F81AA27`		

Table 7. Coefficients for Second-order Equalizaton Filters

3.3 Butterworth Filters

Butterworth filters, [2], [3], [4], [19], [20], [21], are well-known and widely-used for high- and low-pass filtering operations. Technically, Butterworth filters are designed in the analog domain, and converted for use in the digital domain. However, these steps are combined into a simplified process presented here. Matlab® and other tools have built-in functions to design Butterworth filters, but the Matlab® functions presented in the appendix of this paper use only standard mathematical functions so that they will be useful as pseudo code to guide the implementation of these filters in any language.

3.3.1 First Order Low- and High-Pass

The denominator of a first-order analog Butterworth filter is

$$s + \omega_c$$

where ω_c is $2\pi f_c$,

f_c is the desired cut frequency.

For low-pass, the numerator is ω_c, and for high-pass it is s:

$$H_L(s) = \frac{\omega_c}{s + \omega_c}; \quad H_H(s) = \frac{s}{s + \omega_c} \tag{18}$$

Converting to Digital

A digital filter can be created from analog coefficients through the bilinear transform [2], [3], [4]

$$H(z) = H(s)\Big|_{s=\frac{k(z-1)}{z+1}}$$

$$\text{where } k = \frac{2\pi f_c}{\tan\left(\pi f_c / F_s\right)} \tag{19}$$

and F_s is the sampling frequency.

In other words,

$$H_L(z) = \frac{\omega_c + \omega_c z^{-1}}{k + \omega_c + (\omega_c - k)z^{-1}}$$

$$H_H(z) = \frac{k - kz^{-1}}{k + \omega_c + (\omega_c - k)z^{-1}}$$

In implementation the leading term of the denominator must be equal to 1. Therefore, the above equations must be normalized. The following is the equation for a first-order digital Butterworth low-pass filter:

$$b_0 = \frac{\omega_c}{k + \omega_c}$$

$$b_1 = \frac{\omega_c}{k + \omega_c}$$

$$a_1 = \frac{\omega_c - k}{k + \omega_c}$$

$$\text{B} = [b_0\ b_1\ 0] \tag{20}$$

$$\text{A} = [1\ a_1\ 0]$$

The numerator coefficients for the high-pass case are as shown in Eq. (21); the denominator coefficients are the same as those for the low-pass case, in Eq. (20):

$$b_0 = \frac{k}{k + \omega_c}$$

$$b_1 = -\frac{k}{k + \omega_c}$$

$$a_1 = \frac{\omega_c - k}{k + \omega_c}$$

$$B = [b_0 \; b_1 \; 0] \qquad (21)$$

The magnitude and phase responses for first-order Butterworth low-pass filters at cut-off frequencies 100, 400, 1600, and 6400 Hz are shown in Fig. 8 and 9. The coefficients for the 1600 Hz case are shown in Table 8.

The magnitude and phase responses for first-order Butterworth high-pass filters at cut-off frequencies 100, 400, 1600, and 6400 Hz are shown in Fig. 10 and 11. The coefficients for the 100 Hz case are shown in Table 9.

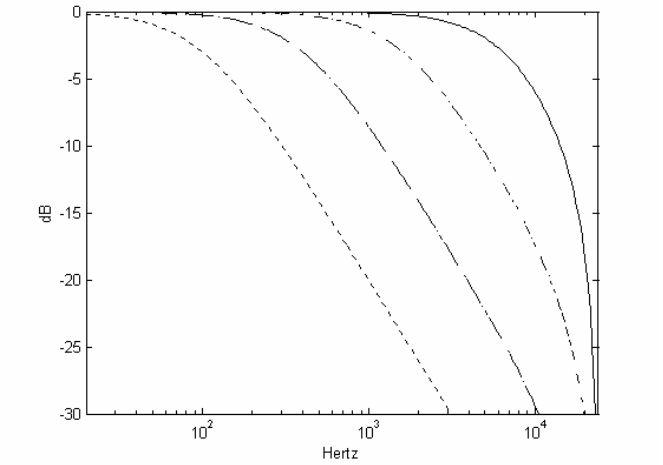

Fig. 8. Magnitude Responses for First-order Butterworth Low-pass Filters

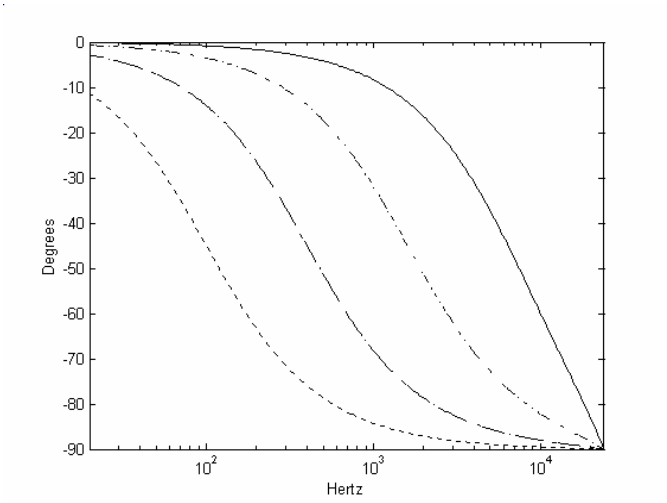

Fig. 9. Phase Responses for First-order Butterworth Low-pass Filters

f_c = 1600 Hz	Fs = 48 kHz
B Decimal	0.095107983 0.095107983 0
A Decimal	(1.0) -0.809784033 0
B 4.20 Hex	01858F 01858F 000000
A 4.20 Hex	F30B20 000000
B 5.23 Hex	00C2C7F 00C2C7F 0000000
A' 5.23 Hex	067A700 0000000

Table 8. Coefficients for First-order Butterworth Low-pass Filters

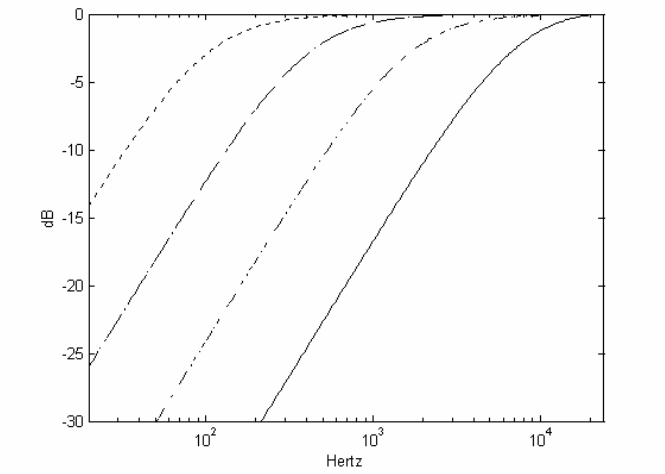

Fig. 10. Magnitude Responses for First-order Butterworth High-pass Filters

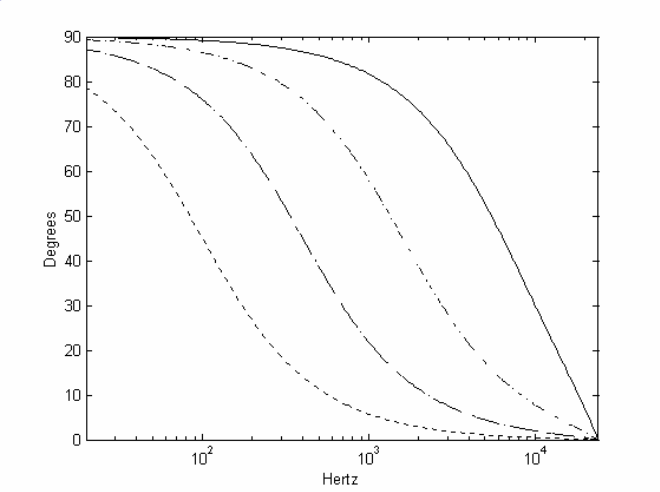

Fig. 11. Phase Responses for First-order Butterworth High-pass Filters

f_c = 100 Hz	Fs = 48 kHz
B Decimal	0.993497481 -0.993497481 0
A Decimal	(1.0) -0.986994963 0
B 4.20 Hex	0FE55D F01AA3 000000
A 4.20 Hex	F03545 000000
B 5.23 Hex	07F2AEC F80D514 0000000
A' 5.23 Hex	07E55D9 0000000

Table 9. Coefficients for First-order Butterworth High-pass Filters

3.3.2 Second Order Low- and High-Pass

The equations for second-order high- and low-pass filters can be derived using the methodology shown for the first-order case above. Here, only the results are shown. The equations for the digital low-pass filter coefficients are shown in Eq. (19).

$$b_0 = \frac{\omega_c^2}{\omega_c^2 + k^2 + \sqrt{2}k\omega_c}$$

$$b_1 = \frac{2\omega_c^2}{\omega_c^2 + k^2 + \sqrt{2}k\omega_c}$$

$$b_2 = \frac{\omega_c^2}{\omega_c^2 + k^2 + \sqrt{2}k\omega_c}$$

$$a_1 = \frac{2\omega_c^2 - 2k^2}{\omega_c^2 + k^2 + \sqrt{2}k\omega_c}$$

$$a_2 = \frac{\omega_c^2 + k^2 - \sqrt{2}k\omega_c}{\omega_c^2 + k^2 + \sqrt{2}k\omega_c}$$

$$B = [b_0\ b_1\ b_2]$$

$$A = [1\ a_1\ a_2] \quad (22)$$

where k and ω_c are defined as in the beginning of the previous section.

The numerator coefficients for the high-pass case are shown in Eq. (23); the denominator coefficients are the same as for the low-pass case, in Eq. (22).

$$b_0 = \frac{k^2}{\omega_c^2 + k^2 + \sqrt{2}k\omega_c}$$

$$b_1 = \frac{-2k^2}{\omega_c^2 + k^2 + \sqrt{2}k\omega_c}$$

$$b_2 = \frac{k^2}{\omega_c^2 + k^2 + \sqrt{2}k\omega_c}$$

$$B = [b_0\ b_1\ b_2] \quad (23)$$

Figs. 12 and 13 show respectively the magnitude and phase responses for second-order Butterworth low-pass filters; Table 10 shows the coefficients for the 400 Hz case at a sampling rate of 48 kHz.

The high-pass filter cases are shown in Figs. 14 and 15; Table 11 shows the coefficients for the 6.4 kHz case with a sampling frequency of 48 kHz.

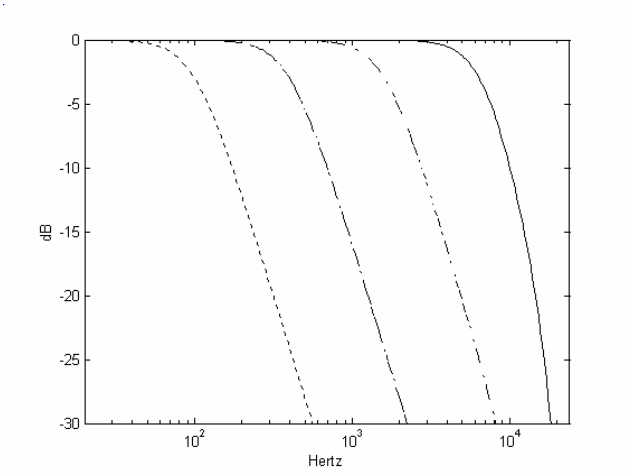

Fig. 12. Magnitude Responses for Second-order Butterworth Low-pass Filters

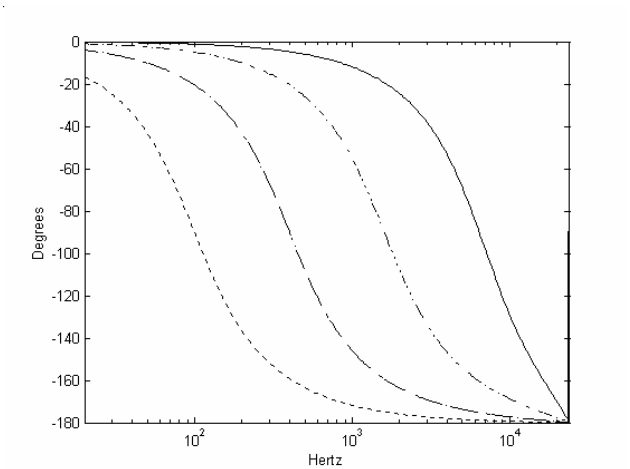

Fig. 13. Phase Responses for Second-order Butterworth Low-pass Filters

f_c = 400 Hz	*Fs* = 48 kHz
B Decimal	0.000661 0.001322 0.000661
A Decimal	(1.0) -1.925984 0.928627
B 4.20 Hex	0002B4 000569 0002B4
A 4.20 Hex	E12F2C 0EDBA8
B 5.23 Hex	00015A7 0002B4E 00015A7
A' 5.23 Hex	0F686A4 F8922C0

