Title:	LPF Designer Documentation
Author:	J.A. Crawford
Abstract:	Design details for the LPF Designer are provided herein.
Date:	24 April 2015
Version:	1.30

1 Overview

A variety of lumped-element LC lowpass filter families are considered in this monograph including

- Butterworth
- Chebyshev
- Inverse Chebyshev
- Gaussian to 6 dB and 12 dB, including adjustable Gaussian
- Bessel
- Linear Phase to 0.5^o and 0.05^o
- Transitional Filters
- Elliptical

Detailed design information is developed for each filter type. The balance of this section looks at several filter design fundamentals.

1.1 General Approach

The approach followed herein begins with posing the lowpass filter design problem initially as an approximation problem based upon filter poles and zeros in the complex plane. The poles and zeros may correspond to a classical filter type like those listed in the previous section or may stem from manual efforts to meet requirements posed in terms of the desired attenuation characteristic and or the filters' group delay characteristics.

Once the approximation problem solution has been obtained in terms of poles and zeros, the synthesis step may begin. While closed-form network solutions exist for a number of the filter types listed earlier, in general these are only available for equally-terminated filters or unloaded filters. On occasion it is advantageous to have a design approach that is easily amendable to the general unequally-terminated filter case. Although polynomial calculations can be done in the spirit of Darlington's approach to filter synthesis, an iterative numerical method[1] is used here because of its general applicability as well as the ease with which it accommodates redundant circuit elements.

The remainder of this section revisits lossless filter theory in the context of using *ABCD* matrix descriptions. This information is crucial for the polynomial-based filter design method of Darlington, but is not required to use the iterative synthesis approach. It is nevertheless included here for completeness.

This section concludes with a brief look at the pole-zero formulation and the impact of differing load resistances upon the Darlington synthesis method.

1.2 Lossless Two-Port Filter Design

Consider the linear two-port network in Figure 1 represented by its *ABCD* matrix description as [15]

$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} V_2 \\ I_2 \end{bmatrix} \tag{2.1}$$

[1] Motivated by [20].

The *transducer gain function* $| \, T(s) \, |^2$ is defined as the ratio of the maximum power available from the generator to the actual power delivered to the load R_2 in Figure 1. As such, the maximum power available from the source is given by

$$P_{Avail} = \frac{1}{R_1}\left(\frac{E}{2}\right)^2 = \frac{E^2}{4R_1} \tag{2.2}$$

where E is the amplitude of the applied input signal which is taken to be $E \exp(j\omega t)$. Similarly, the power delivered to the load R_2 is given by

$$P_{Load} = \frac{|V_2|^2}{R_2} \tag{2.3}$$

Consequently,

$$|T(s)|^2 = \frac{E^2}{4R_1}\frac{R_2}{|V_2|^2} = \frac{R_2}{4R_1}\left|\frac{E}{V_2}\right|^2 \tag{2.4}$$

leading to

$$T(s) = \frac{E}{2V_2}\sqrt{\frac{R_2}{R_1}} \tag{2.5}$$

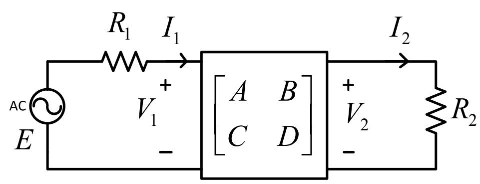


Figure 1 *ABCD* network description

For a purely reactive network, *A* and *D* must be even functions of *s* while *B* and *C* must be odd functions of *s*. From (2.1),

$$\begin{bmatrix} V_2 \\ I_2 \end{bmatrix} = \frac{\begin{bmatrix} D & -B \\ -C & A \end{bmatrix}}{AD - BC} \tag{2.6}$$

For a reciprocal network[2], the determinant of the ABCD matrix ($AD - BC$) must be unity.

From Figure 1, it is clear that

$$V_2 = I_2 R_2 \tag{2.7}$$

$$E = I_1 R_1 + V_1 \tag{2.8}$$

from which (2.5) can be re-expressed as

$$T(s) = \frac{(AR_2 + DR_1) + (B + CR_1R_2)}{2\sqrt{R_1 R_2}} \tag{2.9}$$

Based upon earlier remarks, $T(s)$ can be broken into distinct even and odd portions as

[2] A reciprocal network exhibits the same loss characteristics starting from either port.

$$T_e(s) = \frac{AR_2 + DR_1}{2\sqrt{R_1 R_2}}$$
$$T_o(s) = \frac{B + CR_1 R_2}{2\sqrt{R_1 R_2}} \quad (2.10)$$

In the context of the lossless network [14] shown in Figure 2,

$$|T|^2 = \frac{P_{avail}}{P_{del}} \quad (2.11)$$

The *characteristic function* $K(s)$ is defined as

$$K(s) = \rho_1(s) T(s) \quad (2.12)$$

where $\rho_1(s)$ is the reflection coefficient as viewed from port 1. For $s = j\omega$,

$$|K|^2 = |\rho_1|^2 |T|^2 \quad (2.13)$$

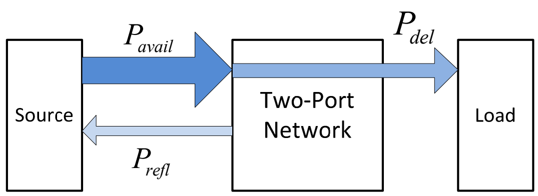


Figure 2 Power flow available, delivered to the load, and reflected back to the source from the lossless network

Substituting (2.11) into (2.13) produces

$$|K|^2 = |\rho_1|^2 \frac{P_{avail}}{P_{del}} = \frac{P_{refl}}{P_{del}} \quad (2.14)$$

Note that

$$1 + |K|^2 = 1 + \frac{P_{refl}}{P_{del}} = \frac{P_{del} + P_{refl}}{P_{del}} = \frac{P_{avail}}{P_{del}} = |T|^2$$
$$\therefore 1 + |K|^2 = |T|^2 \quad (2.15)$$

where the last equation is known as the famous *Feldtkeller equation* which is a statement of energy conservation for the lossless network (i.e., power must be either reflected back to the source or delivered to the load).

Now making use of (2.10) in (2.15),

$$\begin{aligned} |T|^2 = |T_e + T_o|^2 &= [T_e + T_o][T_e + T_o^*] \\ &= [T_e + T_o][T_e - T_o] \\ &= T_e^2 - T_o^2 \\ &= \frac{(AR_2 + DR_1)^2 - (B + CR_1 R_2)^2}{4R_1 R_2} \end{aligned} \quad (2.16)$$

Starting from (2.16), making use of (2.15) and the reciprocal network requirement *AD* – *BC* = 1,

$$\left|T\right|^2 = \frac{A^2R_2^2 + 2ADR_1R_2 + D^2R_1^2 - B^2 - 2BCR_1R_2 - C^2R_1^2R_2^2}{4R_1R_2} \tag{2.17}$$

$$\frac{4ADR_1R_2 - 4BCR_1R_2}{4R_1R_2} = 1 \tag{2.18}$$

Subtracting (2.18) from (2.17) produces

$$\begin{aligned}\left|T\right|^2 - 1 &= \frac{A^2R_2^2 + 2ADR_1R_2 + D^2R_1^2 - B^2 - 2BCR_1R_2 - C^2R_1^2R_2^2}{4R_1R_2} - \frac{4ADR_1R_2 - 4BCR_1R_2}{4R_1R_2} \\ &= \frac{A^2R_2^2 - 2ADR_1R_2 + D^2R_1^2 - B^2 + 2BCR_1R_2 - C^2R_1^2R_2^2}{4R_1R_2} \\ &= \frac{\left(AR_2 - DR_1\right)^2 - \left(B - CR_1R_2\right)^2}{4R_1R_2} = \left|K\right|^2\end{aligned} \tag{2.19}$$

Similarly then, the characteristic equation can be broken into its even and odd portions as

$$K_e\left(s\right) = \frac{AR_2 - DR_1}{2\sqrt{R_1R_2}} \qquad K_o\left(s\right) = \frac{B - CR_1R_2}{2\sqrt{R_1R_2}} \tag{2.20}$$

Using (2.10) and (2.20), it is also true that

$$\begin{aligned} A &= 2\sqrt{\frac{R_1}{R_2}}\left(T_e + K_e\right) \qquad & B &= \sqrt{R_1R_2}\left(T_o + K_o\right) \\ C &= \frac{1}{\sqrt{R_1R_2}}\left(T_o - K_o\right) \qquad & D &= 2\sqrt{\frac{R_2}{R_1}}\left(T_e - K_e\right)\end{aligned} \tag{2.21}$$

Many other relationships exist between the *ABCD*, admittance, and impedance parameters associated with the lossless network[3]. In the case of admittance parameters, for example,

$$\begin{aligned} y_{11} &= \frac{1}{R_1}\frac{T_e + K_e}{T_o - K_o} \\ y_{12} &= \frac{1}{\sqrt{R_1R_2}}\frac{1}{T_o - K_o} \\ y_{22} &= \frac{1}{R_2}\frac{T_e - K_e}{T_o - K_o}\end{aligned} \tag{2.22}$$

It will prove useful to have voltage-gain relationships in terms of the *ABCD* matrix for the iterative filter synthesis step. These details are left to §11.

[3] See chapter 6 of [14].

1.3 Poles and Zeros

The transducer gain function can be expressed in terms of its poles and zeros as

$$T(s) = t_0 \frac{\prod_n (s - t_n)}{\prod_m (s - p_m)} = \frac{E(s)}{P(s)} \tag{2.23}$$

where the zeros are represented by the t_n, the poles are represented by p_m, and t_0 is a real constant of proportionality. From the latter portion of (2.15), the characteristic function can be written in a similar manner as

$$K(s) = s_0 \frac{\prod_n (s - s_n)}{\prod_m (s - p_m)} = \frac{F(s)}{P(s)} \tag{2.24}$$

$K(s)$ and $T(s)$ must clearly have the same poles based upon (2.23) and (2.24). The t_n and s_n values must be real or conjugate imaginary pairs and lie in the left-half plane. The p_m are conjugate pairs and purely imaginary for a ladder-type filter.

Based upon $|T(s)|^2 = 1 + |K(s)|^2$ given earlier in (2.15), these last two equations make it possible to write

$$E(s)E(-s) = P(s)P(-s) + F(s)F(-s) \tag{2.25}$$

From (2.13),

$$|\rho_1|^2 = \left| \frac{Z_{in} - R_{source}}{Z_{in} + R_{source}} \right|^2 = \left| \frac{K}{T} \right|^2 = \frac{F(s)F(-s)}{E(s)E(-s)}$$

(2.26)

$$\rho_1 = \frac{Z_{in} - R_{source}}{Z_{in} + R_{source}} = \frac{F(s)}{E(s)}$$

From this, it easily follows

$$Z_{in} = R_{source} \frac{E(s) + F(s)}{E(s) - F(s)} \tag{2.27}$$

This driving point impedance can be used to synthesize the elliptic filter in terms of its constituent capacitor and inductor values.

When negative elements must be avoided, it is necessary to introduce additional attenuation poles at zero, infinity, or both [16].

Some prefer to initiate a design based upon the characteristic function $K(s)$ because there are almost no restrictions on the placement of its zeros. This allows the zeros to be placed in the passband region thereby making the passband response rather insensitive to element variations [16].

1.4 Arbitrary Load Impedance

In the context of Figure 1, the maximum power which can be delivered to the load is given by

$$P_{avail} = \left(\frac{E}{R_1 + R_2} \right)^2 R_2 \tag{2.28}$$

In the general case where the load is replaced by a general impedance $z = a + j\,b$ with $E \equiv 1$ and $R_1 = 1$, the power delivered is given by

$$P_{del} = i_2^2\, a = \frac{a}{\left(1+a\right)^2 + b^2} \tag{2.29}$$

and $P_{avail} = 1/4$. Continuing,

$$\frac{P_{del}}{P_{avail}} = \frac{4a}{\left(1+a\right)^2 + b^2} \tag{2.30}$$

Since the reflection coefficient Γ is given by

$$\Gamma = \frac{z-1}{z+1} = \frac{a+jb-1}{a+jb+1} \tag{2.31}$$

it is easy to show that

$$\frac{P_{del}}{P_{avail}} = 1 - \left|\Gamma\right|^2 \tag{2.32}$$

Continuing in this vein but with $R_2 \neq R_1$,

$$P_{avail} = \frac{4R_2}{\left(1+R_2\right)^2} \tag{2.33}$$

resulting in

$$\frac{P_{del}}{P_{avail}} = \frac{\left(1+R_2\right)^2}{4R_2}\left(1-\left|\Gamma\right|^2\right) \tag{2.34}$$

For this case then,

$$\frac{P_{del}}{P_{avail}} = \frac{1}{1+K\left(s\right)K\left(-s\right)} = \frac{1}{1+\dfrac{F\left(s\right)F\left(-s\right)}{P\left(s\right)P\left(-s\right)}} = \frac{\left(1+R_2\right)^2}{4R_2}\left(1-\left|\Gamma\right|^2\right) \tag{2.35}$$

which leads directly to

$$\frac{P\left(s\right)P\left(-s\right)\left[1-\dfrac{4R_2}{\left(1+R_2\right)^2}\right]+F\left(s\right)F\left(-s\right)}{P\left(s\right)P\left(-s\right)+F\left(s\right)F\left(-s\right)} = \left|\Gamma\left(s\right)\right|^2 = \frac{M\left(s\right)M\left(-s\right)}{E\left(s\right)E\left(-s\right)} \tag{2.36}$$

where a new polynomial $M\left(s\right)$ emerges. Following the same path as used with (2.26) and (2.27), (2.36) leads to a driving point impedance function given by

$$Z_{in} = \frac{E\left(s\right)-M\left(s\right)}{E\left(s\right)+M\left(s\right)} \tag{2.37}$$

where the function-zeros of $E(s)$, $F(s)$, and $P(s)$ were computed earlier. When $R_2 \neq R_1$, the zeros of $F(s)$ are effectively perturbed in (2.27) as given by the numerator portion of (2.36).

2 Butterworth Lowpass Filters

The Laplace domain voltage transfer function for a filter can be represented by

$$H(s)=\frac{V_{out}(s)}{V_{in}(s)}=\frac{N(s)}{D(s)} \tag{3.1}$$

where *N*( ) and *D*( ) are polynomials in the complex frequency variable $s = \sigma + j\omega$. The attenuation characteristic of the filter (in dB) can be written as

$$A(\omega)=10\log_{10}\left[\frac{1}{\left|H(\omega)\right|^2}\right]=10\log_{10}\left[L(\omega^2)\right] \tag{3.2}$$

The Butterworth filter is the most simple lowpass filter approximation to an ideal lowpass characteristic. It is frequently referred to as a maximally-flat filter because the attenuation characteristic has all of its derivatives with respect to ω equal to zero at DC. Writing the loss function as

$$L(\omega^2)=\sum_{k=0}^{N} B_k \omega^{2k} \tag{3.3}$$

setting *L*(0) = 1, and requiring all of the derivatives of $L(\omega^2)$ to be zero at DC requires all of the B_k to be zero except for the highest-order term. Consequently[4],

$$L_{Butterworth}(\omega^2)=1+\omega^{2N} \tag{3.4}$$

The solutions to (3.4) are given by the 2*N* roots of unity as

$$p_k^{2N}=-1=\left[\exp(-j\pi+j2\pi k)\right] \text{ for arbitrary integer } k \tag{3.5}$$

which leads to

$$p_k=\exp\left[j\frac{\pi(2k-1)}{2N}+j\frac{\pi}{2}\right] \quad \text{for } k\in\{1,2,...,N\} \tag{3.6}$$

The poles must all fall within the left-half portion of the *s*-plane which requires that $n \in \{1,2,\ldots,N\}$. (The additional j π / 2 is convenient to keep the index range for n as given.) Butterworth poles for even- and odd-order cases[5] are shown in Figure 3 and Figure 4.

The voltage transfer function is given by

$$H(s)=\prod_{k=1}^{N}\left(\frac{-p_k}{s-p_k}\right) \tag{3.7}$$

---

[4] The zeros of $L_{Butterworth}$ are the attenuation poles of the filter.

[5] From u18217_butterworth_poles.m.

If the passband loss at frequency f_{pass} is given by A_{pass} (dB) and the minimum stopband loss at f_{stop} is required to be A_{stop} (dB), the minimum order for the Butterworth filter is then

$$N_{\min} \geq \frac{1}{2}\frac{\log_{10}\left(\dfrac{10^{A_{stop}/10}-1}{10^{A_{pass}/10}-1}\right)}{\log_{10}\left(\dfrac{f_{stop}}{f_{pass}}\right)} \tag{3.8}$$

It is straight forward to show that the group delay for any all-pole filter like the Butterworth filter is given by

$$\tau_g(\omega) = -\sum_{k=1}^{N}\left[\frac{\sigma_k}{\sigma_k^2+(\omega-\omega_k)^2}\right] \tag{3.9}$$

Group delay can also be calculated using Hilbert transforms as discussed later in §14.

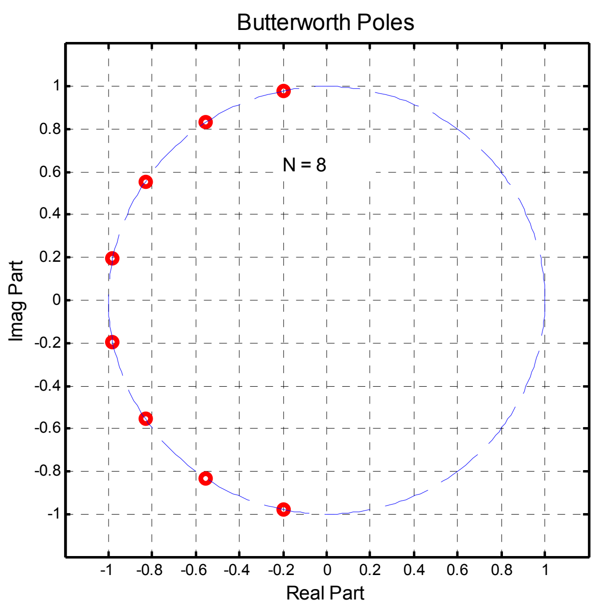


Figure 3 Normalized Butterworth poles for N = 8

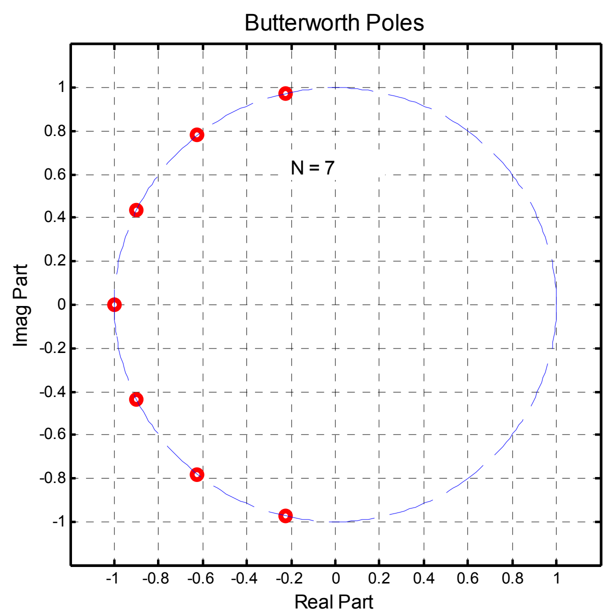


Figure 4 Normalized Butterworth poles for *N* = 7

Attenuation nomographs for the Butterworth lowpass filter family are provided in Figure 5 and Figure 6.

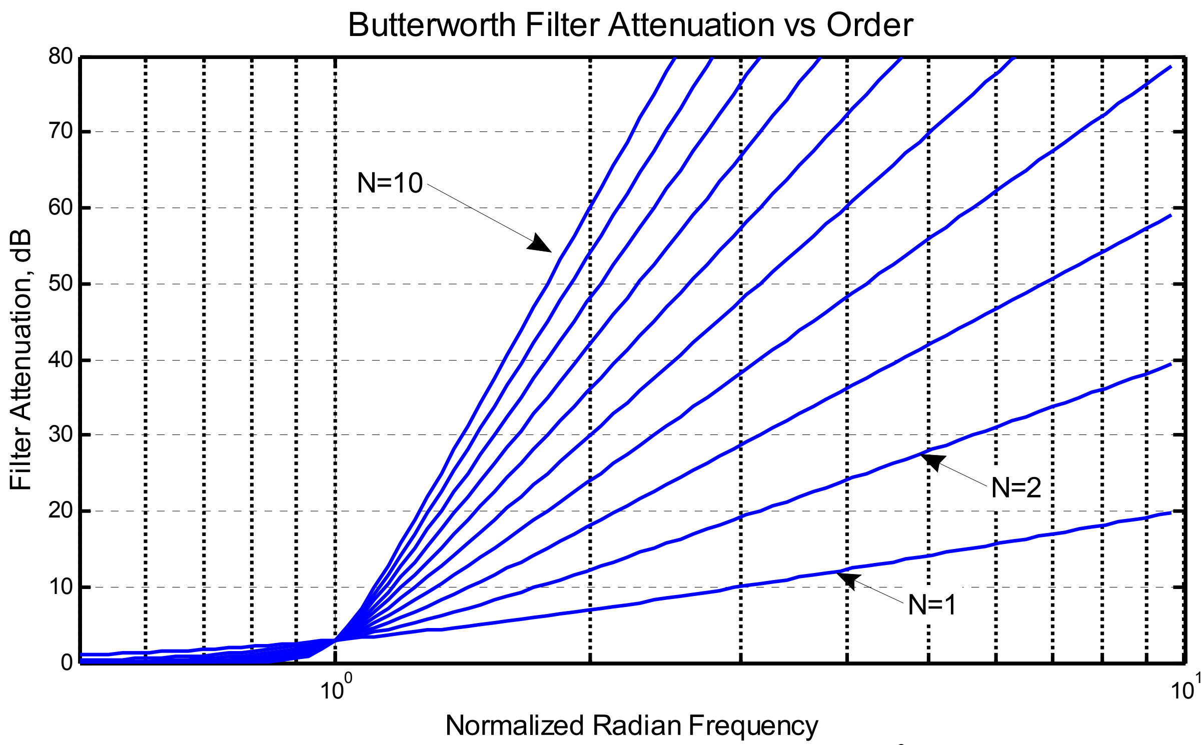


Figure 5 Butterworth filter attenuation versus normalized frequency and order[6]

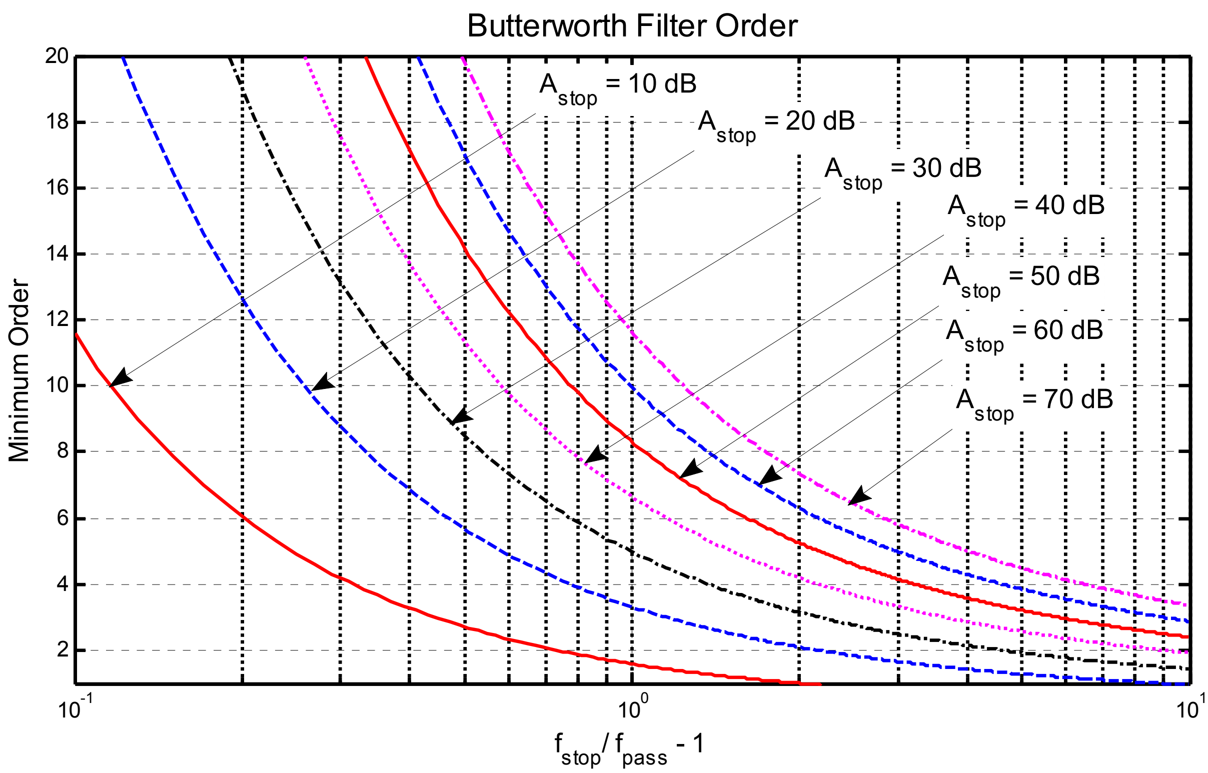


Figure 6 Butterworth filter nomograph for filter order versus stopband attenuation requirement[7]

[6] Computed using u18217_butterworth_poles.m.

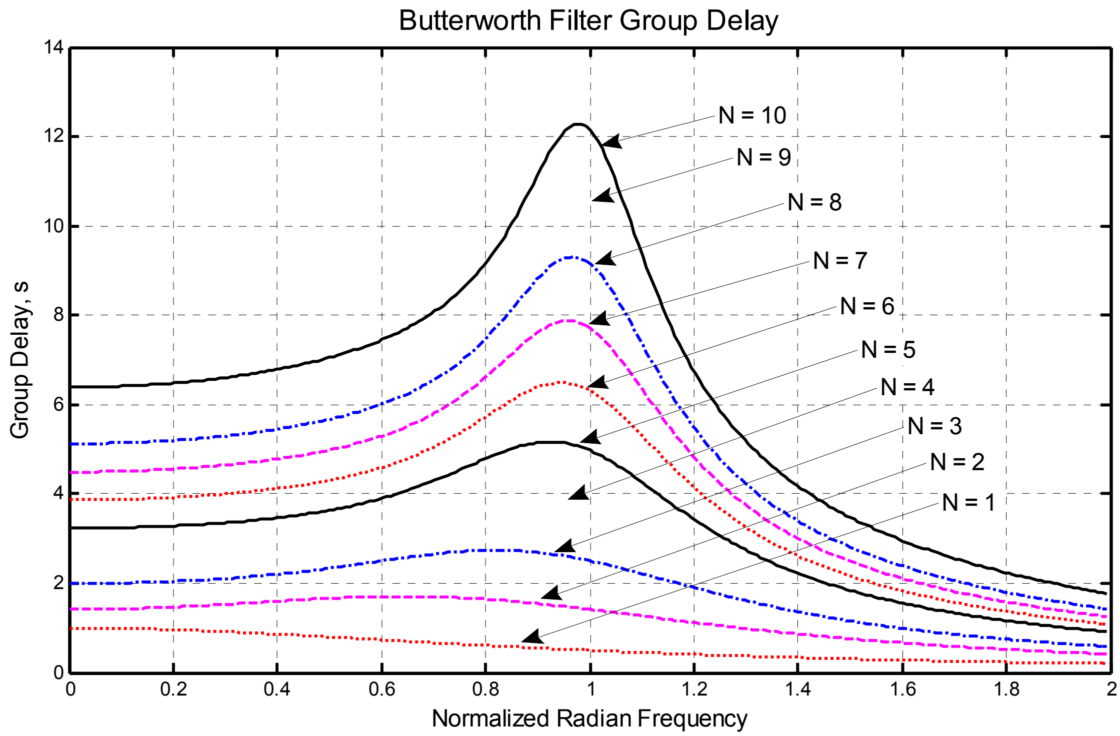


Figure 7 Butterworth filter group delay[8]

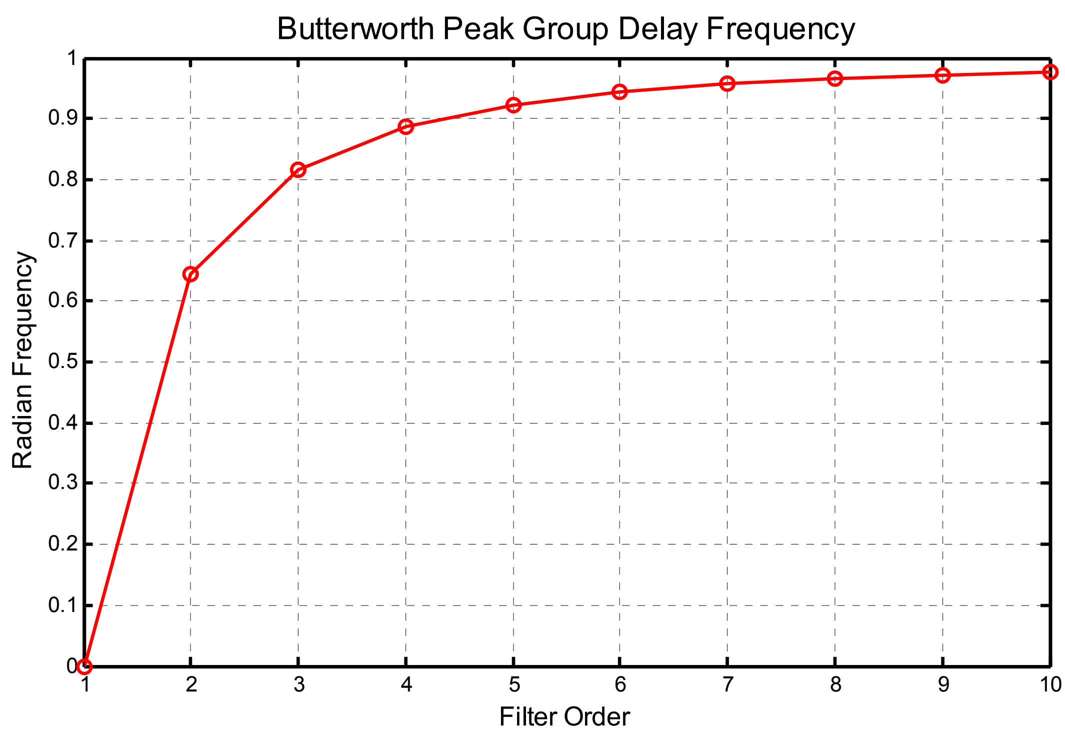


Figure 8 Radian frequency of peak group delay in Figure 7 versus Butterworth filter order[9]

[7] Computed using u18217_butterworth_poles.m.
[8] Ibid.

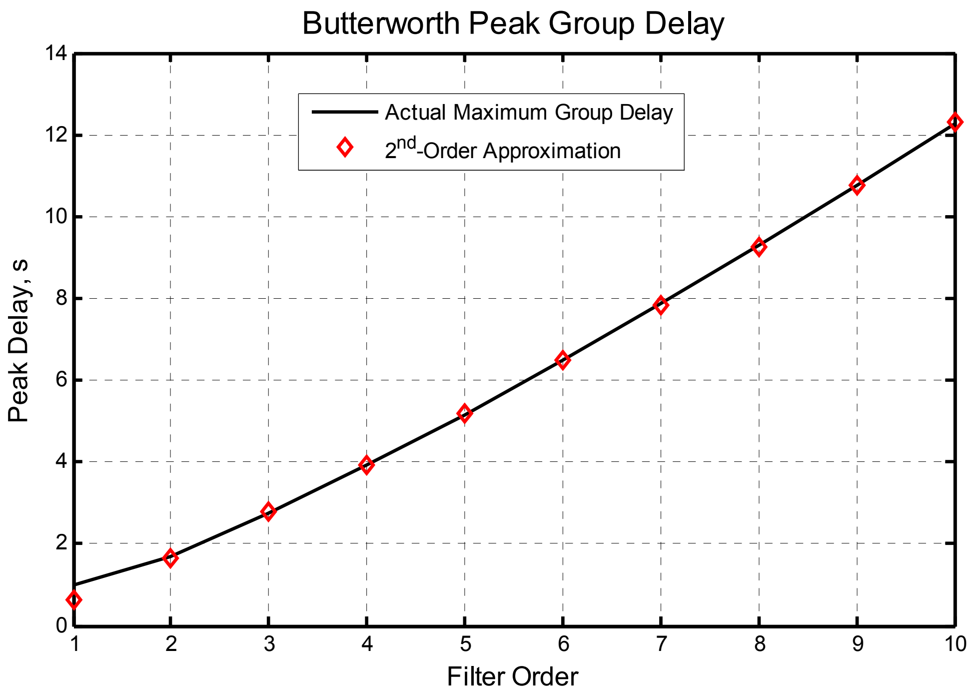


Figure 9 Peak group delay versus order for Butterworth filters

The group delay characteristics for normalized Butterworth filters are shown in Figure 7 through Figure 10. The frequency at which the maximum group delay occurs asymptotically approaches 1 rad/s as shown in Figure 8, with the corresponding peak-delay value as shown in Figure 9. The peak-delay value is closely approximated by the 2nd-order approximation

$$\tau_{\max}\left(N\right) = -0.3692 + 0.9529N + 0.0316N^2 \quad \text{sec} \tag{3.10}$$

whereas the Butterworth filter group delay at DC is closely approximated by

$$\tau_{DC}\left(N\right) = 0.1303 + 0.6245N \tag{3.11}$$

The Butterworth filter time-domain impulse response can be found directly from knowledge of the pole locations given by (3.6). The residue method is particularly easy to employ for all-pole filters like the Butterworth family because none of the poles are repeated and there are no transmission zeros. Once the transfer function (3.7) has been expanded into a sum of partial fractions as

$$H\left(f\right) = \sum_{k=1}^{N} C_k \exp\left(\sigma_k t\right) \exp\left(j\omega_k t\right) \tag{3.12}$$

where the poles $p_k = \sigma_k + j\omega_k$ and the C_k are given by

$$C_k = -p_k \prod_{\substack{n=1 \\ n\neq k}}^{N} \left(\frac{-p_n}{s-p_n}\right)\Bigg|_{s=p_k} \tag{3.13}$$

[9] Ibid.

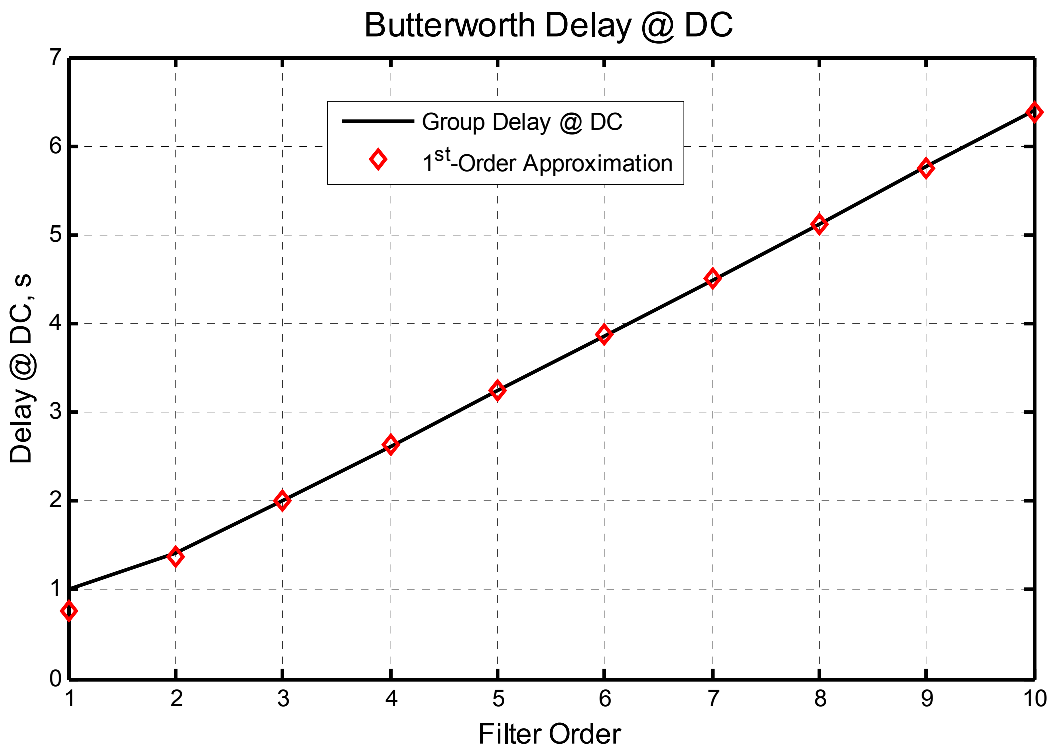


Figure 10 Group delay at DC for normalized Butterworth lowpass filters versus filter order

the corresponding time-domain response is given by

$$f(t) = \sum_{k=1}^{N} C_k \exp(p_k t) \quad \text{for } t \geq 0 \tag{3.14}$$

Since complex poles must appear in conjugate pairs, (3.14) can be simplified to

$$f(t) = \sum_{m} g_m(t) \tag{3.15}$$

where

$$g_m(t) = \begin{cases} e^{\sigma_m t}\left[2a_m \cos(\omega_m t) - 2b_m \sin(\omega_m t)\right] & \text{for complex pole, } \omega_m > 0 \\ a_m e^{\sigma_m t} & \text{for real pole} \end{cases} \tag{3.16}$$

with $C_m = a_m + j\, b_m$, $p_k = \sigma_k + j\, \omega_k$, and $m \in$ { poles with positive or zero values for ω_k }. Impulse responses for the first seven Butterworth order lowpass filters are provided in Table 1 and shown graphically in Figure 11.

Table 1 Butterworth (Normalized) Filter Impulse Responses[10]

Filter Order, *N*	Impulse Response
1	e^{-t}
2	$1.4142\, e^{-0.7071t} \sin(0.7071t)$
3	$e^{-t} - e^{-t/2}\cos(0.86603t) + 0.57735\, e^{-t/2}\sin(0.86603t)$
4	$0.92388\, e^{-0.92388t}\cos(1.1152t) - 2.2304\, e^{-0.92388t}\sin(1.1152t)$ $-0.92388\, e^{-0.38268t}\cos(0.19134t) - 0.38268\, e^{-0.38268t}\sin(0.19134t)$
5	$1.8944\, e^{-t} - 0.27639\, e^{-0.30902t}\cos(0.95106t) - 0.85065\, e^{-0.30902t}\sin(0.95106t)$ $-1.618\, e^{-0.80902t}\cos(0.58779t) + 2.227\, e^{-0.80902t}\sin(0.58779t)$
6	$0.40825\, e^{-0.25882t}\cos(0.96593t) - 0.70711\, e^{-0.25882t}\sin(0.96593t)$ $-3.0472\, e^{-0.70711t}\cos(0.70711t) + 2.639\, e^{-0.96593t}\cos(0.25882t)$ $4.5708\, e^{-0.96593t}\sin(0.25882t)$
7	$4.3119\, e^{-t} + 0.73698\, e^{-0.22252t}\cos(0.97493t) - 0.16821\, e^{-0.22252t}\sin(0.97493t)$ $-2.065\, e^{-0.62349t}\cos(0.78183t) - 2.5894\, e^{-0.62349t}\sin(0.78183t)$ $-2.984\, e^{-0.90097t}\cos(0.43388t) + 6.1962\, e^{-0.90097t}\sin(0.43388t)$

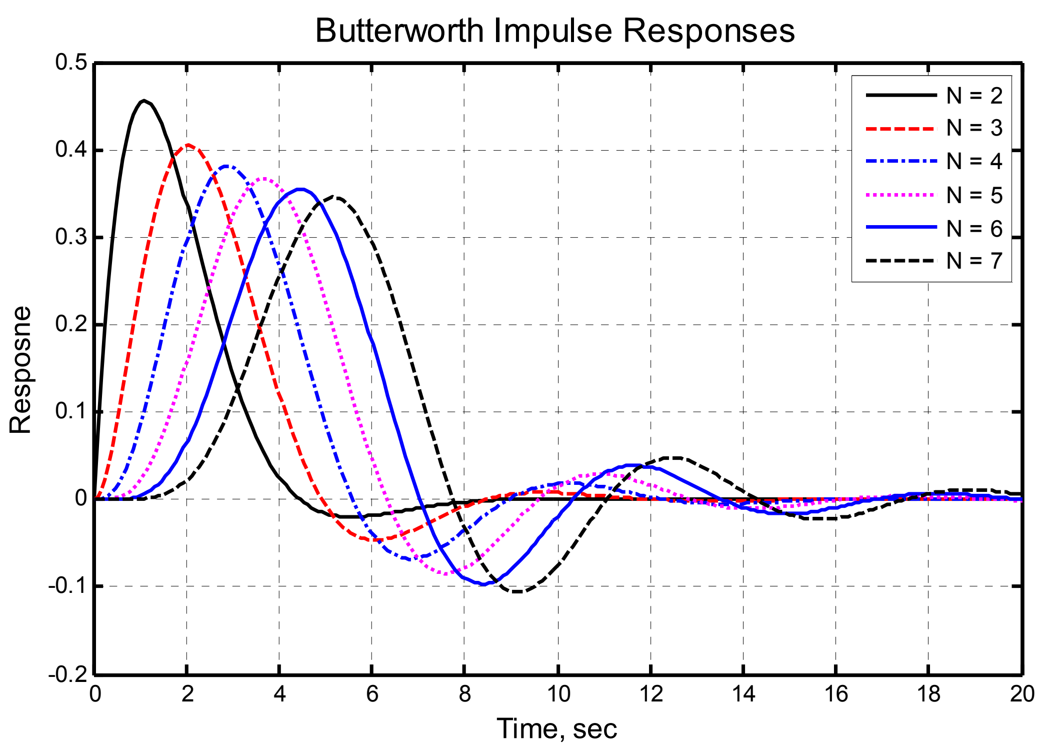


Figure 11 Butterworth impulse responses corresponding to Table 1

[10] Computed using u18217_butterworth_poles.m.