Table 10. Coefficients for Second-order Butterworth Low-pass Filters

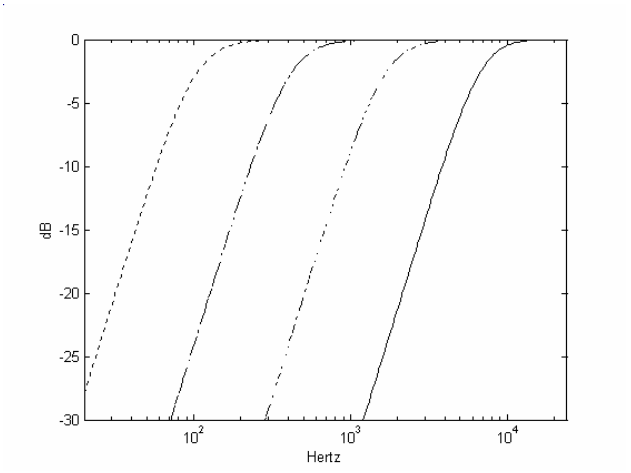

Fig. 14. Magnitude Responses for Second-order Butterworth High-pass Filters

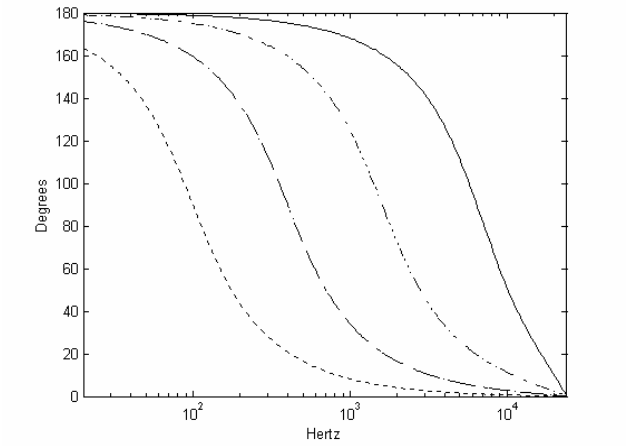

Fig. 15. Phase Responses for Second-order Butterworth Low-pass Filters

f_c = 6400 Hz	*Fs* = 48 kHz
B Decimal	0.547083 -1.094166 0.547083
A Decimal	(1.0) -0.877271 0.311060
B 4.20 Hex	08C0D9 EE7E4D 08C0D9
A 4.20 Hex	F1F6B4 04FA1A
B 5.23 Hex	04606CE F73F263 04606CE
A' 5.23 Hex	0704A67 FD82F2D

Table 11. Coefficients for Second-order Butterworth High-pass Filters

3.3.3 Third Order Low- and High-Pass

The equations for third-order low- and high-pass filters are shown in Eqs. (24) and (25) below. Note that the denominator coefficients are the same for either high-pass or low-pass, and are, therefore, shown only in Eq. (24).

Figs. 16 through 19 are the magnitude and phase plots for third-order Butterworth low- and high-pass filters. Tables 12 and 13 show the decimal coefficients. Implementation as second-order sections will requiring factoring, which is discussed in Section 5 of this paper.

$$b_0 = \frac{\omega_c^3}{\omega_c^3 + k^3 + 2\omega_c^2 k + 2\omega_c k^2}$$

$$b_1 = \frac{3\omega_c^3}{\omega_c^3 + k^3 + 2\omega_c^2 k + 2\omega_c k^2}$$

$$b_2 = \frac{3\omega_c^3}{\omega_c^3 + k^3 + 2\omega_c^2 k + 2\omega_c k^2}$$

$$b_3 = \frac{\omega_c^3}{\omega_c^3 + k^3 + 2\omega_c^2 k + 2\omega_c k^2}$$

$$a_1 = \frac{3\omega_c^3 - 3k^3 + 2\omega_c^2 k - 2\omega_c k^2}{\omega_c^3 + k^3 + 2\omega_c^2 k + 2\omega_c k^2}$$

$$a_2 = \frac{3\omega_c^3 + 3k^3 - 2\omega_c^2 k - 2\omega_c k^2}{\omega_c^3 + k^3 + 2\omega_c^2 k + 2\omega_c k^2}$$

$$a_3 = \frac{3\omega_c^3 - 3k^3 + 2\omega_c^2 k - 2\omega_c k^2}{\omega_c^3 - k^3 - 2\omega_c^2 k + 2\omega_c k^2}$$

$$B = [b_0\ b_1\ b_2\ b_3]$$

$$A = [1\ a_1\ a_2\ a_3] \qquad (24)$$

where k and ω_c are defined as in the beginning of the section on first-order Butterworth filters.

$$b_0 = \frac{k^3}{\omega_c^3 + k^3 + 2\omega_c^2 k + 2\omega_c k^2}$$

$$b_1 = \frac{-3k^3}{\omega_c^3 + k^3 + 2\omega_c^2 k + 2\omega_c k^2}$$

$$b_2 = \frac{3k^3}{\omega_c^3 + k^3 + 2\omega_c^2 k + 2\omega_c k^2}$$

$$b_3 = \frac{-k^3}{\omega_c^3 + k^3 + 2\omega_c^2 k + 2\omega_c k^2}$$

$$B = [b_0\ b_1\ b_2\ b_3] \qquad (25)$$

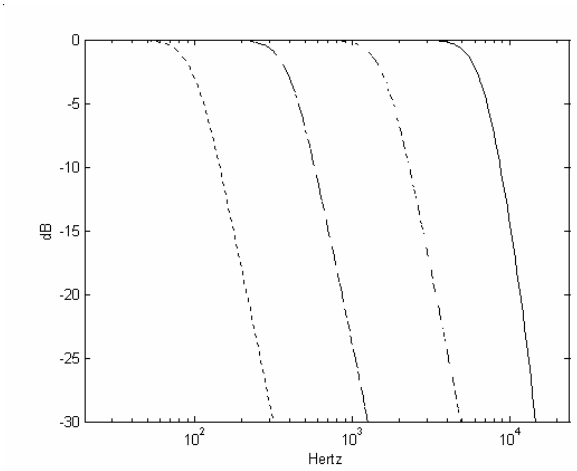

Fig. 16. Magnitude Responses for Third-order Butterworth Low-pass Filters

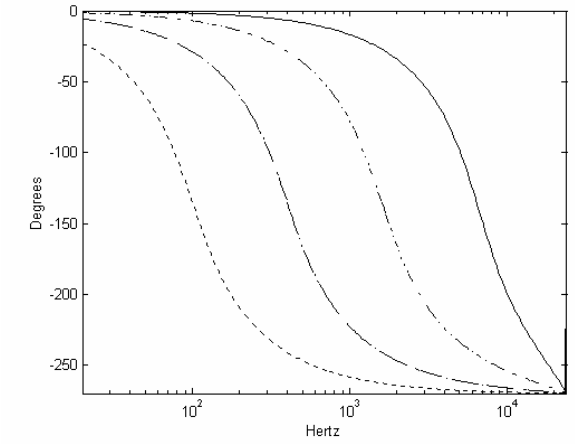

Fig. 17. Phase Responses for Third-order Butterworth Low-pass Filters

f_c = 1600 Hz		Fs = 48 kHz
B Decimal	0.0009 0.0028 0.0028 0.0009	
A Decimal	(1.0) -2.5819 2.2467 -0.6573	

Table 12. Coefficients for Third-order Butterworth Low-pass Filters

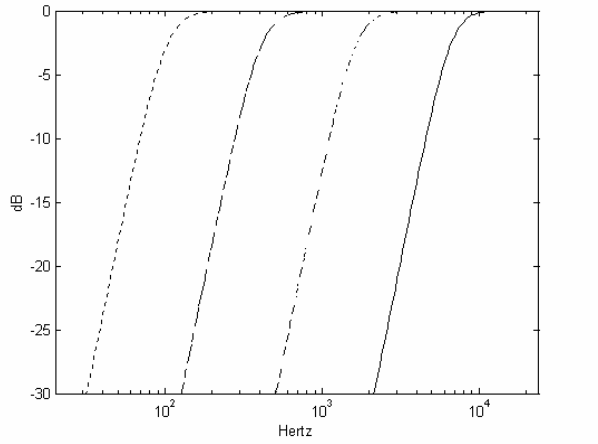

Fig. 18. Magnitude Responses for Third-order Butterworth High-pass Filters

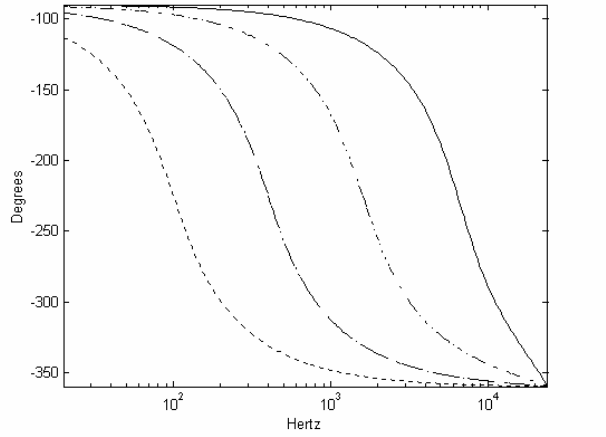

Fig. 19. Phase Responses for Third-order Butterworth High-pass Filters

f_c = 100 Hz		Fs = 48 kHz
B Decimal	0.987 -2.961 2.961 -0.987	
A Decimal	(1.0) -2.974 2.948 -0.975	

Table 13. Coefficients for Third-order Butterworth High-pass Filters

3.3.4 Fourth Order Low- and High-Pass

The equations for fourth-order low- and high-pass filters are shown in Eqs. (26) and (27) below. Note that the denominator coefficients, shown in Eq. (26), are the same for both cases.

$$b_0 = \frac{\omega_c^4}{k^4 + \alpha\omega_c k^3 + \beta\omega_c^2 k^2 + \alpha\omega_c^3 k + \omega_c^4}$$

$$b_1 = \frac{4\omega_c^4}{k^4 + \alpha\omega_c k^3 + \beta\omega_c^2 k^2 + \alpha\omega_c^3 k + \omega_c^4}$$

$$b_2 = \frac{6\omega_c^4}{k^4 + \alpha\omega_c k^3 + \beta\omega_c^2 k^2 + \alpha\omega_c^3 k + \omega_c^4}$$

$$b_3 = \frac{4\omega_c^4}{k^4 + \alpha\omega_c k^3 + \beta\omega_c^2 k^2 + \alpha\omega_c^3 k + \omega_c^4}$$

$$b_4 = \frac{\omega_c^4}{k^4 + \alpha\omega_c k^3 + \beta\omega_c^2 k^2 + \alpha\omega_c^3 k + \omega_c^4}$$

$$a_1 = \frac{-4k^4 - 2\alpha\omega_c k^3 + 2\alpha\omega_c^3 k + 4\omega_c^4}{k^4 + \alpha\omega_c k^3 + \beta\omega_c^2 k^2 + \alpha\omega_c^3 k + \omega_c^4}$$

$$a_2 = \frac{6k^4 - 2\beta\omega_c^2 k^2 + 6\omega_c^4}{k^4 + \alpha\omega_c k^3 + \beta\omega_c^2 k^2 + \alpha\omega_c^3 k + \omega_c^4}$$

$$a_3 = \frac{-4k^4 + 2\alpha\omega_c k^3 - 2\alpha\omega_c^3 k + 4\omega_c^4}{k^4 + \alpha\omega_c k^3 + \beta\omega_c^2 k^2 + \alpha\omega_c^3 k + \omega_c^4}$$

$$a_4 = \frac{k^4 - \alpha\omega_c k^3 + \beta\omega_c^2 k^2 - \alpha\omega_c^3 k + \omega_c^4}{k^4 + \alpha\omega_c k^3 + \beta\omega_c^2 k^2 + \alpha\omega_c^3 k + \omega_c^4}$$

where $\alpha = 2\left[\cos\left(\frac{\pi}{8}\right) + \cos\left(\frac{3\pi}{8}\right)\right]$,

$\beta = 2\left[1 + 2\cos\left(\frac{\pi}{8}\right)\cos\left(\frac{3\pi}{8}\right)\right]$

$B = [b_0\ b_1\ b_2\ b_3\ b_4]$ (26)