2.1 Design of Passive LC Butterworth Lowpass Filters

Butterworth as well as other all-pole passive LC filters can be efficiently implemented using the ladder structure shown in Figure 12. The ladder structure has been shown to exhibit minimum sensitivity to component variations and is therefore widely used. In this normalized form, the source resistance G_0 is always taken to be 1.0 whereas the load resistance $R_L = G_{N+1}$ can be equal to or less than 1.0 as given in the figure.

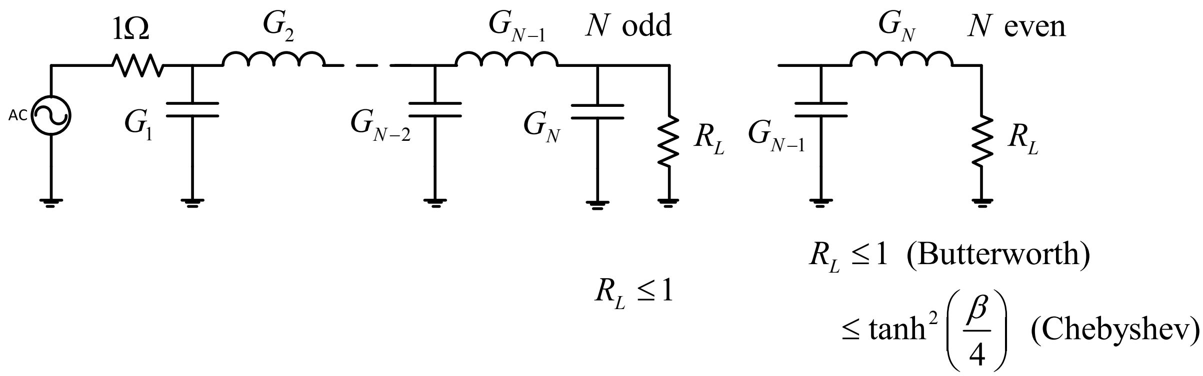


Figure 12 Lowpass ladder network

The prototype Butterworth filter design formula are given below [1]. Several design examples are provided here to facilitate computer program verification in §2.1.1 and §2.1.2.

$$A_t = \frac{4R_L}{\left(1+R_L\right)^2} \tag{3.17}$$

$$\gamma = 1 \tag{3.18}$$

$$d = \left(1-A_t\right)^{1/(2N)} \tag{3.19}$$

$$b_k = \gamma^2 + d^2 - 2\gamma d \cos\left(\frac{k\pi}{N}\right) \quad \text{for } k = 1, 2, ..., N \tag{3.20}$$

$$a_k = \sin\left[\frac{\left(2k-1\right)\pi}{2N}\right] \text{ for } k = 1, 2, ..., N \tag{3.21}$$

$$G_1 = \frac{2a_1}{\gamma - d} \tag{3.22}$$

$$G_k = \frac{4a_k a_{k-1}}{b_{k-1}G_{k-1}} \text{ for } k = 2, 3, ..., N \tag{3.23}$$

2.1.1 Scenario #1: Equally Terminated

U18214 Tabulated LPF Prototype Design						
Butterworth Lowpass Filter Design				Rs= 1.0	Rl=	1
Order=	5	(<= 12)				
At=	1					
gamma=	1					
d=	0					
k=	1.0000	2.0000	3.0000	4.0000	5.0000	
bk=	1.0000	1.0000	1.0000	1.0000	1.0000	
ak=	0.3090	0.8090	1.0000	0.8090	0.3090	
Gk=	0.6180	1.6180	2.0000	1.6180	0.6180	

Figure 13 Calculation details for 5th-order equally-terminated Butterworth lowpass filter

U18214 Tabulated LPF Prototype Design								
Butterworth Lowpass Filter Design				Rs= 1.0	Rl=	1		
Order=	8	(<= 12)						
At=	1							
gamma=	1							
d=	0							
k=	1.0000	2.0000	3.0000	4.0000	5.0000	6.0000	7.0000	8.0000
bk=	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
ak=	0.1951	0.5556	0.8315	0.9808	0.9808	0.8315	0.5556	0.1951
Gk=	0.3902	1.1111	1.6629	1.9616	1.9616	1.6629	1.1111	0.3902

Figure 14 Calculation details for 8th-order equally-terminated Butterworth lowpass filter

2.1.2 Scenario #2: Unequally Terminated

U18214 Tabulated LPF Prototype Design						
Butterworth Lowpass Filter Design				Rs= 1.0	Rl=	0.5
Order=	5	(<= 12)				
At=	0.888888889					
gamma=	1					
d=	0.802741562					
k=	1.0000	2.0000	3.0000	4.0000	5.0000	
bk=	0.3455	1.1483	2.1405	2.9433	3.2499	
ak=	0.3090	0.8090	1.0000	0.8090	0.3090	
Gk=	3.1331	0.9237	3.0510	0.4955	0.6857	

Figure 15 Calculation details for 4th-order unequally-terminated Butterworth lowpass filter

U18214 Tabulated LPF Prototype Design								
Butterworth Lowpass Filter Design				Rs= 1.0	Rl=	**0.5**		
Order=	**8**	(<= 12)						
At=	0.888888889							
gamma=	1							
d=	0.871685543							
k=	1.0000	2.0000	3.0000	4.0000	5.0000	6.0000	7.0000	8.0000
bk=	0.1492	0.5271	1.0927	1.7598	2.4270	2.9926	3.3705	3.5032
ak=	0.1951	0.5556	0.8315	0.9808	0.9808	0.8315	0.5556	0.1951
Gk=	**3.0408**	**0.9558**	**3.6678**	**0.8139**	**2.6863**	**0.5003**	**1.2341**	**0.1042**

Figure 16 Calculation details for 8th-order unequally-terminated Butterworth lowpass filter

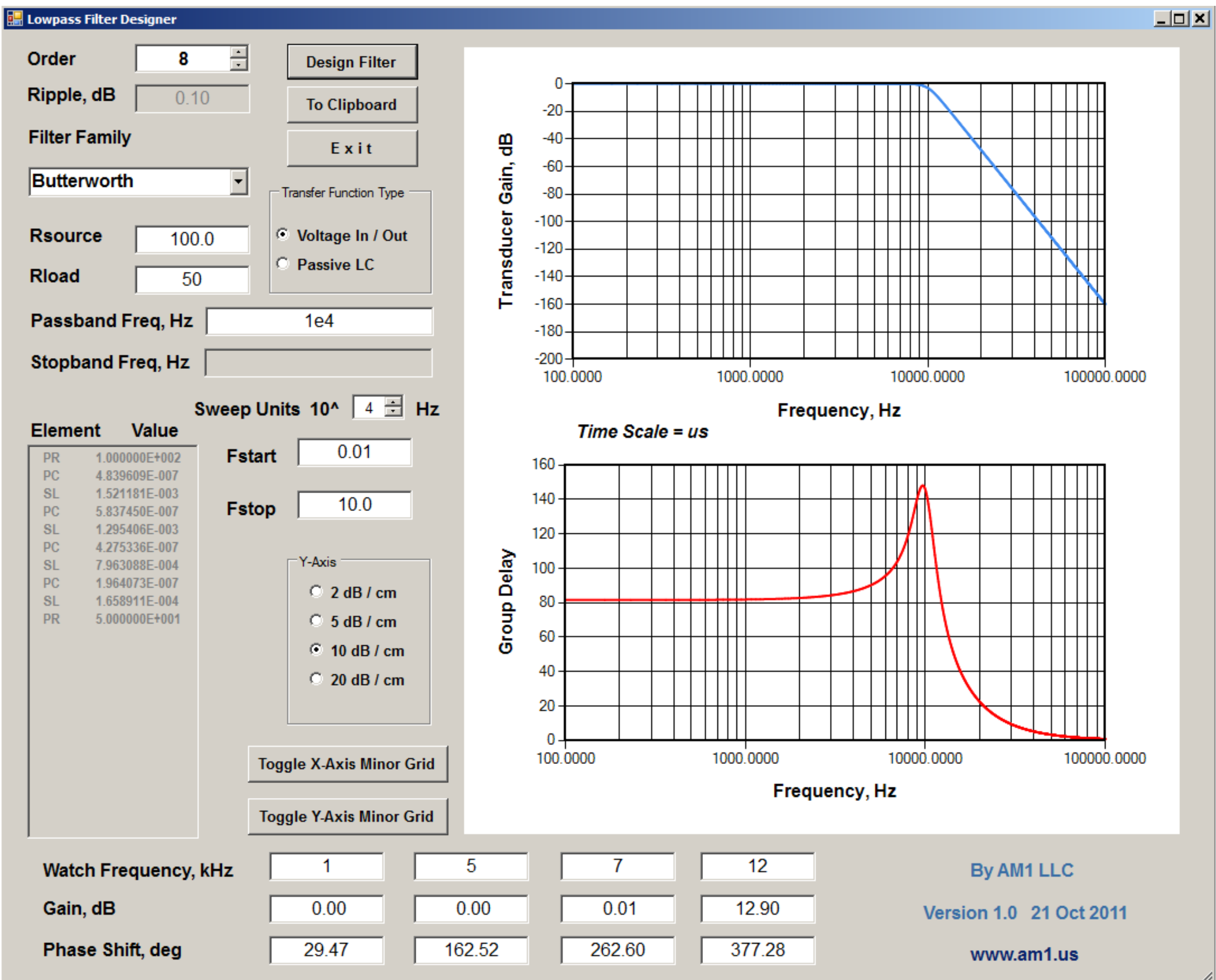


Figure 17 LPF designer appearance for Butterworth case

3 Chebyshev Lowpass Filters

An insightful derivation of the Chebyshev filter approximation is given in [8] and [10] as briefly outlined here. The filter loss is again given by (3.2) but with *L*(ω^2) given by

$$L\left(\omega^2\right)=1+\varepsilon^2F^2\left(\omega\right) \tag{4.1}$$

where

$$\varepsilon^2=10^{A_{pass}/10}-1 \tag{4.2}$$

The 4th-order normalized lowpass filter attenuation characteristic shown in Figure 18 facilitates the derivation greatly.

Chebyshev filters are specifically designed to exhibit *equal-ripple* attenuation in their passband region as shown in Figure 18 and this imposes several simple requirements on the behavior of *F*(ω) and *L*(ω) as follows:

Requirement #1: $F\left(\omega\right)=0$ at radian frequencies $\pm\Psi_1$ and $\pm\Psi_3$

Requirement #2: $F^2\left(\omega\right)=1$ at radian frequencies 0, $\pm\Psi_2$, ±1

Requirement #3: $dL\left(\omega^2\right)/d\omega=0$ at radian frequencies 0, $\pm\Psi_1$, $\pm\Psi_2$, $\pm\Psi_3$

All of the Ψ_k will temporarily be assumed to be unknown.

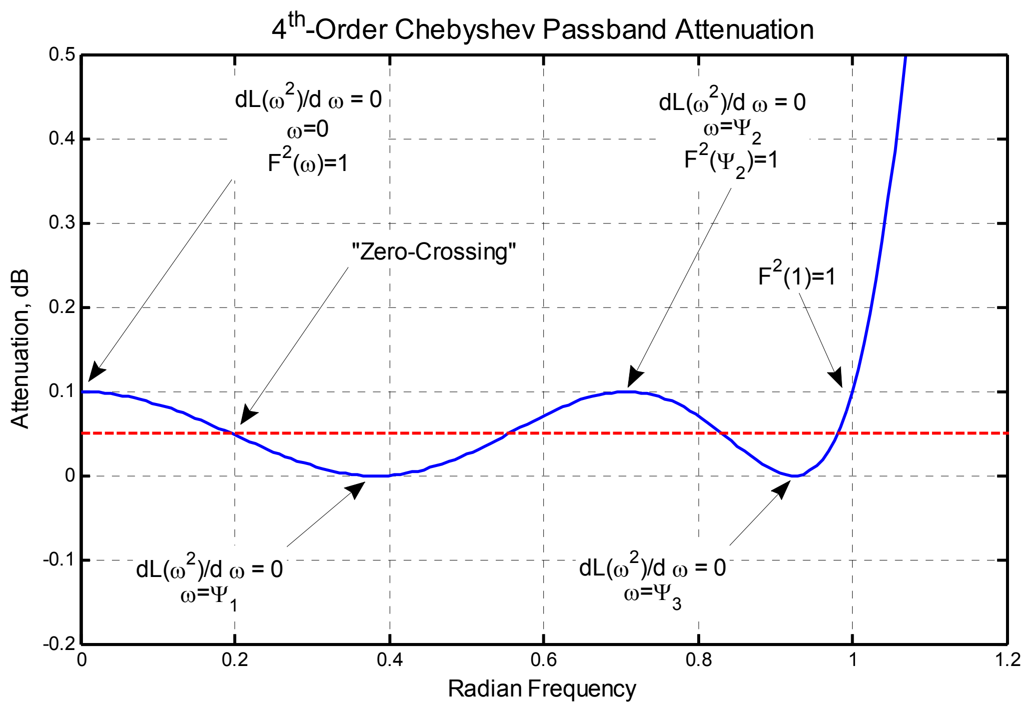


Figure 18 Example passband attenuation characteristic for a normalized *N* = 4 Chebyshev lowpass filter having a passband ripple of 0.1 dB. Filter order is easily identified by noting the number of *zero-crossings* which occur between the attenuation characteristic and an auxiliary line drawn at one-half of the ripple magnitude as shown.

Requirement #1 dictates that *F*(ω) be a polynomial given by

$$F(\omega) = M_1(\omega^2 - \Psi_1^2)(\omega^2 - \Psi_3^2) \tag{4.3}$$

where M_1 is a constant to be determined. From Requirement #2

$$1 - F^2(\omega) = M_2\omega(\omega^2 - \Psi_2^2)(1 - \omega^2) \tag{4.4}$$

From Requirement #3,

$$\frac{dL}{d\omega} = \frac{d}{d\omega}\left[1 + \varepsilon^2 F^2(\omega)\right] = \varepsilon^2 2F(\omega)\frac{dF}{d\omega} \tag{4.5}$$

From Requirement #1, $F(\omega)$ must have zeros at $\pm\Psi_1$ and $\pm\Psi_3$ whereas $dL / d\omega$ is required to have additional zeros at 0 and $\pm\Psi_2$ which implies that $dF / d\omega$ must have the form

$$\frac{dF}{d\omega} = M_3\omega(\omega^2 - \Psi_2^2) \tag{4.6}$$

In order for these latter zeros to survive the derivative with respect to ω in $F(\omega)$, these zeros must, however, be double-roots in $F(\omega)$ which means that (4.4) must be modified to

$$1 - F^2(\omega) = M_2\omega^2(\omega^2 - \Psi_2^2)^2(1 - \omega^2) \tag{4.7}$$

from which follows

$$\frac{1 - F^2(\omega)}{1 - \omega^2} = M_2\omega^2(\omega^2 - \Psi_2^2)^2 \tag{4.8}$$

Comparing the factors in (4.8) with those in (4.6) makes it possible to write

$$\frac{1 - F^2(\omega)}{1 - \omega^2} = M_4\left(\frac{dF}{d\omega}\right)^2 \tag{4.9}$$

Applying a square-root and separation of variables to (4.9) produces

$$M_5\frac{dF}{\sqrt{1 - F^2}} = \frac{d\omega}{\sqrt{1 - \omega^2}} \tag{4.10}$$

which in turn can be written in terms of definite integrals as

$$M_5\int_0^F \frac{dF}{\sqrt{1 - F^2}} + M_6 = \int_0^\omega \frac{d\omega}{\sqrt{1 - \omega^2}} \tag{4.11}$$

Making use of the substitution $u = \cos(\theta)$ in (4.11) produces

$$M_5\cos^{-1}(F) + M_6 = \cos^{-1}(\omega) \tag{4.12}$$

which can be rewritten as

$$F(\omega)\Big|_{\omega=\cos(\theta)} = \cos\left(\frac{\theta - M_6}{M_5}\right) \tag{4.13}$$

and only the constants remain to be identified. From Requirement #2, $F(\omega) = 1$ for $\omega = 1$ which corresponds to $\theta = 0$ thereby dictating that $M_6 \equiv 0$. Similarly, the value of F for $\theta = \pi / 2$ dictates that $M_5 \equiv 1/4$ thereby leading to the final result

$$F(\omega) = \cos\left[4\cos^{-1}(\omega)\right] \tag{4.14}$$

This result can be generalized for an N^{th}-order Chebyshev filter as

$$F_N(\omega) = \cos\left[N\cos^{-1}(\omega)\right] \tag{4.15}$$

This result (4.14) can be expanded in terms of cos(θ) as

$$F(\omega) = 1 - 8\omega^2 + 8\omega^4 \tag{4.16}$$

where the right-hand side of (4.16) corresponds to the 4^{th}-order Chebyshev polynomial represented by $T_4(\omega)$. The first several Chebyshev polynomials along with their simple recursive construction formula are given by

$$\begin{aligned} T_0(\omega) &= 1 \\ T_1(\omega) &= \omega \\ T_2(\omega) &= 2\omega^2 - 1 \\ T_3(\omega) &= 4\omega^3 - 3\omega \\ T_4(\omega) &= 8\omega^4 - 8\omega^2 + 1 \\ T_{n+1}(\omega) &= 2\omega T_n(\omega) - T_{n-1}(\omega) \end{aligned} \tag{4.17}$$

The first few Chebyshev polynomials are plotted in Figure 19 for illustrative purposes.

3.1 Required Chebyshev Filter Order

If the passband ripple up to frequency f_{pass} is given by A_{pass} (dB) and the minimum stopband loss at f_{stop} is required to be A_{stop} (dB), the minimum order for the Chebyshev filter is given by

$$N \ge \frac{\cosh^{-1}\left(\sqrt{\dfrac{10^{A_{stop}/10} - 1}{10^{A_{pass}/10} - 1}}\right)}{\cosh^{-1}\left(\dfrac{f_{stop}}{f_{pass}}\right)} \tag{4.18}$$

where a convenient relationship for $\cosh^{-1}(x)$ is

$$\cosh^{-1}(x) = \log_e\left(x + \sqrt{x^2 - 1}\right) \tag{4.19}$$

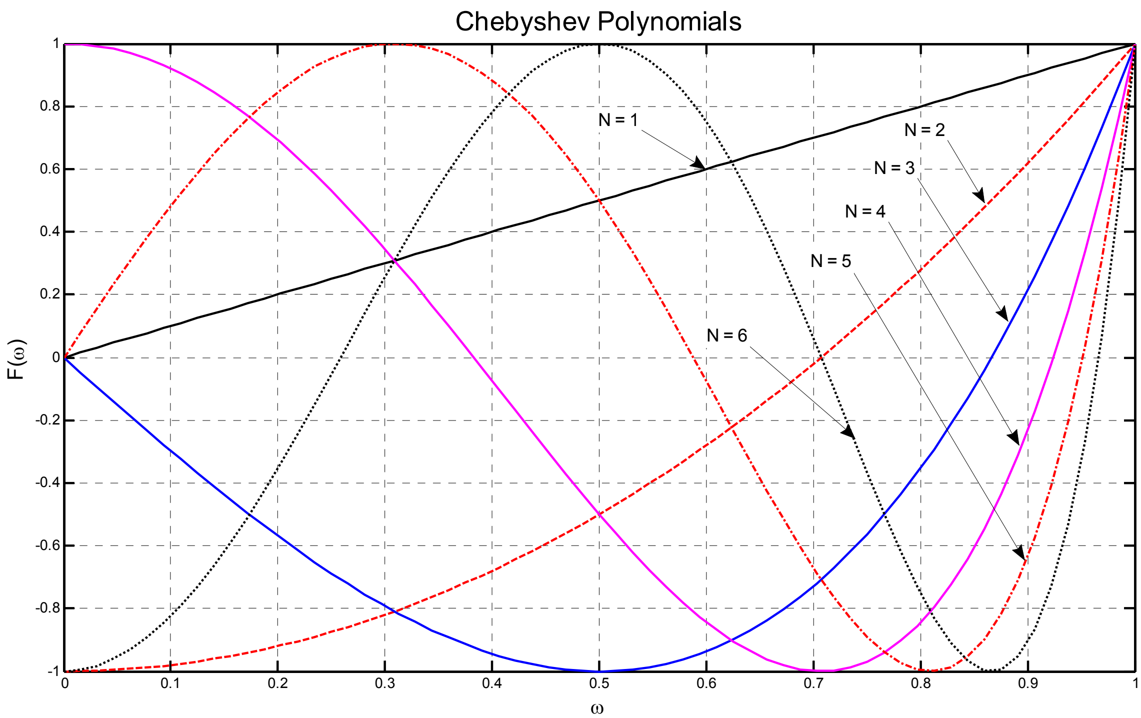


Figure 19 Chebyshev polynomials[11] 1st through 6th-order

3.2 Chebyshev Pole Locations

The N^{th}-order Chebyshev filter loss function can be rewritten using (4.1) while incorporating (4.15) as

$$L_N(\omega) = 1 + \varepsilon^2 \left\{ \cos\left[N \cos^{-1}(\omega) \right] \right\}^2 \tag{4.20}$$

Defining

$$\begin{aligned} \theta &= \cos^{-1}(\omega) \\ y &= \exp(j\theta) \end{aligned} \tag{4.21}$$

permits (4.20) to be re-written as

$$\begin{aligned} L_N(\theta) &= 1 + \varepsilon^2 \cos^2(N\theta) = 1 + \varepsilon^2 \left(\frac{e^{jN\theta} + e^{-jN\theta}}{2} \right)^2 \\ &= 1 + \left(\frac{\varepsilon}{2} \right)^2 \left[\left(e^{j\theta} \right)^N + \left(\frac{1}{e^{j\theta}} \right)^N \right]^2 \end{aligned} \tag{4.22}$$

which then leads to

$$L_N(y) = 1 + \left(\frac{\varepsilon}{2} \right)^2 \left[y^N + \frac{1}{y^N} \right]^2 \tag{4.23}$$

This form is convenient for discussing elliptic lowpass filters as well as deriving the pole locations for Chebyshev filters. The roots which satisfy (4.23) can be found by solving

[11] Using u18260_chebyshevPolynomials.m.

$$1+\frac{\varepsilon^2}{4}\left(y^N+\frac{1}{y^N}\right)^2=0 \tag{4.24}$$

This can be rewritten as

$$\left(y^N+\frac{1}{y^N}\right)^2=-\frac{4}{\varepsilon^2}$$

$$y^N+\frac{1}{y^N}=\pm j\frac{2}{\varepsilon} \tag{4.25}$$

$$y^{2N}\mp j\frac{2}{\varepsilon}y^N+1=0 \quad \text{(a quadratic in } y^N \text{)}$$

Solving the quadratic leads to

$$y^N=\pm j\frac{1}{\varepsilon}\pm j\sqrt{1+\frac{1}{\varepsilon^2}} \tag{4.26}$$

The solutions to (4.26) are given by

$$y_n=r\exp\left[j\frac{\pi}{N}\left(n-\frac{1}{2}\right)\right] \text{ for integers n} \tag{4.27}$$

with $n \in \{ 1, 2,\ldots, N \}$. The magnitude of the roots given by

$$\left|y\right|=r=\left(\frac{1}{\varepsilon}+\sqrt{1+\frac{1}{\varepsilon^2}}\right)^{1/N} \tag{4.28}$$

From (4.21), $\cos\left(\theta_k\right)=\omega_k$ the *s*-plane poles follow by noting that

$$\cos\left(\theta_k\right)=\frac{\exp\left(j\theta_k\right)+\exp\left(-j\theta_k\right)}{2}=\frac{1}{2}\left(y_k+\frac{1}{y_k}\right)=\omega_k=\frac{s_k}{j} \tag{4.29}$$

which results in

$$s_k=\frac{j}{2}\left(y_k+\frac{1}{y_k}\right) \tag{4.30}$$

It is not obvious in this form, however, that the Chebyshev poles lie on an ellipse in the complex *s*-plane. Returning to (4.20), the poles s_k must satisfy

$$\cos\left[N\cos^{-1}\left(\frac{s_k}{j}\right)\right]=\pm\frac{j}{\varepsilon} \tag{4.31}$$

Let $s_k=\sigma_k+j\omega_k$ in (4.31) and note that

$$\begin{aligned}\cos^{-1}\left(\frac{\sigma_k+j\omega_k}{j}\right)&=\cos^{-1}\left(\omega_k-j\sigma_k\right)=u+jv\\ \omega_k-j\sigma_k&=\cos\left(u+jv\right)=\cos\left(u\right)\cosh\left(v\right)-j\sin\left(u\right)\sinh\left(v\right)\end{aligned} \tag{4.32}$$

thereby leading to

$$\begin{aligned} \omega_k &= \cos(u)\cosh(v) \\ \sigma_k &= -\sin(u)\sinh(v) \end{aligned} \tag{4.33}$$

Also from (4.31), write

$$\cos\left[N(u+jv)\right] = \pm\frac{j}{\varepsilon} \tag{4.34}$$

$$\cos(Nu)\cosh(Nv) - j\sin(Nu)\sinh(Nv) = \pm\frac{j}{\varepsilon} \tag{4.35}$$

The solutions to (4.35) then become

$$\begin{aligned} &\cos(Nu)\cosh(Nv) = 0 \\ &\sin(Nu)\sinh(Nv) = \pm\frac{1}{\varepsilon} \end{aligned} \tag{4.36}$$

The solution to the first portion of (4.36) requires that

$$u = \frac{(2k-1)}{2N}\pi \text{ for } k = 1,2,... \tag{4.37}$$

whereas the second portion requires that

$$v = \pm\frac{1}{N}\sinh^{-1}\left(\frac{1}{\varepsilon}\right) \tag{4.38}$$

Using (4.37) and (4.38) in (4.33) finally results in

$$\omega_k = \cosh\left[\frac{1}{N}\sinh^{-1}\left(\frac{1}{\varepsilon}\right)\right]\cos\left[\frac{(2k-1)\pi}{2N}\right] \text{ for } k = 1,2,...,2N \tag{4.39}$$

$$\sigma_k = -\sinh\left[\frac{1}{N}\sinh^{-1}\left(\frac{1}{\varepsilon}\right)\right]\sin\left[\frac{(2k-1)\pi}{2N}\right] \text{ for } k = 1,2,...,2N \tag{4.40}$$

From this final pair of results then,

$$\frac{\omega_k^2}{\cosh^2(v)} + \frac{\sigma_k^2}{\sinh^2(v)} = 1 \tag{4.41}$$

and it becomes clear that the poles lie on an ellipse having parameters

$$\begin{aligned} a &= \cosh\left[\frac{1}{N}\sinh^{-1}\left(\frac{1}{\varepsilon}\right)\right] \quad \text{(major-axis)} \\ b &= \sinh\left[\frac{1}{N}\sinh^{-1}\left(\frac{1}{\varepsilon}\right)\right] \quad \text{(minor-axis)} \end{aligned} \tag{4.42}$$

Two different Chebyshev lowpass filter examples are shown in Figure 20 through Figure 23. In the first case, the passband ripple is purposely made large (1 dB) in order to illustrate that this leads to higher quality poles (poles closer to the jω-axis). As shown in the second case, the poles still lie on a very elliptical perimeter even for small passband ripple cases (0.1 dB).

Chebyshev lowpass attenuation characteristics for orders 1 through 10 are shown in Figure 24 through Figure 28 for passband ripple parameters of 0.01 dB, 0.1 dB, 0.25 dB, 0.5 dB, and 1.0 dB respectively.

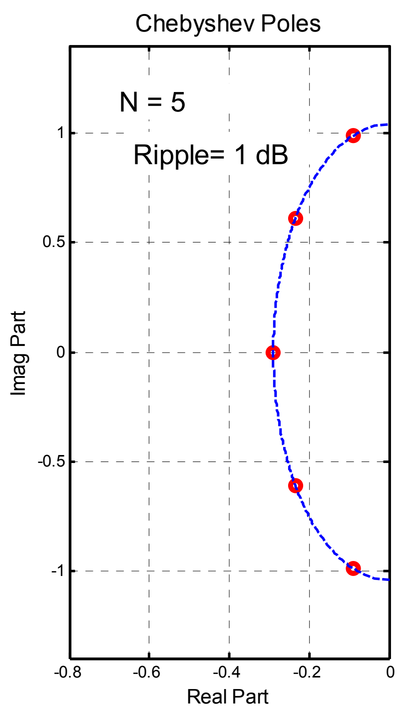


Figure 20 N = 5 Chebyshev lowpass filter with 1 dB passband ripple[12]

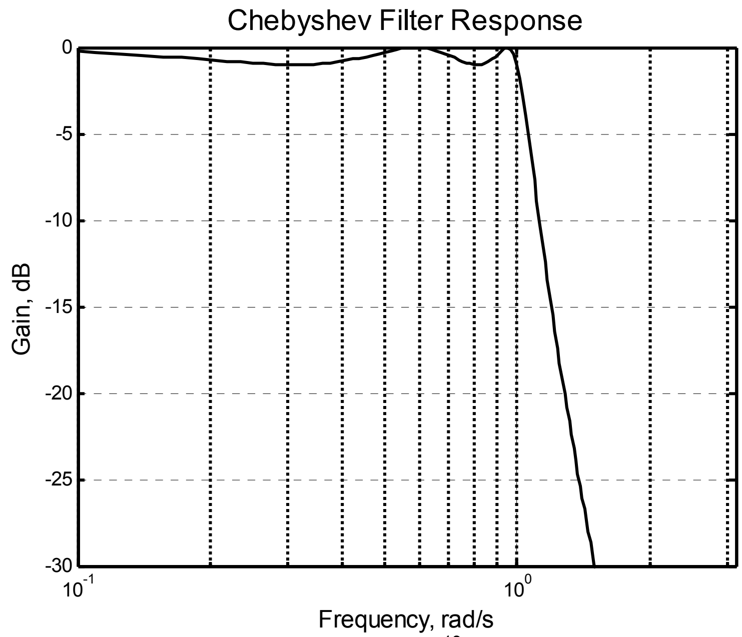


Figure 21 Filter gain characteristic[13] corresponding to poles locations in Figure 20

Group delay and impulse transient responses are shown in Figure 29 through Figure 37. Table 2 provides the 0.1 dB ripple Chebyshev impulse responses in mathematical form for $N \leq 7$.

[12] u18218_chebyshev_poles.m.
[13] Ibid.

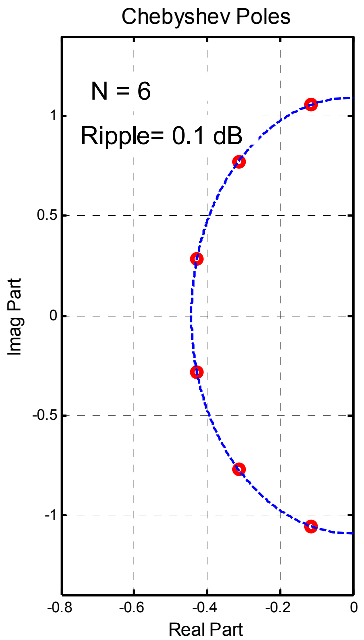


Figure 22 *N* = 6 Chebyshev lowpass filter with 0.1 dB passband ripple

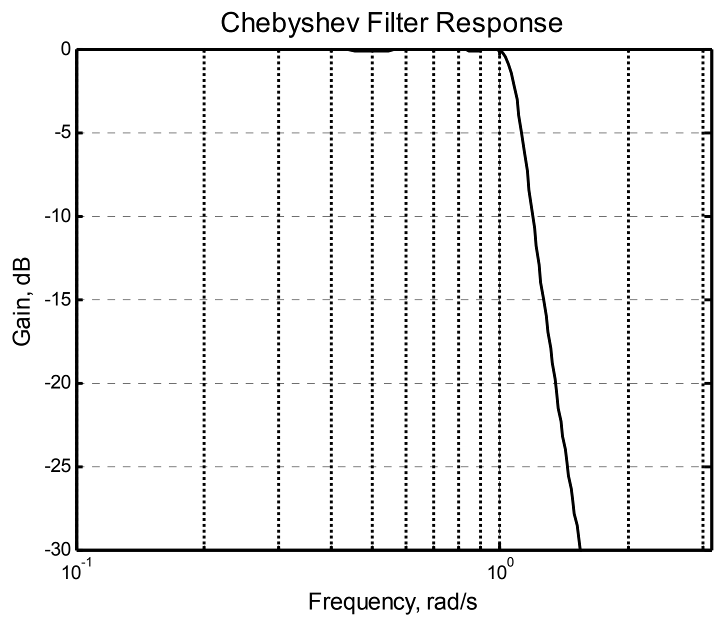


Figure 23 Filter gain characteristic corresponding to poles locations in Figure 22

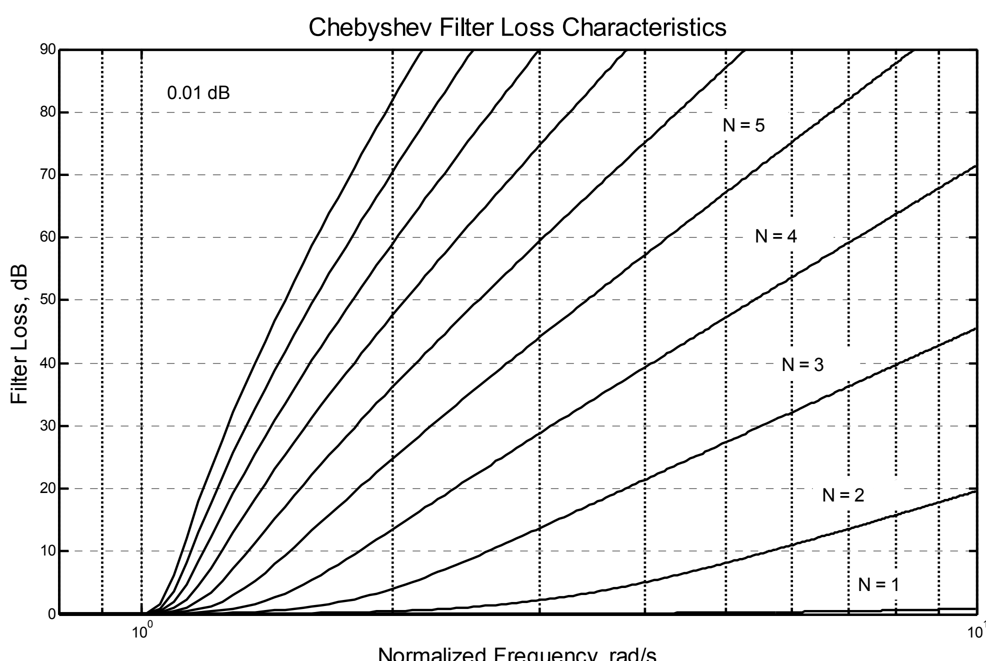


Figure 24 Chebshev lowpass filter stopband attenuation characteristics[14] versus order for 0.01 dB passband ripple filters

[14] Ibid.

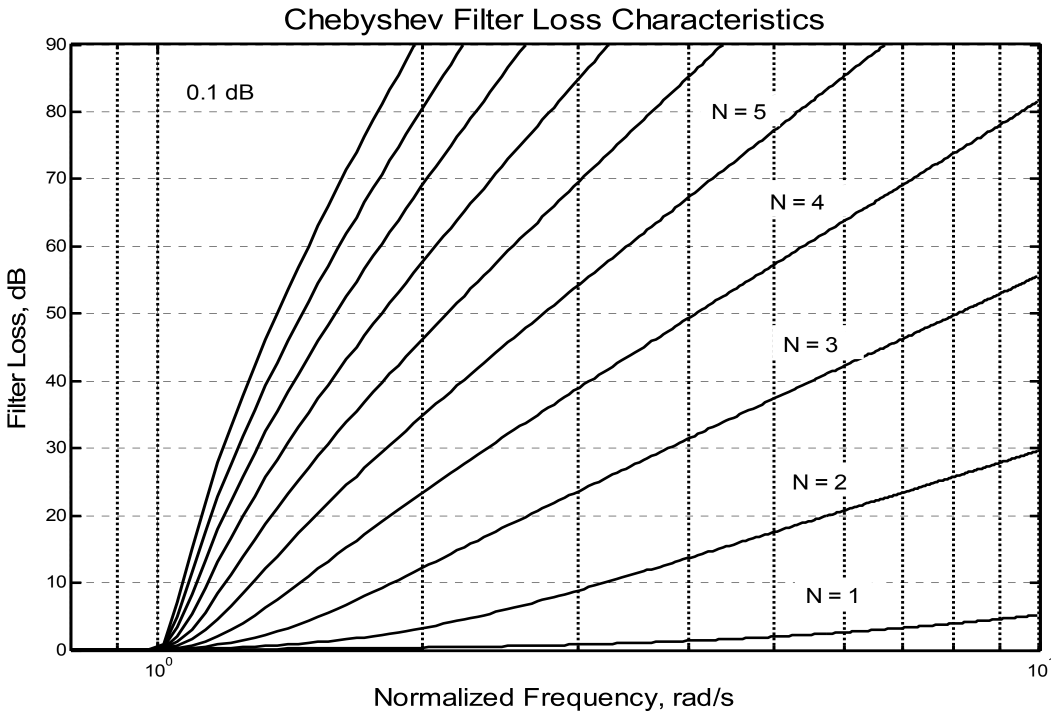


Figure 25 Chebyshev lowpass filter stopband attenuation characteristics versus order for 0.10 dB passband ripple filters

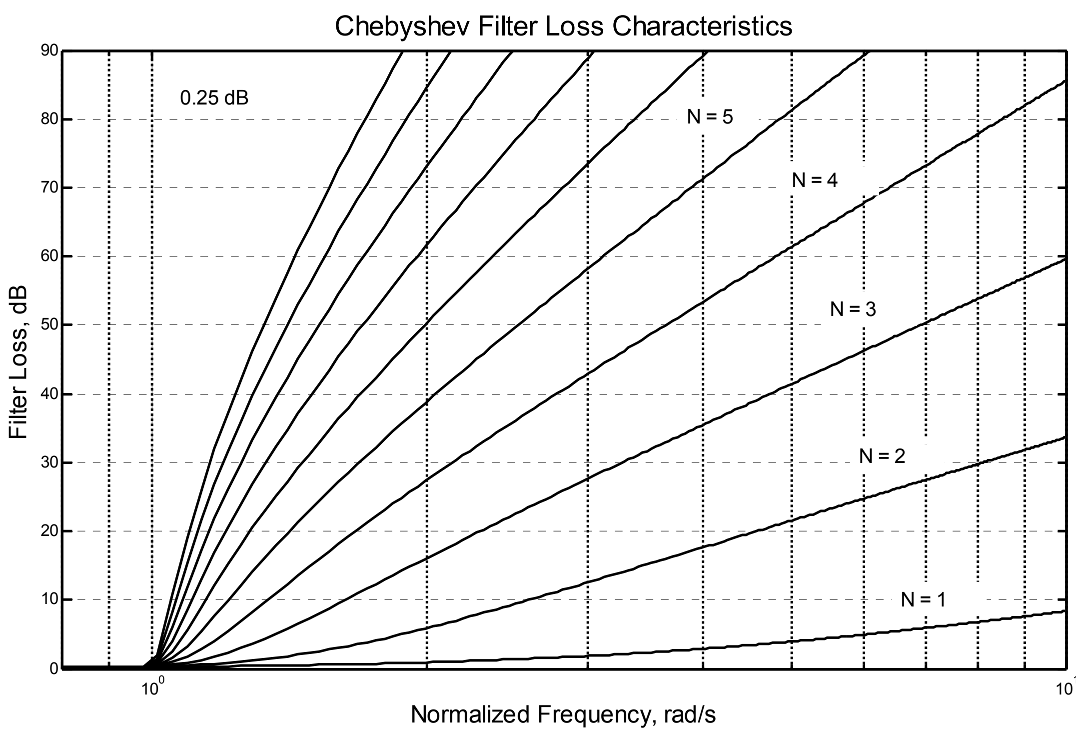


Figure 26 Chebyshev lowpass filter stopband attenuation characteristics versus order for 0.25 dB passband ripple filters

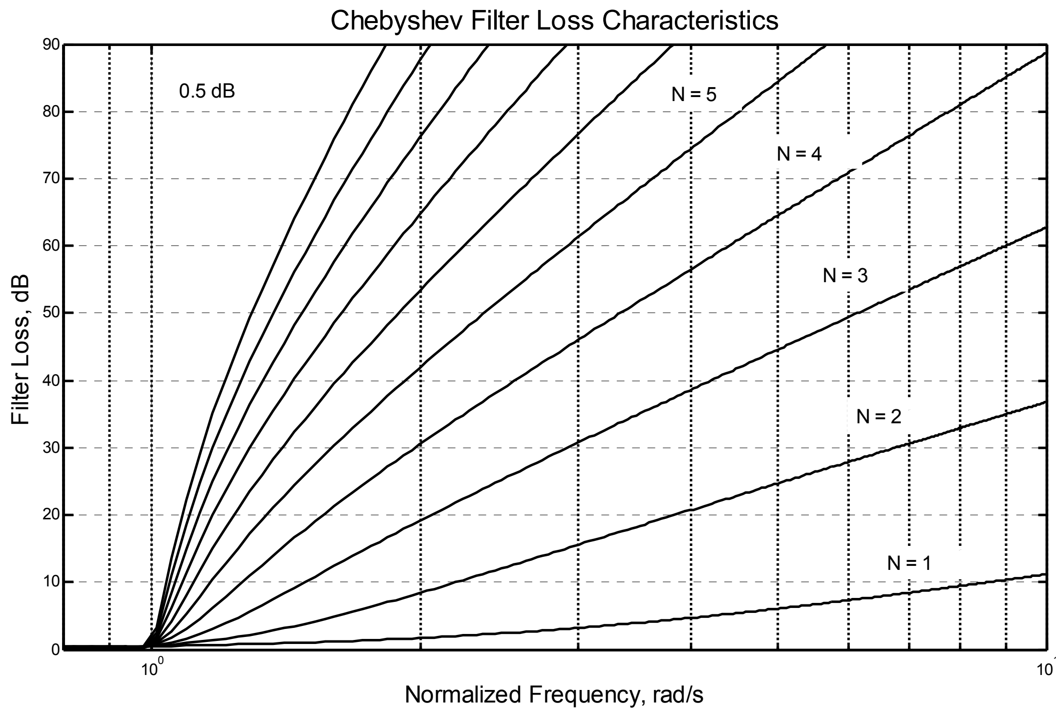


Figure 27 Chebyshev lowpass filter stopband attenuation characteristics versus order for 0.50 dB passband ripple filters

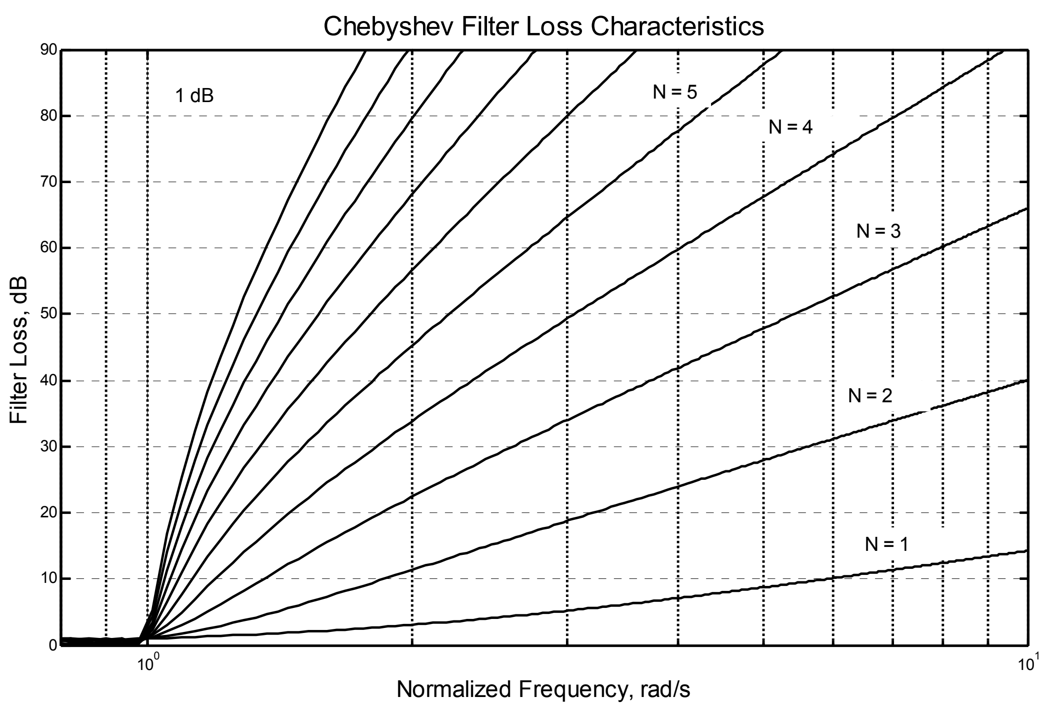


Figure 28 Chebyshev lowpass filter stopband attenuation characteristics versus order for 1.0 dB passband ripple filters

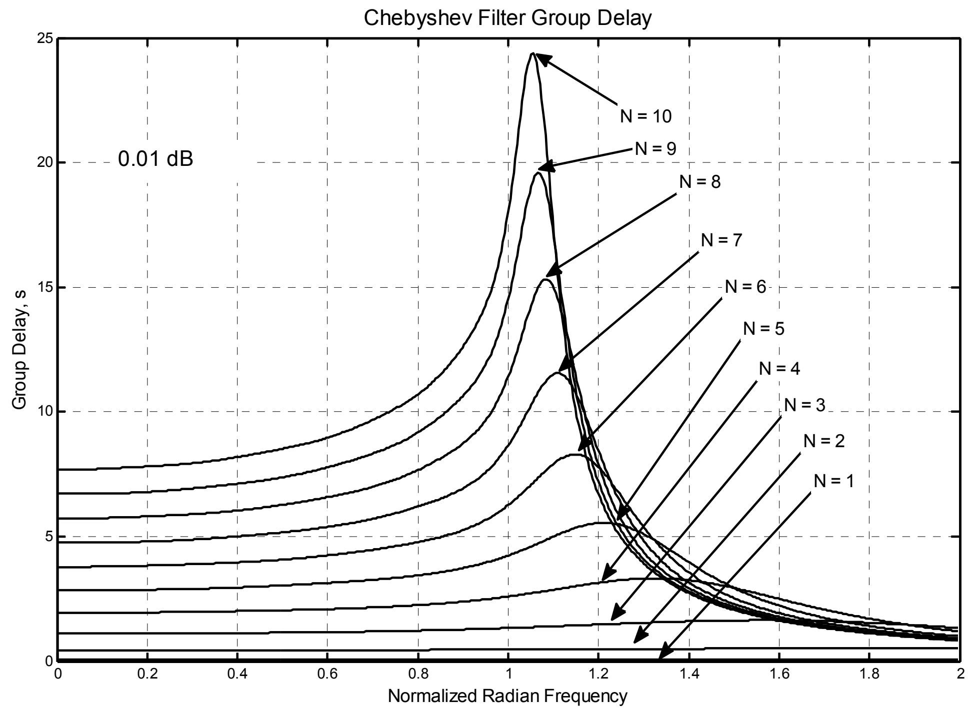


Figure 29 Chebyshev lowpass filter group delay characteristics for 0.01 dB passband ripple case

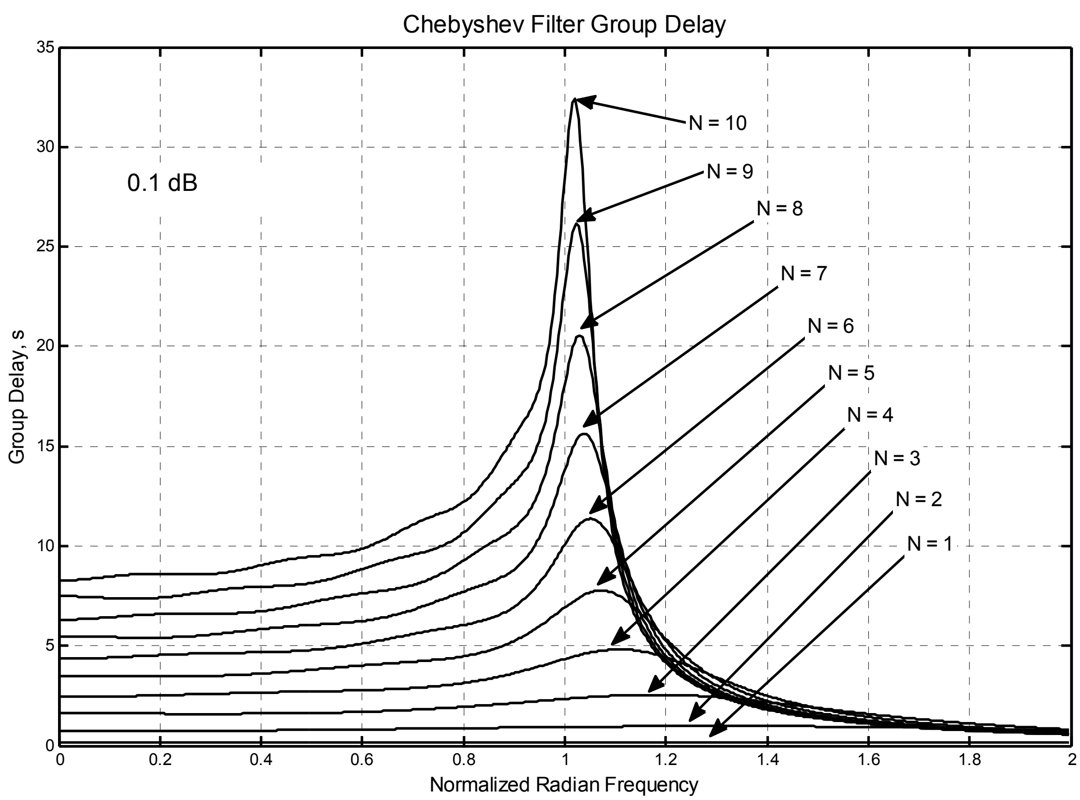


Figure 30 Chebyshev lowpass filter group delay characteristics for 0.1 dB passband ripple case

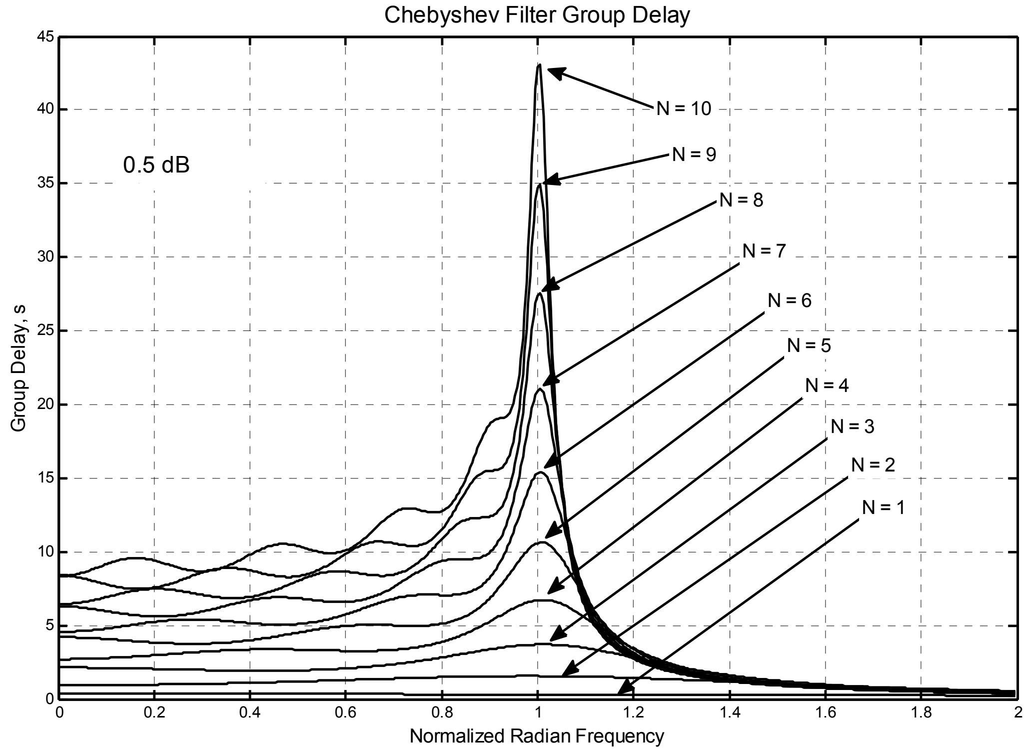


Figure 31 Chebyshev lowpass filter group delay characteristics for 0.5 dB passband ripple case

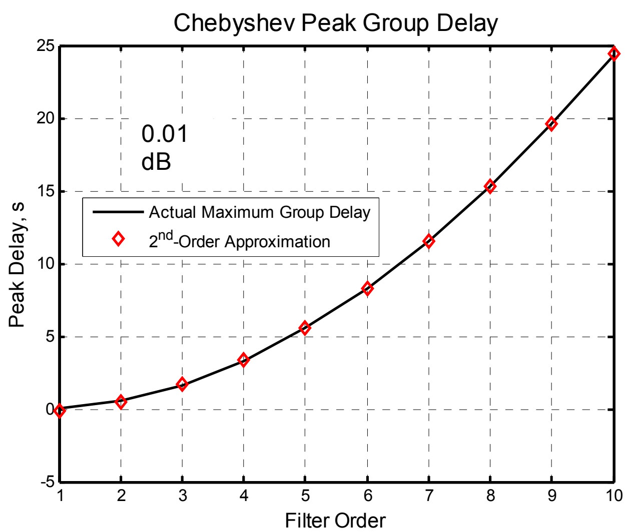


Figure 32 Peak group delay for 0.01 dB ripple Chebyshev lowpass filters versus filter order

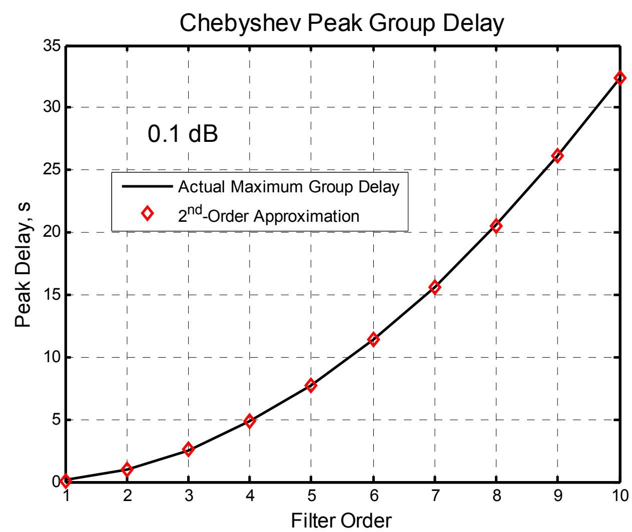


Figure 33 Peak group delay for 0.1 dB ripple Chebyshev lowpass filters versus filter order

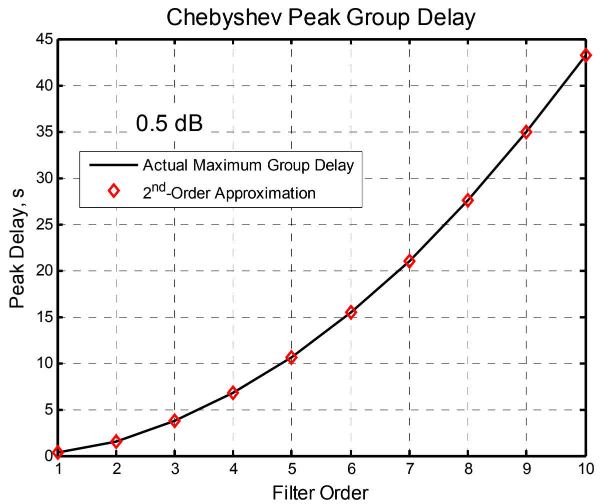


Figure 34 Peak group delay for 0.5 dB ripple Chebyshev lowpass filters versus filter order

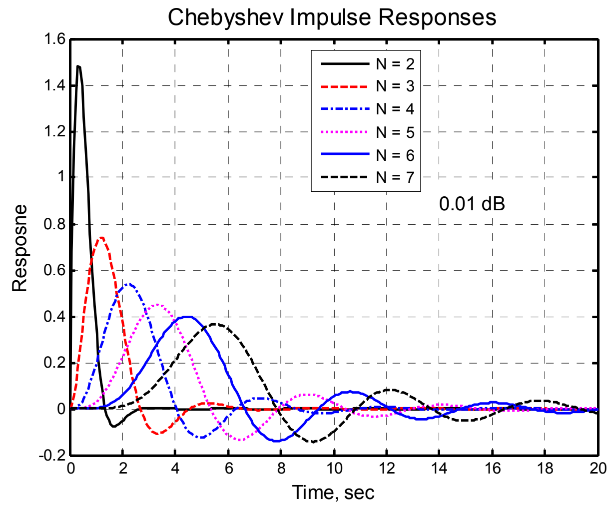


Figure 35 Impulse response for 0.01 dB ripple Chebyshev lowpass filters versus filter order

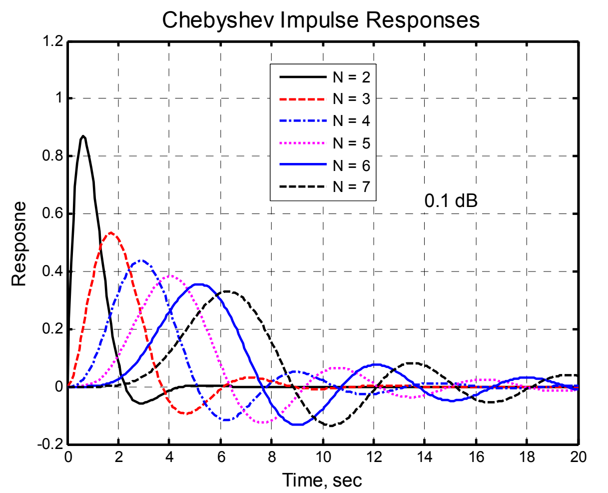


Figure 36 Impulse response for 0.1 dB ripple Chebyshev lowpass filters versus filter order

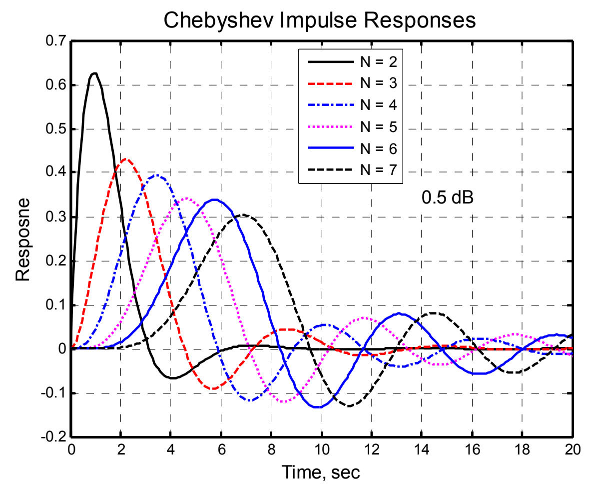


Figure 37 Impulse response for 0.5 dB ripple Chebyshev lowpass filters versus filter order

Table 2 Chebyshev (Normalized) Filter Impulse Responses[15] for 0.1 dB Passband Ripple Case

Filter Order *N*	Impulse Response[16]
1	
2	$f(t) = 2.3998\ e^{-1.1862t} \sin(1.3809t)$
3	$f(t) = 0.96941\ e^{-0.96941t} - 0.96941 e^{-0.4847t} \cos(1.2062t) + 0.38956\,e^{-0.4847t} \sin(1.2062t)$
4	$f(t) = 1.386\,e^{-0.63773t} \sin(0.465t) + 0.40683\,e^{-0.63773t} \cos(0.465t)$ $- 0.4387\,e^{-0.26416t} \sin(1.1226t) - 0.40683\,e^{-0.26416t} \cos(1.1226t)$
5	$f(t) = 0.68706\,e^{-0.53891t} + 0.56938\,e^{-0.43599t} \sin(0.66771t) - 0.87198\,e^{-0.43599t} \cos(0.66771t)$ $- 0.33256\,e^{-0.16653t} \sin(1.0804t) + 0.18492\,e^{-0.16653t} \cos(1.0804t)$
6	$f(t) = 1.1455\,e^{-0.42804t} \sin(0.28309t) + 0.32543\,e^{-0.42804t} \cos(0.28309t)$ $- 0.52372\,e^{-0.31335t} \sin(0.77343t) - 0.58624\,e^{-0.31335t} \cos(0.77343t)$ $0.062738\,e^{-0.11469t} \sin(1.0565t) + 0.26081 e^{-0.11469t} \cos(1.0565t)$
7	$f(t) = 0.56253\,e^{-0.37678t} + 0.48721 e^{-0.33947t} \sin(0.46366t)$ $- 0.84662\,e^{-0.33947t} \cos(0.46366t) - 0.52433\,e^{-0.23492t} \sin(0.83549t)$ $- 0.28409\,e^{-0.23492t} \cos(0.83549t) + 0.19529\,e^{-0.083841t} \sin(1.0418t)$

3.3 Design of Passive LC Chebyshev Lowpass Filters

The design formula for Chebyshev filters are understandably similar to those for the Butterworth case. The formula adopted here are based upon the work provided in [2].[17] The filter configuration is again shown in Figure 12.

Assuming that the filter order is given by *N*, let the passband ripple be represented by A_{rip} in dB. Then define

$$\varepsilon = \sqrt{10^{A_{rip}/10} - 1} \tag{4.43}$$

and

$$a = \begin{cases} \dfrac{4R_{load}}{\left(R_{source} + R_{load}\right)^2} & \text{for } N \text{ odd} \\ \dfrac{4R_{load}}{\left(R_{source} + R_{load}\right)^2}\left(1+\varepsilon^2\right) & \text{for } N \text{ even} \end{cases} \tag{4.44}$$

Then compute

[15] Computed using u18218_chebyshev_poles.m.
[16] Calculated in u18218_chebyshev_poles.m.
[17] In spite of multiple attempts, the design formula provided in [1] were not valid for resistive load cases with $R_L < 1.0$.

$$\alpha_i = 2\sin\left(\frac{i\pi}{2N}\right)$$
$$\beta_i = 2\cos\left(\frac{i\pi}{2N}\right) \quad (4.45)$$

for i = 1,2,…,N. Define two additional parameters

$$\gamma = \left(\frac{1}{\varepsilon} + \sqrt{\frac{1}{\varepsilon^2}+1}\right)^{1/N}$$
$$\delta = \left(\sqrt{\frac{1-a}{\varepsilon^2}} + \sqrt{\frac{1-a}{\varepsilon^2}+1}\right)^{1/N} \quad (4.46)$$

and from here

$$x = \gamma - \frac{1}{\gamma}$$
$$y = \delta - \frac{1}{\delta} \quad (4.47)$$

The first prototype filter value is given by

$$G_1 = \frac{2\alpha_1}{x-y} \quad (4.48)$$

whereas the remaining prototype parameters are recursively given by

$$G_k = \frac{4\alpha_{nm(k-1)}\alpha_{nm(k-1)+2}}{b(k-1,x,y)G_{k-1}} \text{ for } k = 2,3,...,N \quad (4.49)$$

where

$$nm(j) = 2j-1 \quad (4.50)$$
$$b(j,x,y) = x^2 - \beta_{2j}xy + y^2 + \alpha_{2j}^2 \quad (4.51)$$

Several design examples are provided here to facilitate computer program verification in Figure 38 through Figure 41.

U18215 Tabulated Chebyshev LPF Prototype Des						
Chebyshev Lowpass Filter Design				Rs= 1.0	Rl=	0.5
Order=	3	(<= 10)		Ripple, dB=	0.1	
epsilon=	0.152620419					
At=	0.888888889					
gamma=	2.362153866					
d=	1.66143653					
x=	1.938811418					
y=	1.059547753					
k=	1.0000	2.0000	3.0000	4.0000	5.0000	
alpha_k=	1.0000	1.7321	2.0000	1.7321	1.0000	
beta_k=	1.7321	1.0000	0.0000	-1.0000	-1.7321	
b()=	5.8274	9.9359	8.9902	9.9359	5.8274	
nm()=	1.0000	3.0000	5.0000	7.0000	9.0000	
Gk=	2.2746	0.6035	1.3341			

Figure 38 Calculation details for 3rd-order unequally-terminated Chebyshev lowpass filter

U18215 Tabulated Chebyshev LPF Prototype Des							
Chebyshev Lowpass Filter Design				Rs= 1.0	Rl=	0.5	
Order=	4	(<= 10)		Ripple, dB=	0.1		
epsilon=	0.152620419						
At=	0.909593771						
gamma=	1.905377961						
d=	1.4298148						
x=	1.380547706						
y=	0.730423523						
k=	1.0000	2.0000	3.0000	4.0000	5.0000	6.0000	7.0000
alpha_k=	0.7654	1.4142	1.8478	2.0000	1.8478	1.4142	0.7654
beta_k=	1.8478	1.4142	0.7654	0.0000	-0.7654	-1.4142	-1.8478
b()=	3.0134	6.4394	5.8655	4.4562	5.8655	6.4394	3.0134
nm()=	1.0000	3.0000	5.0000	7.0000	9.0000	11.0000	13.0000
Gk=	2.3545	0.7973	2.6600	0.3626			

Figure 39 Calculation details for 4th-order unequally-terminated Chebyshev lowpass filter

U18215 Tabulated Chebyshev LPF Prototype Des										
Chebyshev Lowpass Filter Design				Rs= 1.0	Rl=	0.5				
Order=	5	(<= 10)		Ripple, dB=	0.1					
epsilon=	0.152620419									
At=	0.888888889									
gamma=	1.674884679									
d=	1.356095432									
x=	1.077828648									
y=	0.618684202									
k=	1.0000	2.0000	3.0000	4.0000	5.0000	6.0000	7.0000	8.0000	9.0000	
alpha_k=	0.6180	1.1756	1.6180	1.9021	2.0000	1.9021	1.6180	1.1756	0.6180	
beta_k=	1.9021	1.6180	1.1756	0.6180	0.0000	-0.6180	-1.1756	-1.6180	-1.9021	
b()=	1.8475	4.7504	5.5746	4.0054	2.8782	4.0054	5.5746	4.7504	1.8475	
nm()=	1.0000	3.0000	5.0000	7.0000	9.0000	11.0000	13.0000	15.0000	17.0000	
Gk=	**2.6921**	**0.8042**	**3.3882**	**0.6853**	**1.4572**					

Figure 40 Calculation details for 5^{th}-order unequally-terminated Chebyshev lowpass filter

U18215 Tabulated Chebyshev LPF Prototype Des												
Chebyshev Lowpass Filter Design				Rs= 1.0	Rl=	0.5						
Order=	6	(<= 10)		Ripple, dB=	0.1							
epsilon=	0.152620419											
At=	0.909593771											
gamma=	1.536930013											
d=	1.269170178											
x=	0.886282299											
y=	0.481253776											
k=	1.0000	2.0000	3.0000	4.0000	5.0000	6.0000	7.0000	8.0000	9.0000	10.0000	11.0000	
alpha_k=	0.5176	1.0000	1.4142	1.7321	1.9319	2.0000	1.9319	1.7321	1.4142	1.0000	0.5176	
beta_k=	1.9319	1.7321	1.4142	1.0000	0.5176	0.0000	-0.5176	-1.0000	-1.4142	-1.7321	-1.9319	
b()=	1.2783	3.5906	5.0171	4.4436	2.7559	1.8702	2.7559	4.4436	5.0171	3.5906	1.2783	
nm()=	1.0000	3.0000	5.0000	7.0000	9.0000	11.0000	13.0000	15.0000	17.0000	19.0000	21.0000	
Gk=	**2.5561**	**0.8962**	**3.3962**	**0.8761**	**2.8071**	**0.3785**						

Figure 41 Calculation details for 6^{th}-order unequally-terminated Chebyshev lowpass filter

4 Inverse Chebyshev Lowpass Filters

The Chebyshev loss characteristic was given earlier by (4.1) where $F(\omega)$ was given by (4.15) which was an N^{th}-order Chebyshev polynomial. The power-gain characteristic for the frequency-normalized inverse Chebyshev lowpass filter is given by

$$P(\omega) = 10\log_{10}\left\{ \frac{\left|\delta F(\omega)\right|^2}{1+\left|\delta F(\omega)\right|^2} \right\} \text{ dB} \tag{5.1}$$

where

$$\delta = 10^{-0.05 A_{stopdB}} \tag{5.2}$$

$$F(\omega) = \cos\left[N \cos^{-1}\left(\frac{1}{\omega} \right) \right] \tag{5.3}$$

and A_{stopdB} is the minimum equal-ripple stopband attenuation required in dB. Parameter N is the order of the filter. The ε corresponding to the associated Chebyshev filter is given by

$$\varepsilon = \frac{\delta}{\sqrt{1-\delta^2}} \tag{5.4}$$

The (normalized) –3 dB passband frequency is given by

$$\omega_{3dB} = \frac{1}{\cosh\left[\frac{1}{N} \cosh^{-1}\left(\frac{1}{\delta} \right) \right]} \tag{5.5}$$

The (normalized) radian frequency at which the passband gain is A_{passdB} is given by

$$\omega_{pass} = \frac{1}{\cosh\left[\frac{1}{N} \cosh^{-1}\left(\frac{1}{\delta} \sqrt{\frac{\alpha_0}{1-\alpha_0}} \right) \right]} \tag{5.6}$$

where

$$\alpha_0 = 10^{-0.1 A_{passdB}} \tag{5.7}$$

The (normalized) poles for the associated Chebyshev filter are given by

$$\begin{aligned} \sigma_{poles1_{kk}} &= -\sinh(\nu_0)\sin\left[\frac{(2k+1)\pi}{2N} \right] \\ \omega_{poles1_{kk}} &= \cosh(\nu_0)\cos\left[\frac{(2k+1)\pi}{2N} \right] \end{aligned} \tag{5.8}$$

for $k = \{0,1,...,N-1\}$ where

$$\nu_0 = \frac{1}{N}\sinh^{-1}\left(\frac{1}{\varepsilon}\right) \tag{5.9}$$

The (normalized) poles for the inverse Chebyshev filter are found from the associated Chebyshev poles as

$$\begin{aligned} \sigma_{poles_k} &= \frac{\sigma_{poles1_k}}{\left(\sigma_{poles1_k}\right)^2 + \left(\omega_{poles1_k}\right)^2} \\ \omega_{poles_k} &= \frac{\omega_{poles1_k}}{\left(\sigma_{poles1_k}\right)^2 + \left(\omega_{poles1_k}\right)^2} \end{aligned} \tag{5.10}$$

The (normalized) zeros for the inverse Chebyshev filter are given by

$$\omega_{zeros_k} = \frac{1}{\cos\left[\frac{(2k+1)\pi}{2N}\right]} \tag{5.11}$$

Unlike the Butterworth or Chebyshev poles, the poles of the inverse Chebyshev filter do not follow a recognizable pattern. This fact is illustrated[18] in Figure 42 and Figure 43 with the associated attenuation characteristics shown in Figure 44 and Figure 45.

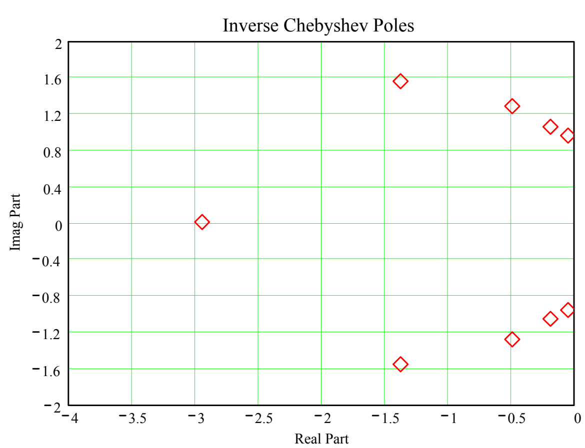


Figure 42 Inverse Chebyshev normalized pole locations for *N* = 9 and 20 dB minimum stopband attenuation

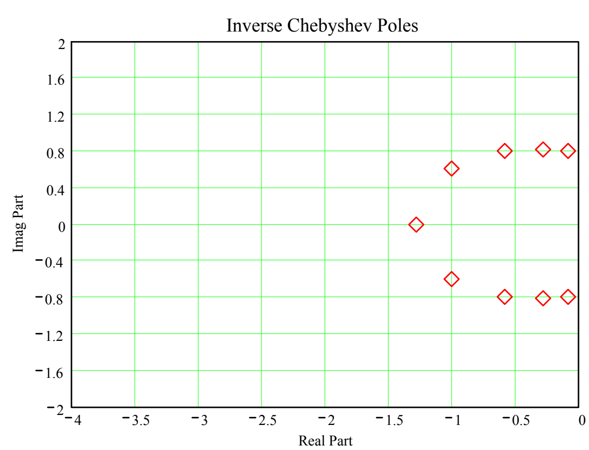


Figure 43 Inverse Chebyshev normalized pole locations for *N* = 9 and 50 dB minimum stopband attenuation

[18] From U22136 Inverse Chebyshev.mcd.

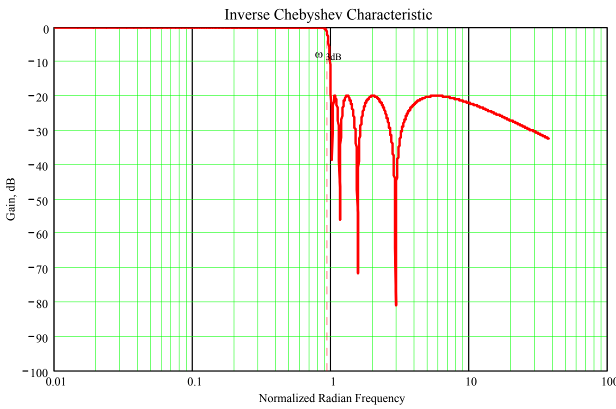


Figure 44 *N* = 9 inverse Chebyshev filter with 20 dB minimum stopband attenuation

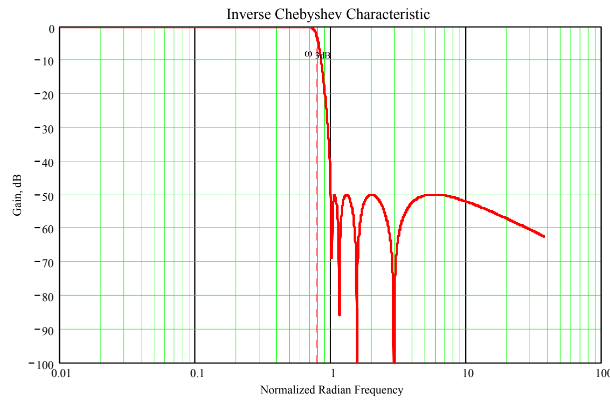


Figure 45 *N* = 9 inverse Chebyshev filter with 50 dB minimum stopband attenuation

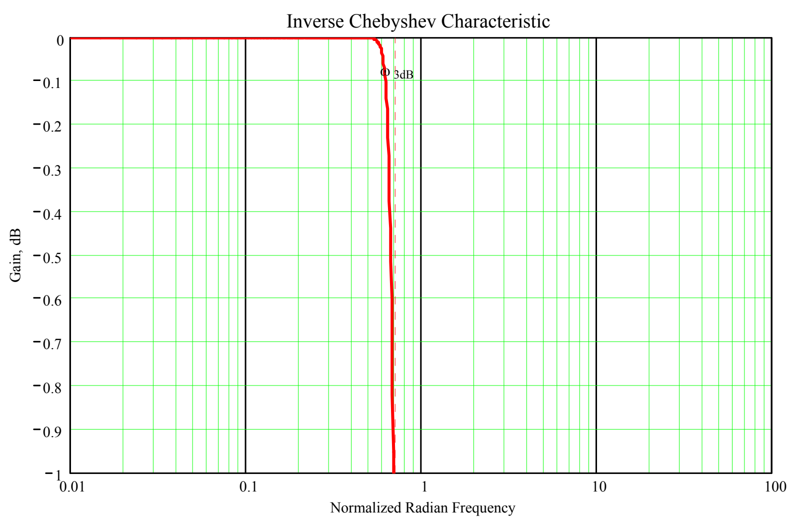


Figure 46 Close-up of passband characteristic for *N* = 9 inverse Chebyshev filter exhibiting 50 dB minimum stopband attenuation

The attenuation characteristic of some inverse Chebyshev filters like that shown in Figure 45 could easily be mistaken for an elliptical filter. The distinguishing characteristic between the two filters is that the elliptical filters are equal-ripple in the passband as well as the stopband whereas the inverse Chebyshev filters are not. A close-up of the passband characteristic associated with Figure 45 is shown in Figure 46.

Interestingly enough, the *inverse Butterworth* characteristic (if attempted) results in the original Butterworth filter! This occurs because all of the (normalized) Butterworth poles lay on a unit-circle which results in the denominator values in (5.10) all being unity.

5 Gaussian Lowpass Filters

The Paley-Weiner criterion determines whether a specified amplitude response can be physically realized by a causal filter or not [19]. If the amplitude response in question is represented by $\left|H\left(j\omega\right)\right|$, realizability demands that

$$\int_{-\infty}^{+\infty} \frac{\log_e\left[\left|H\left(j\omega\right)\right|\right]}{1+\omega^2} d\omega < \infty \tag{5.12}$$

For the true Gaussian-shaped attenuation characteristic,

$$\left|H_{Gauss}\left(j\omega\right)\right| = k_0 \exp\left[-k_1\omega^2\right] \tag{5.13}$$

Consequently,

$$\int_{-\infty}^{+\infty} \frac{\log_e\left(k_0\right) - k_1\omega^2}{1+\omega^2} d\omega \to \infty \tag{5.14}$$

This result means that the true Gaussian filter shape can only be approximated over a finite frequency range in order for the filter to be physically realizable.