$A = [1\ a_1\ a_2\ a_3\ a_4]$

$$b_0 = \frac{k^4}{k^4 + \alpha\omega_c k^3 + \beta\omega_c^2 k^2 + \alpha\omega_c^3 k + \omega_c^4}$$

$$b_1 = \frac{-4k^4}{k^4 + \alpha\omega_c k^3 + \beta\omega_c^2 k^2 + \alpha\omega_c^3 k + \omega_c^4}$$

$$b_2 = \frac{6k^4}{k^4 + \alpha\omega_c k^3 + \beta\omega_c^2 k^2 + \alpha\omega_c^3 k + \omega_c^4}$$

$$b_3 = \frac{-4k^4}{k^4 + \alpha\omega_c k^3 + \beta\omega_c^2 k^2 + \alpha\omega_c^3 k + \omega_c^4}$$

$$b_4 = \frac{k^4}{k^4 + \alpha\omega_c k^3 + \beta\omega_c^2 k^2 + \alpha\omega_c^3 k + \omega_c^4}$$

$$B = [b_0\ b_1\ b_2\ b_3\ b_4] \quad (27)$$

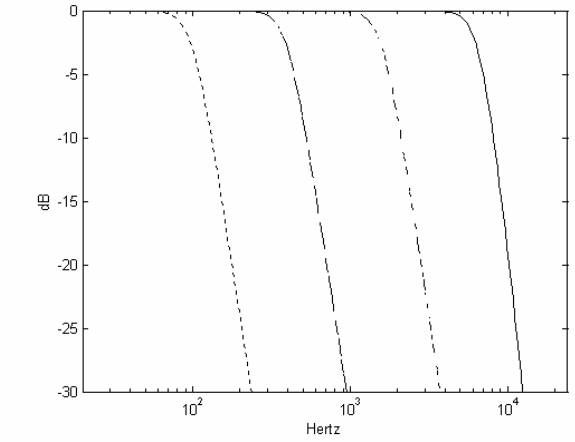

Fig. 20. Magnitude Responses for Fourth-order Butterworth Low-pass Filters

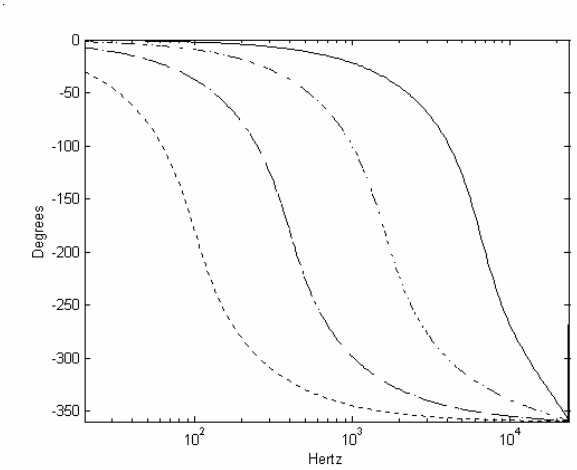

Fig. 21. Phase Responses for Fourth-order Butterworth Low-pass Filters

f_c = 400 Hz	*Fs* = 48 kHz
B Decimal x10^6	0.439 1.76 2.63 1.76 0.439
A Decimal	(1) -3.86 5.60 -3.61 0.872

Table 14. Coefficients for Fourth-order Butterworth Low-pass Filters

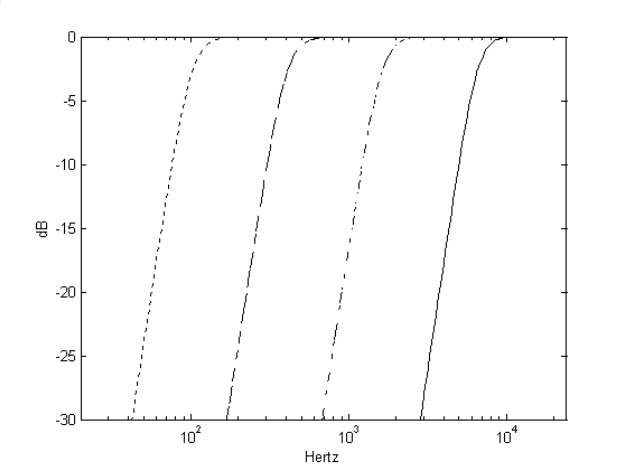

Fig. 22. Magnitude Responses for Fourth-order Butterworth High-pass Filters

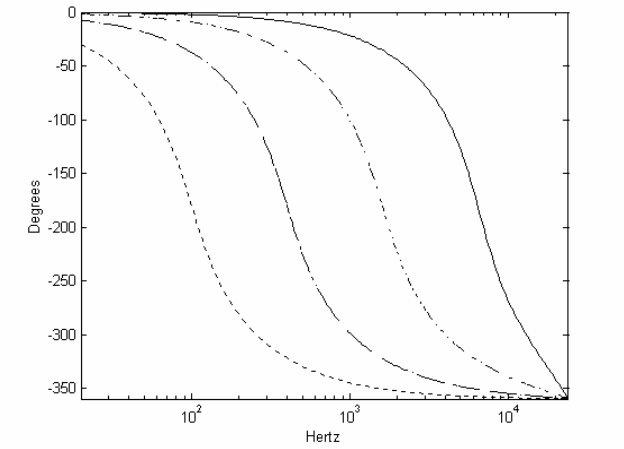

Fig. 23. Phase Responses for Fourth-order Butterworth High-pass Filters

f_c = 6400 Hz	*Fs* = 48 kHz
B Decimal	0.322 -1.29 1.93 -1.29 0.322
A Decimal	(1) -1.84 1.57 -0.636 0.104

Table 15. Coefficients for Fourth-order Butterworth High-pass Filters

3.4 Linkwitz-Riley Filters

Although not so generally well-known as Butterworth filters, Linkwitz-Riley high-pass and low-pass filters are frequently used for crossovers, and are, therefore, well-known at least in audio circles [19]. Like Butterworth filters, Linkwitz-Riley filters are designed in the analog domain, and converted for use in the

digital domain. However, these steps are combined into a simplified process presented here. Once again functions to implement these filters in Matlab® are presented in the appendix.

3.4.1 Second Order Low- and High-Pass

The equations for second-order low- and high-pass filters are shown in Eqs. (28) and (29) below. Since the denominator coefficients are the same for both cases, they are shown only in Eq. (28).

$$b_0 = \frac{\omega_c^2}{\omega_c^2 + k^2 + 2k\omega_c}$$

$$b_1 = \frac{2\omega_c^2}{\omega_c^2 + k^2 + 2k\omega_c}$$

$$b_2 = \frac{\omega_c^2}{\omega_c^2 + k^2 + 2k\omega_c}$$

$$a_1 = \frac{2\omega_c^2 - 2k^2}{\omega_c^2 + k^2 + 2k\omega_c}$$

$$a_2 = \frac{\omega_c^2 + k^2 - 2k\omega_c}{\omega_c^2 + k^2 + 2k\omega_c}$$

B = [b_0 b_1 b_2]

A = [1 a_1 a_2] (28)

$$b_0 = \frac{k^2}{\omega_c^2 + k^2 + 2k\omega_c}$$

$$b_1 = \frac{-2k^2}{\omega_c^2 + k^2 + 2k\omega_c}$$

$$b_2 = \frac{k^2}{\omega_c^2 + k^2 + 2k\omega_c}$$

B = [b_0 b_1 b_2] (29)

Figs. 24 and 25 show respectively the magnitude and phase responses for second-order Linkwitz-Riley low-pass filters; Table 16 shows the coefficients for the 100 Hz case at a sampling rate of 192 kHz.

The high-pass filter cases are shown in Figs. 26 and 27; Table 17 shows the coefficients for the 200 Hz case with a sampling frequency of 192 kHz.

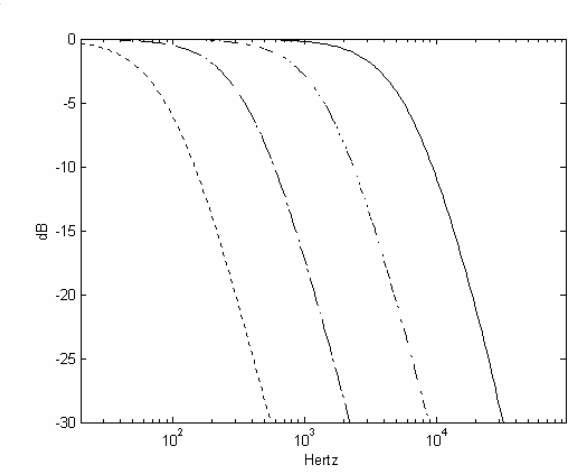

Fig. 24. Magnitude Responses for Second-order Linkwitz-Riley Low-pass Filters

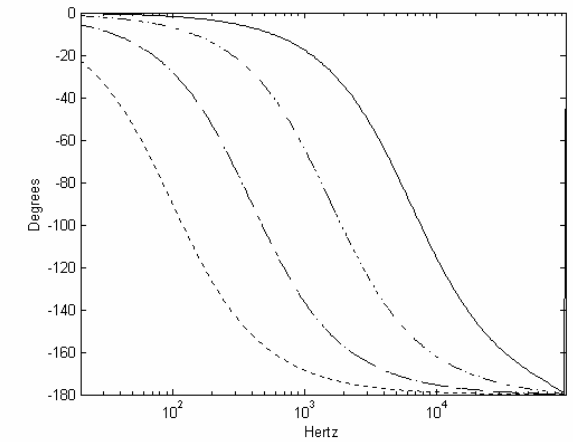

Fig. 25. Phase Responses for Second-order Linkwitz-Riley Low-pass Filters

f_c = 100 Hz	Fs = 192 kHz
B Decimal x10⁶	2.668566 5.337133 2.668566
A Decimal	(1.0) -1.9934657 0.9934764
B 4.20 Hex	000002 000005 000002
A 4.20 Hex	E01AC4 0FE547
B 5.23 Hex	0000016 000002C 0000016
A' 5.23 Hex	0FF29E2 F80D5C5

Table 16. Coefficients for Second-order Linkwitz-Riley Low-pass Filters

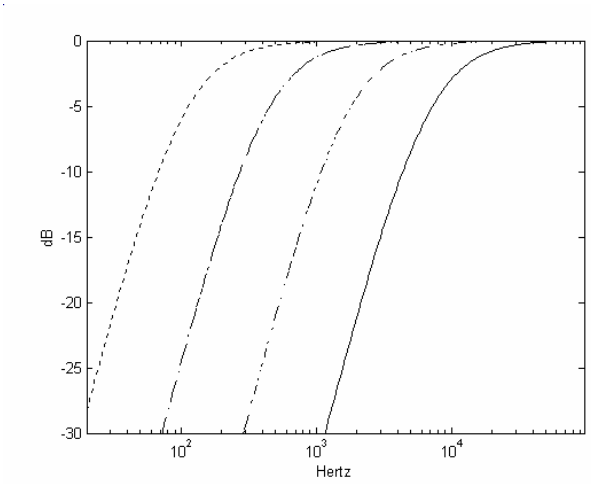

Fig. 26. Magnitude Responses for Second-order Linkwitz-Riley High-pass Filters

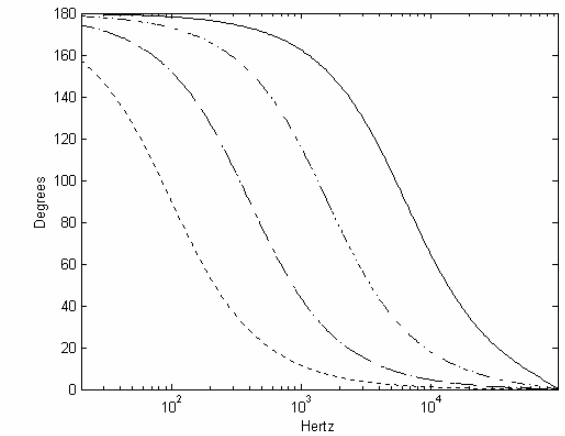

Fig. 27. Phase Responses for Second-order Linkwitz-Riley High-pass Filters

f_c = 400 Hz	Fs = 192 kHz
B Decimal	0.987037 -1.974074 0.987037
A Decimal	(1.0) -1.9739899 0.9741591
B 4.20 Hex	0FCAE7 E06A31 0FCAE7
A 4.20 Hex	E06A8A 0F9627
B 5.23 Hex	07E573C F035187 07E573C
A' 5.23 Hex	0FCABB3 F834EC2

Table 17. Coefficients for Second-order Linkwitz-Riley High-pass Filters

3.4.1 Fourth Order Low- and High-Pass

The equations for fourth-order low- and high-pass filters are shown in Eqs. (30) and (31) below. Since the denominator coefficients are the same for both cases, they are shown only in Eq. (30).

$$b_0 = \frac{\omega_c^4}{\omega_c^4 + 2\sqrt{2}\omega_c^3 k + 4\omega_c^2 k^2 + 2\sqrt{2}\omega_c k^3 + k^4}$$

$$b_1 = \frac{4\omega_c^4}{\omega_c^4 + 2\sqrt{2}\omega_c^3 k + 4\omega_c^2 k^2 + 2\sqrt{2}\omega_c k^3 + k^4}$$

$$b_2 = \frac{6\omega_c^4}{\omega_c^4 + 2\sqrt{2}\omega_c^3 k + 4\omega_c^2 k^2 + 2\sqrt{2}\omega_c k^3 + k^4}$$

$$b_3 = \frac{4\omega_c^4}{\omega_c^4 + 2\sqrt{2}\omega_c^3 k + 4\omega_c^2 k^2 + 2\sqrt{2}\omega_c k^3 + k^4}$$

$$b_4 = \frac{\omega_c^4}{\omega_c^4 + 2\sqrt{2}\omega_c^3 k + 4\omega_c^2 k^2 + 2\sqrt{2}\omega_c k^3 + k^4}$$

$$a_1 = \frac{4\omega_c^4 + 4\sqrt{2}\omega_c^3 k - 4\sqrt{2}\omega_c k^3 - 4k^4}{\omega_c^4 + 2\sqrt{2}\omega_c^3 k + 4\omega_c^2 k^2 + 2\sqrt{2}\omega_c k^3 + k^4}$$

$$a_2 = \frac{6\omega_c^4 - 8\omega_c^2 k^2 - 6k^4}{\omega_c^4 + 2\sqrt{2}\omega_c^3 k + 4\omega_c^2 k^2 + 2\sqrt{2}\omega_c k^3 + k^4}$$

$$a_3 = \frac{4\omega_c^4 - 4\sqrt{2}\omega_c^3 k + 4\sqrt{2}\omega_c k^3 - 4k^4}{\omega_c^4 + 2\sqrt{2}\omega_c^3 k + 4\omega_c^2 k^2 + 2\sqrt{2}\omega_c k^3 + k^4}$$

$$a_4 = \frac{\omega_c^4 - 2\sqrt{2}\omega_c^3 k + 4\omega_c^2 k^2 - 2\sqrt{2}\omega_c k^3 + k^4}{\omega_c^4 + 2\sqrt{2}\omega_c^3 k + 4\omega_c^2 k^2 + 2\sqrt{2}\omega_c k^3 + k^4} \quad (30)$$

$$b_0 = \frac{k^4}{\omega_c^4 + 2\sqrt{2}\omega_c^3 k + 4\omega_c^2 k^2 + 2\sqrt{2}\omega_c k^3 + k^4}$$

$$b_1 = \frac{-4k^4}{\omega_c^4 + 2\sqrt{2}\omega_c^3 k + 4\omega_c^2 k^2 + 2\sqrt{2}\omega_c k^3 + k^4}$$

$$b_2 = \frac{6k^4}{\omega_c^4 + 2\sqrt{2}\omega_c^3 k + 4\omega_c^2 k^2 + 2\sqrt{2}\omega_c k^3 + k^4}$$

$$b_3 = \frac{-4k^4}{\omega_c^4 + 2\sqrt{2}\omega_c^3 k + 4\omega_c^2 k^2 + 2\sqrt{2}\omega_c k^3 + k^4}$$

$$b_4 = \frac{k^4}{\omega_c^4 + 2\sqrt{2}\omega_c^3 k + 4\omega_c^2 k^2 + 2\sqrt{2}\omega_c k^3 + k^4} \quad (31)$$

Figs. 28 and 29 show respectively the magnitude and phase responses for fourth-order Linkwitz-Riley low-pass filters; Table 18 shows the coefficients for the 1600 Hz case at a sampling rate of 192 kHz.

The high-pass filter cases are shown in Figs. 30 and 31; Table 19 shows the coefficients for the 6400 Hz case with a sampling frequency of 192 kHz.

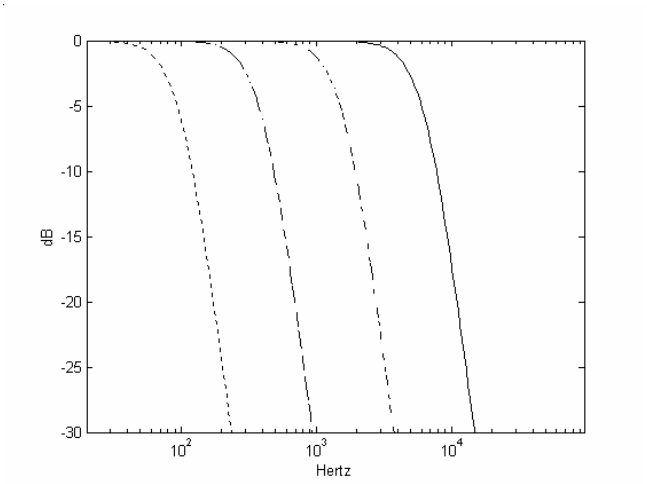

Fig. 28. Magnitude Responses for Fourth-order Linkwitz-Riley Low-pass Filters

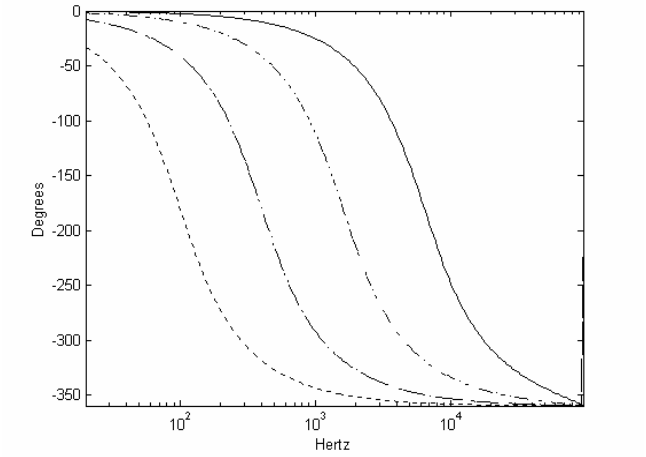

Fig. 29. Phase Responses for Fourth-order Linkwitz-Riley Low-pass Filters

f_c = 1600 Hz	**Fs = 192 kHz**
B Decimal x10⁶	0.437 1.75 2.62 1.75 0.437
A Decimal	(1.0) -3.85 5.57 -3.58 0.862

Table 18. Coefficients for Fourth-order Linkwitz-Riley Low-pass Filters

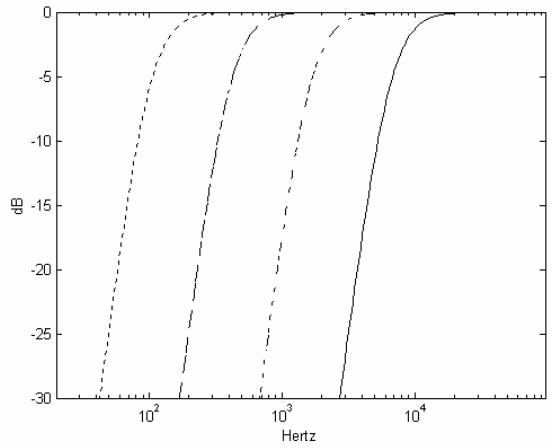

Fig. 30. Magnitude Responses for Fourth-order Linkwitz-Riley High-pass Filters

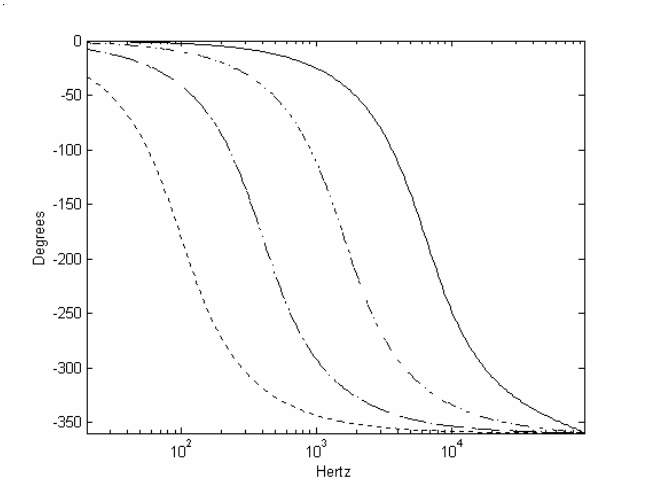

Fig. 31. Phase Responses for Fourth-order Linkwitz-Riley High-pass Filters

f_c = 6400 Hz	Fs = 192 kHz
B Decimal	`0.744 -2.97 4.46 -2.97 0.744`
A Decimal	`(1.0) -3.41 4.39 -2.54 0.553`

Table 19. Coefficients for Fourth-order Linkwitz-Riley High-pass Filters

3.5 Bass and Treble Shelf Filters

Often used in tone control applications, bass and treble shelf filters are an important class of audio filters. Here both first [6] and second [22] order designs are discussed. Once again functions to implement these filters in Matlab® are presented in the appendix.

3.5.1 First Order Bass and Treble Shelves

The equations for first-order bass and treble shelf filters are shown in Eq. (32) below.

if $g > 1$

$$a = \frac{\tan\left(\frac{\pi f_c}{F_s}\right) - 1}{\tan\left(\frac{\pi f_c}{F_s}\right) + 1}$$

if bass

$$b_0 = 1 + \frac{(1+a)(g-1)}{2}$$

$$b_1 = a + \frac{(1+a)(g-1)}{2}$$

else %treble

$$b_0 = 1 + \frac{(1-a)(g-1)}{2}$$

$$b_1 = a + \frac{(a-1)(g-1)}{2}$$

else $\%g \le 1$

if bass

$$a = \frac{\tan\left(\frac{\pi f_c}{F_s}\right) - g}{\tan\left(\frac{\pi f_c}{F_s}\right) + g}$$

$$b_0 = 1 + \frac{(1+a)(g-1)}{2}$$

$$b_1 = a + \frac{(1+a)(g-1)}{2}$$

else %treble

$$a = \frac{g \cdot \tan\left(\frac{\pi f_c}{F_s}\right) - 1}{g \cdot \tan\left(\frac{\pi f_c}{F_s}\right) + 1}$$

$$b_0 = 1 + \frac{(1-a)(g-1)}{2}$$

$$b_1 = a + \frac{(a-1)(g-1)}{2}$$

B = [b_0 b_1 0]

A = [1 a_1 0] (32)

Figs. 32 and 33 show respectively the magnitude and phase responses for boost and cut first-order shelf filters, with corners at 100 Hz and 1600 Hz; Table 20 shows the coefficients for the 100 Hz boost case at a sampling rate of 32 kHz.

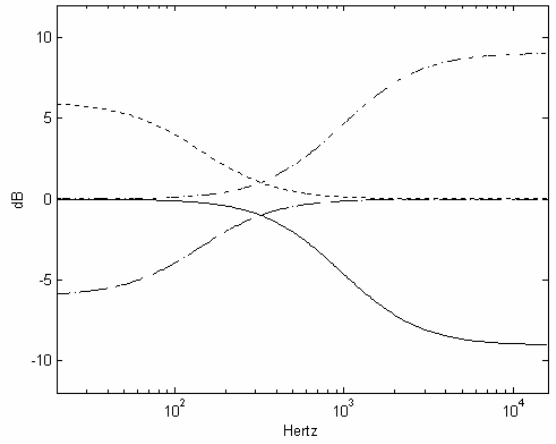

Fig. 32. Magnitude Responses for First-order Shelf Filters

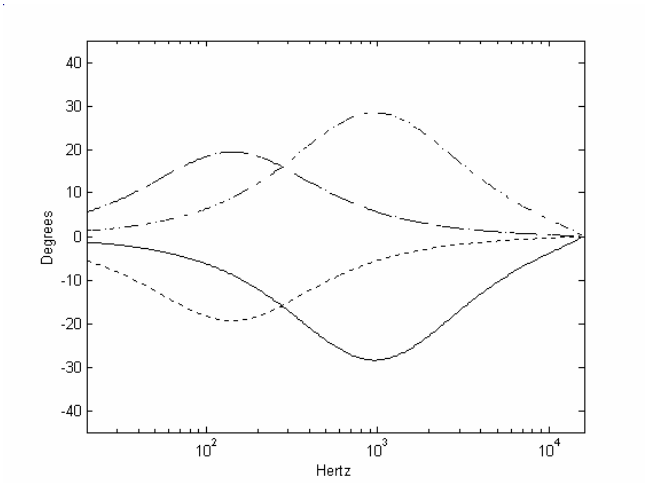

Fig. 33. Phase Responses for First-order Shelf Filters

f_c = 100 Hz	*Fs* = 32 kHz	g=1.995 (6 dB)
B Decimal	1.0096763 -0.9708790 0.0	
A Decimal	(1.0) -0.9805553 0.0	
B 4.20 Hex	1027A2 F07748 000000	
A 4.20 Hex	F04FA6 000000	
B 5.23 Hex	0813D12 F83BA3D 0000000	
A' 5.23 Hex	07D82D6 0000000	

Table 20. Coefficients for First-order Bass Shelf Filters

3.5.1 Second Order Bass and Treble Shelves

The equations for second-order bass and treble shelf filters are shown in Eq. (33) below.