The only design parameters for the approximate Gaussian filters are (i) the extent of the approximation which is usually taken as attenuation levels of 6 dB or 12 dB, and (ii) the order of the filter. Williams [4] refers to these filters as *transitional filters* in that the characteristics lie between the Chebyshev and Bessel filter families. Other so-called transitional filters can be constructed between the Butterworth and Bessel filter families of course. The derivation details behind the transitional filters in Williams is sketchy at best and seems to have been lost in antiquity! Williams comments that these filters were generated by mathematical techniques which involve interpolation of pole locations, but no other details are provided. These approximate Gaussian filters are all-pole in nature and the filter poles are given in Table 3.

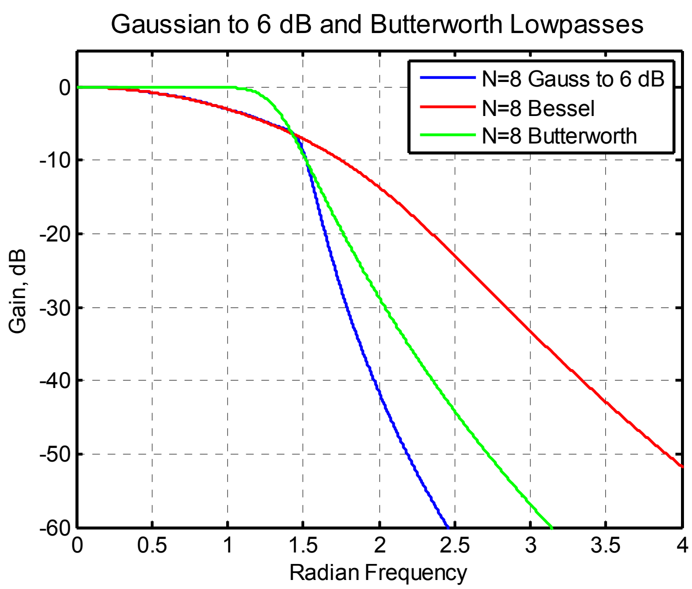


Figure 47 8th-order Gaussian, Butterworth, and Bessel filters compared[19]

[19] From u22365_transitional_filters.m.

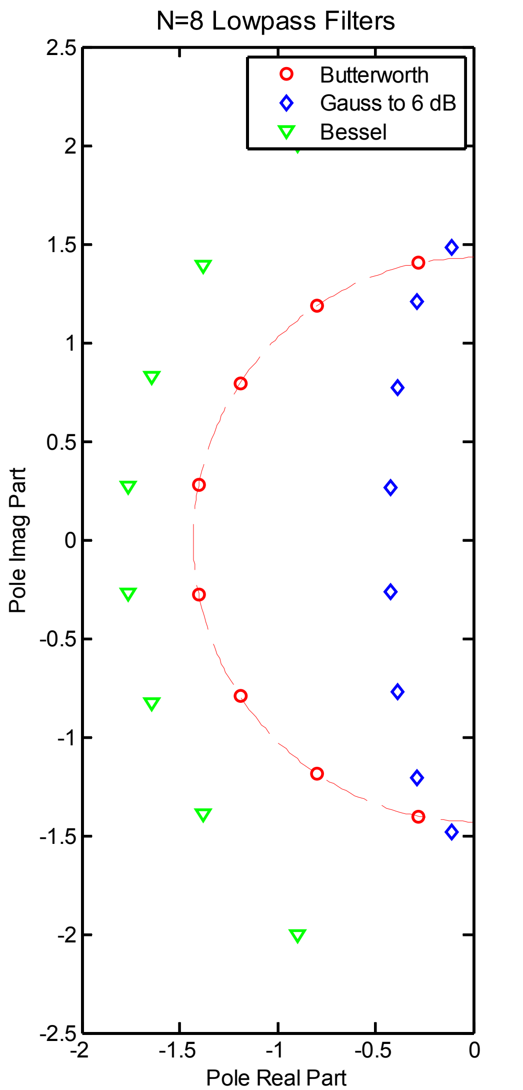


Figure 48 Poles locations of 8th-order Bessel, Butterworth, and Gaussian filters compared[20]

The attenuation characteristics of the 8th-order Butterworth, Bessel, and Gaussian to 6 dB filters are compared in Figure 47. The pole locations for these same filters are compared in Figure 48. Group delay characteristics for the filters are compared in Figure 49.

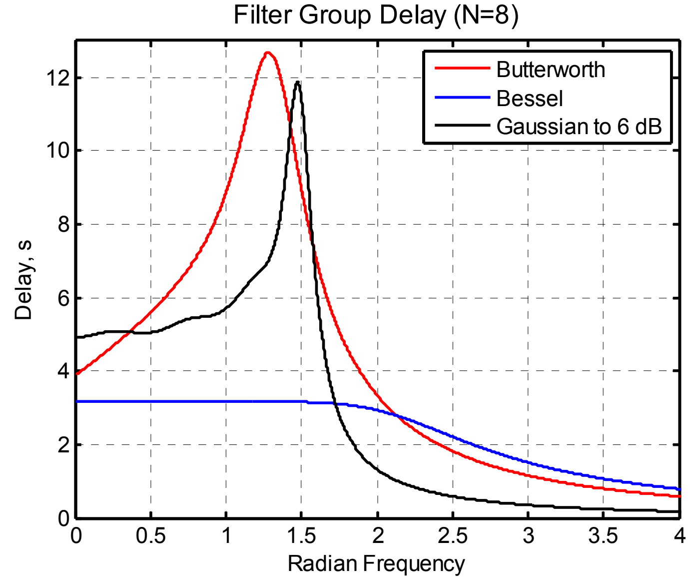


Figure 49 Group delay filter characteristics compared[21]

Several of the Gaussian to 6 dB filters are compared to the ideal Gaussian filter shape in Figure 50. The filters break from the ideal Gaussian shape at different radian frequencies depending upon the order of the filter. Within the 6 dB filter bandwidth, the approximate Gaussian filters approximate the ideal Gaussian shape in an almost equal-ripple manner as shown in Figure 51.

Table 3 Normalized Gaussian Filter Poles[22]

Gaussian to 6 dB				Gaussian to 12 dB		
Order	$-\sigma$	$\pm\omega$		**Order**	$-\sigma$	$\pm\omega$
3	0.9622 0.9776	1.2214		3	0.9360 0.9630	1.2168
4	0.7940 0.6304	0.5029 1.5407		4	0.9278 0.9192	1.6995 0.5560
5	0.619 0.3559 0.6650	0.8254 1.5688		5	0.8075 0.7153 0.8131	0.9973 2.0532
6	0.5433	0.3431		6	0.7019	0.4322

20 From u22365_transitional_filters.m.
21 From u22365_transitional_filters.m.
22 Tables 12-50 and 12-51 from [4].

Gaussian to 6 dB				Gaussian to 12 dB		
Order	$-\sigma$	$\pm\omega$		**Order**	$-\sigma$	$\pm\omega$
	0.4672 0.2204	0.9991 1.5067			0.6667 0.4479	1.2931 2.1363
7	0.4580 0.3649 0.1522 0.4828	0.5932 1.1286 1.4938		7	0.6155 0.5486 0.2905 0.6291	0.7703 1.5154 2.1486
8	0.4222 0.3833 0.2878 0.1122	0.2640 0.7716 1.2066 1.4798		8	0.5441 0.5175 0.4328 0.1978	0.3358 0.9962 1.6100 2.0703
9	0.3700 0.3230 0.2309 0.08604 0.3842	0.4704 0.9068 1.2634 1.4740		9	0.4961 0.4568 0.3592 0.1489 0.5065	0.6192 1.2145 1.7429 2.1003
10	0.3384 0.3164 0.2677 0.1849 0.06706	0.2101 0.6180 0.9852 1.2745 1.4389		10	0.4535 0.4352 0.3886 0.2908 0.1136	0.2794 0.8289 1.3448 1.7837 2.0599

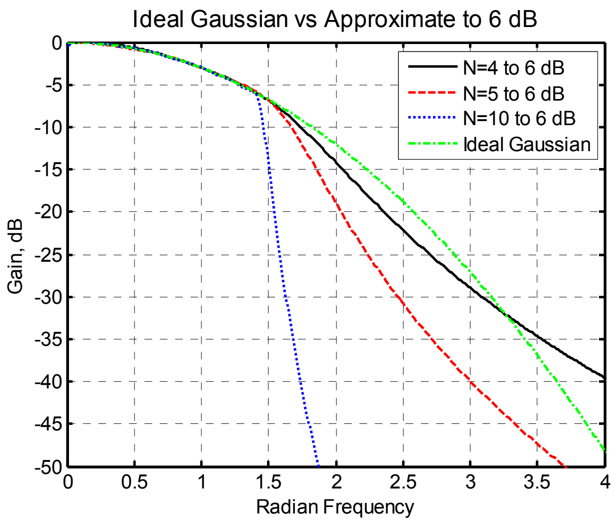


Figure 50 Gaussian to 6 dB filters compared to the ideal Gaussian filter shape[23]

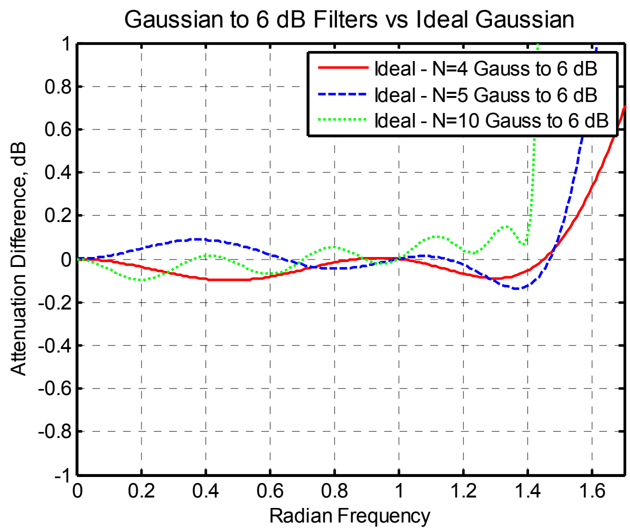


Figure 51 Gaussian to 6 dB filters compared to the ideal Gaussian shape.[24] The former approximate the ideal Gaussian shape in nearly a Chebyshev manner as shown.

[23] Using u22357_gaussian_to_xdb.m or u22365_transitional_filters.m.
[24] Ibid.

5.1 Approximate Gaussian Filters

The Gaussian to *x* dB filter approximations in Table 3 appear to be some kind of Chebyshev curve-fit in the passband with the ideal Gaussian shape, transitioning to a stopband shape that has the steepness of traditional Chebyshev filters. As mentioned earlier, however, the precise objective filter shape seems to have been lost in antiquity.

Some experimentation with a candidate function has proved promising, however[25]. The passband frequency edge f_p is defined here as the frequency at which the ideal Gaussian shape exhibits x_{dB} of attenuation. More specifically,

$$\begin{aligned} -x_{dB} &= 10\log_{10}\left\{\exp\left[-\left(\gamma f_p\right)^2\right]\right\} \text{ dB} \\ &= 10\frac{\log_e\left\{\exp\left[-\left(\gamma f_p\right)^2\right]\right\}}{\log_e(10)} = -\frac{10}{\log_e(10)}\left(\gamma f_p\right)^2 \end{aligned} \tag{5.15}$$

The –3 dB frequency is consequently given by $\gamma f_{-3dB} = \sqrt{\frac{3\log_e(10)}{10}} = 0.83113$ and this relationship can be used to compute γ for a specified –3 dB bandwidth value.

Similarly, the N^{th}-order Chebyshev stopband attenuation is given by

$$A_{cheby\_dB}(\omega) = 10\log_{10}\left\{1+\varepsilon^2\cosh^2\left[N\cosh^{-1}\left(\frac{\omega}{\omega_{rip}}\right)\right]\right\} \text{ dB} \tag{5.16}$$

where ω_{rip} is the radian frequency associated with the ripple bandwidth and

$$\varepsilon = \sqrt{10^{A_{rip}/10}-1} \tag{5.17}$$

for a passband ripple of A_{rip} dB. The radian frequency at which the Chebyshev filter attenuation is x_{dB} is given by

$$\frac{\omega_{xdB}}{\omega_{rip}} = \cosh\left[\frac{1}{N}\cosh^{-1}\left(\sqrt{\frac{10^{x_{dB}/10}-1}{10^{A_{rip}/10}-1}}\right)\right] \tag{5.18}$$

Equations (5.15) and (5.18) can be used to ensure that the objective attenuation characteristic is piecewise continuous in nature. Assuming that the –3 dB radian frequency for the Gaussian filter is known, the value for γ is given by $\gamma = 0.83113 / f_{-3dB}$ and the frequency associated with x_{dB} of attenuation is

$$f_p = \frac{1}{\gamma}\sqrt{\frac{\log_e(10)}{10}x_{dB}} \tag{5.19}$$

From here, ω_{xdB} is calculated directly using (5.18).

The complete objective attenuation function is then given by

[25] u22357_gaussian_to_xdb.m.

$$A_{dB}\left(f\right)=\begin{Bmatrix}\dfrac{10}{\log_e\left(10\right)}\left(\gamma f\right)^2 & f\le f_p\\ A_{cheby\_dB}\left(2\pi f\right) & f>f_p\end{Bmatrix} \tag{5.20}$$

The attenuation for the all-pole filter can be written as

$$\begin{aligned}A_{dB}\left(\omega\right)&=-20\log_{10}\left[\prod_{k=1}^{N}\frac{-p_k}{s-p_k}\right]\\&=\begin{Bmatrix}-20\log_{10}\left(\left|\dfrac{\sigma_{\frac{N+1}{2}}}{j\omega-\sigma_{\frac{N+1}{2}}}\right|\right)-10\log_{10}\left[\displaystyle\prod_{k=1}^{\frac{N-1}{2}}\frac{\left(\sigma_k^2+\omega_k^2\right)^2}{\left(\omega_k^2+\sigma_k^2-\omega^2\right)^2+\left(2\sigma_k\omega\right)^2}\right] & \text{for } N \text{ odd}\\ -10\log_{10}\left[\displaystyle\prod_{k=1}^{\frac{N}{2}}\frac{\left(\sigma_k^2+\omega_k^2\right)^2}{\left(\omega_k^2+\sigma_k^2-\omega^2\right)^2+\left(2\sigma_k\omega\right)^2}\right] & \text{for } N \text{ even}\end{Bmatrix}\end{aligned} \tag{5.21}$$

where the filter poles p_k are assumed to be ordered appropriately.

For an individual complex pole, it is straight forward to show that the partial derivatives of interest are

$$\frac{\partial A_{dB}\left(\omega\right)}{\sigma_k}=\frac{40}{\log_e\left(10\right)}\frac{\sigma_k}{\sigma_k^2+\omega_k^2}-\frac{40}{\log_e\left(10\right)}\frac{\left(\sigma_k^2+\omega_k^2-\omega^2\right)\sigma_k+2\sigma_k\omega^2}{\left(\sigma_k^2+\omega_k^2-\omega^2\right)^2+\left(2\sigma_k\omega\right)^2} \tag{5.22}$$

$$\frac{\partial A_{dB}\left(\omega\right)}{\omega_k}=\frac{40}{\log_e\left(10\right)}\frac{\omega_k}{\sigma_k^2+\omega_k^2}-\frac{40}{\log_e\left(10\right)}\frac{\left(\sigma_k^2+\omega_k^2-\omega^2\right)\omega_k+2\omega_k\omega^2}{\left(\sigma_k^2+\omega_k^2-\omega^2\right)^2+\left(2\sigma_k\omega\right)^2} \tag{5.23}$$

A simple gradient-based least-mean-square solution usually finds a very good solution but the objective function choice still results in a bit more ripple near the passband edge than the original transitional filters given by Zverev and Williams.

6 Adjustable Gaussian Lowpass Filters

The folks at Iowa Hills Software[26] have scripted a new type of filter they call *adjustable Gaussian lowpass filters*. The filter family is said to be a compromise between a Gaussian filter and a Butterworth filter by way of a single parameter γ which will be described shortly.

This filter is an all-pole filter. The polynomial associated with its loss function (e.g., see (3.4)) is given by

$$P(s)=1-s^2+\left(\frac{1}{2!}\right)^{\gamma}s^4-\left(\frac{1}{3!}\right)^{\gamma}s^6+\left(\frac{1}{4!}\right)^{\gamma}s^8-\left(\frac{1}{5!}\right)^{\gamma}s^{10}\pm \bullet\bullet\bullet \tag{5.24}$$

where the maximum order of *s* is equal to two-times the order of the filter. The poles of (5.24) reside in both halves of the complex *s*-plane whereas only the poles in the left-half plane are retained for the physical filter implementation.

For programming purposes, Iowa Hills constrains the user's γ such that $-1\le\gamma\le 1$ but the sign of the value is subsequently flipped, and the value multiplied by two if the user's value is greater than zero. Iowa Hills also scales-up the imaginary portion of each pole by a factor of 1.1 to improve filter group delay flatness.

A number of design examples follow[27]. One notable difference compared to the Iowa Hills results is that the computed filter poles are frequency-scaled so that the maximum pole modulus is always unity.

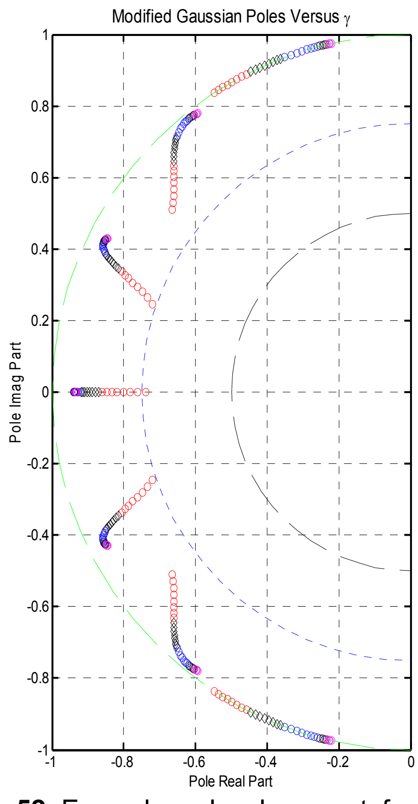


Figure 52 Example pole placement for 7th-order filter

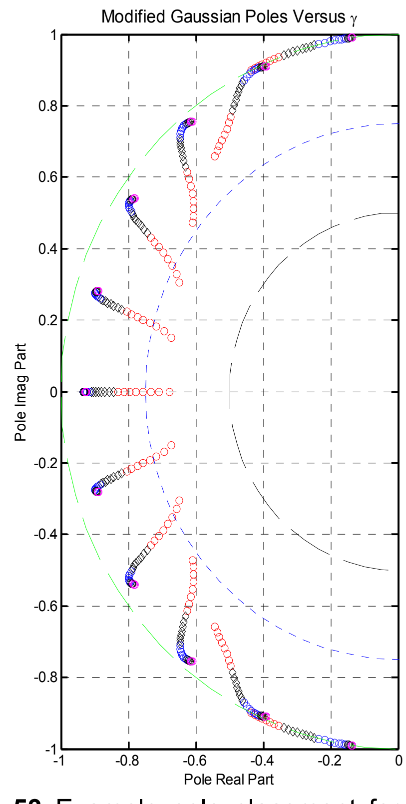
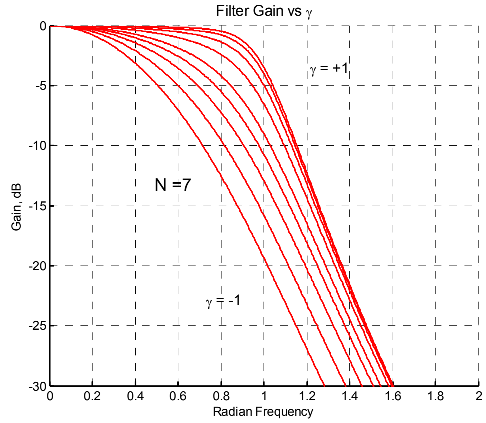


Figure 53 Example pole placement for 11th-order filter

[26] http://www.iowahills.com/7AAdjGaussAlgorithm.html .
[27] u22176_adjustable_gaussian.m.

Table 4 Attenuation and Group Delay for 7th-Order Filters Versus γ



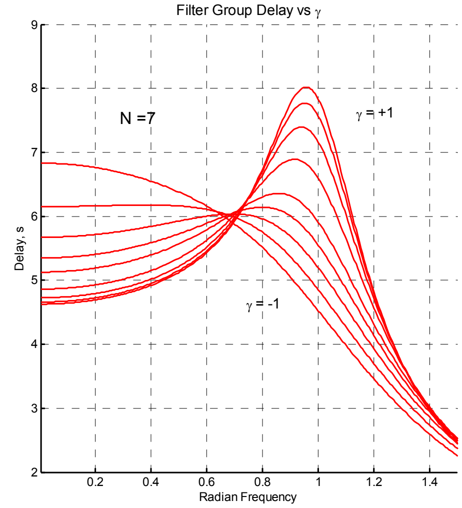
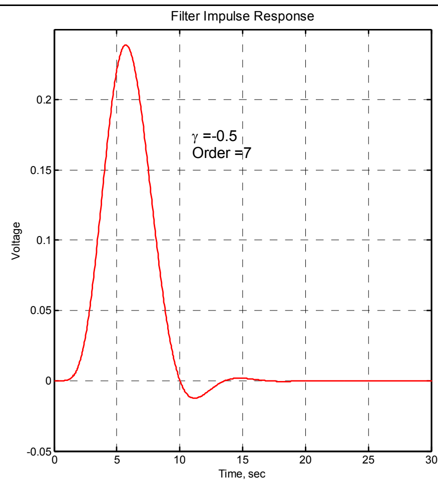
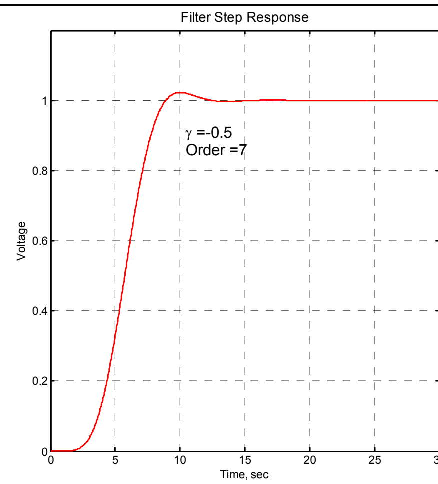
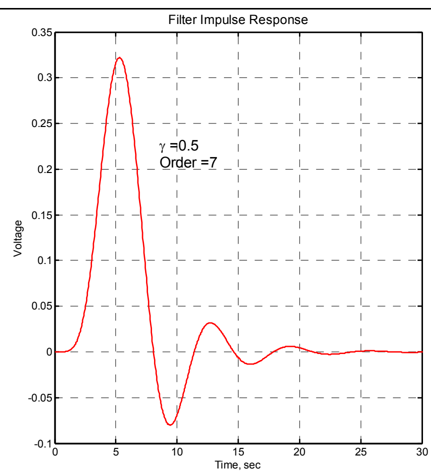
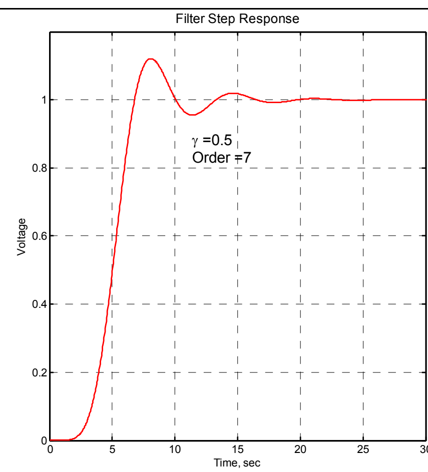


Table 5 Impulse and Step-Responses for 7th-Order Filters Versus γ









7 Bessel Filters

Bessel filters are known for their very flat group delay characteristic and associated pristine impulse response. Bessel filters are all-pole filters in which the poles can be determined by equally spacing the imaginary parts of the poles and choosing the real part of the poles such that they all lie on a circle [21]. Pole placement for a 5th-order Bessel filter is shown in Figure 54.

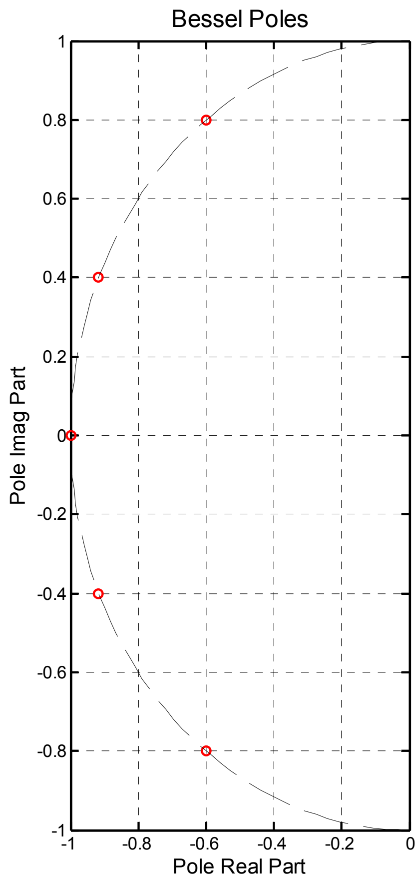


Figure 54 Pole placement for 5th-order Bessel lowpass filter

The transfer function of the Bessel filter is a rational function whose denominator is a reverse Bessel polynomial, such as those given in Table 6.[28]

The (reverse) Bessel polynomials of Table 6 are mathematically given by

$$\theta_n(s) = \sum_{k=0}^{n} a_{n,k} s^k \qquad (5.25)$$

where

$$a_{n,k} = \frac{(2n-k)!}{2^{n-k} k!(n-k)!} \qquad (5.26)$$

The (reverse) Bessel polynomials may also be formulated using a recursion formula where

$$\begin{aligned} \theta_0(x) &= 1 \\ \theta_1(x) &= x+1 \\ \theta_n(x) &= (2n-1)\theta_{n-1}(x) + x^2\theta_{n-2}(x) \end{aligned} \qquad (5.27)$$

For an n^{th} order filter, the first n – 1 terms in the series expansion for the group delay are zero, thereby maximizing the flatness at zero frequency.

Table 6 Reverse Bessel Polynomials

n	Reverse Bessel Polynomial
1	$s+1$
2	s^2+3s+3
3	$s^3+6s^2+15s+15$
4	$s^4+10s^3+45s^2+105s+105$
5	$s^5+15s^4+105s^3+420s^2+945s+945$

[28] Wikipedia, “Bessel filter”.

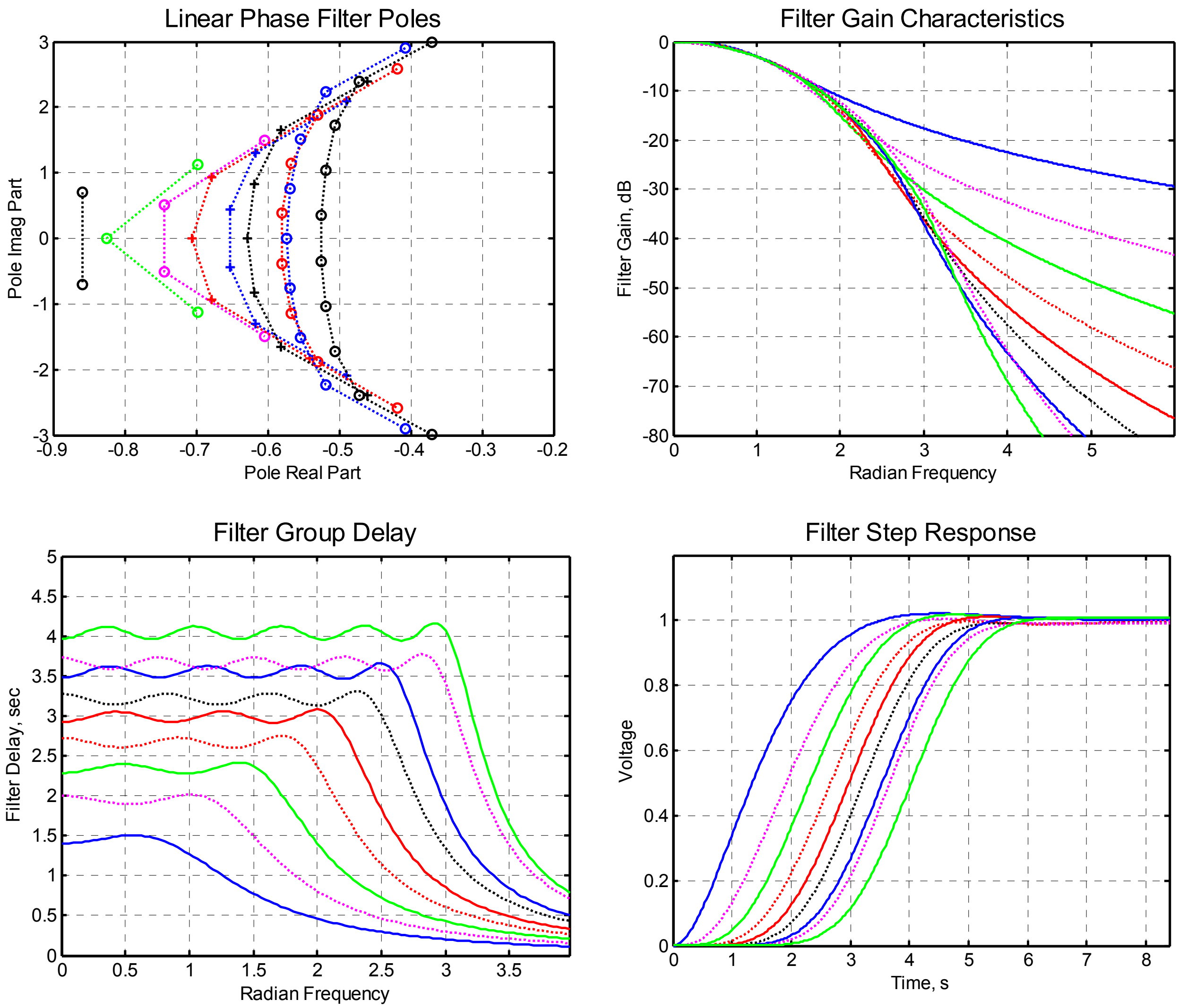
8 Linear Phase Filters

A closed-form method for computation of the pole locations is not available for linear phase filters. The pole locations are developed by iterative techniques [4]. Poles for equiripple linear phase filters of 0.05^o and 0.50^o error are provided in Table 7.

Table 7 Linear Phase Filter Poles from [4]

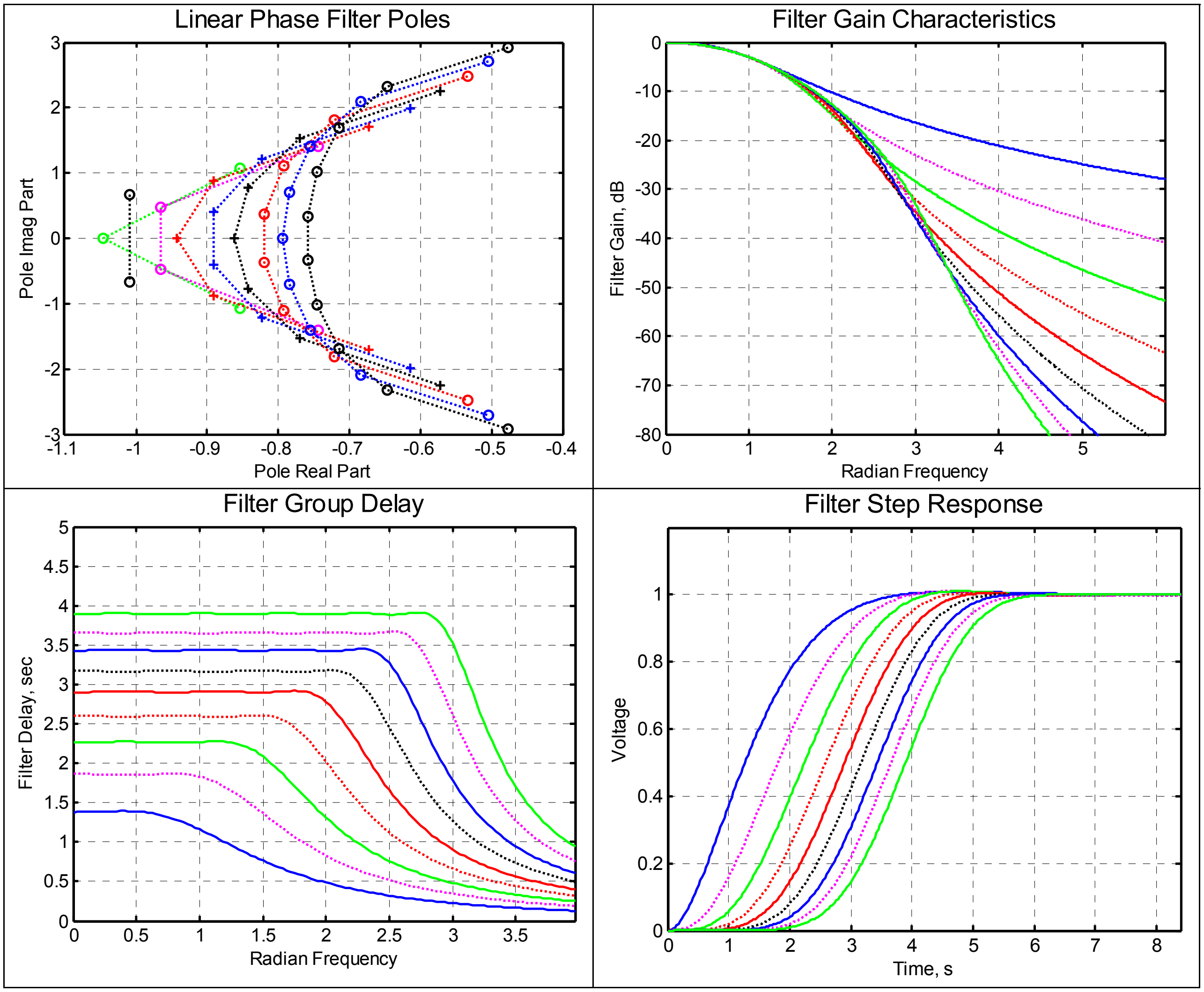
	0.05^o Error			***0.50^o Error***	
N	***Real Part*** $-\alpha$	***Imag Part*** $\pm j\beta$		***Real Part*** $-\alpha$	***Imag Part*** $\pm j\beta$
2	1.0087	0.6680		0.8590	0.6981
3	0.8541 1.0459	1.0725		0.6969 0.8257	1.1318
4	0.9648 0.7448	0.4748 1.4008		0.7448 0.6037	0.5133 1.4983
5	0.8915 0.6731 0.9430	0.8733 1.7085		0.6775 0.5412 0.7056	0.9401 1.8256
6	0.8904 0.8233 0.6152	0.4111 1.2179 1.9810		0.6519 0.6167 0.4893	0.4374 1.2963 2.0982
7	0.8425 0.7708 0.5727 0.8615	0.7791 1.5351 2.2456		0.6190 0.5816 0.4598 0.6283	0.8338 1.6453 2.3994
8	0.8195 0.7930 0.7213 0.5341	0.3711 1.1054 1.8134 2.4761		0.5791 0.5665 0.5303 0.4184	0.3857 1.1505 1.8914 2.5780
9	0.7853 0.7555 0.6849 0.5060 0.7938	0.7125 1.4127 2.0854 2.7133		0.5688 0.5545 0.5179 0.4080 0.5728	0.7595 1.5089 2.2329 2.9028
10	0.7592 0.7467 0.7159 0.6475 0.4777	0.3413 1.0195 1.6836 2.3198 2.9128		0.5249 0.5193 0.5051 0.4711 0.3708	0.3487 1.0429 1.7261 2.3850 2.9940

Table 8 Linear Phase to 0.50° Filter Characteristics[29]



[29] u22437_linphase_0pt50.m.

Table 9 Linear Phase to 0.05° Filter Characteristics[30]



30 u22437_linphase_0pt50.m.

9 Transitional Filters

A fairly wide variety of *transitional filters* can be found in the literature. For example, *BeBut* filters are defined in [22] as a filter family which transition from the Bessel shape to the Butterworth shape. These filters are obtained via the recurrent relationship

$$u_{n+1}(s) = \left[\frac{2n+d}{s}\right] u_n(s) + u_{n-1}(s) \tag{5.28}$$

where $n \geq 1$, d is a design parameter between 0 and 1, and

$$\begin{aligned} u_0(s) &= 1 \\ u_1(s) &= 1 + \frac{1}{s} \end{aligned} \tag{5.29}$$

When d = 1, the resulting polynomials correspond to Bessel filters whereas d = 0 corresponds to filters very similar to Butterworth filters. For the d = 1 case (Bessel), the first few polynomials are

$$\begin{aligned} w_0(s) &= 1 \\ w_1(s) &= 1 + \frac{1}{s} \\ w_2(s) &= 1 + \frac{3}{s} + \frac{3}{s^2} \\ w_3(s) &= 1 + \frac{6}{s} + \frac{15}{s^2} + \frac{15}{s^3} \\ w_4(s) &= 1 + \frac{10}{s} + \frac{45}{s^2} + \frac{105}{s^3} + \frac{105}{s^4} \end{aligned} \tag{5.30}$$

The normal procedure is to cast a given polynomial into is normal form as

$$p_n(s) = s^n + a_{n-1}s^{n-1} + a_{n-2}s^{n-2} + \cdots + a_0 \tag{5.31}$$

from which the transfer function follows as

$$T_n(s) = \frac{a_0}{p_n(s)} = \frac{a_0}{s^n + a_{n-1}s^{n-1} + a_{n-2}s^{n-2} + \cdots + a_0} \tag{5.32}$$

In the case where d = 0, the 4th-order transfer function is given by

$$T_4(s) = \frac{48}{s^4 + 8s^3 + 32s^2 + 48s + 48} \tag{5.33}$$

Example pole loci[31] for a 5th-order and 8th-order BeBut filters as a function of parameter *d* are shown in Figure 55 and Figure 56. The pole locations do not mimic the Butterworth filter case very well, but the step-response of the filters are quite reasonable as shown in Figure 57 and Figure 58. The associated frequency-domain responses are shown in Figure 59 and Figure 60.

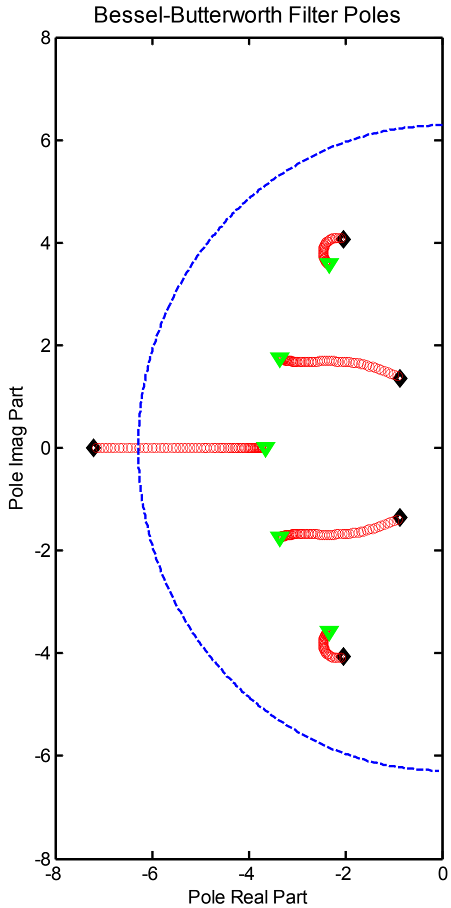


Figure 55 Pole loci [32] for 5th-order BeBut filter. *d* = 0 corresponds to the black diamonds whereas *d* = 1 correspond to the green triangles.

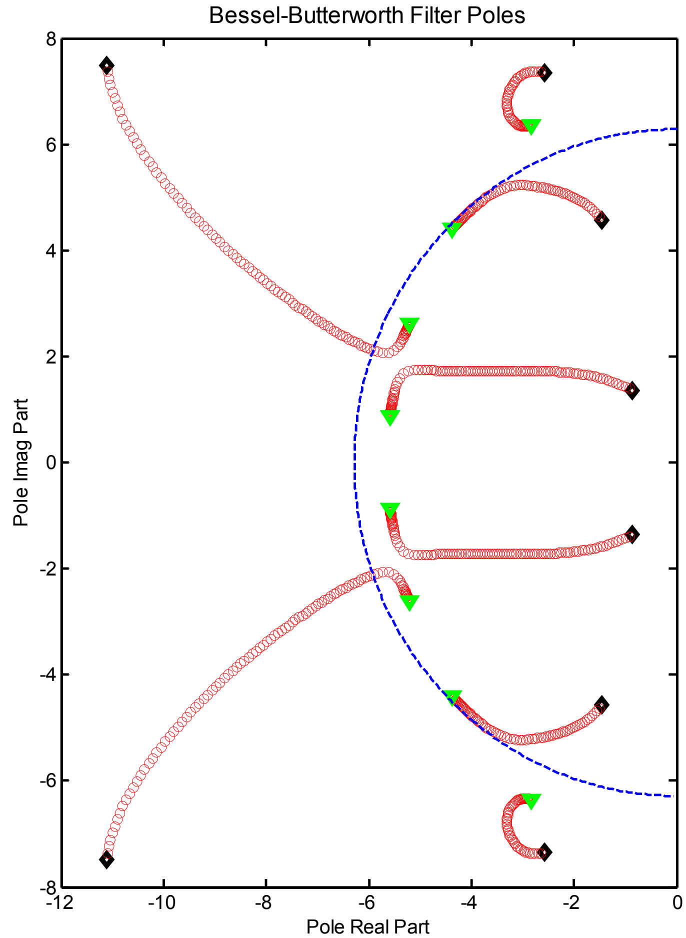


Figure 56 Pole loci for 8th-order BeBut filter. *d* = 0 corresponds to the black diamonds whereas *d* = 1 correspond to the green triangles.

[31] Computed using u22393_bebut_filters.m.
[32] From u22393_bebut_filters.m.

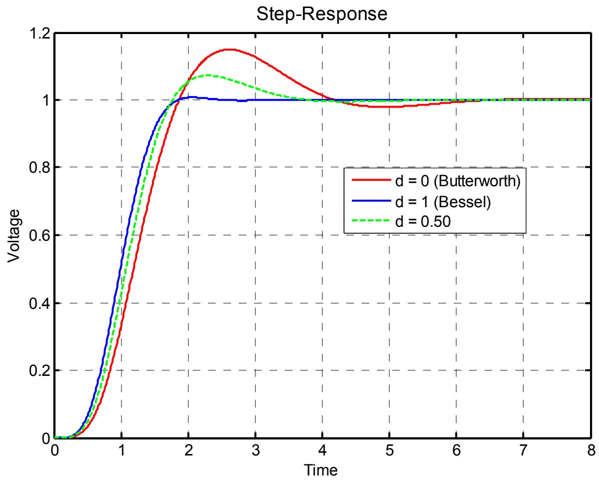


Figure 57 5th-order filter case step-response[33]

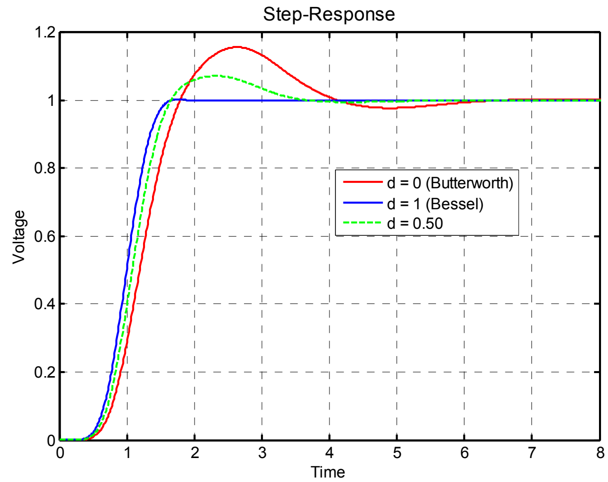


Figure 58 8th-order filter case step-response

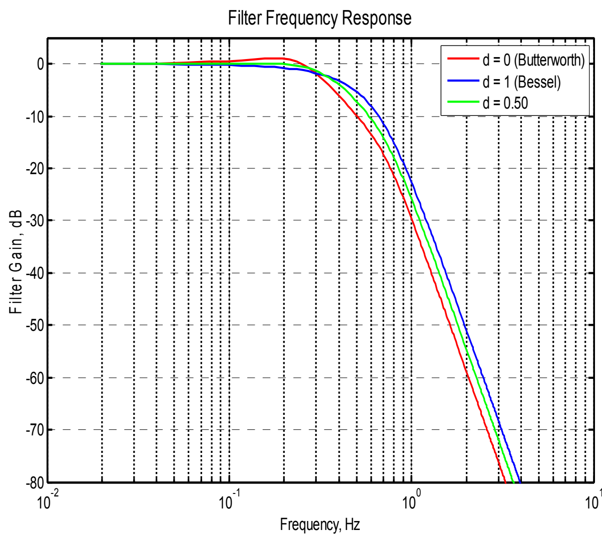


Figure 59 5th-order filter case[34]

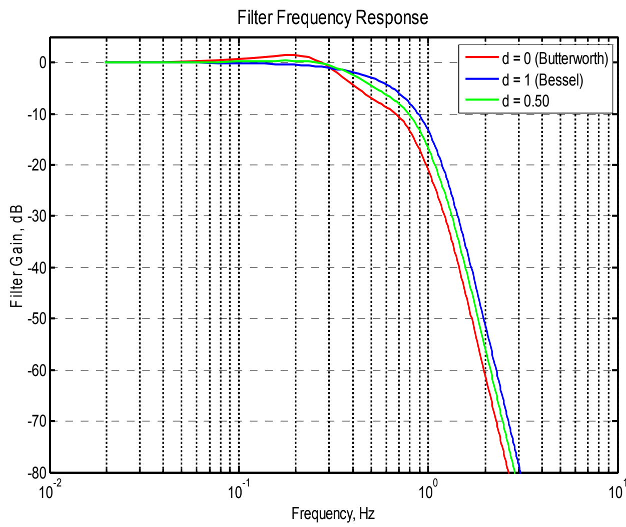


Figure 60 8th-order filter case

[33] Computed using u22393_bebut_filters.m.
[34] Computed using u22393_bebut_filters.m.

10 Elliptic Lowpass Filters[35]

Elliptic filters exhibit equal loss maximums in the passband and equal loss minimums in the stopband; they are often said to be equal-ripple in the passband and stopband. This filter type is more complicated than the Butterworth and Chebyshev filters considered thus far, so the discussion which follows is fairly lengthy.

Elliptic filters are first introduced here by considering a 5th-order elliptic lowpass filter. Most of the discussion is focused on the filter's loss characteristic denoted by $L(\omega^2)$. This naturally leads to material about the Jacobi elliptic functions and how they can be used to compute the poles and zeros of the transducer gain $T(s)$ function (See §1.2).

Early contributors (e.g., Saal & Ulbrich [18]) were more inclined to study elliptic filters in terms of their *characteristic function* which is denoted here by $K(s)$. This notation is unfortunate given that the complete elliptic integral of the first kind is denoted by K, but retaining the functional dependence on s should be sufficient to keep these two uses clearly separated. The ideal elliptic loss characteristic is not realizable without mutually-coupled transformers and or at least one negative component value for even-order LC filters. This difficulty is circumvented by developing multiple filter types (denoted by types *a, b,* and *c*) as discussed later in §10.3.1 and §10.3.2 for the even-order case. Odd-order elliptic filters are naturally symmetric and therefore more straight forward to design. Elliptic filter synthesis has traditionally been based upon the characteristic function approach (e.g., [18]) whereas the methodology due to Amstutz [11] is adopted here for the filter synthesis portion of this paper.

10.1 5th-Order Elliptic Filter Loss Characteristic

A representative loss characteristic for a 5th-order elliptic lowpass filter is shown in Figure 61. The passband and stopband frequency edges are respectively defined as

$$\omega_{pass} = \sqrt{k} \qquad \omega_{stop} = \frac{1}{\sqrt{k}} \tag{6.1}$$

where the ratio of passband to stopband frequencies is given by

$$k = \frac{\omega_{pass}}{\omega_{stop}} < 1 \tag{6.2}$$

Elliptic filters are frequently referred to in terms of their order, maximum passband reflection coefficient, and their modular angle θ. The passband reflection coefficient magnitude and passband attenuation ripple are related by

$$A_{pass} = -10\log_{10}\left(1-\left|\rho\right|^2\right) \text{ dB} \tag{6.3}$$

whereas the modular phase angle is given by

$$\theta = \sin^{-1}\left(\frac{\omega_{pass}}{\omega_{stop}}\right) \tag{6.4}$$

[35] There are a number of excellent treatises on the design of elliptic filters, notably [7], [8], [10], and [11].

It is convenient to further define

$$\varepsilon_p = \sqrt{10^{A_{pass}/10} - 1}$$
$$\varepsilon_s = \sqrt{10^{A_{stop}/10} - 1} \tag{6.5}$$

$$k_1 = \sqrt{\frac{10^{0.1A_{pass}} - 1}{10^{0.1A_{stop}} - 1}} = \frac{\varepsilon_p}{\varepsilon_s} \ll 1 \tag{6.6}$$

Analogous with the Chebyshev lowpass case, define the loss function as

$$L\left(\omega^2\right) = 1 + \varepsilon_p^2 F^2\left(\omega\right) \tag{6.7}$$

which is given in decibel form as

$$A_{dB}\left(\omega\right) = 10\log_{10}\left[L\left(\omega^2\right)\right] \text{ dB} \tag{6.8}$$

As true earlier for the Chebyshev filter case, $F(\omega)$, $L(\omega^2)$, and $L(-s^2)$ are all given by polynomial ratios. Following the lead information provided in Figure 61, the 5th-order elliptic lowpass filter must exhibit the following characteristics:

Requirement #1: $F\left(\omega\right) = 0$ at $\omega = 0, \pm\Psi_2, \pm\Psi_4$

Requirement #2: $F\left(\omega\right) = \infty$ at $\omega = \pm\Psi_5, \pm\Psi_7, \pm\infty$

Requirement #3: $F^2\left(\omega\right) = 1$ at $\omega = \pm\Psi_1, \pm\Psi_3, \pm\omega_{pass} = \sqrt{k}$

Requirement #4: $F^2\left(\omega\right) = \frac{1}{k_1^2}$ at $\omega = \pm\Psi_6, \pm\Psi_8, \pm\omega_{stop} = \frac{1}{\sqrt{k}}$

Requirement #5: $\frac{dL\left(\omega^2\right)}{d\omega} = 0$ at $\omega = \pm\Psi_1, \pm\Psi_3, \pm\Psi_6, \pm\Psi_8$

From Requirements #1 and #2, $F(\omega)$ must have the form

$$F\left(\omega\right) = M_1 \frac{\omega\left(\omega^2 - \Psi_2^2\right)\left(\omega^2 - \Psi_4^2\right)}{\left(\omega^2 - \Psi_5^2\right)\left(\omega^2 - \Psi_7^2\right)} \tag{6.9}$$

where the M_n are arbitrary constants. From Requirement #2 and #3, $1 - F^2(\omega)$ must be zero at the specified frequencies so that

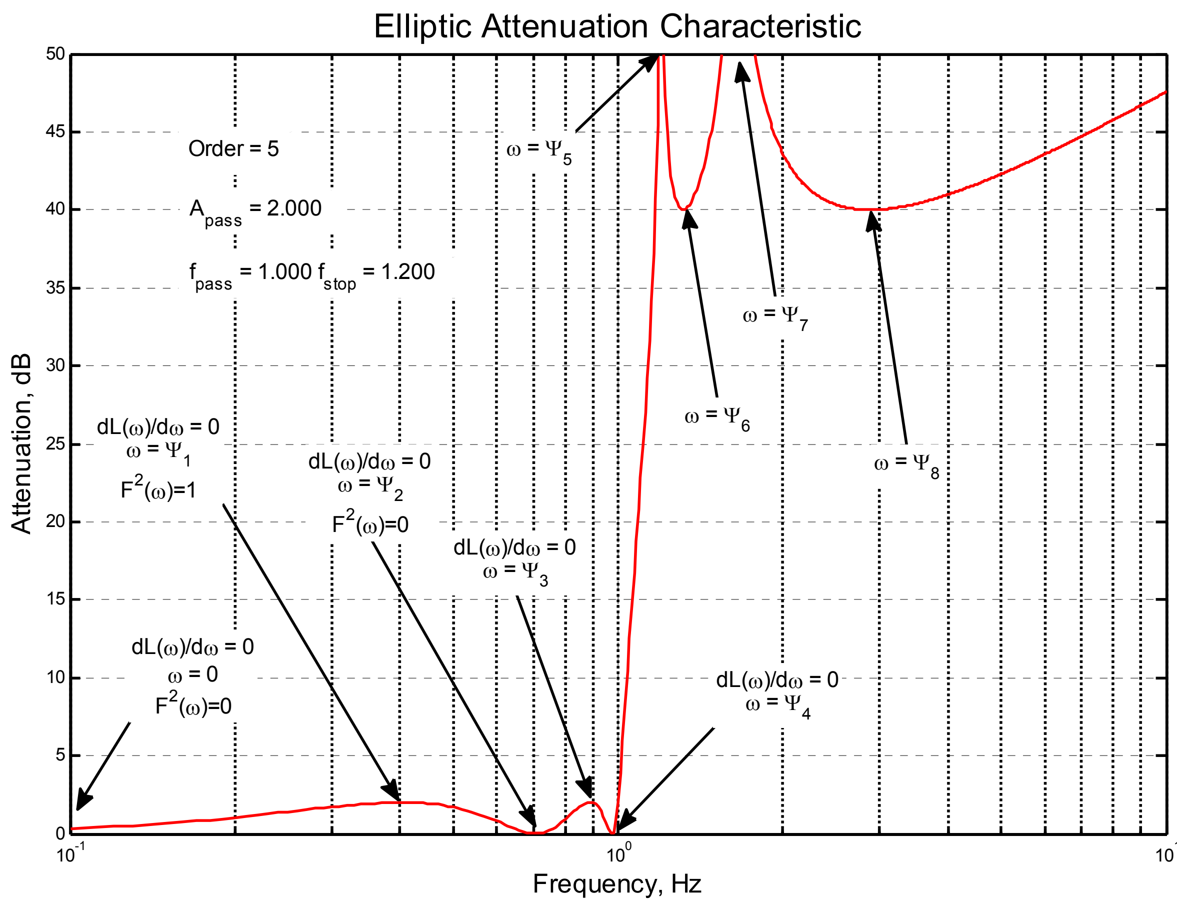


Figure 61 5th-order elliptic lowpass filter[36] with A_{pass} = 2 dB, A_{stop} = 40 dB, f_{pass} = 1 Hz, f_{stop} = 1.2 Hz

$$1-F^2(\omega)=M_2\frac{\left(\omega^2-\Psi_1^2\right)^2\left(\omega^2-\Psi_3^2\right)^2\left(\omega^2-k\right)^2}{\left(\omega^2-\Psi_5^2\right)^2\left(\omega^2-\Psi_7^2\right)^2} \tag{6.10}$$

The additional squaring of the two numerator terms is in anticipation of Requirement #5. Similarly from Requirements #2, #4, and #5

$$1-k_1^2F^2(\omega)=M_3\frac{\left(\omega^2-\Psi_6\right)^2\left(\omega^2-\Psi_8\right)^2\left(\omega^2-\frac{1}{k}\right)^2}{\left(\omega^2-\Psi_5\right)^2\left(\omega^2-\Psi_7\right)^2} \tag{6.11}$$

Requirement #5 along with the denominator portion of $F(\omega)$ already present in (6.9) dictates that

$$\begin{aligned}\frac{dF}{d\omega}&=M_4\frac{\left(\omega^2-\Psi_1^2\right)\left(\omega^2-\Psi_3^2\right)\left(\omega^2-\Psi_6^2\right)\left(\omega^2-\Psi_8^2\right)}{\left(\omega^2-\Psi_5^2\right)^2\left(\omega^2-\Psi_7^2\right)^2}\\&=M_4\left[\frac{\left(\omega^2-\Psi_1^2\right)\left(\omega^2-\Psi_3^2\right)}{\left(\omega^2-\Psi_5^2\right)^2\left(\omega^2-\Psi_7^2\right)^2}\right]\times\left(\omega^2-\Psi_6^2\right)\left(\omega^2-\Psi_8^2\right)\end{aligned} \tag{6.12}$$

[36] Computed using u18310_multi_lpf_designer.m.

Upon squaring (6.12) and then making use of (6.10) and (6.11), equation (6.12) can be rewritten as

$$\begin{aligned}\left|\frac{dF}{d\omega}\right|^2 &= M_5\left[\frac{1-F^2(\omega)}{\left(\omega^2-k\right)^2}\right]\times\left[\frac{\left(\omega^2-\Psi_6^2\right)^2\left(\omega^2-\Psi_8^2\right)^2}{\left(\omega^2-\Psi_5^2\right)^2\left(\omega^2-\Psi_7^2\right)^2}\right] \\ &= M_6\left[\frac{1-F^2(\omega)}{\left(1-\frac{\omega^2}{k}\right)^2}\right]\times\left[\frac{1-k_1^2F^2(\omega)}{\left(1-k\omega^2\right)^2}\right]\end{aligned} \tag{6.13}$$

In differential form, (6.13) can be further simplified as

$$\frac{dF}{\sqrt{\left(1-F^2\right)\left(1-k_1^2F^2\right)}}=\sqrt{M_6}\frac{d\omega}{\sqrt{\left(1-\frac{\omega^2}{k}\right)\left(1-k\omega^2\right)}} \tag{6.14}$$

Substituting $y=\omega/\sqrt{k}$ into the right-hand side of (6.14) and performing the implied definite integration leads to

$$\int_0^F\frac{dx}{\sqrt{\left(1-x^2\right)\left(1-k_1^2x^2\right)}}=M_7\int_0^{\omega/\sqrt{k}}\frac{dy}{\sqrt{\left(1-y^2\right)\left(1-k^2y^2\right)}}+M_8 \tag{6.15}$$

Both sides of (6.15) involve an elliptic integral which becomes more obvious by substituting $x=\sin(\phi)$ into the left-hand side, and $y=\sin(\theta)$ into the right-hand side. These substitutions transform (6.15) into

$$\int_0^{\phi=\sin^{-1}(F)}\frac{d\phi}{\sqrt{1-k_1^2\sin^2(\phi)}}=M_7\int_0^{\theta=\sin^{-1}\left(\omega/\sqrt{k}\right)}\frac{d\theta}{\sqrt{1-k^2\sin^2(\theta)}}+M_8 \tag{6.16}$$

Defining

$$z=\int_0^{\theta}\frac{d\varphi}{\sqrt{1-k^2\sin^2(\varphi)}} \tag{6.17}$$

the solution to (6.16) can be expressed in terms of two simultaneous equations given as

$$\frac{\omega}{\sqrt{k}}=\sin(\theta)=sn(z,k) \tag{6.18}$$

$$F=\sin(\phi)=sn\left(M_7z+M_8,k_1\right) \tag{6.19}$$

where *sn* (*u, v*) is known as the *elliptic sine* function.

Based upon the information developed thus far along with the material in §10.7.1, §10.7.4, and §10.7.3, it can be shown that (for *N* is odd)

$$F(\omega)=\frac{(-1)^{(N-1)/2}\,\omega}{\sqrt{k_1}}\prod_{n=1}^{\frac{N-1}{2}}\frac{\omega^2-\Psi_{zero,n}^2}{1-\omega^2\Psi_{zero,n}^2} \tag{6.20}$$

where the attenuation zeros of the elliptic filter are given by

$$\Psi_{zero,n}=\sqrt{k}\;sn\left(\frac{2Kn}{N},k\right)\quad \text{for n}=1,2,...,\frac{N-1}{2} \tag{6.21}$$

K is the complete elliptic integral (See §10.7.1), and *N* is the filter order. The attenuation poles are given by the reciprocal of the zeros as

$$\Psi_{pole,n}=\frac{1}{\Psi_{zero,n}} \tag{6.22}$$

10.2 Elliptic Filter Poles and Zeros

10.2.1 Loss Function Poles and Zeros (Odd N)

The poles and zeros of interest are in the context of (6.7) and involve the extended loss function given by

$$L\left(-s^2\right)=1+\varepsilon_p^2F^2\left(s\right) \tag{6.23}$$

which can be rewritten in terms of the transformed frequency variable *z* (see equ. (6.17) and Figure 88) as

$$L(z)=1+\varepsilon_p^2F^2(z) \tag{6.24}$$

Based upon (6.19) and other periodicity requirements,

$$F(z)=sn\left(\frac{NK_1z}{K},k_1\right) \tag{6.25}$$

Factoring (6.24) produces

$$L(z)=\left[1+j\,\varepsilon_p\,sn\left(\frac{NK_1z}{K},k_1\right)\right]\left[1-j\,\varepsilon_p\,sn\left(\frac{NK_1z}{K},k_1\right)\right] \tag{6.26}$$

and the zero-solutions are dictated by the solutions to

$$sn\left(\frac{NK_1z}{K},k_1\right)=\frac{j}{\varepsilon_p} \tag{6.27}$$

Continuing, this becomes[37]

$$sn\left(\frac{NK_1z}{K},k_1\right)\cong\sin\left(\frac{NK_1z}{K}\right)=\frac{j}{\varepsilon_p} \tag{6.28}$$

[37] The exact solution is developed in §10.7.7.

since k_1 is almost always extremely small. For example, if the passband ripple is 0.1 dB and the minimum stopband attenuation requirement is 30 dB, k_1 = 0.0048. (See §10.7.7.) If the stopband attenuation is increased to 50 dB, k_1 = 0.000483. It is therefore valid to take $K_1 = \pi / 2$ in (6.28) thereby leading to

$$-j\frac{N\pi z}{2K} \cong \sinh^{-1}\left(\frac{1}{\varepsilon_p}\right) \tag{6.29}$$

Using the identity $\sinh^{-1}(x) = \log_e\left(x+\sqrt{x^2+1}\right)$, one solution-zero to (6.24) is given by

$$z_0 \cong j\frac{K}{N\pi}\log_e\left(\frac{10^{A_{pass}/20}+1}{10^{A_{pass}/20}-1}\right) \tag{6.30}$$

Since the *sn*( ) function in (6.27) has a real period of 4*K* / *N*, all of the zeros are subsequently given by

$$z_n = z_0 + \frac{4K}{N}n \text{ for } n = 0,1,... \tag{6.31}$$

The *s*-plane zeros are finally found by transforming the z_n values in (6.31) by using the transformation between *z* and *s* given by (6.18). In general, however, the z_n values in (6.31) are complex. This issue can be handled by using the *addition formula*[38] for elliptic sines as given without proof by

$$sn(z_1+z_2,k) = \frac{sn(z_1,k)cn(z_2,k)dn(z_2,k)+cn(z_1,k)sn(z_2,k)dn(z_1,k)}{1-k^2sn^2(z_1,k)sn^2(z_2,k)} \tag{6.32}$$

Making use of (6.32) and (6.87) for a complex value $z = a + jb$ produces[39]

$$sn(a+jb,k) = \frac{sn(a,k)dn(b,k')+j\,sn(b,k')cn(a,k)cn(b,k')dn(a,k)}{cn^2(b,k')+k^2sn^2(a,k)sn^2(b,k')} \tag{6.33}$$

This result makes it simple to translate all of the zeros given by (6.31) to the *s*-plane based upon (6.18) with $\omega = -j\,s$ resulting in[40]

$$\sigma_n \pm j\omega_n = j\sqrt{k}\,sn\left(z_0+\frac{4K}{N}n\right) \text{ for } n = 1,2,...,N-1 \tag{6.34}$$

For *N* an odd integer, this may be rewritten as[41]

$$\begin{aligned}\sigma_n \pm j\omega_n &= j\sqrt{k}\,sn\left(z_0+\frac{2K}{N}n\right) \text{ for } n = 1,2,...,\frac{N-1}{2}\\ \sigma_0 &= j\sqrt{k}\,sn(z_0)\end{aligned} \tag{6.35}$$

The (double) poles are located at

38 See equ. (A.22) in [8].
39 Same as equ. (5.27) in [10].
40 Complex poles always appear along with their complex conjugate, hence the ± sign.
41 Result in chapter 5 of [8] includes an additional factor of $(-1)^n$ but this appears to be in error.

$$s_n = \pm \frac{j}{\Omega_n} \qquad \text{with} \qquad \Omega_n = \sqrt{k}\, sn\left(\frac{2K}{N} n, k\right) \tag{6.36}$$

and

$$F(\omega) = \frac{(-1)^r}{\sqrt{k_1}} \omega \prod_{n=1}^{r} \frac{\omega^2 - \Omega_n^2}{1 - \omega^2 \Omega_n^2} \;\; \text{with } r = \frac{N-1}{2} \tag{6.37}$$

The poles and zeros can be directly scaled to a ripple bandwidth of ω_p rad/sec by replacing $\sqrt{k}$ with ω_p in (6.34) and (6.36). See §16 for a number of detailed design examples.

10.2.2 Loss Function Poles and Zeros (Even *N*)

For even-order filters, the zeros of *L*( ) are given by

$$\sigma_n \pm j\omega_n = j\sqrt{k}\, sn\left[z_0 + K\left(\frac{2n-1}{N}\right), k \right] \;\; \text{for } n = 1, 2, ..., \frac{N}{2} \tag{6.38}$$

Similarly, the poles of *L*( ) are given by

$$s_n = \pm \frac{j}{\Omega_n} \qquad \text{with} \qquad \Omega_n = \sqrt{k}\, sn\left[K\left(\frac{2n-1}{N}\right), k \right] \text{ for } n = 1, 2, ..., \frac{N}{2} \tag{6.39}$$

The corresponding expression for *F*( ) is

$$F(\omega) = \frac{(-1)^r}{\sqrt{k_1}} \prod_{n=1}^{r} \frac{\omega^2 - \Omega_n^2}{1 - \omega^2 \Omega_n^2} \;\; \text{with } r = \frac{N}{2} \tag{6.40}$$

10.2.3 Characteristic Function Poles & Zeros (Odd *N*)

The characteristic function is given by $K(s) = \varepsilon_p F(s)$ from (6.7). The significance of knowing *K* (*s*) is that it plays an integral part in computing the input impedance of the filter versus frequency as given by (2.24) and (2.27). This function plays a vital role in traditional filter synthesis, but to a lesser extent in the Amstutz synthesis method which is used in §10.8. Based upon the Feldtkeller equation (2.15), the poles of *K*(*s*) must be the same as the poles of $T(s)$ which were just computed in §10.2 since $|T(s)|^2 = L(-s^2)$. The zeros of $K(s)$ are given by

$$s_n = \pm j\Omega_n \quad \text{with} \quad \Omega_n = \sqrt{k}\, sn\left(\frac{2Kn}{N}, k\right) \text{ for } n = 1, 2, ...r \tag{6.41}$$

where *r* is the number of elliptic sections involved, namely *floor* [(*N* – 1) / 2]. The Amstutz elliptic filter synthesis method is addressed in §10.8.

10.2.4 Characteristic Function Poles & Zeros (Even *N*)

As just described in §10.2.3, the poles of *K*(*s*) must be the same as those for *T*( ), namely those given by (6.39). The zeros of *K*(*s*) for the even-order case are given by

$$s_n = \pm j\Omega_n \quad \text{with} \quad \Omega_n = \sqrt{k}\, sn\left[K\left(\frac{2n-1}{N}\right), k \right] \text{ for } n = 1, 2, \ldots \frac{N}{2} \tag{6.42}$$

10.2.5 Complete Elliptic Integrals and the Jacobi Elliptic Functions

Numerical evaluation of the complete elliptic integral as well as the twelve Jacobi elliptic functions are discussed further in §10.7. For a more detailed discussion about elliptic functions, Appendix A of [8] is highly recommended as is reference [9]. Additional material is developed based upon Amstutz's work [11] in §17.

10.2.6 *N* = 5 Elliptic Lowpass Filter Design Example[42]

Assume the following:

$$N = 5$$

$$k = 0.5 \quad \Rightarrow \omega_{pass} = \sqrt{\frac{1}{2}}, \quad \omega_{stop} = \sqrt{2}$$

$$A_{pass} = 0.1 \text{ dB}$$

$$A_{stop} = 50 \text{ dB}$$

Then these results follow:

$$\varepsilon_p = 0.15262041895$$

$$\varepsilon_s = 794.32760526133$$

$$k_1 = 0.000192138$$

$$z_0 = j0.55348751887$$

$L(\ )$ Zeros:

-0.30341367575 ± j0.51005682043

-0.09822601916 ± j0.75912684028

-0.41785310753

$L(\ )$ Poles:

± j2.29866617127

± j1.47732036935

Figure 62 5th-order elliptic lowpass filter example

42 Computed in u18602_equation_check1.m.

10.3 Physically Realizable Even-Order Elliptic Filters

Elliptic lowpass filters are characterized by four different types commonly referred to as *a, b, c,* and *s*. Type–*s* elliptic filters are odd-order filters which can be physically implemented in a ladder network without requiring ideal transformers or negative LC values. Type-s filters exhibit a symmetric topology, with an N^{th}-order lowpass having (*N* – 1) / 2 trap sections. In the context of the filter's *ABCD* matrix description (See §1.2), symmetry requires that

$$AR_{load} = DR_{source} \tag{6.43}$$

Elliptic filter types *a, b,* and *c* are referred to as *antimetric* filters and are even-order filters. A filter is antimetric provided that

$$B = R_{source} R_{load} C \tag{6.44}$$

The *type-a* filter must include ideal transformers or at least one negative element in order to be physically realizable [1]. The *type-b* filter eliminates the need for negative circuit elements by moving the highest finite-frequency stopband attenuation pole to infinity. This frequency transformation reduces the transition rate from the passband to the stopband somewhat, but makes the filter physically realizable. The *type-c* filter additionally transforms the lowest passband attenuation zero to the origin so that the termination impedances can be made equal.

10.3.1 Even-Order Type-B Filters

Design of the type-b filter begins by following the details provided earlier in §10.2.2 and §10.2.4. These formula produce the poles and zeros for a lowpass filter having a passband frequency of $\omega_p = \sqrt{k}$ and implied stopband frequency of $\omega_s = 1/\sqrt{k}$. The design is transformed to a type-b filter attenuation characteristic with a passband frequency of $\omega_p = 1$ by making use of the frequency transformation function

$$s_b = \frac{\gamma_o}{\sqrt{k}} \frac{s_a}{\sqrt{1 + \left(s_a \Omega_1\right)^2}} \tag{6.45}$$

where s_a and s_b represent complex frequency variables for the original filter design and the type-b filter design respectively, Ω_1 is the lowest-frequency zero given by (6.39), and

$$\gamma_o = \sqrt{1 - k\Omega_1^2} \tag{6.46}$$

A detailed design example for a *N* = 10 filter type-b filter follows.

Example[43]**:** $N = 10$, $\varepsilon_p = 0.20$, $k = \sin\left(60^o\right)$, $A_{stop} = 80.8$ dB

Ω_1 = 0.198007183
γ_o = 0.982876328

Loss Poles Type-A $\omega_p = \sqrt{k}$ **rad/sec**
j 5.05032182896
j 1.85942796588
j 1.31607401295
j 1.13993579946
j 1.08092053456

Loss Zeros Type-A $\omega_p = \sqrt{k}$ rad/sec
–0.28865998524 + j 0.21542621550
–0.20765911214 + j 0.57214146700
–0.11699931195 + j 0.78822155729
–0.05433928213 + j 0.89514103296
–0.01540932827 + j 0.93713410053

Loss Poles Type-B ω_p **= 1 rad/sec**
j 2.112246661
j 1.439741558
j 1.235858726
j 1.168717739

Loss Zeros Type-B ω_p = **1** rad/sec
–0.305204244 + j 0.226618383
–0.223410566 + j 0.606614441
–0.128188023 + j 0.842111589
–0.060202020 + j 0.960452517
–0.017153023 + j 1.007250333

Figure 63 N = 10, ρ = 20%, *k* = sin(60°) type-b filter[44]

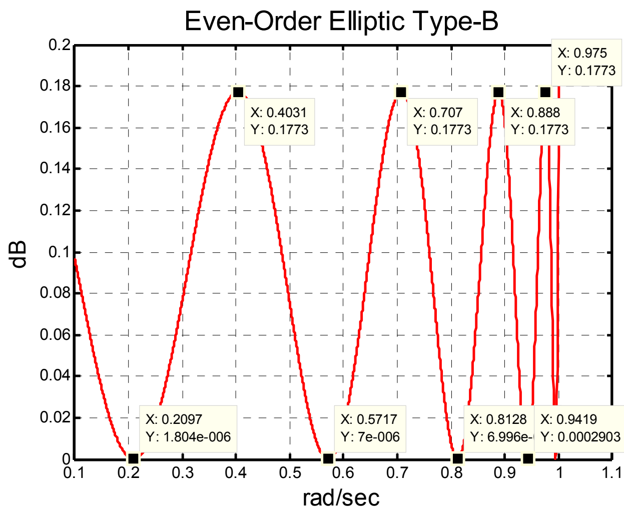


Figure 64 N = 10, ρ = 20%, *k* = sin(60°) type-b filter**.** Passband close-up.

[43] These trap frequencies match those given in [18] *exactly* (to within the 7-digit published precision).
[44] Computed using u18602_equation_check1.m.

10.3.2 Even-Order Type-C Filters

Design of a type-c filter begins by following the details provided earlier in §10.2.2 and §10.2.4. These formula produce the poles and zeros for a lowpass filter having a passband frequency of $\omega_p = \sqrt{k}$ and implied stopband frequency of $\omega_s = 1/\sqrt{k}$. The design is transformed to a type-c filter attenuation characteristic with a passband frequency of $\omega_p = 1$ by making use of the frequency transformation function

$$s_c = \frac{\gamma_o}{\sqrt{k}} \sqrt{\frac{s_a^2 + \Omega_1^2}{1 + \left(s_a \Omega_1\right)^2}} \tag{6.47}$$

where γ_o is initially set to unity. The transformation of the passband zero to DC causes the passband frequency ω_p to shift very slightly (e.g., < 1%) away from 1 rad/sec making a polishing step for parameter γ_o necessary if an exact numerical match with [18] is desired. This can be done by employing a simple Newton-Raphson type solution where γ_o is iteratively adjusted based upon the filter attenuation at 1 rad/sec. It is convenient to express the loss function as

$$A_{dB}(s) = 10\log_{10}\left\{ \left| A_0 \prod_{n=1}^{N/2} \left[\frac{(s - z_n)(s - z_n^*)}{(s - p_n)(s - p_n^*)} \right] \right|^2 \right\} \tag{6.48}$$

where z_n and p_n represent the transformed zeros and poles from (6.47), and

$$A_0 = \prod_{n=1}^{N/2} \left| \frac{p_n}{z_n} \right|^2 \tag{6.49}$$

The filter attenuation at the passband edge (1 rad/sec) should equal A_{pass} exactly. A detailed design example for a 10th-order type-c elliptic filter follows.

Example (from §10.3.1) Continued[45]: *N* = 10, $\varepsilon_p = 0.20$, $k = \sin\left(60^o\right)$, $A_{stop} = 80.8$ dB

γ_o = 1.005910031

Loss Poles Type-C ω_p = 1 rad/sec	**Loss Zeros Type-C ω_p = 1** rad/sec
j 2.149455379	–0.360305306 + j 0.200755509
j 1.456709508	–0.241284178 + j 0.587408240
j 1.245593971	–0.135230486 + j 0.834823971
j 1.175866923	–0.063074909 + j 0.958700060
	–0.017932772 + j 1.007588230

Figure 65 N = 10, ρ = 20%, *k* = sin(60°) type-c filter[46]

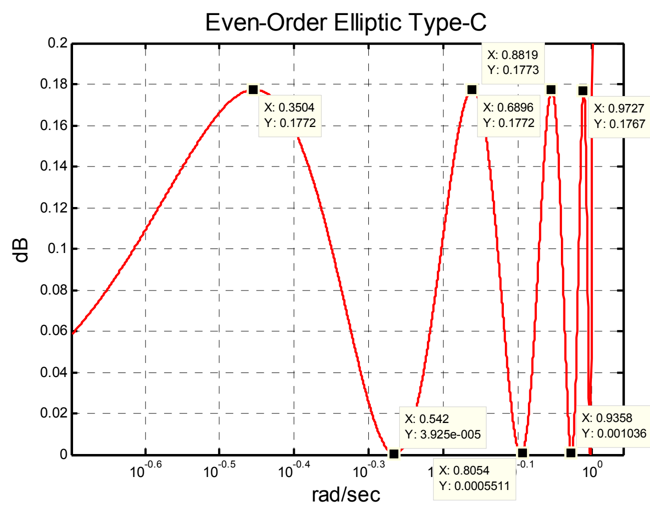


Figure 66 N = 10, ρ = 20%, *k* = sin(60°) type-c filter**.** Passband closeup.

10.4 Elliptic Filter Group Delay

The group delay for an all-pole filter was developed earlier in (3.9). Assuming that the voltage transfer function poles are represented by $\sigma_n + j\,\omega_n$ and the zeros are represented by $u_m + j\,v_m$, the filter group delay can be calculated as

$$\tau_g\left(\omega\right) = -\sum_{m=1}^{N_{poles}}\left[\frac{\sigma_m}{\sigma_m^2+\left(\omega-\omega_m\right)^2}\right]+\sum_{m=1}^{N_{zeros}}\left[\frac{u_m}{\sigma_m^2+\left(\omega-v_m\right)^2}\right] \tag{6.50}$$

Only odd-order elliptic filter results are shown here for brevity, and for the more common passband ripple values of 0.01 dB, 0.1 dB, and 0.25 dB. The transfer function gain-nulls are due to ideal poles located at ± $j\,\omega$ which contribute nothing to the group delay since the real-parts of these poles are identically zero. Amstutz [11] develops a result for $dZ_{in}/d\omega$ which is directly related to the filter group delay (6.137).

Two different perspectives are offered in the plots which follow. The passband and stopband attenuation parameters are kept fixed in Figure 67 through Figure 69 and only the filter shape factor ($k = f_{pass} / f_{stop}$) allowed to vary. In all cases, the ripple bandwidth is held fixed at 1 rad/sec. In Figure 70 through Figure 81, the group delay is plotted for different passband and stopband attenuation levels versus filter order.

45 These trap frequencies match those given in [18] *exactly* (to within the 7-digit published precision).
46 Computed using u18602_equation_check1.m.

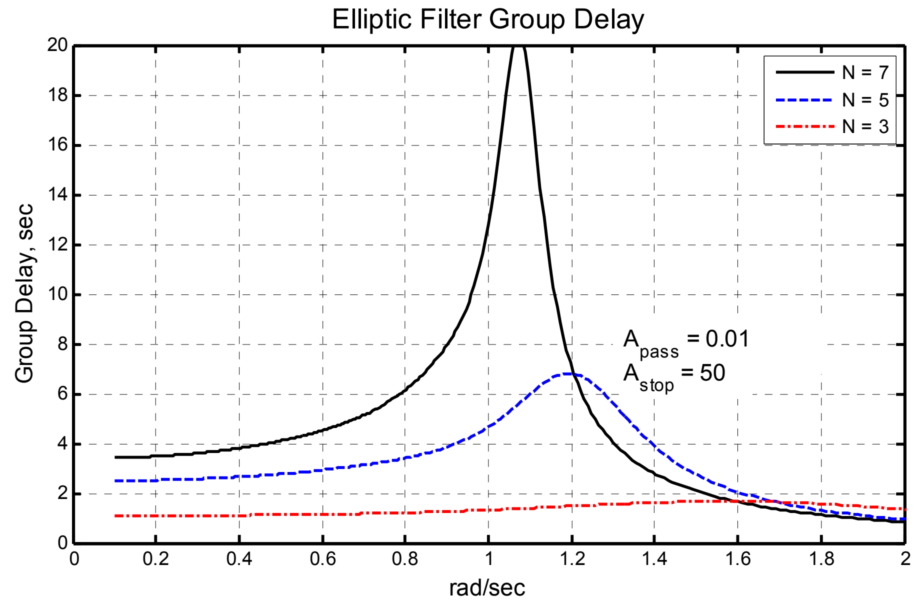


Figure 67 Elliptic filter group delay for fixed passband and stopband attenuation (k_1 is constant) allowing the filter shape factor ($k = f_{pass} / f_{stop}$) to vary with filter order[47]. A_{pass} = 0.01 dB.