if $g \geq 0.5$ & $g \leq 2$

$$F = \sqrt{g}$$

elseif $g > 1$

$$F = \frac{g}{\sqrt{2}}$$

else

$$F = g\sqrt{2}$$

end

$$g_d = \frac{F^2 - 1}{\sqrt[4]{g^2 - F^2}}$$

$$g_n = g_d \sqrt{g}$$

$$a = \tan\left(\pi\left(\frac{f_c}{F_s} - \frac{1}{4}\right)\right)$$

$$\sigma = \frac{\sqrt{2}}{2}; \quad \rho = \frac{\pi}{2}; \quad \phi = \frac{f_c}{F_s}\pi$$

if bass

$$b_0 = -\frac{-1 + 2a - 2\sigma g_n - 2g_n^2 a + 2\sigma g_n a^2 - a^2 - g_n^2 - g_n^2 a^2}{1 - 2a + 2\sigma g_d - 2\sigma g_d a^2 + 2g_d^2 a + a^2 + g_d^2 + g_d^2 a^2}$$

$$b_1 = -\frac{2 - 4a - 4g_n^2 a + 2a^2 - 2g_n^2 - 2g_n^2 a^2}{1 - 2a + 2\sigma g_d - 2\sigma g_d a^2 + 2g_d^2 a + a^2 + g_d^2 + g_d^2 a^2}$$

$$b_2 = \frac{1 - 2a - 2\sigma g_n + 2g_n^2 a + 2\sigma g_n a^2 + a^2 + g_n^2 + g_n^2 a^2}{1 - 2a + 2\sigma g_d - 2\sigma g_d a^2 + 2g_d^2 a + a^2 + g_d^2 + g_d^2 a^2}$$

$$a_1 = \frac{-2 + 4a + 4g_d^2 a - 2a^2 + 2g_d^2 + 2g_d^2 a^2}{1 - 2a + 2\sigma g_d - 2\sigma g_d a^2 + 2g_d^2 a + a^2 + g_d^2 + g_d^2 a^2}$$

$$a_2 = \frac{1 - 2a - 2\sigma g_d + 2g_d^2 a + 2\sigma g_d a^2 + a^2 + g_d^2 + g_d^2 a^2}{1 - 2a + 2\sigma g_d - 2\sigma g_d a^2 + 2g_d^2 a + a^2 + g_d^2 + g_d^2 a^2}$$

else

$$b_0 = \frac{1 + 2a + 2\sigma g_n - 2g_n^2 a - 2\sigma g_n a^2 + a^2 + g_n^2 + g_n^2 a^2}{1 + 2a + 2\sigma g_d - 2\sigma g_d a^2 - 2g_d^2 a + a^2 + g_d^2 + g_d^2 a^2}$$

$$b_1 = \frac{2 + 4a + 4g_n^2 a + 2a^2 - 2g_n^2 - 2g_n^2 a^2}{1 + 2a + 2\sigma g_d - 2\sigma g_d a^2 - 2g_d^2 a + a^2 + g_d^2 + g_d^2 a^2}$$

$$b_2 = \frac{1 + 2a - 2\sigma g_n - 2g_n^2 a + 2\sigma g_n a^2 + a^2 + g_n^2 + g_n^2 a^2}{1 + 2a + 2\sigma g_d - 2\sigma g_d a^2 - 2g_d^2 a + a^2 + g_d^2 + g_d^2 a^2}$$

$$a_1 = \frac{2 + 4a + 4g_d^2 a + 2a^2 - 2g_d^2 - 2g_d^2 a^2}{1 + 2a + 2\sigma g_d - 2\sigma g_d a^2 - 2g_d^2 a + a^2 + g_d^2 + g_d^2 a^2}$$

$$a_2 = \frac{1 + 2a - 2\sigma g_d - 2g_d^2 a + 2\sigma g_d a^2 + a^2 + g_d^2 + g_d^2 a^2}{1 + 2a + 2\sigma g_d - 2\sigma g_d a^2 - 2g_d^2 a + a^2 + g_d^2 + g_d^2 a^2}$$

B = [b_0 b_1 b_2]

A = [1 a_1 a_2] (33)

Figs. 34 and 35 show the magnitude and phase responses for boost and cut first-order shelf filters, with corners at 100 Hz and 1600 Hz; Table 20 shows the coefficients for the 100 Hz boost case at a sampling rate of 32 kHz.

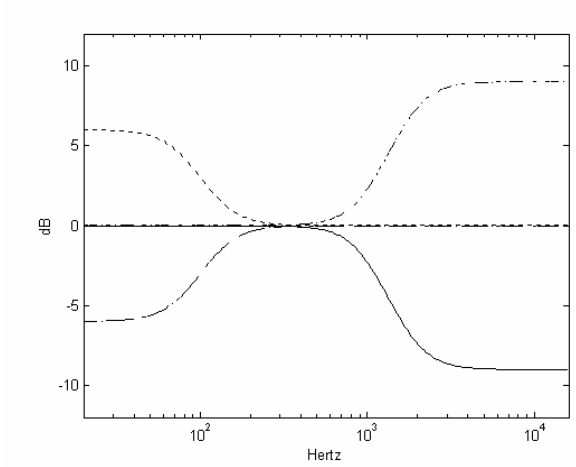

Fig. 34. Magnitude Responses for Second-order Shelf Filters

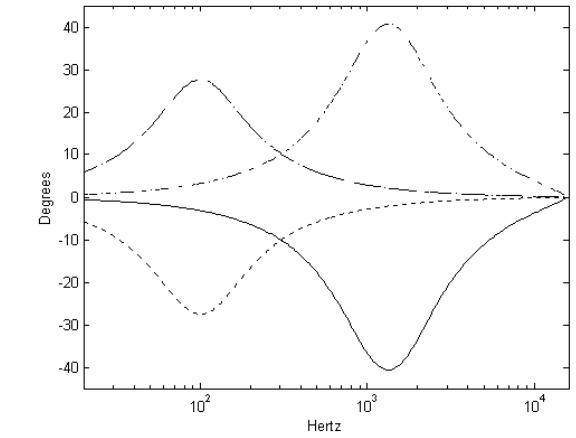

Fig. 35. Phase Responses for Second-order Shelf Filters

f_c = 1600 Hz	*Fs* = 32 kHz	g=0.355 (-9 dB)
B Decimal	0.39051 -0.59723 0.24239	
A Decimal	(1.0) -1.71565 0.75132	
B 4.20 Hex	063F8C F671C4 03E0CF	
A 4.20 Hex	E48CB8 0C0568	
B 5.23 Hex	031FC62 FB38E1E 01F067D	
A' 5.23 Hex	0DB9A45 F9FD4BE	

Table 21. Coefficients for Second-order Treble Shelf Cut Filter

4 IMPLEMENTING FILTERS OF OTHER ORDERS

Using second-order IIR structures, IIR filters of any order, up to and including 2 times the number of second-order filters available, can be implemented. In this section the method of implementing other-than-second-order filters is described.

4.1 Implementing First-order Filters

First-order filters are not only important in their own right, but they are an essential part of implementing filters of higher, odd orders. A first-order filter can be implemented in the second-order structure by simply downloading zero for the values of b_2 and a_2. However, if more than one first-order filter is to be implemented, two of them can be implemented together in a single second-order structure. This is done by multiplying the coefficients in the following fashion:

First-order Filter 1: $[b_{10}, b_{11}], [1, a_{11}]$

First-order Filter 2: $[b_{20}, b_{21}], [1, a_{21}]$

Resulting second-order Filter:

$$[b_{10}\, b_{20},\ b_{10}\, b_{21} + b_{20}\, b_{11},\ b_{11}\, b_{21}]$$
$$[1,\ a_{10}\, a_{21} + a_{20}\, a_{11},\ a_{11}\, a_{21}] \quad (34)$$

4.1.1 First-order Filter Example

Let filter 1 be a first-order Butterworth high-pass filter at 100 Hz, to be implemented at a sample rate of 48 kHz. The response of this filter is shown in Fig. 36.

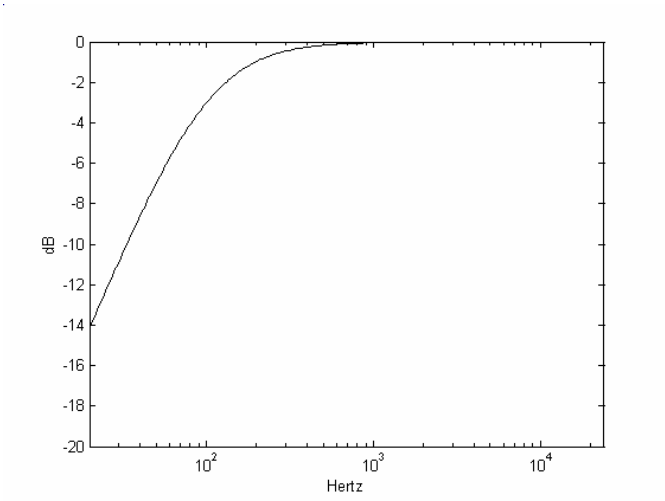

Fig. 36. Filter 1

For Filter 1,

B = [0.99349748134078 -0.99349748134078],

A = [1.00000000000000 -0.98699496268155]

To implement this filter alone in a second-order section, the following coefficients should be downloaded:

B = [0.99349748134078 -0.99349748134078 0],

A = [1.00000000000000 -0.98699496268155 0]

Now let Filter 2 be a 5 dB, first-order bass shelf at 500 Hz as shown in Fig. 37.

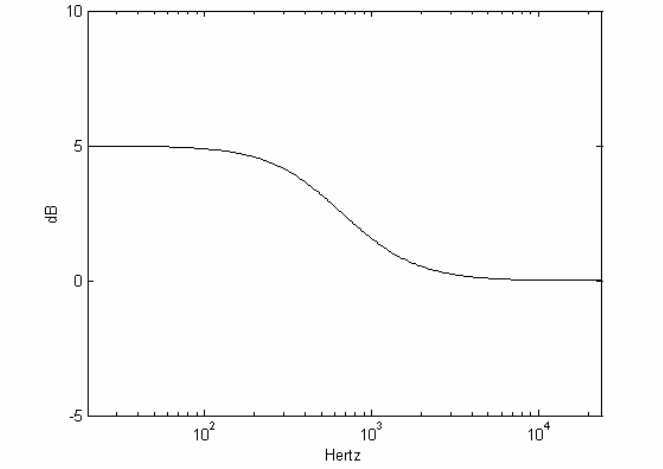

Fig. 37. Filter 2

For Filter 2,

B = [1.02467059808085 -0.91193160991121],

A = [1.00000000000000 -0.93660220799206]

To implement this filter alone in a second-order section the following should be downloaded:
B = [1.02467059808085 -0.91193160991121 0],
A = [1.00000000000000 -0.93660220799206 0]

Now, if these two filters are to be combined and implemented as a single second-order filter, the coefficients, as derived by Eq. (31) will be as follows:
B=[1.01800765839728 -1.92400941599910 0.90600175760182]
A=[1.0000000000000 -1.92359717067361 0.92442166132458]

Whether implemented separately in two second-order sections, or jointly in one, the composite filter response will be that shown in Fig. 38.

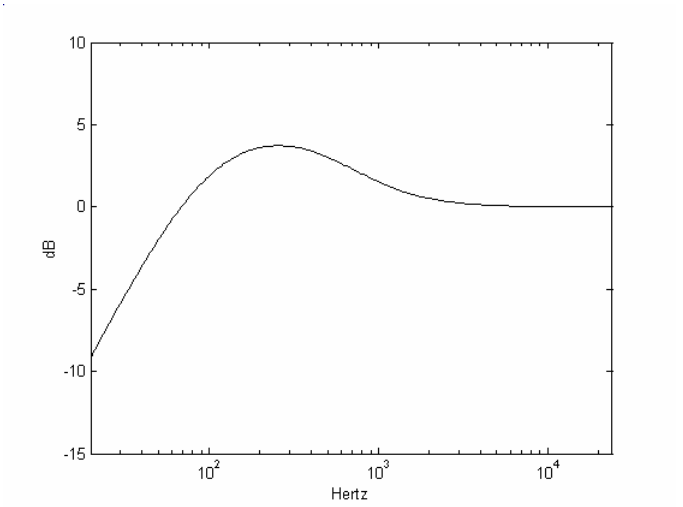

Fig. 38. Composite Filter

4.2 Implementing Higher-order Filters

The procedure for implementing filters of higher orders as cascades of second-order filter sections is discussed in this section.

Consider an n[th]-order filter as shown in Eq. (35):

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-n}}{1 + a\ z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}} \tag{35}$$

This equation can be factored and written in the following form:

$$H(z) = b_0 \cdot \frac{(z - Z_{1R} - Z_{1I} i)(z - Z_{1R} + Z_{1I} i)}{(z - P_{1R} - P_{1I} i)(z - P_{1R} + P_{1I} i)} \cdot \frac{(z - Z_{2R} - Z_{2I} i)(z - Z_{2R} + Z_{2I} i)\cdots}{(z - P_{2R} - P_{2I} i)(z - P_{2R} + P_{2I} i)\cdots} \cdot \frac{(z - Z_{nR} - Z_{nI} i)(z - Z_{nR} + Z_{nI} i)}{(z - P_{nR} - P_{nI} i)(z - P_{nR} + P_{nI} i)} \tag{36}$$

Where Z and P represent Zeros and Poles and where R and I represent their Real and Imaginary parts. In addition, b_0 is the first numerator coefficient of the original equation. This contains scaling information that is lost when factoring.

As shown in Eq. (36), all of the complex poles and zeroes will appear in complex conjugate pairs. For filters of odd order, at least one real pole and one real zero will remain. In some cases there might also be other real poles or real zeros. The real poles and zeros can be implemented as described in the section above discussing first-order filters. That is, if there is more than one real pole and one real zero, they can be combined. Otherwise, the single, real, first-order filter is implemented using a second-order section.