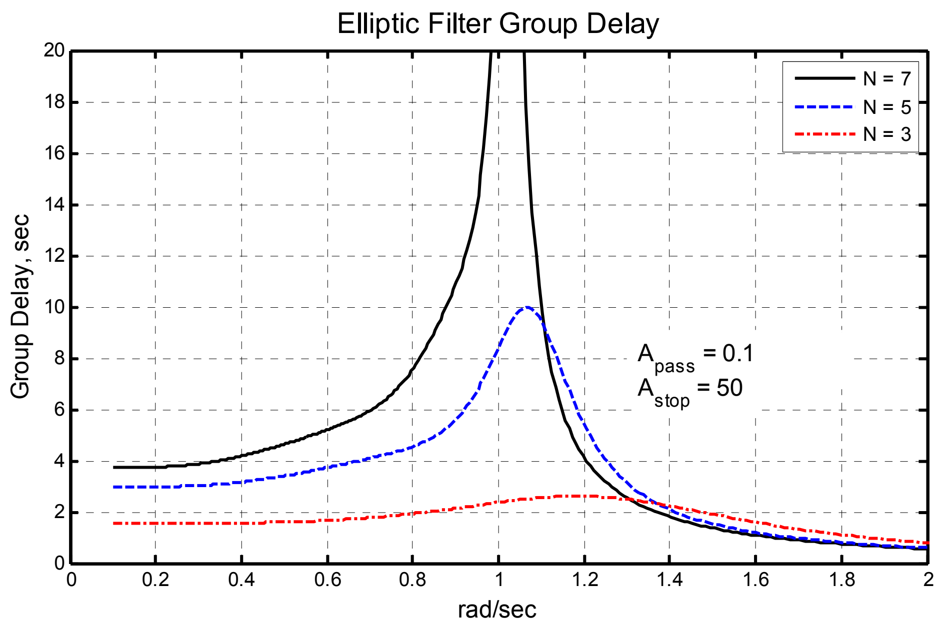


Figure 68 Elliptic filter group delay for fixed passband and stopband attenuation (k_1 is constant) allowing the filter shape factor ($k = f_{pass} / f_{stop}$) to vary with filter order[48]. A_{pass} = 0.1 dB.

[47] Computed using u18426_elliptic_group_delay.m.
[48] Computed using u18426_elliptic_group_delay.m.

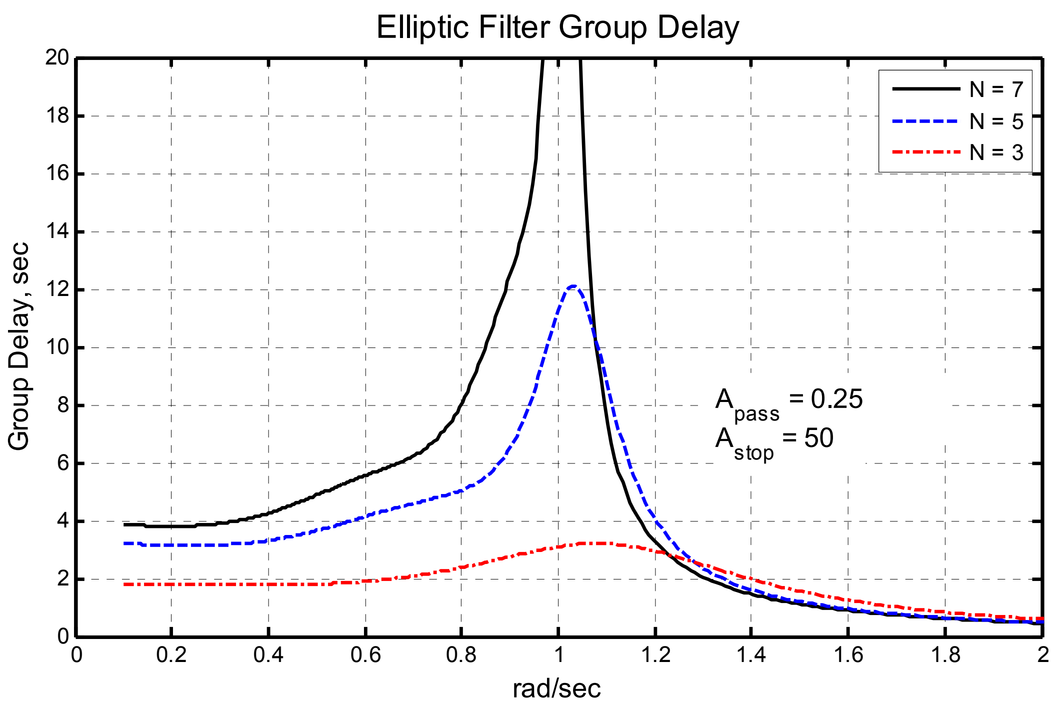


Figure 69 Elliptic filter group delay for fixed passband and stopband attenuation (k_1 is constant) allowing the filter shape factor ($k = f_{pass} / f_{stop}$) to vary with filter order[49]. A_{pass} = 0.25 dB.

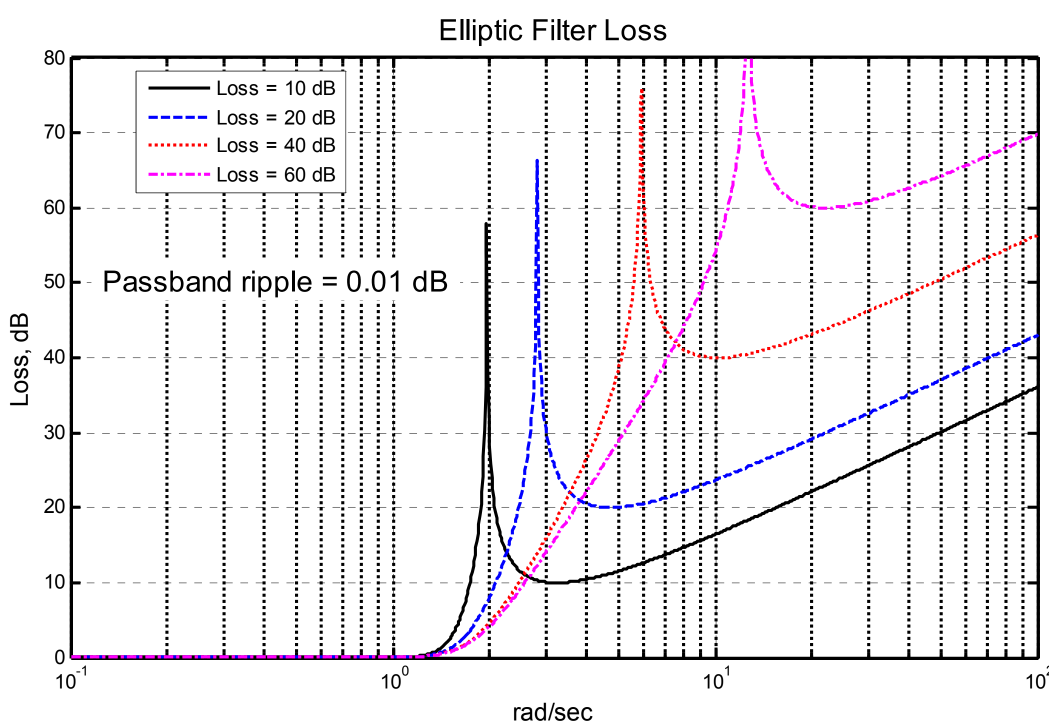


Figure 70 N = 3 elliptic loss characteristics[50] with different stopband attenuation levels. Associated group delay characteristics are shown in Figure 71.

---

49 Computed using u18426_elliptic_group_delay.m.
50 Computed using u18426_elliptic_group_delay.m.

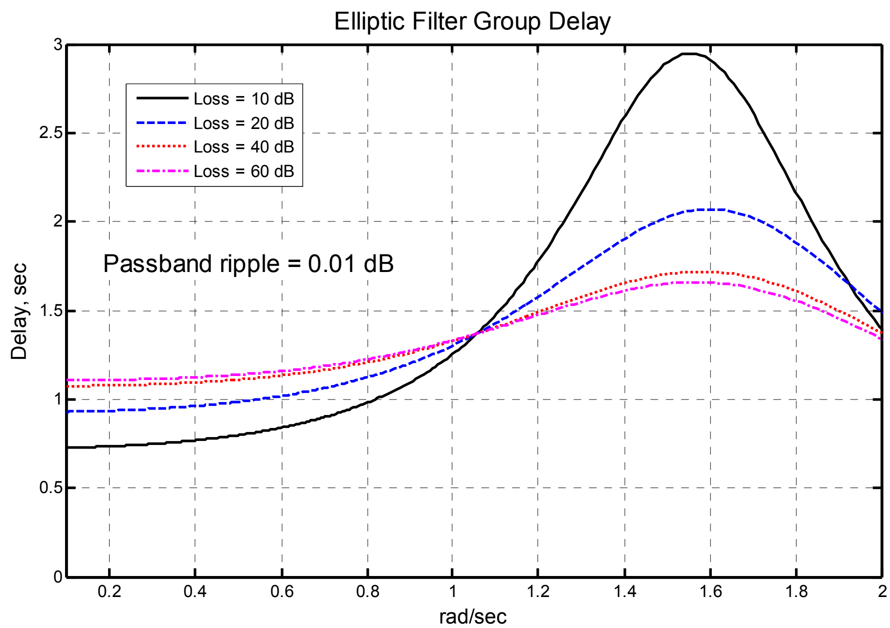


Figure 71 Group delay characteristics for *N* = 3 elliptic lowpass filter loss characteristics shown in Figure 70

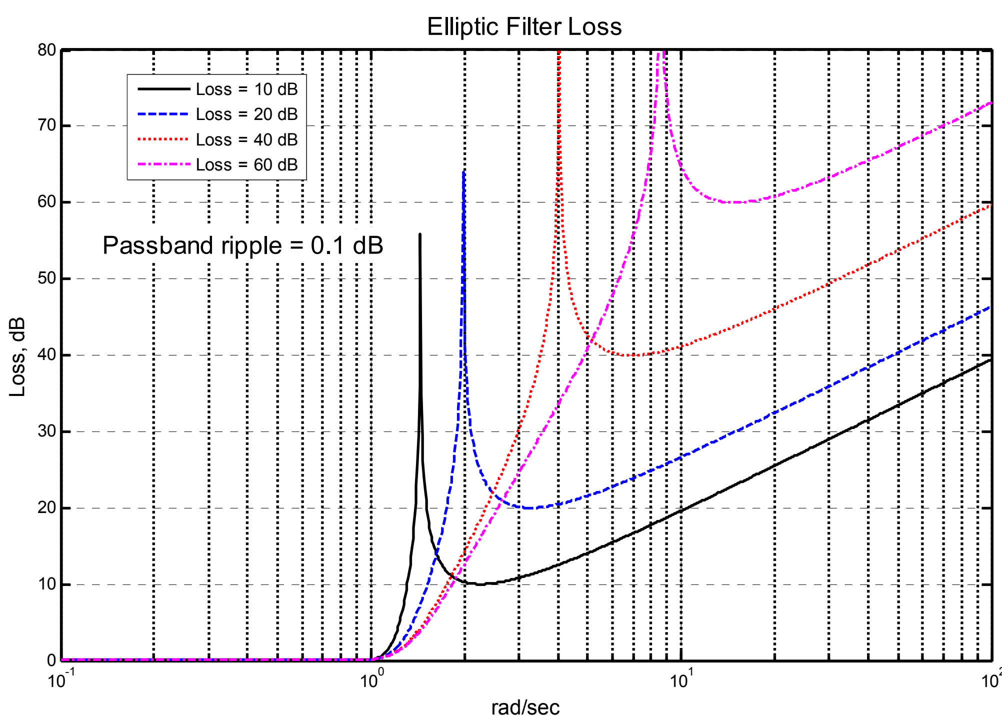


Figure 72 *N* = 3 elliptic loss characteristics[51] with different stopband attenuation levels. Associated group delay characteristics are shown in Figure 73.

[51] Computed using u18426_elliptic_group_delay.m.

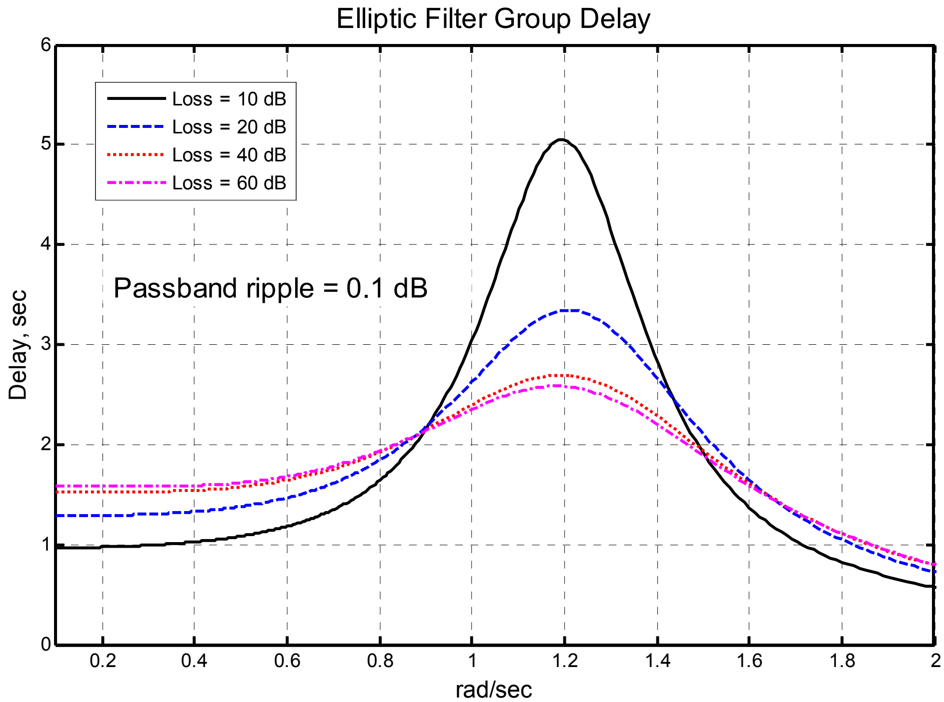


Figure 73 Group delay characteristics for *N* = 3 elliptic lowpass filter loss characteristics shown in Figure 72

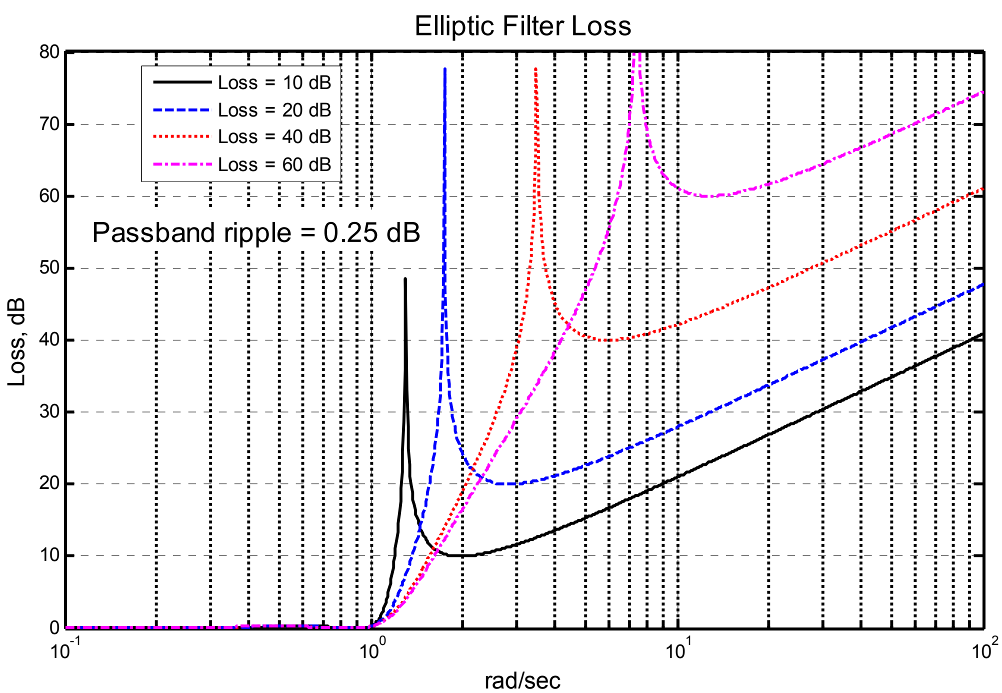


Figure 74 *N* = 3 elliptic loss characteristics[52] with different stopband attenuation levels. Associated group delay characteristics are shown in Figure 75.

52 Computed using u18426_elliptic_group_delay.m.

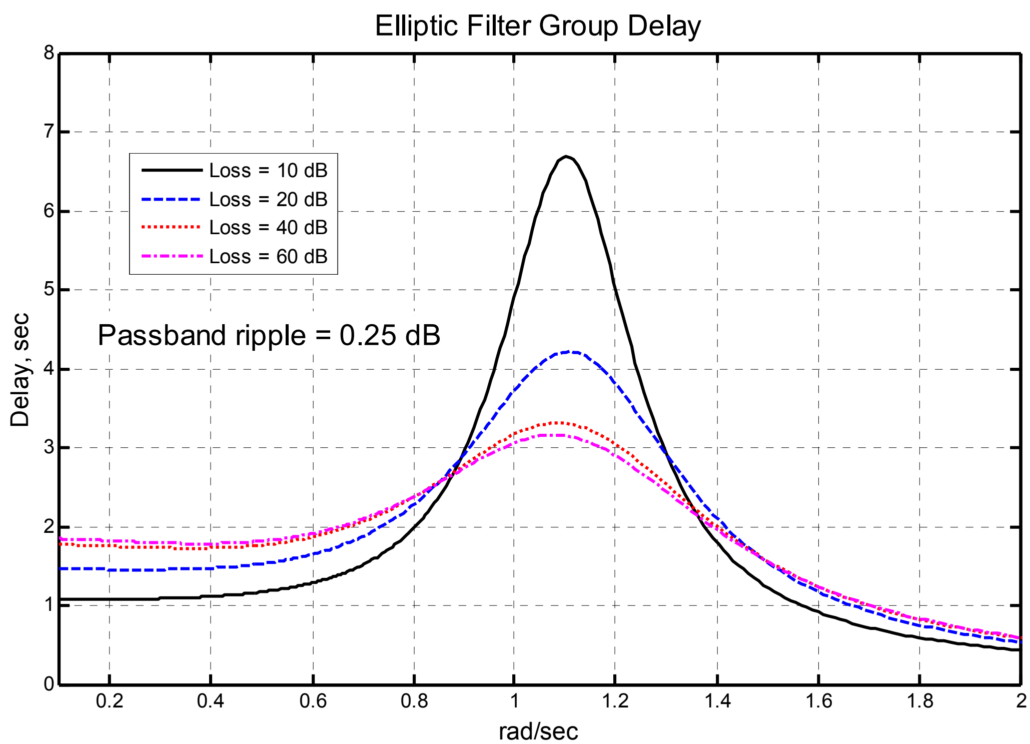


Figure 75 Group delay characteristics for N = 3 elliptic lowpass filter loss characteristics shown in Figure 74

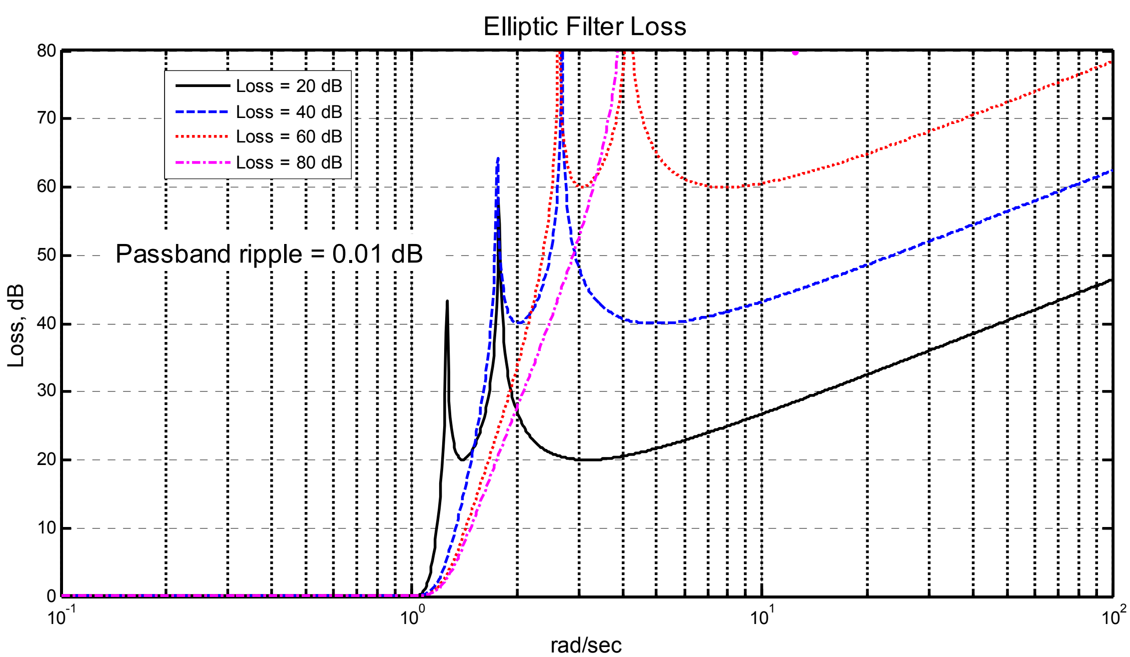


Figure 76 N = 5 elliptic loss characteristics[53] with different stopband attenuation levels. Associated group delay characteristics are shown in Figure 77.

53 Computed using u18426_elliptic_group_delay.m.

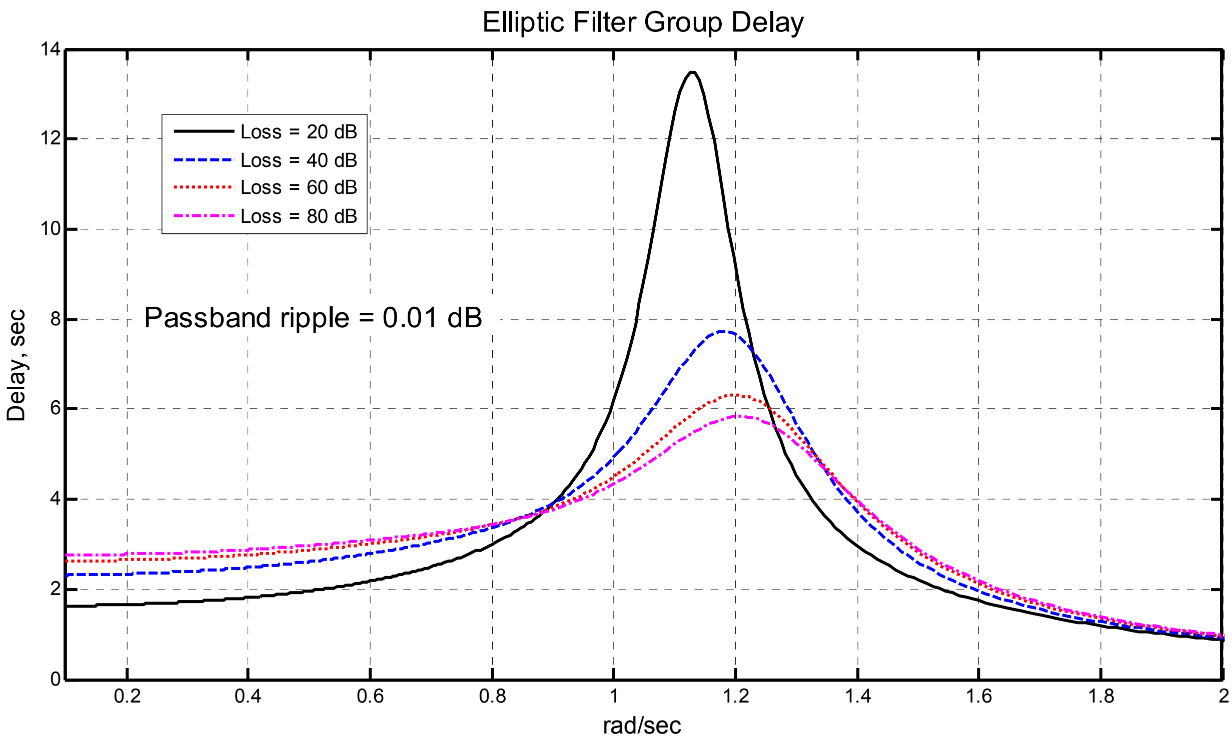


Figure 77 Group delay characteristics for *N* = 5 elliptic lowpass filter loss characteristics shown in Figure 76

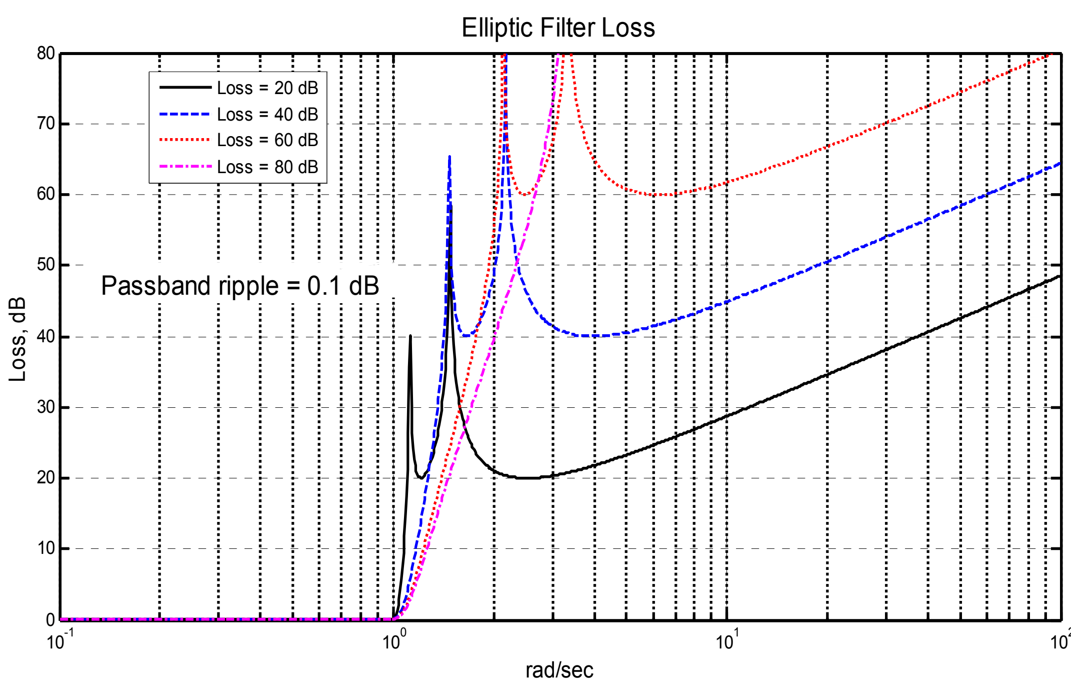


Figure 78 *N* = 5 elliptic loss characteristics[54] with different stopband attenuation levels. Associated group delay characteristics are shown in Figure 79.

[54] Computed using u18426_elliptic_group_delay.m.

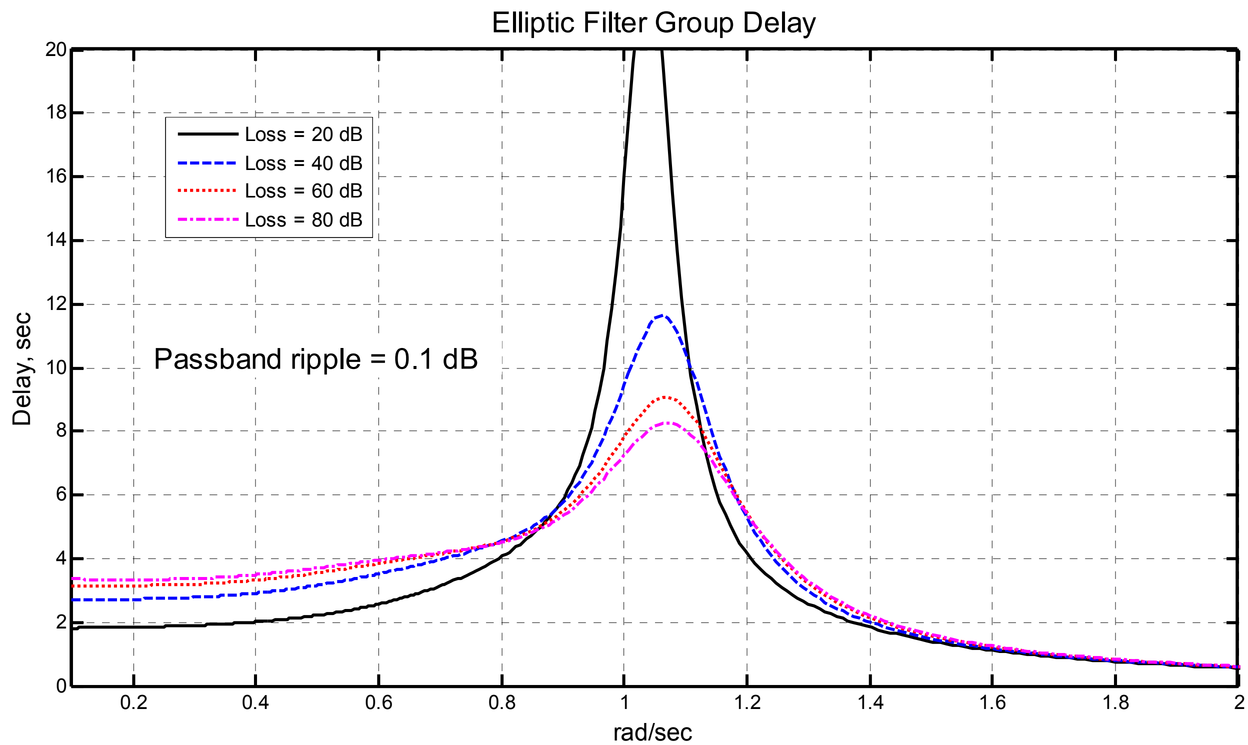


Figure 79 Group delay characteristics for *N* = 5 elliptic lowpass filter loss characteristics shown in Figure 78

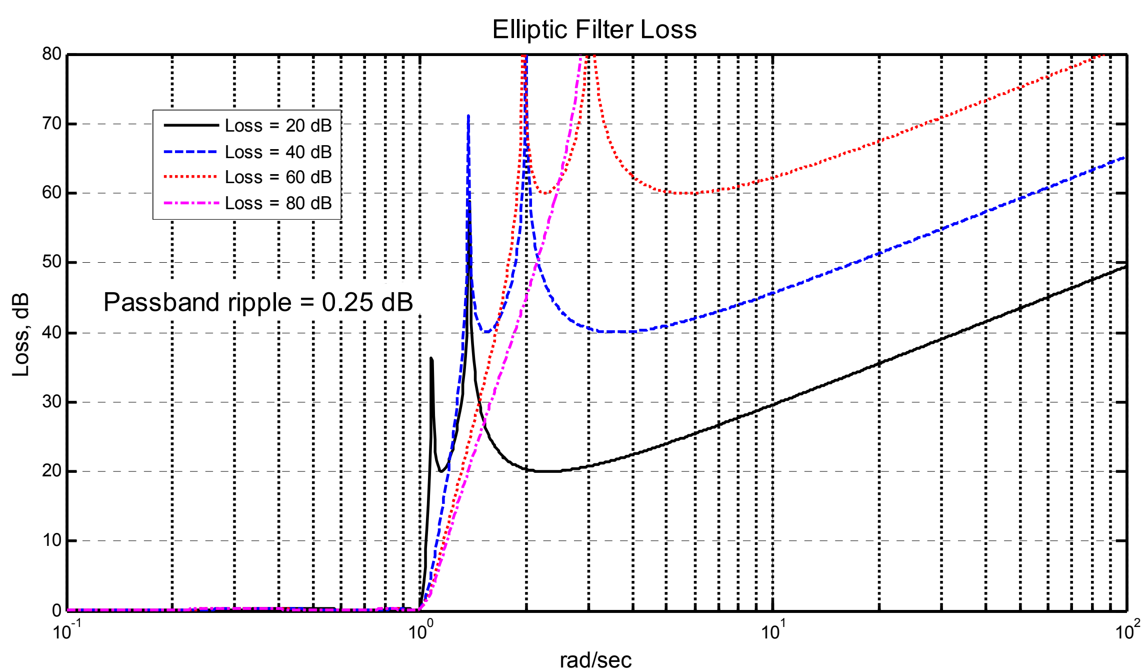


Figure 80 *N* = 5 elliptic loss characteristics[55] with different stopband attenuation levels. Associated group delay characteristics are shown in Figure 81.

[55] Computed using u18426_elliptic_group_delay.m.

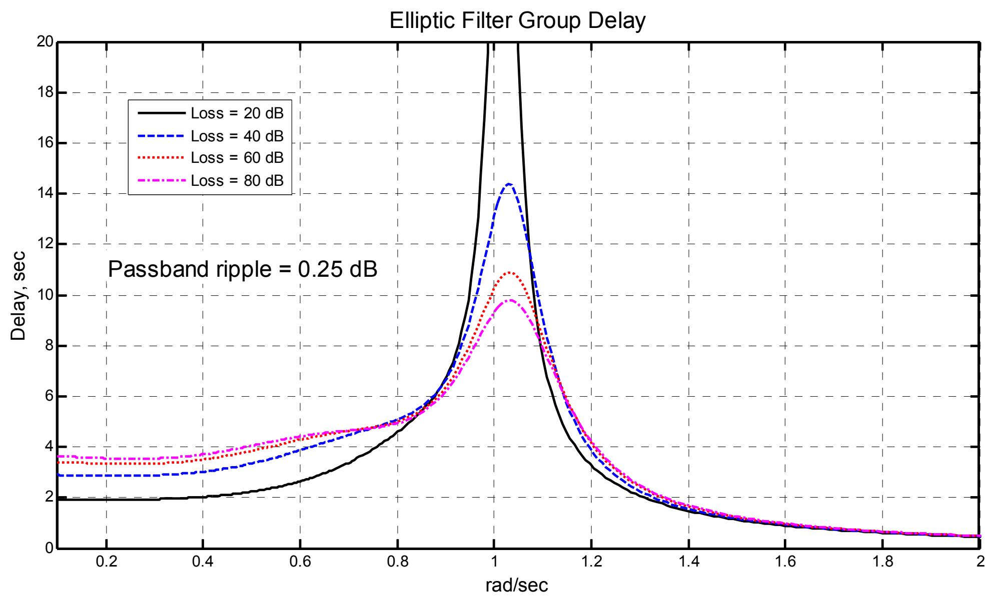


Figure 81 Group delay characteristics for N = 5 elliptic lowpass filter loss characteristics shown in Figure 80

10.5 Elliptic Filter Transient Responses

Since elliptic filters are normally selected for their excellent stopband attenuation characteristics, transient response performance is generally of secondary importance. Elliptic filters also contain resonant LC sections which are prone to substantial ringing. The residue method can be used to calculate the impulse response of elliptic filters as done earlier for the Butterworth and Chebyshev filter cases. In general, the oscillatory ringing becomes more severe as the shape factor becomes more abrupt (i.e., $k \rightarrow 1$). Two example results are shown below in Figure 82 and Figure 83 for illustrative purposes.

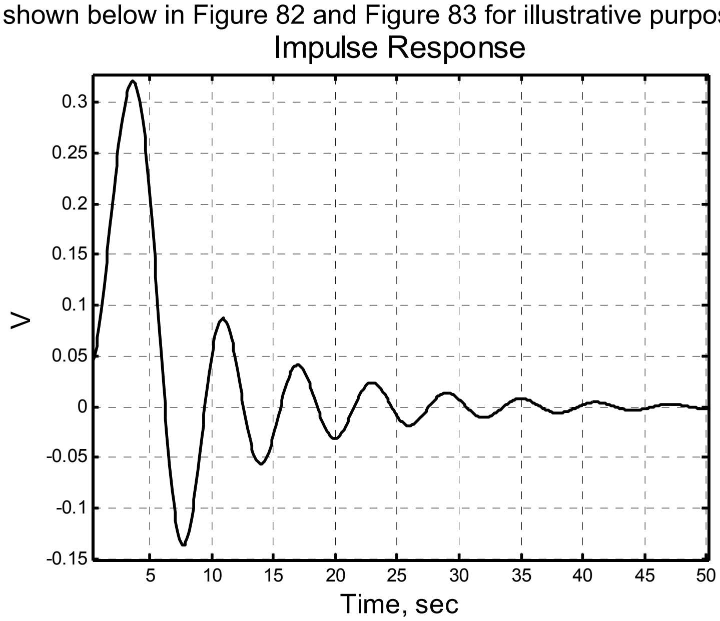


Figure 82 Impulse response for N = 5 lowpass filter, A_{pass} = 0.1778 dB, A_{stop} = 40 dB, k = 0.73412

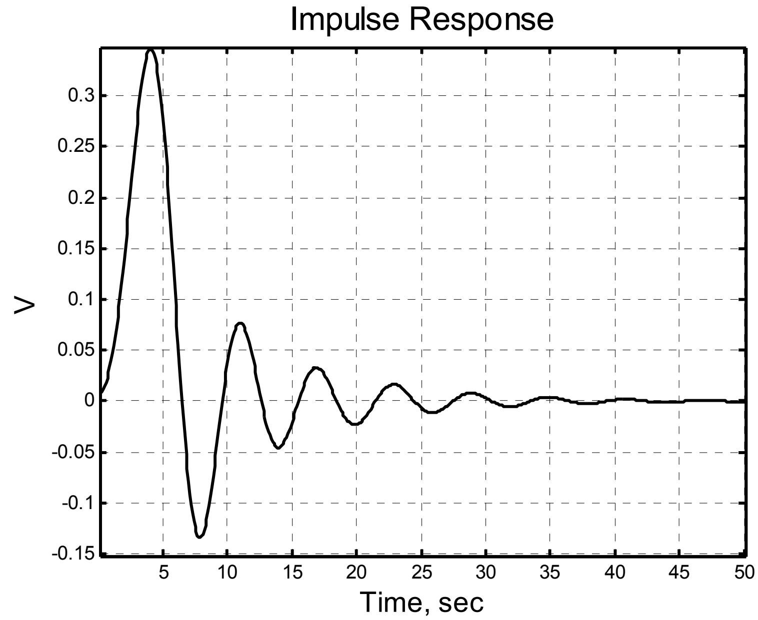


Figure 83 Impulse response for N = 5 lowpass filter, A_{pass} = 0.1778 dB, A_{stop} = 60 dB, k = 0.51445

10.6 Elliptic Filter Design Parameters

Owing to the appearance of elliptic integrals on both sides of (6.16), one involved with the value of *F* and the other with ω, it should not be surprising to see symmetry in the *filter order* equation here which is stated without proof as[56]

$$N \geq \frac{K_1'}{K_1}\frac{K}{K'} \tag{6.51}$$

where K and K' are given by (6.56) and (6.76) using k, and K_1 and K_1' are calculated using the same two equations but with k_1 as the modulus rather than k.

If on the other hand, the filter order and passband ripple are known, and tradeoffs between stopband attenuation and filter shape factor are needed, a more convenient result for the minimum stopband loss is given by[57]

$$A_{stopband} = 10\log_{10}\left[\left(10^{0.1A_{pass}} - 1\right)\exp\left(N\pi\frac{K'}{K}\right)\right] - 12.04 \text{ dB} \tag{6.52}$$

where N is the filter order, K and K' are the complete elliptic integrals given by (6.56) and (6.76), and A_{pass} is the passband ripple in dB. For an example, assume that N = 5, k = 0.80, and A_{pass} = 0.1 dB. This results in K = 1.9953, K' = 1.7508, and a minimum stopband attenuation of 31.49 dB. Equation (6.52) is shown for several of the most commonly used passband ripple cases in Figure 84 through Figure 86.