After dealing with the real poles and zeros, the remaining complex conjugate pairs can then be combined into second-order sections with real coefficients as shown in Eq. (37):

$$\begin{aligned} &(z - Z_{1R} - Z_{1I} i)(z - Z_{1R} + Z_{1I} i) \\ &= (z - Z_1)(z - Z_1^*) \\ &= z^2 - (Z_1 + Z_1^*) z + Z_1 Z_1^* \\ &= z^2 - 2 Z_{1R} z + Z_{1R}^2 + Z_{1I}^2 \end{aligned} \tag{37}$$

Now, by combining a pair of complex zeroes with a pair of complex poles, and dividing through by z^2, a filter in the form of Eq. (1) appears. The following is general advice for creating the best filter implementations:

1. Usually the greatest success is achieved when the second-order sections each appear

to be a viable filter in its own right; massive boosts or other scaling disparities between filters can lead to numerical problems. In general, if the poles and zeros are ordered according to magnitude, and greatest-magnitude poles are paired with greatest-magnitude zeroes, the resulting filters will be well-behaved.

2. Notice the b_0 in Eq. (36), the b_0 coefficient from the original, higher-order filter. This scale factor from the original filter is lost in the factoring process and must be re-applied. Since this discussion is specific to cascade filter implementations, this scale factor needs to be applied to only 1 of the resulting filters. If the scale factor is less than one it should be applied to the filter with the greatest magnitude (or largest numerator coefficients). Otherwise, it can be applied to a smaller filter. Sometimes it might make the most sense to distribute the scale factor across multiple filters. For example, when factoring a fourth-order filter, it might make sense to scale both resulting second-order sections by the square root of b_0.

3. As with any filter implementation, care should be taken to see that there is adequate headroom for each of the filters being implemented. If one filter provides a cut that compensates another filter's boost, placing the cut first in the chain will help prevent clipping between filters. Artificially scaling the numerators, and using subsequent filters or other system gains to compensate for the scaling can also help assure adequate headroom.

4.2.1 Higher-order Filter Example

In this example a 500 Hz, fourth-order Butterworth high-pass filter is designed for a 44.1 kHz sampling rate:

```
B = [0.91110246841372   -3.64440987365487
5.46661481048230  -3.64440987365487  0.91110246841372]
A = [1.00000000000000  -3.81386538359704
5.45872379150560  -3.47494261156512 0.83010770795173]
```

When the poles and zeros are computed, two real zeros are found, and all poles are complex:

```
Zeros:
  1.00022566526664
  0.99977439304055
  0.99999997084641 - 0.00022563610571i
  0.99999997084641 + 0.00022563610571i
Poles:
  0.97101464699516 + 0.06401591243324i
  0.97101464699516 - 0.06401591243324i
  0.93591804480336 + 0.02555784867173i
  0.93591804480336 - 0.02555784867173i
```

Taking care to assure that the real zeros are handled as a pair, in order to allow all complex roots to be handled in complex conjugate pairs, Eq. (37) is applied to derive two sets of second-order equations. The coefficients of these are shown below:

```
B1 = [0.9545168769664 -1.9090336982776 0.9545168699073]
A1 = [1.000000000000  -1.8718360896067   0.8765957902173]
B2 = [0.9545168769664  -1.909033809588   0.9545168840256]
A2 = [1.000000000000  -1.9420292939903   0.9469674817238]
```

Notice that the b_0 coefficients are equal in these two sets of coefficients. This is because each of the derived numerator polynomials was multiplied though by the square root of the b_0 coefficient of the original equation.

Fig. 39 shows the two second-order filters in dotted and dashed lines, and the fourth-order filter resulting from cascading them, in other words, the original fourth-order filter, as a solid line.

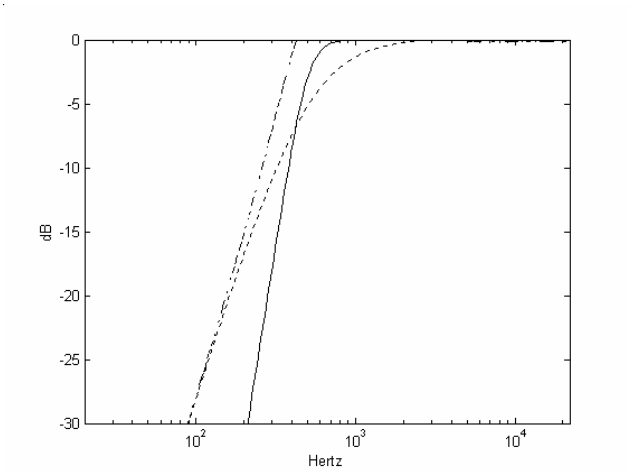

Fig. 39. Factored and reconstructed fourth-order Butterworth filter

4.3 Converting Filters from Other Structures for Second-order Cascade Implementation

Although it is possible to generate any desired filter function using a cascade structure, sometimes the designer will be called upon to convert filters that have already been designed for a parallel or combination cascade-parallel structure to be implemented fully in a cascade structure. The process for doing this is described in this section.

First, consider the filter structure shown in Fig. 40.

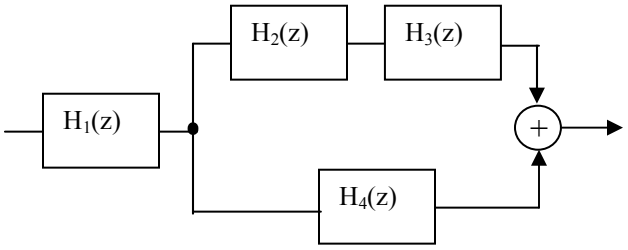

Fig. 40. Cascade-Parallel Filter Structure

This structure has an overall transfer function, from input to output, of

$$H_1(z)\left[H_2(z)H_3(z)+H_4(z)\right] \qquad (38)$$

To implement this transfer function in a cascade structure, this overall transfer function is computed, then factored into second-order sections and implemented.

4.3.1 Structure Conversion Example

Suppose that the structure of Fig. 40 is populated with the following transfer functions:

$H_1(z)$ is a first-order Butterworth highpass filter with a cutoff frequency of 80 Hz. This filter has the following coefficients:

B = [0.99479123765938 -0.99479123765938]

A = [1.00000000000000 -0.98958247531875]

$H_2(z)$ is a second-order Butterworth highpass filter with a cutoff frequency of 300 Hz. It has the coefficients shown :

B=[0.97261389849984 -1.94522779699969 0.97261389849984]

A=[1.0000000000000 -1.94447765776709 0.94597793623228]

$H_3(z)$ is a second-order Butterworth lowpass filter with a cutoff frequency of 3 kHz. Its coefficients are as shown:

B=[0.02995458220809 0.05990916441618 0.02995458220809]

A=[1.0000000000000 -1.45424358625159 0.57406191508395]

$H_4(z)$ is a first-order Butterworth lowpass filter with a cutoff frequency of 5 kHz. It has the coefficients shown:

B=[0.25342728698435 0.25342728698435]

A=[1.00000000000000 -0.49314542603130]

The magnitude responses of these four filters are shown in Fig. 41, and the resulting composite filter is shown in Fig. 42.

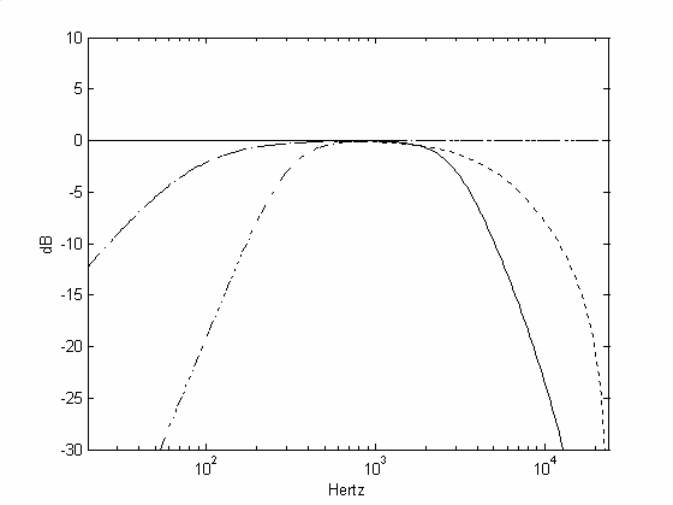

Fig. 41. Original Filters One Through Four

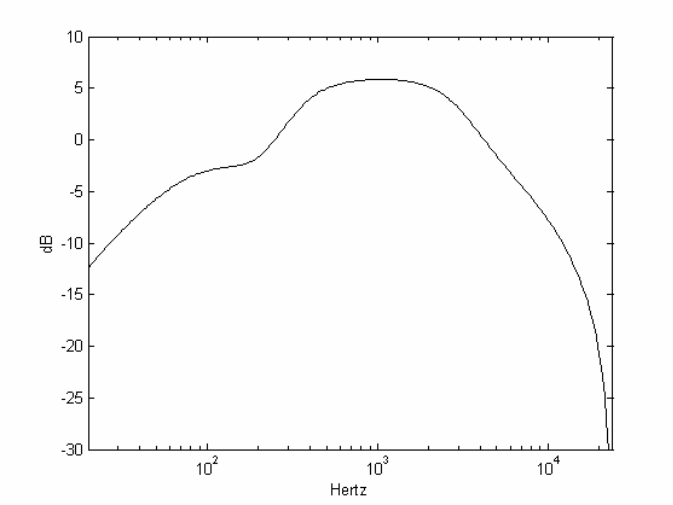

Fig. 42. Composite Magnitude Response

All of these bundled together form the sixth-order filter whose transfer function is shown in Eq. (38). To convert this to a cascade structure, first the transfer function as a simplified ratio of polynomials is sought. This process can be much simplified by use of a symbolic manipulation program such as Maple®.

Plugging in the general formulae for first- and second-order filters, Eq. (38) becomes

$$\frac{B_{10}+B_{11}z^{-1}}{1+A_{11}z^{-1}}\cdot\left[\frac{B_{20}+B_{21}z^{-1}+B_{22}z^{-2}}{1+A_{21}z^{-1}+A_{22}z^{-2}}\cdot\frac{B_{30}+B_{31}z^{-1}+B_{32}z^{-2}}{1+A_{31}z^{-1}+A_{32}z^{-2}}+\frac{B_{40}+B_{41}z^{-1}}{1+A_{41}z^{-1}}\right] \tag{39}$$

How to proceed from here differs somewhat due to available tools, personal preferences, and desired generality of the solution. Obviously, the coefficients could be substituted and Eq. (39) simplified. Or, especially if using a symbolic tool, this equation can be simplified symbolically. This has the advantage that the algebra does not have to be repeated every time one of the original filters changes.

It is also worthwhile to notice that, if the orders of the four filters will sometimes change, Eq. (39) can be written with the highest possible orders, simplified symbolically, and then when deriving the final equation, substituting zeros for any unused higher-order coefficients.

In addition, take note that H_1 is already in cascade form and can be implemented as such. Therefore, the combination of H_2 through H_4 can be simplified and reduced to two second-order sections and one first-order section. These four total sections (including H_1) can then be implemented using four second-order structures, or the two first-order sections can be combined at that time for a three-section implementation. Additionally, the entire equation can be reduced to one simplified ratio of polynomials, and then factored.

For purposes of this example, it is assumed that filters will not change, so the coefficients can be substituted into Eq. 39 and the resulting equation simplified. The resulting sixth-order equation has this form

$$H(z)=\frac{b_0+b_1z^{-1}+\cdots+b_6z^{-6}}{1+a_1z^{-1}+\cdots+a_6z^{-6}} \tag{40}$$

The coefficients are

B=[0.28108973410751 -0.90011731939953 0.80032820573604 0.31515805072053 -0.95880370670137 0.58495926867901 -0.12261423314218]

A=[1.00000000000000 -4.88144914536874 9.87517090253539 -10.59711698022950 6.35966205774405 -2.02127862265964 0.26501273714634]

If a sixth-order structure is available, this filter can be implemented as is to achieve the same transfer function as if implemented by the structure of Fig. 40.

To implement this in a cascade of second-order filters, however, the process described in section 5.2 is used: the numerator polynomials are factored, complex conjugate pole and zero pairs are combined into second-order filter coefficients, and any real poles and zeros may also be combined. This example has the following zeros:

-1.00000000000000

0.99999999999696

0.98594396121424 + 0.02327116735616i

0.98594396121424 - 0.02327116735616i

0.61517698021458 + 0.26465822467728i

0.61517698021458 - 0.26465822467728I

The poles are shown below:

0.98958247534274

0.49314542603125

0.97223882887180 + 0.02701103189409i

0.97223882887180 - 0.02701103189409i

0.72712179312558 + 0.21296904245800i

0.72712179312558 - 0.21296904245800I

Combining these poles and zeros, in the order shown, and scaling the numerator of the first filter with the b_0 coefficient of the the sixth-order filter (0.28108973410751 for this example), the following filter coefficients result:

B=[0.28108973410751 0.00000000000085 -0.28108973410666]

A=[1.0000000000000 -1.48272790137399 0.48800807139595]

B=[1.00000000000000 -1.97188792242849 0.97262704188495]

A=[1.0000000000000 -1.94447765774359 0.94597793620998]

B=[1.00000000000000 -1.23035396042916 0.44848669287526]

A=[1.0000000000000 -1.45424358625116 0.57406191508364]

The magnitude responses of these three filters are shown in Fig. 43. The composite magnitude response achieved upon cascading these three filters is identically that of Fig. 42.

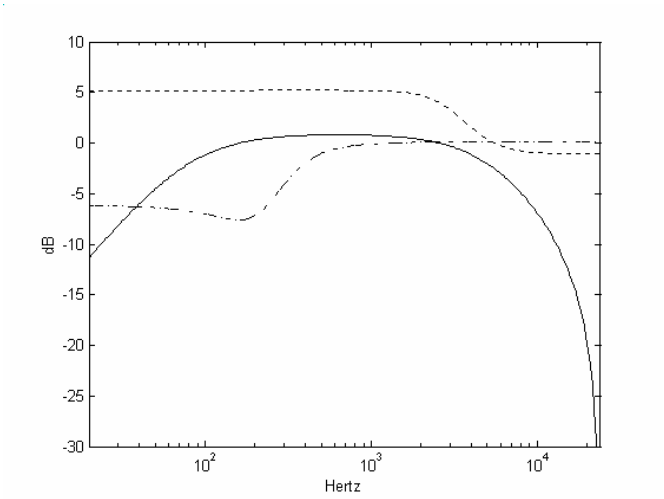

Fig. 43. Magnitude Responses of the Three Cascade Filters

5 PLOTTING THE MAGNITUDE RESPONSE OF SECOND-ORDER IIR FILTERS

A second-order digital filter has the transfer function shown in Eq. (1). The magnitude response, as a function of frequency, admits the following functional form:

$$\left|H(\omega)\right| = \left|H(z)\right|_{z=e^{i\omega}} = \left|\frac{b_0 + b_1 e^{-iw} + b_2 e^{-i2\omega}}{1 + a_1 e^{-i\omega} + a_2 e^{-i2\omega}}\right| \quad (41)$$

where $\omega = 2\pi f/Fs$, Fs = sampling frequency.

Typically the magnitude response will be plotted on a log-frequency axis, so only a few hundred frequency points need to be computed. The log spacing can be computed as follows:

$$M = 10^{\frac{\log_{10}(F_2)-\log_{10}(F_1)}{N-1}} \quad (42)$$

Where F_1 is the beginning frequency point; F_2 is the ending frequency point, and N is the desired number of frequency points.

For example, to plot 300 points between 10 Hz and 22050 Hz, a multiplier of

$$M = 10^{\frac{\log_{10}(22050)-\log_{10}(10)}{299}} = 1.026082$$

is used resulting in frequency values of 10, 10.26082, 10.52844, 10.80304, 11.0848, 11.37391, …, 19892.1, 20410.93, 20943.28, 21489.52, 22050.

Now, all that remains is to compute the frequency values. Typically, these are plotted in dB, so the computation for the first point, assuming a sample rate of 44.1 kHz is as follows:

$$20\log_{10}\left|H(\frac{2\pi 10}{44100})\right| = 20\log_{10}\left|\frac{b_0 + b_1 e^{-i\frac{2\pi 10}{44100}} + b_2 e^{-i\frac{4\pi 10}{44100}}}{1 + a_1 e^{-i\frac{2\pi 10}{44100}} + a_2 e^{-i\frac{4\pi 10}{44100}}}\right| \quad (43)$$

Using a math library that can deal with complex exponentials, substitute the coefficients into Eq. (43), compute the response for each of the frequency points, and plot. If complex exponentials are not available, the equation can be rewritten as follows. This should be computable using most math libraries:

$$\left|H(\omega)\right| = \left|\frac{b_0 + b_1 e^{-iw} + b_2 e^{-i2\omega}}{1 + a_1 e^{-i\omega} + a_2 e^{-i2\omega}}\right| = \quad (44)$$

$$\left|\frac{b_0 + b_1\left[\cos(\omega) - i\sin(\omega)\right] + b_2\left[\cos(2\omega) - i\sin(2\omega)\right]}{1 + a_1\left[\cos(\omega) - i\sin(\omega)\right] + a_2\left[\cos(2\omega) - i\sin(2\omega)\right]}\right|$$

Note: if necessary the following definition can be used after substituting in the various parameters and simplifying:

$$\left|\frac{\alpha + i\beta}{\gamma + i\delta}\right| = \sqrt{\frac{\alpha^2 + \beta^2}{\gamma^2 + \delta^2}} \quad (45)$$

Although Matlab® has a resident tool, the FREQZ command, for deriving this information, this paper includes a Matlab® function written so as to not require that command. This function can be used as pseudocode for implementations in languages where filtering tools are not available. By way of testing its functionality, it can be used to plot any of the second-order filters presented in this paper.

6 SUMMARY

This paper is meant as a companion to the digital audio systems designer. It covers the information needed to design and implement any IIR filter as a cascade of second-order IIR filter sections. It presents the equations needed to compute many filters, and presents them in a straightforward format, using familiar parameters, and without requiring transformations to derive the actual coefficients for implementation.

7 ACKNOWLEDGEMENT

The author is grateful to the McDonald's and Jack-in-the-Box restaurants near his home in Plano, Texas for making their facilities available during the preparation of this paper.

8 REFERENCES

[1] P. H. Bauer and L.-J. Leclerc, "A Computer-aided Test for the Absence of Limit Cycles in Fixed-point Digital Filters," *IEEE Trans. Signal Proc.*, vol. 39, (1991 Nov.).

[2] E. C. Ifeachor and B. W. Jervis, *Digital Signal Processing: A Practical Approach*, (Addison-Wesley, 1993).

[3] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, (Prentice-Hall, 1975).

[4] J. G. Proakis and D. G. Manolakis, *Introduction to Digital Signal Processing*, (Macmillan, 1988).

[5] Y. Hirata, "Digitalization of Conventional Analog Filters for Recording Use". *J. Audio Eng. Soc.*, vol. 29, pp. 333-337, (1981).

[6] P. A. Regalia and S. K. Mitra, "Tuneable Digital Frequency Response Equalization Filters". *IEEE Tran. Acoust., Speech, and Signal Proc.*, vol. 35, pp. 118-120, (1987 Jan.)

[7] U. Zolzer and T. Boltze, "Parametric Digital Filter Structures". 99th Convention Audio Engineering Society, (1995 Oct.).

[8] P. A. Regalia et al., "The Digital Allpass Filter: a Versatile Signal Processing Building Block," *Proc. IEEE*, vol. 76, pp. 19-37, (1988 Jan.).

[9] A. N. Willson, Jr. and H. Orchard, "Insights into Digital Filters Made as the Sum of Two Allpass Functions," *IEEE Trans. Circuits Sys.*, vol. 42, pp. 129-137, (1995 Mar.).

[10] L. Gazsi, "Explicit Formulas for Lattice Wave Digital Filters," *IEEE Trans. Circuits Sys*., vol. 32, pp. 68-88, (1985).

[11] R. Ansari and B. Liu, "A Class of Low-Noise Computationally Efficient Recursive Digital Filters with Applications to Sampling Rate Alterations," *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. 33, pp. 90-97, (1985 Feb.).

[12] R. Bristow-Johnson, "The Equivalence of Various Methods of Computing Biquad Coefficients for Audio Parametric Equalizers," 97th Convention Audio Engineering Society, (1994 Nov.), preprint 3906.

[13] D. Zaucha "Importance of Precision on Performance for Digital Audio Filters". *112th Convention Audio Engineering Society*. Munich, (2002 May).

[14] R. J. Clark et al., "Techniques for Generating Digital Equalizer Coefficients," *J. Audio Eng. Soc.*, vol. 48, pp. 281-298, (2000).

[15] S. J. Orfanidis "Digital Parametric Equalizer Design with Prescribed Nyquist-Frequency Gain," 101st Convention Audio Engineering Society, (1996 Nov.), preprint 4361.

[16] J. McNally, *Digital Audio: Recursive Digital Filtering for High Quality Audio Signals*. BBC Research Department Report, 1981/10, (1981).

[17] S. A. White, "Design of a Digital Biquadratic Peaking or Notch Filter for Digital Audio Equalization," *J. Audio Eng. Soc.*, vol. 34, pp. 479-, (1986).

[18] D. J. Shpak, "Analytical Design of Biquadradic Filter Sections for Parametric Filters," *J. Audio Eng. Soc*., vol. 40, pp. 876-885, (1992 Nov.).

[19] S. P. Lipshitz and J. Vanderkooy, "A Family of Linear-Phase Crossover Networks of High Slope Derived by Time Delay". *J. Audio Eng. Soc.*, vol. 31, pp. 374-392, (1983).

[20] T. W. Parks and C. S. Burrus, *Digital Filter Design*, (Wiley, 1987).

[21] R. W. Hamming, *Digital Filters*, (Prentice-Hall, 1977).

[22] J. A. Moorer, "The Manifold Joys of Conformal Mapping". *J. Audio Eng. Soc.*, vol. 31, pp. 826-841, (1983).

APPENDIX - MATLAB® FUNCTIONS

(Electronic versions (*.m files) available from the author.)

First-Order Allpass Filters

```
function [B, A] = FO_allpass(fc, Fs)
alpha = (tan(pi*fc/Fs) - 1) /
(tan(pi*fc/Fs) + 1);
B = [alpha 1 0];
A = [1 alpha 0];
```

Second-Order Allpass Filters

```
function [B, A] = SO_allpass(b, fc, Fs)
a = (tan(pi*b/Fs) - 1) / (tan(pi*b/Fs)
+ 1);
d = -cos(2*pi*fc/Fs);
B = [-a d*(1 - a) 1];
A = [1 d*(1 - a) -a];
```

Second-Order Equalization Filters

```
function [B, A] = SO_EQ(g, b, fc, Fs)
if g < 1
   a = (tan(pi*b/Fs)-
g)/(tan(pi*b/Fs)+g);
else
   a = (tan(pi*b/Fs)-
1)/(tan(pi*b/Fs)+1);
end
H = g - 1;
d = -cos(2*pi*fc/Fs);
b0 = 1 + (1+a)*H/2;
b1 = d*(1-a);
a1 = b1;
b2 = -a-(1+a)*H/2;
a2 = -a;

B = [b0 b1 b2];
A = [1 a1 a2];
```

First-Order Butterworth Filters

```
function [B, A] = fo_butter(fc, Fs, HL)
wc = 2*pi*fc;
k = 2*pi*fc/tan(pi*fc/Fs);
if HL(1:3) == 'low'
    b0 = wc/(k+wc);
    b1 = wc/(k+wc);
else
    b0 = k/(k+wc);
    b1 = -k/(k+wc);
end
a1 = (wc-k)/(k+wc);
B = [b0 b1 0]
A = [1 a1 0]
```

Second-Order Butterworth Filters

```
function [B, A] = so_butter(fc, Fs, HL)
wc = 2*pi*fc;
k = 2*pi*fc/tan(pi*fc/Fs);
if HL(1:3) == 'low'
   b0 = wc^2/(wc^2+k^2+sqrt(2)*wc*k);
   b1 = 2*wc^2/(wc^2+k^2+sqrt(2)*wc*k);
   b2 = wc^2/(wc^2+k^2+sqrt(2)*wc*k);
else
   b0 = k^2/(wc^2+k^2+sqrt(2)*wc*k);
   b1 = -2*k^2/(wc^2+k^2+sqrt(2)*wc*k);
   b2 = k^2/(wc^2+k^2+sqrt(2)*wc*k);
end
a1 = (2*wc^2-
2*k^2)/(wc^2+k^2+sqrt(2)*wc*k);
a2 = (wc^2+k^2-
sqrt(2)*wc*k)/(wc^2+k^2+sqrt(2)*wc*k);
B = [b0 b1 b2];
A = [1 a1 a2];
```

Third-Order Butterworth Filters

```
function [B, A] = to_butter(fc, Fs, HL)
wc = 2*pi*fc;
k = 2*pi*fc/tan(pi*fc/Fs);
if HL(1:3) == 'low'
b0=wc^3/(wc^3+k^3+2*wc^2*k+2*wc*k^2);
b1=3*wc^3/(wc^3+k^3+2*wc^2*k+2*wc*k^2);
b2=3*wc^3/(wc^3+k^3+2*wc^2*k+2*wc*k^2);
b3=wc^3/(wc^3+k^3+2*wc^2*k+2*wc*k^2);
else
b0=k^3/(wc^3+k^3+2*wc^2*k+2*wc*k^2);
b1=-3*k^3/(wc^3+k^3+2*wc^2*k+2*wc*k^2);
b2=3*k^3/(wc^3+k^3+2*wc^2*k+2*wc*k^2);
b3=-k^3/(wc^3+k^3+2*wc^2*k+2*wc*k^2);
```

```
end

a1=(3*wc^3-3*k^3+2*wc^2*k-2*wc*k^2)
/(wc^3+k^3+2*wc^2*k+2*wc*k^2);

a2=(3*wc^3+3*k^3-2*wc^2*k-2*wc*k^2)
/(wc^3+k^3+2*wc^2*k+2*wc*k^2);

a3=(wc^3-k^3-2*wc^2*k+2*wc*k^2)
/(wc^3+k^3+2*wc^2*k+2*wc*k^2);

B = [b0 b1 b2 b3];
A = [1 a1 a2 a3];
```