[56] See [8], [10], or [11] for details.
[57] [8] equation (5.43).

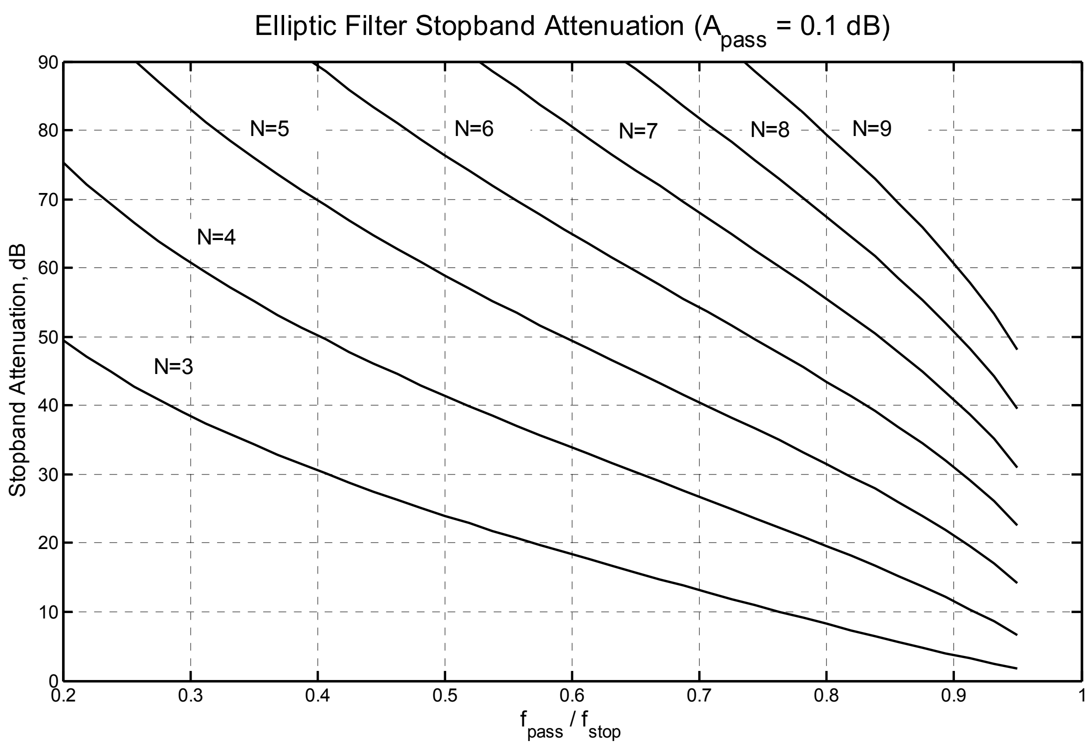


Figure 84 Minimum elliptic filter stopband attenuation versus shape factor for A_{pass} = 0.1 dB

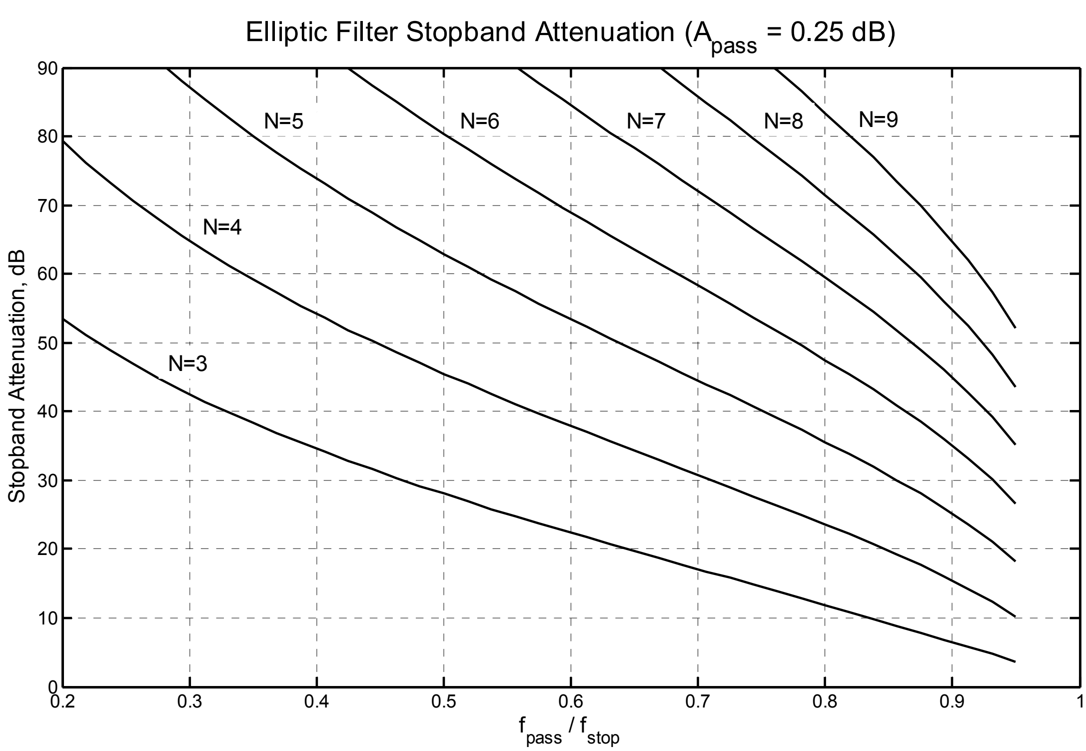


Figure 85 Minimum elliptic filter stopband attenuation versus shape factor[58] for A_{pass} = 0.25 dB

[58] Calculated using u18311_elliptic_pz.m.

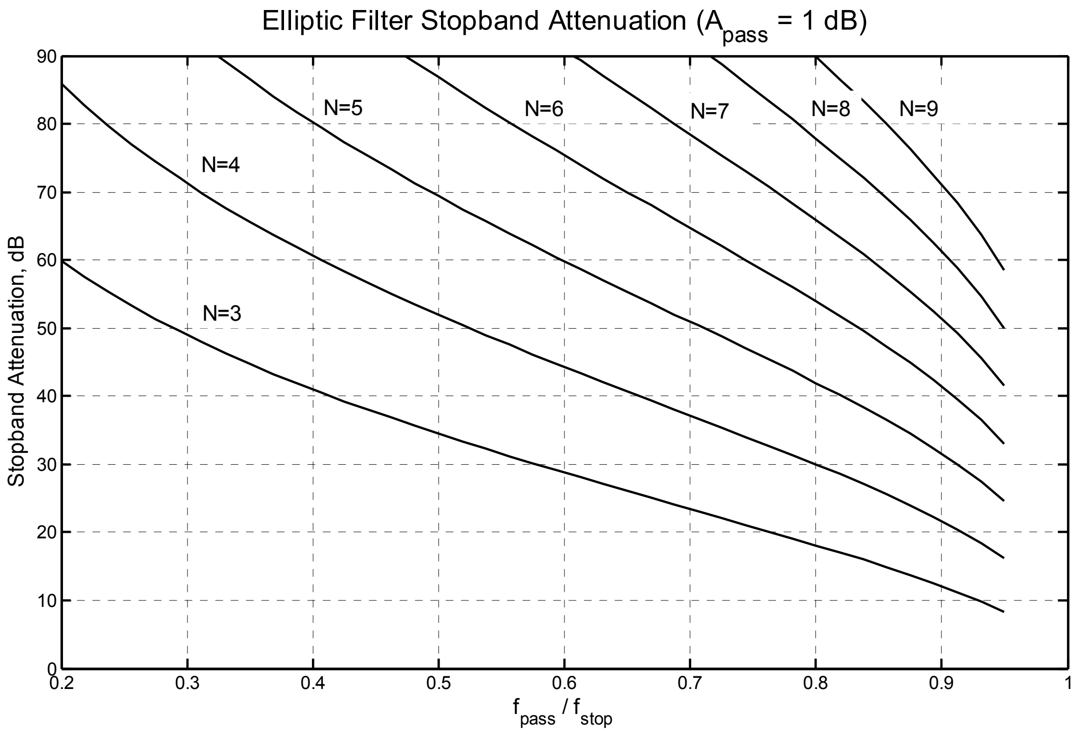


Figure 86 Minimum elliptic filter stopband attenuation versus shape factor for A_{pass} = 1.0 dB

10.6.1 Shortened Elliptic Order Equation

A much less computationally intensive means to compute the minimum required elliptic filter order (without computing complete elliptic functions) is given in chapter 5 of [8] and is provided here without further proof. Given the elliptic modulus value k from (6.2), compute the following:

$$k' = \sqrt{1-k^2}$$
$$q_0 = \frac{1}{2}\left(\frac{1-\sqrt{k'}}{1+\sqrt{k'}}\right) \tag{6.53}$$
$$q = q_0 + 2q_0^5 + 15q_0^9 + 150q_0^{13}$$

Given the allowable passband ripple A_{pass} in dB and minimum desired stopband attenuation A_{stop} again in dB, the remainder of the calculations follow as

$$D = \frac{10^{A_{stop}/10} - 1}{10^{A_{pass}/10} - 1}$$
$$N \geq \frac{\log_e\left(16D\right)}{\log_e\left(\dfrac{1}{q}\right)} \tag{6.54}$$

Additional details are also provided in §10.7.9.

10.6.2 Filter Shape Factor from A_{pass}, A_{stop}, and N

In some cases, the allowable passband ripple A_{pass}, required stopband attenuation A_{stop}, and filter order N are known and it remains to calculate the stopband frequency. Orfanidis provides a concise result for this case in [13] as provided here without further proof. Given k_1 from (6.6), first compute $k_1' = \sqrt{1-k_1^2}$, it's associated complete elliptic integral K', and then

$$k' = \left(k_1'\right)^N \prod_{m=1}^{\lfloor N/2 \rfloor} sn^4\left[\frac{(2m-1)}{N}K_1', k_1'\right] \tag{6.55}$$

The exact result for k (= f_{pass} / f_{stop}) follows as $k = \sqrt{1-(k')^2}$.

10.7 Computing Elliptic Quantities

Efficient computation of the complete elliptic integral of the first kind is addressed first in §10.7.1, followed by computation of the complimentary complete elliptic integral in §10.7.2. The important *s*-plane to *z*-plane mapping function is discussed next in §10.7.3. The 12 Jacobi elliptic functions are introduced in §10.7.4. Landen's transformation is used to compute the elliptic *sne*( ) and *cde*( ) functions in §10.7.5 and their inverses in §10.7.6. The exact solution to (6.27) is discussed in §10.7.7. The elliptic functions can also be computed using theta functions as discussed in §10.7.9.

10.7.1 Complete Elliptic Integral of the First Kind

The complete elliptic integral of the first kind is given by

$$K(k) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1-k^2\sin^2(\theta)}} \tag{6.56}$$

The complete elliptic integral obviously shows up in the context of elliptic filters, but also appears in several other engineering contexts including:

- Characteristic impedance relationship for stripline microwave transmission lines[59]
- Exact time-period of a swinging pendulum[60]

Direct numerical integration of (6.56) is painful for values of $k \to 1$. Fortunately for us, an ingenious method due to Landen can be used to compute the integral quickly and precisely. This method makes use of *arithmetic-geometric means* which make the integral solution very easy to compute.

The elliptic integral of the first kind may also be written in Gauss's formulation with a > b as

[59] §5.05 of *Microwave Filters, Impedance-Matching Networks, and Coupling Structures*, G.L. Matthaei, L. Young, and E.M.T. Jones, Artech House, 1980.
[60] Crawford, J.A., "Pendulums and Elliptic Integrals," 2004.

$$\int_0^{\pi/2} \frac{d\theta}{\sqrt{a^2\cos^2(\theta)+b^2\sin^2(\theta)}} = \int_0^{\pi/2} \frac{d\theta}{\sqrt{a^2\left[1-\sin^2(\theta)\right]+b^2\sin^2(\theta)}}$$
$$= \frac{1}{a}\int_0^{\pi/2} \frac{d\theta}{\sqrt{1-\dfrac{a^2-b^2}{a^2}\sin^2(\theta)}} = \frac{1}{a}\int_0^{\pi/2} \frac{d\theta}{\sqrt{1-k^2\sin^2(\theta)}} \tag{6.57}$$

where

$$k^2 = \frac{a^2-b^2}{a^2} \tag{6.58}$$

The arithmetic-geometric mean relationship can be developed by returning to the first integral in (6.57) and substituting

$$x = b\tan(\theta)$$
$$dx = b\sec^2(\theta)\,d\theta = \frac{b\,d\theta}{\cos^2(\theta)} \tag{6.59}$$

leading to

$$\int_0^{\pi/2} \frac{d\theta}{\sqrt{a^2\cos^2(\theta)+b^2\sin^2(\theta)}} = \int_0^{\infty} \frac{dx}{\sqrt{\left(x^2+a^2\right)\left(x^2+b^2\right)}} \tag{6.60}$$

Making a second substitution into (6.60) of

$$x = t+\sqrt{t^2+ab} \tag{6.61}$$

$$dx = dt + \frac{t\,dt}{\sqrt{t^2+ab}} = \left(\frac{\sqrt{t^2+ab}+t}{\sqrt{t^2+ab}}\right)dt = \frac{x\,dt}{\sqrt{t^2+ab}} \tag{6.62}$$

results in

$$\sqrt{\left(x^2+a^2\right)\left(x^2+b^2\right)} \Rightarrow 2x\sqrt{t^2+\left(\frac{a+b}{2}\right)^2\left(\frac{2t}{x}+\frac{ab}{x^2}\right)}$$
$$= 2x\sqrt{t^2+\left(\frac{a+b}{2}\right)^2} \tag{6.63}$$

because $2t / x + ab / x^2 \equiv 1$. With (6.63) as the denominator and (6.62) as the numerator of the integrand, the *x*-terms cancel out leaving

$$\int_{-\infty}^{+\infty} \frac{dt}{2\sqrt{t^2+ab}\sqrt{t^2+\left(\frac{a+b}{2}\right)^2}} = \int_0^{\infty} \frac{dt}{\sqrt{\left[t^2+\left(\sqrt{ab}\right)^2\right]\left[t^2+\left(\frac{a+b}{2}\right)^2\right]}} \tag{6.64}$$

Therefore, as long as the geometric mean and arithmetic mean of *a* and *b* remain constant, the value of the integral is unchanged! In Gauss's formulation of Landen's transformation, the integral

$$I = \int_0^{\pi/2} \frac{d\theta}{\sqrt{a^2 \cos^2(\theta) + b^2 \sin^2(\theta)}} \tag{6.65}$$

remains unchanged if *a* and *b* are replaced by their arithmetic and geometric means respectively as

$$a_1 = \frac{a+b}{2}; \qquad b_1 = \sqrt{ab} \tag{6.66}$$

The evaluation of $K(k)$ begins then with (6.58) which can be re-written as

$$k^2 = 1 - \left(\frac{b}{a}\right)^2 \tag{6.67}$$

It is convenient to let $a_0 = 1$ (due to the 1 / *a* factor in (6.57)) leading to $b_0 = (1 - k^2)^{1/2}$. The arithmetic and geometric means are then iterated as

$$\begin{aligned} a_{j+1} &= \frac{a_j + b_j}{2} \\ b_{j+1} &= \sqrt{a_j b_j} \end{aligned} \tag{6.68}$$

until such time as $a_j - b_j$ is sufficiently small. At this point (iteration *L*),

$$K(k) = \frac{\pi}{2a_L} \tag{6.69}$$

Note that starting out with

$$\begin{aligned} a_0 &= 1 + k \\ b_0 &= 1 - k \end{aligned} \tag{6.70}$$

gives identical results while avoiding the square-root operation in (6.68). The arithmetic mean of (6.70) is clearly 1 whereas the geometric mean is $(1 - k^2)^{1/2}$ thereby agreeing with the starting values identified with (6.67). The complete elliptic integral *K* and complete complimentary elliptic integral *K′* are shown plotted in Figure 87.

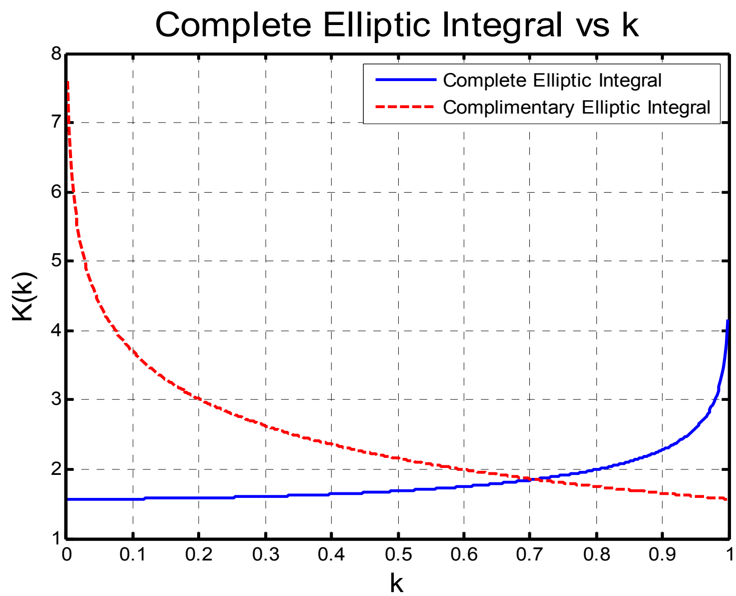


Figure 87 Complete elliptic integral[61] K (6.56) and complementary elliptic integral K' (6.76)

10.7.2 Complementary Complete Elliptic Integral of the First Kind

The previous section only considered real values of $K(k)$ whereas imaginary values also occur. Consider the imaginary value case where

$$j\upsilon = \int_0^{\psi} \frac{d\theta}{\sqrt{1-k^2\sin^2(\theta)}} \tag{6.71}$$

Now applying the transformation

$$\sin(\theta) = j\tan(\theta') \tag{6.72}$$

to (6.71), the differentials are

$$\cos(\theta)\,d\theta = j\sec^2(\theta')\,d\theta' \tag{6.73}$$

and carrying this through,

$$d\theta = \frac{j\,d\theta'}{\cos^2(\theta')\sqrt{1+\tan^2(\theta')}} = \frac{j\,d\theta'}{\cos(\theta')} \tag{6.74}$$

leading to

$$\upsilon = \int_0^{\psi'} \frac{d\theta'}{\sqrt{1-(1-k^2)\sin^2(\theta')}} = \int_0^{\psi'} \frac{d\theta'}{\sqrt{1-(k')^2\sin^2(\theta')}} \tag{6.75}$$

where $\sin(\psi) = j\tan(\psi')$ for the integrand upper-limit. It is therefore helpful to define the *complete complementary elliptic integral of the first-kind* as

[61] Computed using u18311_elliptic_pz.m.

$$K' = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1-(k')^2 \sin^2(\theta)}} \tag{6.76}$$

where $k' = \sqrt{1-k^2}$.

10.7.3 *s*-Plane to *z*-Plane Transformation using *sn*(*z, k*)

The earlier result (6.21) is really a variable transformation which maps points in the z - plane into points within the ω - plane. It is this transformation which is responsible for the elliptic filter's attenuation characteristic versus frequency, especially its rapid transition between passband and stopband.

The *sn*( ) function is *doubly periodic* in that a single point z_p as well as other points given by $z = z_p + 4mK + j2nK'$ (for integer values of *m* and *n*) are all mapped onto the same point within the ω - plane. The *z*-parameter has different periods in the real and imaginary dimensions given by 4*K* and 2*K'*, and these are the complete elliptic integrals discussed earlier in §10.7.1 and §10.7.2 respectively.

The transformation between the *z* - and ω - planes is shown graphically in Figure 88. The *z* – plane nodes are specifically labeled with the letters *S*, *C*, *D*, and *N*, and are directly tied to the names given to the 12 different elliptic functions possible [13]. These nodes correspond to the *z* – plane corner points { 0, *K*, *j K'*, *K* + *j K'* } as shown. An elliptic function *pq*(*z*, *k*) is named such that the first letter *p* can be any of the four possible letters { *s, c, d, n* } and the second letter *q* can be any of the three remaining letters. Each function *pq*(*z, k*) has a *simple zero* at corner *p* and a simple pole at corner *q* in Figure 88. In general, the following relationships hold

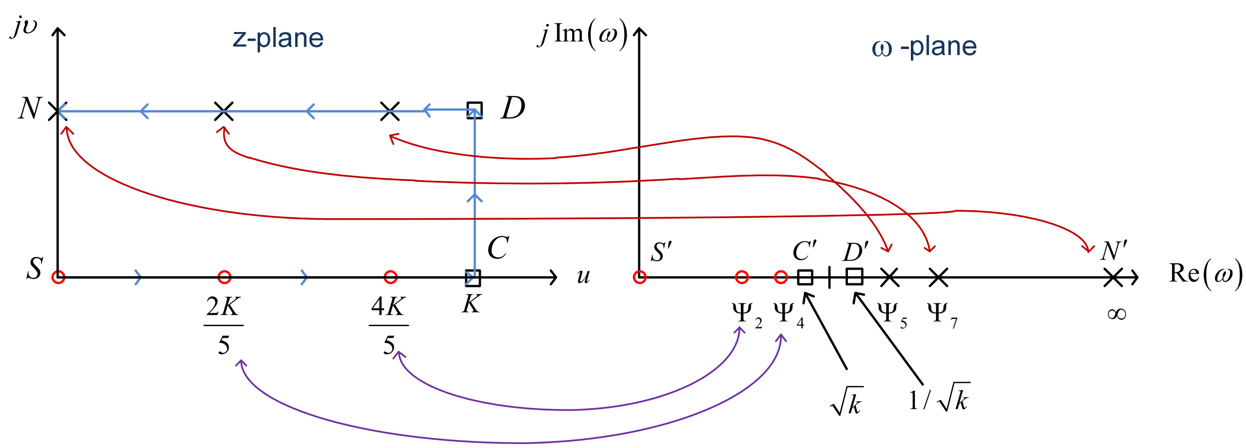


Figure 88 Transformation between *z*-plane and ω-plane by way of (6.18) for *N* = 5. Only the fundamental *z* – plane rectangle is shown.[62]

$$pq(z,k) = \frac{1}{qp(z,k)}, \quad pq(z,k) = \frac{pr(z,k)}{qr(z,k)} \tag{6.77}$$

where letter *r* can be any of the four letters but different than *p* and *q*. All twelve elliptic functions are summarized in terms of *cn*( ), *dn*( ), and *sn*( ) in Table 10 for convenience.

The mathematical symmetry imposed by (6.22) is responsible for delivering the equiripple stopband attenuation characteristic of elliptic filters when the passband is equiripple. The passband zeros and poles can otherwise be chosen independently of each other (e.g., Butterworth and Chebyshev filters

62 For zeros, $z = (2m+1)K + j2nK'$ and for poles $z = (2m+1)K + j(2n+1)K'$ for arbitrary integers *n* and *m*.

have all of their attenuation poles at ∞). Since the passband and stopband edges are given by $\sqrt{k}$ and $1 / \sqrt{k}$ respectively, the passband to stopband transition speed for elliptic filters is arguably optimal, at least in the case of analog filters.

Darlington [7] was one of the first to recognize the similarities between all-pole filters with their poles located at ∞, and the increasing slope of the filter's transition region as some of the poles are moved from infinity to finite frequencies in the context of elliptic filters. He likened the transformation of the pole positions (in the case of an equiripple stopband attenuation characteristic) as equivalent to manipulating elliptic sine values and their moduli. In this respect, Darlington was able to unify the theory between all-pole filters such as the Butterworth and Chebyshev and the elliptic filter family [7].

10.7.4 Jacobi Elliptic Functions

The elliptic sine function from (6.18) is rewritten here for convenience as

$$\sin(\theta) = sn(z,k) \tag{6.78}$$

and similarly for the elliptic function $\cos(\theta) = cn(z,k)$. MATLAB provides a single function call which returns the three primary elliptic function values as $[sn, cn, dn] = ellipj(z, M)$ where $M = k^2$, $sn(z,k) = \sin(\theta)$, $cn(z,k) = \cos(\theta)$. For the third function, taking the derivative of (6.17),

$$\begin{aligned} \frac{dz}{d\theta} &= \sqrt{1-k^2\sin^2(\theta)} = \sqrt{1-k^2 sn^2(z,k)} \\ &= dn(z,k) \end{aligned} \tag{6.79}$$

This function *dn*(*z, k*) is also known as the *difference function* [10]. These elliptic functions are plotted for several values of *k* in Figure 89 through Figure 93. (Note that *z* is *not* normalized to *K* in these equations!)

Several other identities may prove helpful including the following [13]:

$$w = cn(z,k) = \cos(\theta) \tag{6.80}$$

$$sn^2(z,k) + cn^2(z,k) = 1 \tag{6.81}$$

$$cd(z,k) = sn(z+K,k) = sn(K-z,k) \tag{6.82}$$

$$cd\left[z+(2i-1)K,k\right] = (-1)^i\, sn(z,k) \quad \text{for any integer } i \tag{6.83}$$

$$cd(z+2iK,k) = (-1)^i\, cd(z,k) \quad \text{for any integer } i \tag{6.84}$$

$$cd(z+jK',k) = \frac{1}{k\, cd(z,k)} \tag{6.85}$$

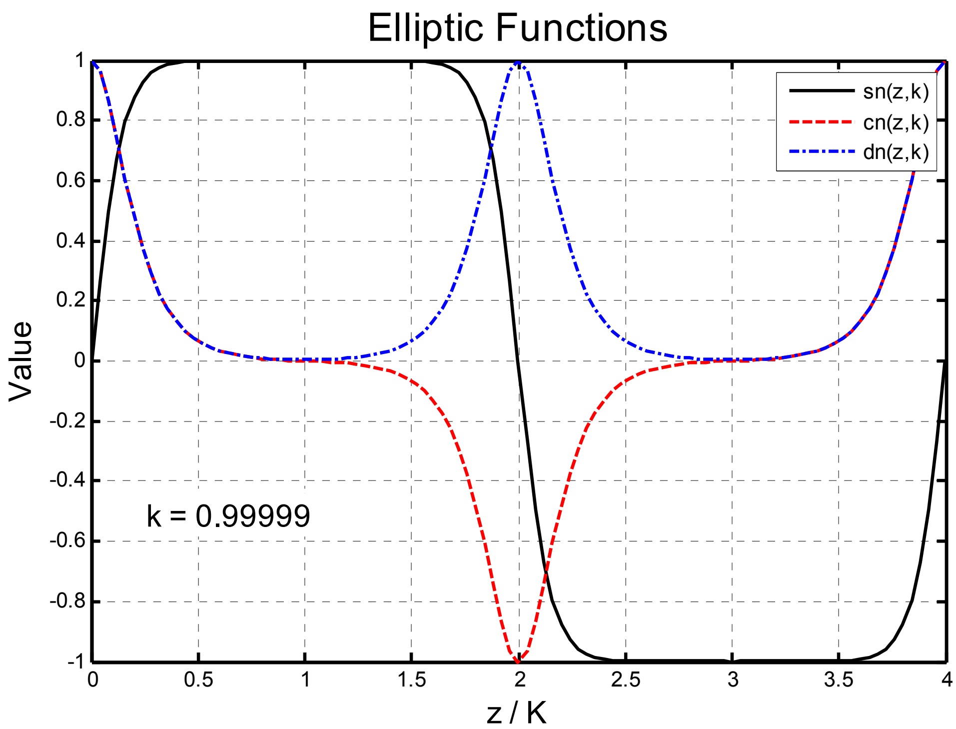


Figure 89 Elliptic functions[63] for k = 0.99999

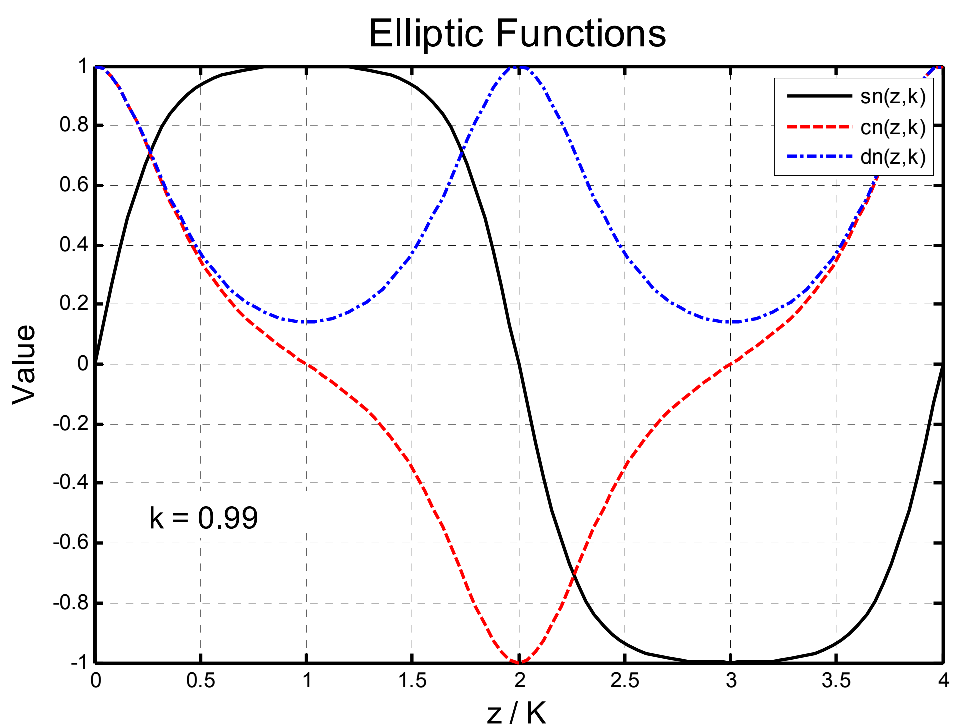


Figure 90 Elliptic functions[64] for k = 0.99

63 Calculated in u18311_elliptic_pz.m.
64 Calculated in u18311_elliptic_pz.m.

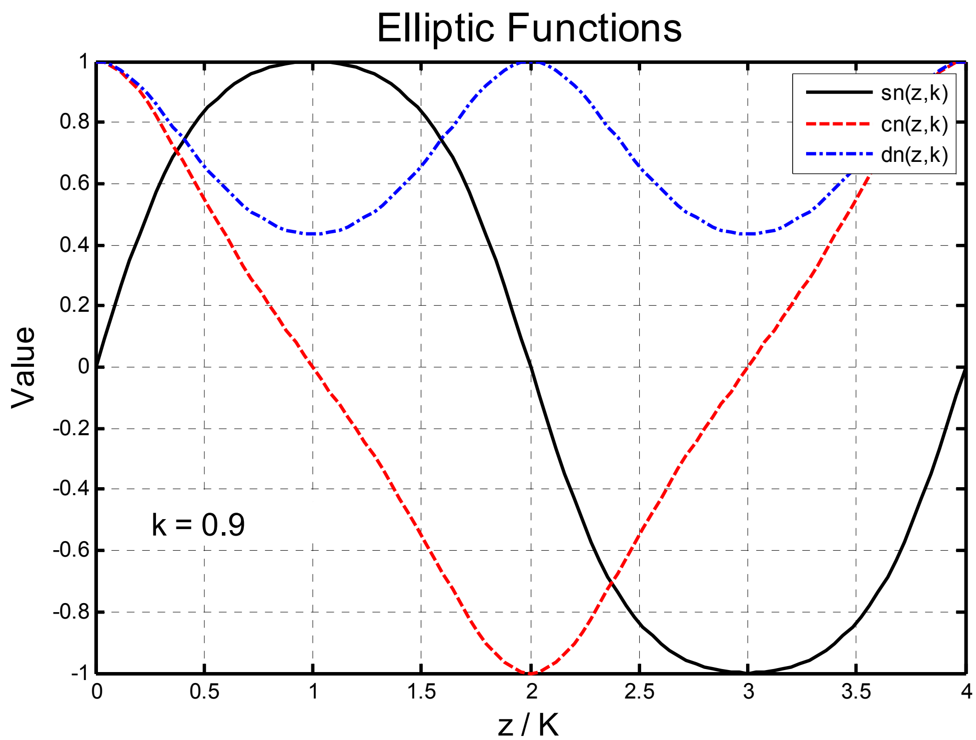


Figure 91 Elliptic functions[65] for *k* = 0.90

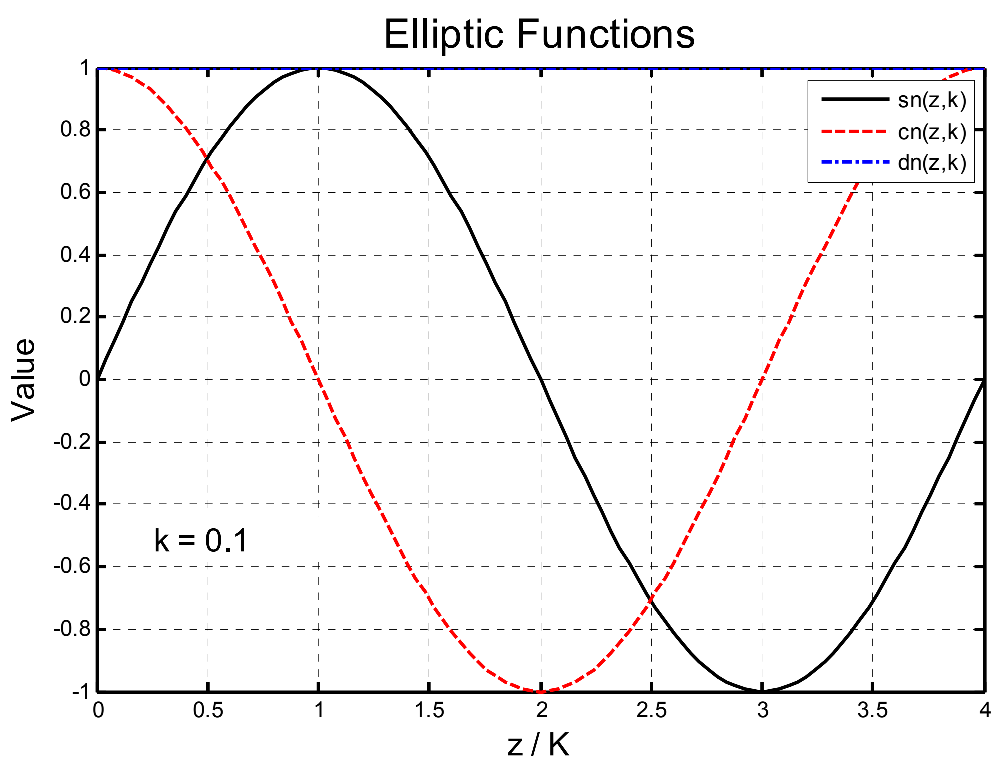


Figure 92 Elliptic functions[66] for *k* = 0.10

65 Calculated in u18311_elliptic_pz.m.
66 Calculated in u18311_elliptic_pz.m.

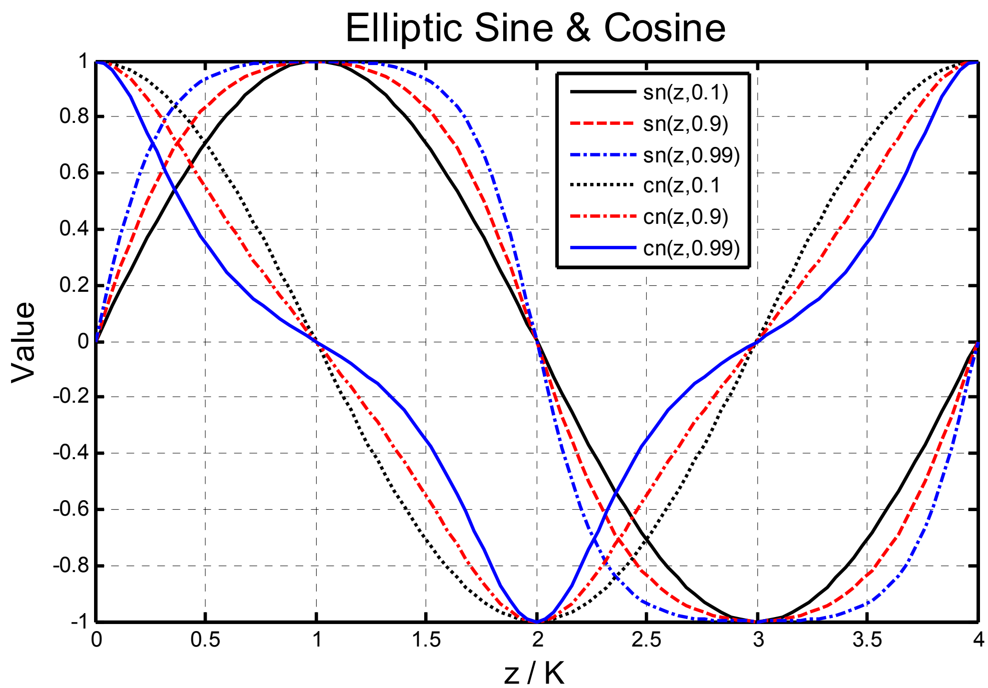


Figure 93 Elliptic *sn*( ) and *cn*( ) functions[67]

Table 10 All 12 Elliptic Functions in Terms of *sn*( ), *cn*( ), and *dn*( ) (First letter of the function on the far left, second letter of the function across the top.)

	s	**c**	**d**	**n**
s	–	$\frac{sn(z,k)}{cn(z,k)}$	$\frac{sn(z,k)}{dn(z,k)}$	$sn(z,k)$
c	$\frac{cn(z,k)}{sn(z,k)}$	–	$\frac{cn(z,k)}{dn(z,k)}$	$cn(z,k)$
d	$\frac{dn(z,k)}{sn(z,k)}$	$\frac{dn(z,k)}{cn(z,k)}$	–	$dn(z,k)$
n	$\frac{1}{sn(z,k)}$	$\frac{1}{cn(z,k)}$	$\frac{1}{dn(z,k)}$	–

As $k \to 0$, the *sn*( ) and *cn*( ) functions become increasingly sinusoidal as shown in Figure 92 and Figure 93 because in the limit, $z = \theta$ thereby resulting in

$$\begin{aligned} sn(z,0) &= \sin(z) \\ cn(z,0) &= \cos(z) \end{aligned} \tag{6.86}$$

[67] Calculated in u18311_elliptic_pz.m.