Fourth-Order Butterworth Filters

```
function [B, A] = four_o_butter(fc, Fs,
HL)

wc = 2*pi*fc;

a = 2*(cos(pi/8)+cos(3*pi/8));

b = 2*(1+2*cos(pi/8)*cos(3*pi/8));

k = 2*pi*fc/tan(pi*fc/Fs);

if HL(1:3) == 'low'

   b0 = wc^4/(k^4 + a*wc*k^3 +
b*wc^2*k^2 + a*wc^3*k + wc^4);

   b1 = 4*wc^4/(k^4 + a*wc*k^3 +
b*wc^2*k^2 + a*wc^3*k + wc^4);

   b2 = 6*wc^4/(k^4 + a*wc*k^3 +
b*wc^2*k^2 + a*wc^3*k + wc^4);

   b3 = 4*wc^4/(k^4 + a*wc*k^3 +
b*wc^2*k^2 + a*wc^3*k + wc^4);

   b4 = wc^4/(k^4 + a*wc*k^3 +
b*wc^2*k^2 + a*wc^3*k + wc^4);

else

   b0 = k^4/(k^4 + a*wc*k^3 +
b*wc^2*k^2 + a*wc^3*k + wc^4);

   b1 = -4*k^4/(k^4 + a*wc*k^3 +
b*wc^2*k^2 + a*wc^3*k + wc^4);

   b2 = 6*k^4/(k^4 + a*wc*k^3 +
b*wc^2*k^2 + a*wc^3*k + wc^4);

   b3 = -4*k^4/(k^4 + a*wc*k^3 +
b*wc^2*k^2 + a*wc^3*k + wc^4);

   b4 = k^4/(k^4 + a*wc*k^3 +
b*wc^2*k^2 + a*wc^3*k + wc^4);

end

a1 = (-4*k^4 - 2*a*wc*k^3 + 2*a*wc^3*k
+ 4*wc^4)/(k^4 + a*wc*k^3 + b*wc^2*k^2
+ a*wc^3*k + wc^4);

a2 = (6*k^4 - 2*b*wc^2*k^2 +
6*wc^4)/(k^4 + a*wc*k^3 + b*wc^2*k^2 +
a*wc^3*k + wc^4);

a3 = (-4*k^4 + 2*a*wc*k^3 - 2*a*wc^3*k
+ 4*wc^4)/(k^4 + a*wc*k^3 + b*wc^2*k^2
+ a*wc^3*k + wc^4);

a4 = (k^4 - a*wc*k^3 + b*wc^2*k^2 -
a*wc^3*k + wc^4)/(k^4 + a*wc*k^3 +
b*wc^2*k^2 + a*wc^3*k + wc^4);

B = [b0 b1 b2 b3 b4];
A = [1 a1 a2 a3 a4];
```

Second-Order Linkwitz-Riley Filters

```
function [B, A] = so_LR(fc, Fs, HL)

wc = 2*pi*fc;

k = 2*pi*fc/tan(pi*fc/Fs);

if HL(1:3) == 'low'

    b0 = wc^2/(k^2+wc^2+2*wc*k);

    b1 = 2*wc^2/(k^2+wc^2+2*wc*k);

    b2 = wc^2/(k^2+wc^2+2*wc*k);

else

    b0 = k^2/(k^2+wc^2+2*wc*k);

    b1 = -2*k^2/(k^2+wc^2+2*wc*k);

    b2 = k^2/(k^2+wc^2+2*wc*k);

end

a1 = (-2*k^2+2*wc^2)/(k^2+wc^2+2*wc*k);

a2=(-
2*wc*k+k^2+wc^2)/(k^2+wc^2+2*wc*k);

B = [b0 b1 b2];
A = [1 a1 a2];
```

Fourth-Order Linkwitz-Riley Filters

```
function [B, A] = four_LR(fc, Fs, HL)

wc = 2*pi*fc;

k = 2*pi*fc/tan(pi*fc/Fs);

if HL(1:3) == 'low'

    b0 = wc^4/(4*wc^2*k^2+2*sqrt(2)*
wc^3*k+k^4+2*sqrt(2)*wc*k^3+wc^4);

    b1 = 4*wc^4/(4*wc^2*k^2+2*sqrt(2)*
wc^3*k+k^4+2*sqrt(2)*wc*k^3+wc^4);

    b2 = 6*wc^4/(4*wc^2*k^2+2*sqrt(2)*
wc^3*k+k^4+2*sqrt(2)*wc*k^3+wc^4);

    b3 = 4*wc^4/(4*wc^2*k^2+2*sqrt(2)*
```

```
wc^3*k+k^4+2*sqrt(2)*wc*k^3+wc^4);
    b4 = wc^4/(4*wc^2*k^2+2*sqrt(2)*
wc^3*k+k^4+2*sqrt(2)*wc*k^3+wc^4);
else
    b0 = k^4/(4*wc^2*k^2+2*sqrt(2)*
wc^3*k+k^4+2*sqrt(2)*wc*k^3+wc^4);
    b1 = -4*k^4/(4*wc^2*k^2+2*sqrt(2)*
wc^3*k+k^4+2*sqrt(2)*wc*k^3+wc^4);
    b2 = 6*k^4/(4*wc^2*k^2+2*sqrt(2)*
wc^3*k+k^4+2*sqrt(2)*wc*k^3+wc^4);
    b3 = -4*k^4/(4*wc^2*k^2+2*sqrt(2)*
wc^3*k+k^4+2*sqrt(2)*wc*k^3+wc^4);
    b4 = k^4/(4*wc^2*k^2+2*sqrt(2)*
wc^3*k+k^4+2*sqrt(2)*wc*k^3+wc^4);
end
a1 = (4*wc^4+4*sqrt(2)*wc^3*k-4*k^4-
4*sqrt(2)*wc*k^3)/(4*wc^2*k^2+2*sqrt(2)
*wc^3*k+k^4+2*sqrt(2)*wc*k^3+wc^4);
a2 = (6*wc^4-8*wc^2*k^2+6*k^4)/(4*wc^2*
k^2+2*sqrt(2)*wc^3*k+k^4+2*sqrt(2)*wc*k
^3+wc^4);
a3 = (-
4*sqrt(2)*wc^3*k+4*wc^4+4*sqrt(2)*
wc*k^3-
4*k^4)/(4*wc^2*k^2+2*sqrt(2)*wc^3*
k+k^4+2*sqrt(2)*wc*k^3+wc^4);
a4 = (k^4-2*sqrt(2)*wc^3*k+wc^4-
2*sqrt(2)*
wc*k^3+4*wc^2*k^2)/(4*wc^2*k^2+2*sqrt(2
)*wc^3*k+k^4+2*sqrt(2)*wc*k^3+wc^4);
B = [b0 b1 b2 b3 b4];
A = [1 a1 a2 a3 a4];
```

First-order Shelf

```
function [B, A] = fo_shelf(g, fc, Fs,
bass)
if g > 1
    a = (tan(pi*fc/Fs) -
1)/(tan(pi*fc/Fs) + 1);
    if bass
        b0 = 1 + (1 + a)*(g-1)/2;
        b1 = a + (1 + a)*(g-1)/2;
    else
        b0 = 1 + (1 - a)*(g-1)/2;
        b1 = a + (a - 1)*(g-1)/2;
    end
else
    if bass
        a = (tan(pi*fc/Fs) -
g)/(tan(pi*fc/Fs) + g);
        b0 = 1 + (1 + a)*(g-1)/2;
        b1 = a + (1 + a)*(g-1)/2;
    else
        a=(g*tan(pi*fc/Fs)-
)/(g*tan(pi*fc/Fs) + 1);
        b0 = 1 + (1 - a)*(g-1)/2;
        b1 = a + (a - 1)*(g-1)/2;
  end
end
B = [b0 b1 0];
A = [1 a 0];
```

Second-order Shelf

```
function [B, A] = so_shelf(fc, Fs, g,
bass)
s = sqrt(2)/2;
rho = pi/2;
phi = (fc/Fs)*pi;
if g > 0.5 & g < 2
   F = sqrt(g);
elseif g > 1
   F = g / sqrt(2);
else
   F = g * sqrt(2);
end
gd = ((F^2 - 1)/(g^2 - F^2))^0.25;
gn = sqrt(g) * gd;
a = tan(pi*(fc/Fs - 1/4));

if bass
    b0 = -(-1-gn^2*a^2-a^2-2*gn^2*a-
gn^2-2*s*gn+2*s*gn*a^2+2*a)/(2*s*gd+1-
```

```
2*s*gd*a^2+gd^2*a^2+2*gd^2*a+a^2+gd^2-
2*a);

    b1 = -(2-4*a-4*gn^2*a-2*gn^2*a^2-
2*gn^2+2*a^2)/(2*s*gd+1-
2*s*gd*a^2+gd^2*a^2+2*gd^2*a+a^2+gd^2-
2*a);

    b2 = (1+2*s*gn*a^2-2*a+gn^2-
2*s*gn+2*gn^2*a+a^2+gn^2*a^2)/(2*s*gd+1
-2*s*gd*a^2+gd^2*a^2+2*gd^2*a+a^2+gd^2-
2*a);

    a0 = 1;

    a1 = (-2+2*gd^2*a^2+4*gd^2*a-
2*a^2+2*gd^2+4*a)/(2*s*gd+1-
2*s*gd*a^2+gd^2*a^2+2*gd^2*a+a^2+gd^2-
2*a);

    a2 = (gd^2*a^2-2*a+1+2*gd^2*a-
2*s*gd+a^2+2*s*gd*a^2+gd^2)/(2*s*gd+1-
2*s*gd*a^2+gd^2*a^2+2*gd^2*a+a^2+gd^2-
2*a);

else

    b0 = (gn^2*a^2+2*s*gn-2*gn^2*a+1-
2*s*gn*a^2+a^2+gn^2+2*a)/(1+gd^2+2*s*gd
-2*s*gd*a^2+gd^2*a^2-2*gd^2*a+a^2+2*a);

    b1 = (2-2*gn^2*a^2+4*gn^2*a+4*a-
2*gn^2+2*a^2)/(1+gd^2+2*s*gd-
2*s*gd*a^2+gd^2*a^2-2*gd^2*a+a^2+2*a);

    b2 = (1+2*s*gn*a^2-
2*s*gn+2*a+a^2+gn^2-
2*gn^2*a+gn^2*a^2)/(1+gd^2+2*s*gd-
2*s*gd*a^2+gd^2*a^2-2*gd^2*a+a^2+2*a);

    a0 = 1;

    a1 = (2-2*gd^2*a^2+4*gd^2*a+2*a^2-
2*gd^2+4*a)/(1+gd^2+2*s*gd-
2*s*gd*a^2+gd^2*a^2-2*gd^2*a+a^2+2*a);

    a2 = (1-2*gd^2*a+2*a+gd^2-
2*s*gd+a^2+gd^2*a^2+2*s*gd*a^2)/(1+gd^2
+2*s*gd-2*s*gd*a^2+gd^2*a^2-
2*gd^2*a+a^2+2*a);

end

B = [b0 b1 b2];
A = [a0 a1 a2];
```

Magnitude Plotting

```
function filt_plot(B, A, beg_freq,
end_freq, num_pts, Fs)
b0 = B(1);
b1 = B(2);
b2 = B(3);
a1 = A(2);
a2 = A(3);
F(1) = beg_freq;


%First magntidue point

H(1) =
20*log10(abs((b0+b1*(cos(2*pi*F(1)/Fs)-
i*sin(2*pi*F(1)/Fs))+b2*(cos(4*pi*F(1)/
Fs)-
i*sin(4*pi*F(1)/Fs)))/(1+a1*(cos(2*pi*F
(1)/Fs)-
i*sin(2*pi*F(1)/Fs))+a2*(cos(4*pi*F(1)/
Fs)-i*sin(4*pi*F(1)/Fs)))));


%Frequency multiplier

M = 10^((log10(end_freq)-
log10(beg_freq))/(num_pts-1));


for j = 2:num_pts

   %Update frequency

   F(j) = F(j-1)*M;

   %Magnitude points

   H(j) =
20*log10(abs((b0+b1*(cos(2*pi*F(j)/Fs)-
i*sin(2*pi*F(j)/Fs))+b2*(cos(4*pi*F(j)/
Fs)-
i*sin(4*pi*F(j)/Fs)))/(1+a1*(cos(2*pi*F
(j)/Fs)-
i*sin(2*pi*F(j)/Fs))+a2*(cos(4*pi*F(j)/
Fs)-i*sin(4*pi*F(j)/Fs)))));

end


%Plot on log-frequency axis

semilogx(F, H)
```