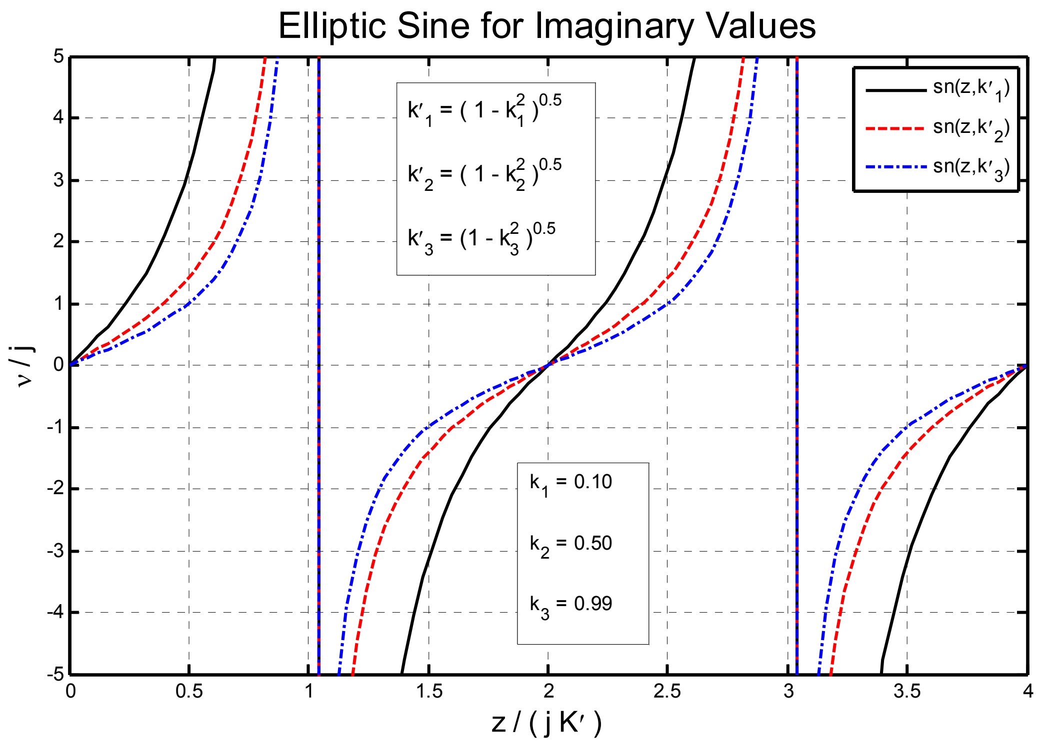


Figure 94 Elliptic sine functions for an imaginary argument[68]

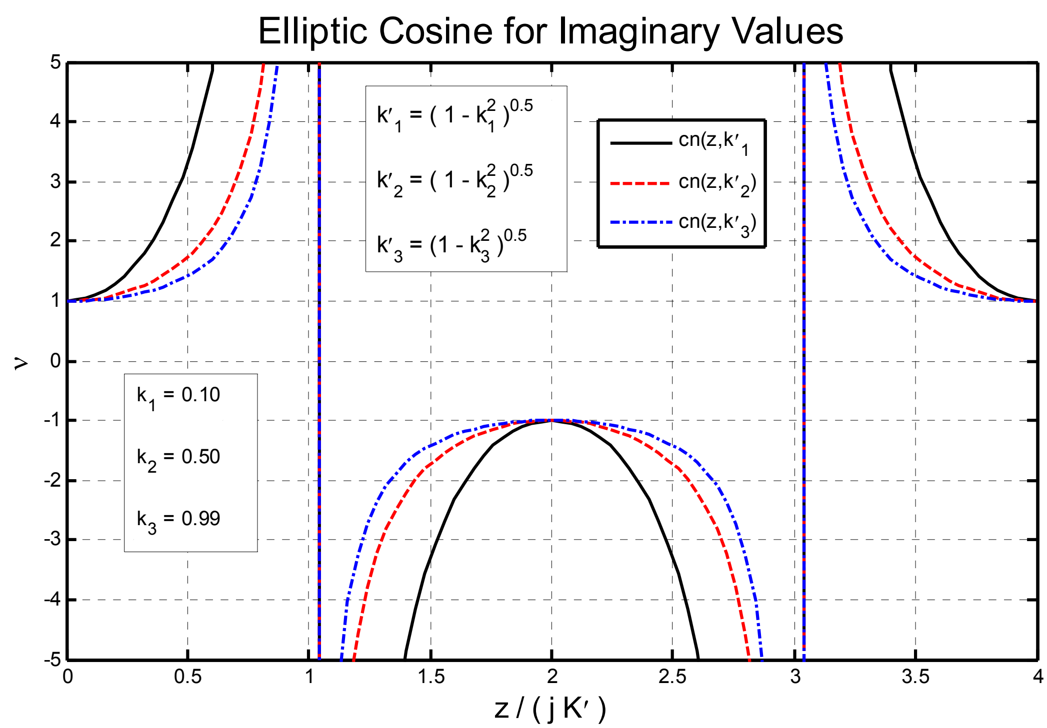


Figure 95 Elliptic cosine functions for an imaginary argument[69]

In the case of imaginary z values where $z = j\,u$, it can be shown [10]

[68] Calculated in u18311_elliptic_pz.m.
[69] Calculated in u18311_elliptic_pz.m.

$$sn(ju,k) = j\frac{sn(u,k')}{cn(u,k')}$$
$$cn(ju,k) = \frac{1}{cn(u,k')} \qquad (6.87)$$
$$dn(ju,k) = \frac{dn(u,k')}{cn(u,k')}$$

Of these three, the *sn*( ) function is of greatest interest in the design of elliptic filters. The elliptic sine function is shown in Figure 94 for several different *k*-moduli as a function of *z* / *j* as is the elliptic cosine function in Figure 95.

10.7.5 Elliptic *sne*( ) and *cde*( ) Functions Using Landen's Transformations[70]

The key tool for evaluating the elliptic functions $w = cd(z,k)$ and $w = sn(z,k)$ at any complex value *z* is the Landen transformation. This transformation begins with a given elliptic modulus *k* and generates a sequence of decreasing moduli k_n via a recursion starting with $k_0 = k$. The recursion is given by

$$k_n = \left(\frac{k_{n-1}}{1+\sqrt{1-k_{n-1}^2}} \right)^2 \text{ for } n = 1,2,...,M \qquad (6.88)$$

The moduli k_n decrease rapidly to zero which permits easy evaluation of the *sn*( ) and *cd*( ) values as shown momentarily. Another form of (6.88) is given by

$$k_n = \frac{1-\sqrt{1-k'_{n-1}}}{1+\sqrt{1-k'_{n-1}}} \qquad (6.89)$$

The inverse recursion of (6.88) is given by

$$k_{n-1} = \frac{2\sqrt{k_n}}{1+k_n} \text{ for } n = M, M-1,...,1 \qquad (6.90)$$

The MATLAB elliptic sine function *sne*( ) uses normalized input values such that

$$sn(u \times K, k) = sne(u,k) \qquad (6.91)$$

In order to compute $w = sne(u,k)$, first initialize

$$w_M = \sin\left(u\frac{\pi}{2}\right) \qquad (6.92)$$

and recursively compute

$$w_{n-1}^{-1} = \frac{1}{1+k_n}\left(\frac{1}{w_n} + k_n w_n\right) \text{ for } n = M, M-1,...,1 \qquad (6.93)$$

leaving the final answer as w_0. For computing $w = cde(u,k)$, initialize

[70] Based upon material in [13].

$$w_M = \cos\left(u\frac{\pi}{2}\right) \quad (6.94)$$

and perform the same recursion given by (6.93) leaving the answer as w_0. Several numerical results are provided in Table 11 to assist in confirming computed results.

Table 11 Computed Elliptic Function Values

k	*K*	$sn(0.2K,k) = sne(0.2,k)$	$sn(0.4K,k) = sne(0.4,k)$	$sn(0.6K,k) = sne(0.6,k)$
0.98	3.0209804455298	0.54113794844234	0.840849536186955	0.95538987217989
0.90	2.28054913842277	0.429472291338501	0.735680640297899	0.903822534082928
0.80	1.99530277555208	0.382521258305844	0.682296930663461	0.872518193323011

10.7.6 Inverse Elliptic *cde*( ) and *sne*( ) Functions Using Landen's Transformation

The inverse of *cde*( ) can be calculated in a very similar fashion as done in §10.7.5. Given a specific value $w = cde(u,k)$ for which the inverse is to be computed, first set $w_0 = w$. The reverse recursion of (6.93) is given by

$$w_n = \frac{2w_{n-1}}{(1+k_n)\left(1+\sqrt{1-k_{n-1}^2 w_{n-1}^2}\right)} \quad \text{for } n = 1,2,...,M \quad (6.95)$$

The repeated recursions will end with $w_M = \cos\left(u\frac{\pi}{2}\right)$ from which the final answer follows as

$$u = \frac{2}{\pi}\cos^{-1}(w_M) \quad (6.96)$$

The only difference involved with computing the inverse *sne*( ) function is that in the final step (6.96) is replaced by

$$u = \frac{2}{\pi}\sin^{-1}(w_M) \quad (6.97)$$

to obtain the final answer.

10.7.7 Exact Solution to Equation (6.27)

Equation (6.27) is repeated here for convenience as

$$sn\left(\frac{NK_1 z}{K}, k_1\right) = \frac{j}{\varepsilon_p} \quad (6.98)$$

The text outlined an earlier close approximate solution as given in (6.30) but it is instructive to follow through with the exact solution here. Letting $z = j\,x$, (6.98) becomes

$$sn\left(j\frac{NK_1}{K}x, k_1 \right) = j\frac{sn\left(\frac{NK_1}{K}x, \sqrt{1-k_1^2} \right)}{cn\left(\frac{NK_1}{K}x, \sqrt{1-k_1^2} \right)} = \frac{j}{\varepsilon_p} \tag{6.99}$$

by using the top identity given in (6.87). Canceling j out on each side and recognizing that $sn^2(x,y) + cn^2(x,y) = 1$, the right-hand portion of this can be rewritten as

$$sn\left(\frac{NK_1}{K}x, \sqrt{1-k_1^2} \right) = \frac{1}{\sqrt{1+\varepsilon_p^2}} \tag{6.100}$$

The solution for x quickly follows as

$$x = \frac{K}{NK_1} sn^{-1}\left(\frac{1}{\sqrt{1+\varepsilon_p^2}}, \sqrt{1-k_1^2} \right) \tag{6.101}$$

where it is assumed that if $sn(uK_x, k_x) = w$, then $uK_x = sn^{-1}(w, k_x)$. Some *sn*( ) inverse function implementations, however, return the *normalized* value u rather than uK_x thereby offering some potential confusion. Defining $asn(w, k_x) = u$, $k_x = \sqrt{1-k_1^2}$, and K_x as the associated complete elliptic integral,

$$x = \frac{K\,K_x}{NK_1} asn\left(\frac{1}{\sqrt{1+\varepsilon_p^2}}, k_x \right) \tag{6.102}$$

A few numerical examples should serve to eliminate any confusion.

Since k_1 is given by (6.6), it is usually quite small for all practical design cases. Repeating the equation here,

$$k_1 = \sqrt{\frac{10^{0.1A_{pass}} - 1}{10^{0.1A_{stop}} - 1}} < 1 \tag{6.103}$$

even if A_{pass} is as large as 0.5 dB (for the passband ripple), k_1 will be less than 0.01 so long as minimum stopband attenuation A_{stop} is greater than 31 dB.

Table 12 Example Calculations

k_1	K_1	N	K	k_x	A_p, dB	ε	x Equ. (6.102)	Approx Equ. (6.30)
0.5	1.685750	5	2.0	0.866025	0.10	0.1526204	0.440627044455	0.6566661
0.1	1.574746	5	2.0	0.994987	0.10	0.1526204	0.632566132275	“
0.01	1.570836	5	2.0	0.999950	0.10	0.1526204	0.656390017576	“
0.001	1.570797	5	2.0	0.9999995	0.10	0.1526204	0.656663291799	“
0.001	1.570797	5	2.0	0.9999995	0.25	0.2434209	0.540006574347	0.5400077
0.001	1.570797	5	2.0	0.9999995	0.50	0.3493114	0.451779250502	0.4517798

10.7.8 MATLAB Script

The MATLAB solution to (6.98) is simply given by

$$z = \frac{K}{N} asne\left(\frac{j}{\varepsilon_p}, k_1 \right) \tag{6.104}$$

since the *asne*( ) function handles complex arguments directly.

10.7.9 Computing Elliptic Sine and Cosine Using Theta-Functions

Elliptic functions can also be represented in terms of series. Many older references use *theta functions* to calculate several of the elliptic functions. The results are presented here as a matter of continuity and without proof[71] as

$$sn(z,k) = \frac{1}{\sqrt{k}} \frac{\theta_1\left(\frac{z}{2K}, q \right)}{\theta_0\left(\frac{z}{2K}, q \right)} \tag{6.105}$$

$$cn(z,k) = \sqrt{\frac{k'}{k}} \frac{\theta_2\left(\frac{z}{2K}, q \right)}{\theta_0\left(\frac{z}{2K}, q \right)} \tag{6.106}$$

$$dn(z,k) = \sqrt{k'} \frac{\theta_3\left(\frac{z}{2K}, q \right)}{\theta_0\left(\frac{z}{2K}, q \right)} \tag{6.107}$$

The *q*-parameter is known as the *modular constant* and is given by

$$q = \exp\left(-\pi \frac{K'}{K} \right) \tag{6.108}$$

and the individual *theta functions* are given as

$$\theta_0\left(\frac{z}{2K}, q \right) = 1 + 2\sum_{m=1}^{\infty} \left[(-1)^m q^{m^2} \cos\left(2m \frac{\pi z}{2K} \right) \right] \tag{6.109}$$

$$\theta_1\left(\frac{z}{2K}, q \right) = 2q^{1/4} \sum_{m=0}^{\infty} \left\{ (-1)^m q^{m(1+m)} \sin\left[(2m+1) \frac{\pi z}{2K} \right] \right\} \tag{6.110}$$

$$\theta_2\left(\frac{z}{2K}, q \right) = 2q^{1/4} \sum_{m=0}^{\infty} \left\{ q^{m(1+m)} \cos\left[(2m+1) \frac{\pi z}{2K} \right] \right\} \tag{6.111}$$

[71] From Appendix A of [8].

$$\theta_0\left(\frac{z}{2K},q\right)=1+2\sum_{m=1}^{\infty}\left[q^{m^2}\cos\left(2m\frac{\pi z}{2K}\right)\right] \tag{6.112}$$

Since $q < 1$, these series converge fairly rapidly and any degree of precision desired can be obtained. The formula are directly applicable for complex z values as well.

As noted elsewhere[72], the rather lengthy calculations represented by (6.108) can be shortened substantially by using the recursive approximation

$$q_m = q_0 + 2q_{m-1}^5 - 5q_{m-1}^9 + 10q_{m-1}^{13} \tag{6.113}$$

for m = 1, 2, … until the desired accuracy has been obtained where

$$q_0 = \frac{1}{2}\frac{1-\sqrt{k'}}{1+\sqrt{k'}} \tag{6.114}$$

It normally suffices to truncate the recursion in (6.113) leading to

$$q \cong q_0 + 2q_0^5 + 15q_0^9 + 150q_0^{13} \tag{6.115}$$

The modular constant (6.108) is plotted versus the elliptic modulus value k in Figure 96 and the approximation error using (6.113) is shown in Figure 97 illustrating that the convergence is indeed rapid.

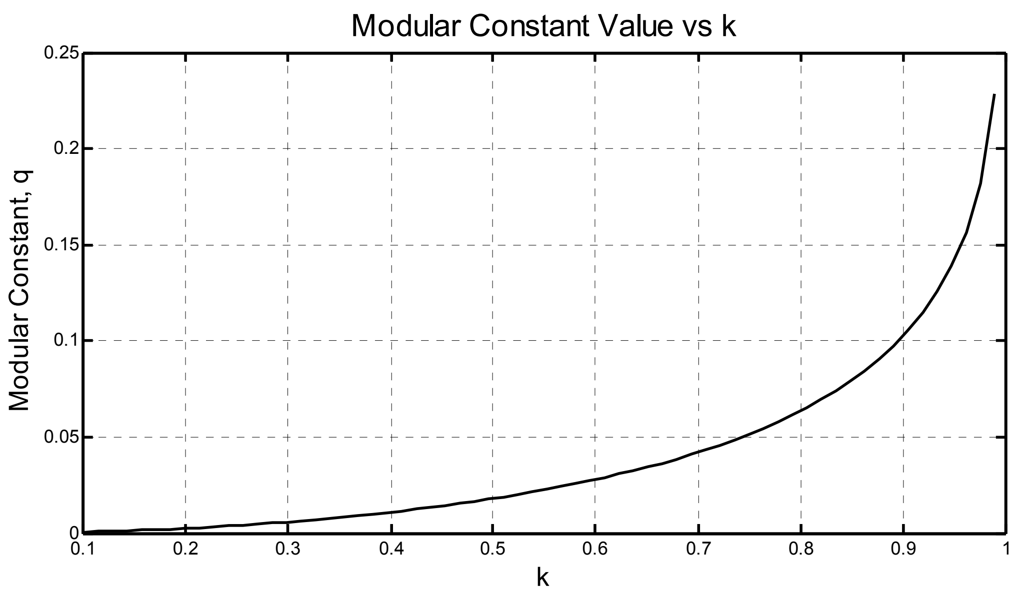


Figure 96 Exact modular function value[73] (6.108) versus elliptic modulus parameter k

[72] Chapter 5 of [8].

[73] Computed using u18404_mod_constant.m.

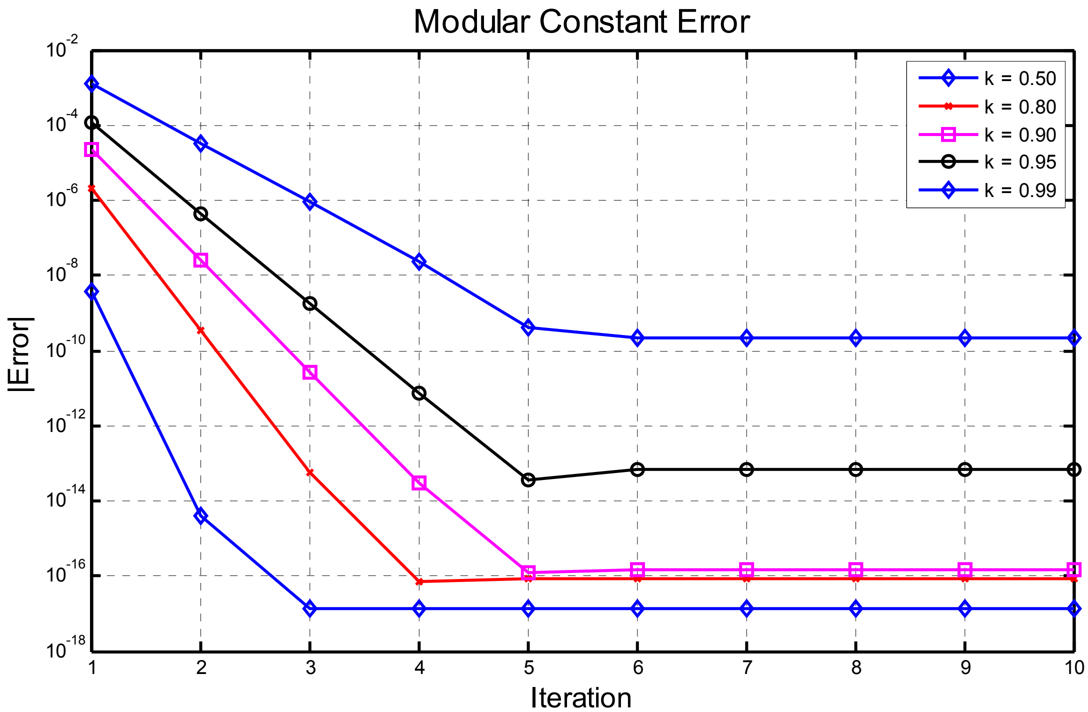


Figure 97 Modular constant approximation error versus iteration number using (6.113)

10.8 Elliptic Filter Synthesis

Modern filter synthesis methods usually rely upon first developing a driving-point impedance function based upon the insertion loss techniques described in §1.2 and §1.3. This first step will also be employed here, but the next step which is used in the synthesis process follows the method proposed by Amstutz in [11].

10.8.1 Amstutz Elliptic Filter Synthesis Method

This method is frequently cited in the literature because it is the only method known which avoids the mounting precision issues involved with polynomial manipulation in the customary synthesis methods. That said, the precision requirements in calculating $dZ / d\omega$ with this method are rather severe and cannot be taken lightly. Two computational methods are discussed shortly.

Consider two cascaded elliptic filter sections as shown in Figure 98. This topology is the basis for the derivations which follow and it is a simple matter to convert the results to the dual topology later. Assume that the radian resonance frequency of inductor M_1 and capacitor C_1 is given by ω_1, and similarly for the second section comprising of M_2 and C_2 which are series-resonant at frequency ω_2. In a neighborhood of $p_1 = j\,\omega_1$, the impedance of the series-resonant section is given by

$$Z_{M_1C_1}\left(p\right) = M_1\left(p - \frac{p_1^2}{p}\right) \tag{6.116}$$

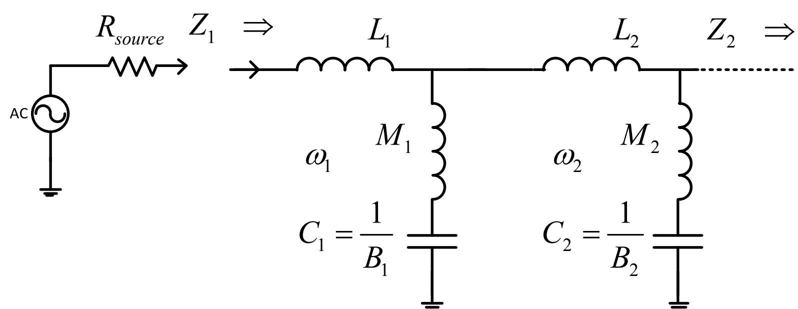


Figure 98 The Amstutz synthesis method relies upon an ingenious permutation of elliptic filter sections. The left-most trap is series resonant at radian frequency ω_1 and the second section series resonant at ω_2.

In general, the impedance on the right-hand side of the first section is not zero, but the input impedance Z_1 in a sufficiently small neighborhood of p_1 is still given very accurately by

$$Z_1\left(p\right) = pL_1 + M_1\left(p - \frac{p_1^2}{p}\right) \tag{6.117}$$

Differentiating (6.117) produces

$$\left.\frac{dZ_1}{dp}\right|_{p=p_1} = L_1 + 2M_1 \tag{6.118}$$

Based upon (6.117),

$$L_1 = \frac{Z_1\left(p_1\right)}{p_1} \tag{6.119}$$

and from (6.118)

$$M_1 = \frac{1}{2}\left[\left. \frac{dZ_1}{dp} \right|_{p=p_1} - L_1 \right] \tag{6.120}$$

Consequently,

$$C_1 = -\frac{1}{M_1 p_1^2} \tag{6.121}$$

Returning to Figure 98, it can be shown that any two-port having this structure has an equivalent two-port network with the same structure but with the resonant circuits associated with the resonant frequencies ω_1 and ω_2 interchanged as shown in Figure 99. Once the input impedance function for the filter is known, the designer can choose whether to place the elliptic section associated with ω_1 first or second, and similarly with the ω_2 section. The input impedance in terms of L's and C's is only easily calculated, however, using (6.119) through (6.121) for the first section. Amstutz recognized these facts and used them to synthesize elliptic filters using two basic steps. In the first step, each of the elliptic filter sections is computed as if it were the first section in the complete filter cascade. In the second step, Amstutz brought in each of these sections from the left (or right) and iteratively permuted their position in the cascade until it finally appeared on the far right (or left) of the cascade. The entire filter was subsequently synthesized by iteratively bringing in one new LC section at a time.

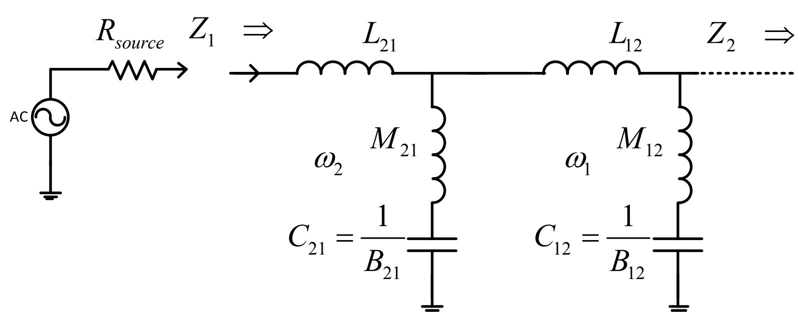


Figure 99 Two cascaded elliptic filter sections where the sections have been permutated

The Amstutz method begins with computing the LC elliptic filter section values associated with each trap-frequency ω_m as if it were the first section in the overall filter cascade. Let these values be denoted by $L_{m,1}$, $M_{m,1}$, and $C_{m,1}$. The second subscript denotes the assumed position for the elliptic filter section within the cascade where the indexing begins from the left (input) side of the filter.

Only one of the sections can in fact be the first section in the cascade of course. Subsequent sections are introduced on the left (right) side and then permutated from left to right (right to left) using the Amstutz algorithm until they are ultimately placed on the far-right (-left) side of the cascade.

Given $L_{1,1}$, $M_{1,1}$, and $C_{1,1}$ for the first section, assume that a second elliptic section is to be appended to the filter having starting values $L_{2,1}$, $M_{2,1}$, and $C_{2,1}$. Once the new section has been permutated to be the second section in the cascade, its component values are denoted by $L_{2,2}$, $M_{2,2}$, and $C_{2,2}$. These component values (with the 22 subscripts) can be calculated from the 11 and 21 subscripted values as follows. First let

$$U = L_{11} - L_{21} \tag{6.122}$$

$$V = \left[\frac{U\,C_{11}}{\left(\omega_2^{-2} - \omega_1^{-2}\right)} - 1 \right]^{-1} \tag{6.123}$$

From these results, then compute

$$\frac{1}{C_{22}} = \frac{V^2}{C_{21}} - \frac{\left(1+V\right)^2}{C_{11}} \tag{6.124}$$

$$M_{22} = \frac{1}{C_{22}\omega_2^2} \tag{6.125}$$

$$L_{22} = UV \tag{6.126}$$

A third elliptic filter section can be brought into the filter section cascade by applying this permutation algorithm two times. Working again from the left side of the filter, the ω_3 and ω_1 sections are first permutated so that the resonant sections left-to-right are ω_1, ω_3, ω_2. Then the algorithm is applied a second time to the last two sections thereby resulting in the sequence ω_1, ω_2, ω_3. Additional details can be found in [1], [3], and of course [11].

10.8.2 Input Impedance Function $Z_{in}(s)$ and $dZ_{in} / d\omega$

A second crucial step in Amstutz's solution is his computation of the filter's input impedance and especially $dZ_{in} / d\omega$ at the trap resonant frequencies. Although Z_{in} is well conditioned, it is not adequately conditioned for direct numerical differentiation with high-order filters. As discussed briefly in §10.8.3, even fairly complicated differentiation techniques fall prey to numerical precision issues and are in general, not reliable for higher order cases.

The input impedance function for the elliptic lowpass filter can be found from the characteristic function and transducer gain function since the reflection coefficient is given by (see (2.12) and (2.26))

$$\rho\left(s\right) = \frac{K\left(s\right)}{T\left(s\right)} \tag{6.127}$$

From (2.23), the transducer gain function is given by

$$T\left(s\right) = \frac{E\left(s\right)}{P\left(s\right)} \tag{6.128}$$

and the characteristic gain function is similarly given by

$$K\left(s\right) = \frac{F\left(s\right)}{P\left(s\right)} \tag{6.129}$$

The form adopted for $T(s)$ is the same as that used in (2.23), namely

$$T\left(s\right) = t_0 \frac{\prod_n \left(s - t_n\right)}{\prod_m \left(s - p_m\right)} = \frac{E\left(s\right)}{P\left(s\right)} \tag{6.130}$$

and similarly for $K(s)$

$$K(s) = s_0 \frac{\prod_n (s - s_n)}{\prod_m (s - p_m)} \tag{6.131}$$

which produces the reflection coefficient given by

$$\rho(s) = \frac{s_0}{t_0} \frac{\prod_n (s - s_n)}{\prod_n (s - t_n)} \tag{6.132}$$

The magnitude of the reflection coefficient at every attenuation pole p_k is unity and the corresponding input impedance can be written as

$$Z_{in}(p_k) = R_{source} \frac{1 + \rho(p_k)}{1 - \rho(p_k)} \tag{6.133}$$

where

$$\rho(p_k) = \varepsilon \exp\left[j \sum_n \arg(p_k - s_n) - j \sum_n \arg(p_k - t_n) \right] \tag{6.134}$$

with $\varepsilon = \pm 1$ where the +1 value applies for symmetric filters and the –1 for antimetric filters. In the symmetric and antimetric elliptic filter case, $\arg(p_k - s_n) = \pi / 2$ which makes it possible to rewrite (6.134) as

$$\begin{aligned} \rho(p_k) &= \varepsilon \exp\left[j \sum_n \left(\frac{\pi}{2} - \arg(p_k - t_n) \right) \right] \\ &= \varepsilon \exp\left[j \sum_n \phi_k^n \right] \end{aligned} \tag{6.135}$$

Using this result in (6.133) then produces

$$\begin{aligned} Z_{in}\left(p_k\right) &= R_{source}\frac{1+\varepsilon\exp\left[j\sum_n \phi_k^n\right]}{1-\varepsilon\exp\left[j\sum_n \phi_k^n\right]} \\ &= R_{source}\frac{\exp\left[-\frac{j}{2}\sum_n \phi_k^n\right]+\varepsilon\exp\left[\frac{j}{2}\sum_n \phi_k^n\right]}{\exp\left[-\frac{j}{2}\sum_n \phi_k^n\right]-\varepsilon\exp\left[\frac{j}{2}\sum_n \phi_k^n\right]} \\ &= \begin{cases} j\,R_{source}\cot\left[\frac{1}{2}\sum_n \phi_k^n\right] & \text{for } \varepsilon = 1 \\ -j\,R_{source}\tan\left[\frac{1}{2}\sum_n \phi_k^n\right] & \text{for } \varepsilon = -1 \end{cases} \end{aligned} \quad (6.136)$$

which is identical to Amstutz (3.4). This result is easily calculated for each p_k value with excellent accuracy. Amstutz derives a similarly important result for $dZ_{in}/d\omega$ as

$$\begin{aligned} \left.\frac{2R_{source}}{R_{source}^2 - Z_{in}^2\left(\omega\right)}\frac{dZ_{in}}{d\omega}\right|_{\omega=p_k} &= \left(\frac{d}{d\omega}\sum_n \arg\left(j\omega - t_n\right)\right)_{\omega=p_k} \\ &= -\sum_n\left[\frac{\sigma_n}{\sigma_n^2+\left(\omega_k-\omega_n\right)^2}\right] \end{aligned} \quad (6.137)$$

The results provided in (6.136) and (6.137) make it possible to accurately compute Z_{in} and $dZ_{in}/d\omega$ on the basis of the p_k and t_n values alone thereby making it possible to compute all of the initial LC sections as described earlier in §10.8.1.

> *NOTE:* $Z_{in}\left(\omega\right)$ must be replaced by $R_{source}R_{load}/Z_{in}\left(\omega\right)$ for the *antimetric* (even-order) case when LC sections are introduced from the output side of the filter rather than the input.

10.8.3 Aside: Computing $dZ_{in}/d\omega$ Using Numerical Differentiation

Calculating the derivative of the input impedance with respect to frequency at $\omega = \omega_m$ is particularly sensitive to numerical imprecision. Using a simple finite-difference to approximate the derivative at each characteristic function zero is insufficient except for the most benign design cases. The approach described here is to first perform a polynomial curve-fit through a set of calculated Z_{in} values at radian frequencies ω_{wrk} = $\left[\omega_m - 2\delta\omega,\quad \omega_m-\delta\omega,\quad \omega_m,\quad \omega_m+\delta\omega,\quad \omega_m+2\delta\omega\right]$, followed by differentiation of this polynomial at radian frequency ω_m which is a specific characteristic function pole of interest. Assuming that Z_{in} is closely approximated by a 4th-order polynomial in ω near ω_m as

$$Z_{in}\left(\omega_m+\delta\omega\right) = a\left(\delta\omega\right)^4 + b\left(\delta\omega\right)^3 + c\left(\delta\omega\right)^2 + d\left(\delta\omega\right) + e \quad (6.138)$$

the 4th-order polynomial which passes through all of the Z_{in} values precisely has coefficients given by[74]

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \frac{1}{24}\begin{bmatrix} 1 & -4 & 6 & -4 & 1 \\ -2 & 4 & 0 & -4 & 2 \\ -1 & 16 & -30 & 16 & -1 \\ 2 & -16 & 0 & 16 & -2 \\ 0 & 0 & 24 & 0 & 0 \end{bmatrix}\begin{bmatrix} Z_{in}\left(\omega_m - 2\delta\omega\right) \\ Z_{in}\left(\omega_m - \delta\omega\right) \\ Z_{in}\left(\omega_m\right) \\ Z_{in}\left(\omega_m + \delta\omega\right) \\ Z_{in}\left(\omega_m + 2\delta\omega\right) \end{bmatrix} \tag{6.139}$$

Differentiating (6.138) with respect to ω at ω_n produces the derivative

$$\frac{dZ_{in}}{d\omega} \cong \frac{\delta Z_{in}}{\delta\omega} \tag{6.140}$$

implying that only the 4th row of (6.139) need actually be computed. In a completely analogous manner, a 6th-order polynomial can be used to curve-fit the Z_{in} values and upon differentiating the resultant polynomial,

$$\frac{dZ_{in}}{d\omega} \cong \frac{1}{\delta\omega}\left[\frac{-1}{60},\frac{3}{20},\frac{-3}{4},0,\frac{3}{4},\frac{-3}{20},\frac{1}{60}\right]\left[Z_x\right]^T \tag{6.141}$$

where Z_x is the impedance row-vector given by calculating Z_{in} at radian frequencies $\omega_m + n\,\delta\omega$ for $n = \{-3,-2,...,3\}$. This level of precision in the impedance derivative is required in order to have accurate results through roughly 11th-order elliptic filters over most stopband / passband attenuation combinations. Even so, this approach is considerably less accurate than Amstutz's method even though it also involves substantially more computation.

More details about the Amstutz method are provided in §17. A thorough study of Amstutz's original paper [11] is highly recommended for anyone who wants to master the mathematical design of elliptic filters.

[74] [19], equations (2.171), 2(.172).

11 Filter Synthesis Using Iterated Analysis[75]

Perhaps the most valuable tenet provided in [20] is the use of ABCD matrices to formulate the design solution. In the case of a lossless two-port network as shown in Figure 100, the ABCD formulation provides

$$\begin{bmatrix} v_1 \\ i_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} v_2 \\ i_2 \end{bmatrix} \tag{142}$$

The associated power-related transfer function of interest is given by[76]

$$H(s) = \frac{2v_2}{E}\sqrt{\frac{R_1}{R_2}} \tag{143}$$

and in terms of decibel power-gain,

$$\begin{aligned} G_{dB} &= 10\log_{10}\left[\frac{4R_1}{R_2}\left|\frac{v_2}{E}\right|^2\right] \\ &= 10\log_{10}\left[\frac{4R_1}{R_2}\right] - 10\log_{10}\left[\left|\frac{E}{v_2}\right|^2\right] \end{aligned} \tag{144}$$

In the situation where $R_2 \to \infty$, the first term in (144) is discarded and attention is focused on the voltage-gain term alone. It is convenient to take E = 1 without any loss of generality. In terms of ABCD components,

$$\frac{E}{v_2} = \frac{1}{v_2} = A + \frac{B}{R_2} + R_1 C + \frac{R_1}{R_2} D \tag{145}$$

The calculation method employed herein ultimately makes use of the Newton method and partial derivatives with respect to each network element value e_n are consequently needed. From (144),

$$\begin{aligned} \frac{\partial G_{dB}}{\partial e_n} &= \frac{-10}{\log_e(10)} \frac{\partial}{\partial e_n}\left[\log_e\left(\frac{1}{v_2 \overline{v_2}}\right)\right] \\ &= \frac{-10}{\log_e(10)} \frac{\partial}{\partial e_n}\left[\log_e\left(\frac{1}{v_2}\right) + \log_e\left(\frac{1}{\overline{v_2}}\right)\right] \\ &= \frac{-10}{\log_e(10)}\left[v_2 \frac{\partial}{\partial e_n}\left(\frac{1}{v_2}\right) + \overline{v_2} \frac{\partial}{\partial e_n}\left(\frac{1}{\overline{v_2}}\right)\right] \\ &= \frac{-20}{\log_e(10)} \operatorname{Re}\left[v_2 \frac{\partial}{\partial e_n}\left(\frac{1}{v_2}\right)\right] \end{aligned} \tag{146}$$

---

[75] Motivated by [20].
[76] This is the *reciprocal* of the relationship used in [20] so do not get confused.

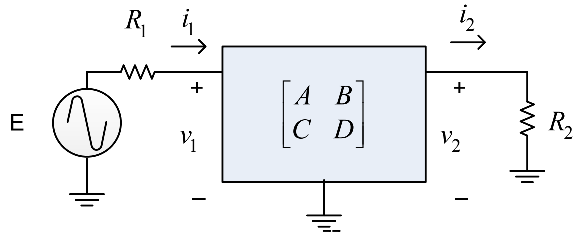


Figure 100 Definition terms for ABCD matrix description[77]

The quantity v_2^{-1} is directly available from (145). In order to compute the partial derivatives required in (146), we turn our attention now to **Figure 101** and **Figure 102**. For the shunt-admittance case in **Figure 101**, the resultant ABCD network is given by

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{Bmatrix} \left[A_1 A_2 + B_1 \left(Y A_2 + C_2 \right) \right] & \left[A_1 B_2 + B_1 \left(Y B_2 + D_2 \right) \right] \\ \left[C_1 A_2 + D_1 \left(Y A_2 + C_2 \right) \right] & \left[C_1 B_2 + D_1 \left(Y B_2 + D_2 \right) \right] \end{Bmatrix} \tag{147}$$

from which

$$\frac{\partial}{\partial Y} \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} B_1 A_2 & B_1 B_2 \\ D_1 A_2 & D_1 B_2 \end{bmatrix} \tag{148}$$

Based upon **Figure 100**, (143), and (148)

$$\frac{E}{v_2} = \frac{1}{v_2} = \left[R_1 C + \frac{R_1}{R_2} D + A + \frac{B}{R_2} \right] \tag{149}$$

which leads to

$$\frac{\partial}{\partial Y} \left(\frac{E}{v_2} \right) = R_1 D_1 A_2 + \frac{R_1}{R_2} D_1 B_2 + B_1 A_2 + \frac{1}{R_2} B_1 B_2 \tag{150}$$

For the series-impedance case shown in **Figure 102**, the resultant derivative is

$$\frac{\partial}{\partial Z} \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} A_1 C_2 & A_1 D_2 \\ C_1 C_2 & C_1 D_2 \end{bmatrix} \tag{151}$$

from which

$$\frac{\partial}{\partial Z} \left(\frac{E}{v_2} \right) = \left[R_1 C_1 C_2 + \frac{R_1}{R_2} C_1 D_2 + A_1 C_2 + \frac{1}{R_2} A_1 D_2 \right] \tag{152}$$

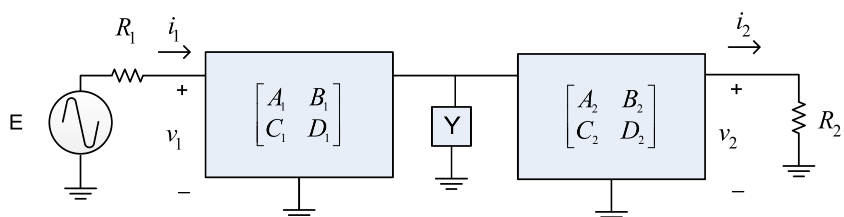


Figure 101 Cascaded network with shunt admittance Y present

[77] From U22332 Figures.vsd.

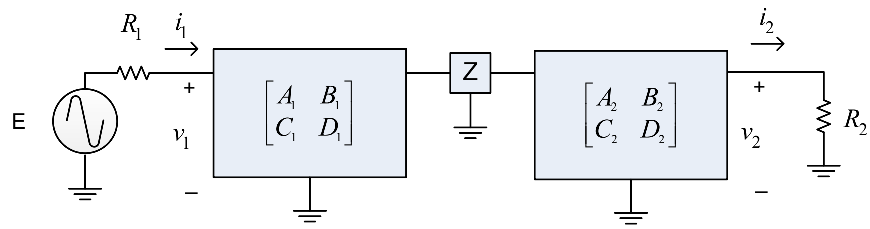


Figure 102 Cascaded network with series impedance Z present

In order to get the partial derivatives with respect to the component values e_k, the chain-rule must be used. In the case where admittance Y is a shunt capacitor C_{shunt}

$$\frac{\partial Y}{\partial C_{shunt}} = j\omega \tag{153}$$

If the shunt admittance is a series-LC trap,

$$Y = \frac{1}{\dfrac{1}{sC_{trap}} + sL_{trap}} = \frac{j\omega C_{trap}}{1 - \left(\dfrac{\omega}{\omega_{trap}} \right)^2} \tag{154}$$

which leads to

$$\frac{\partial Y}{\partial C_{trap}} = \frac{j\omega}{1 - \left(\dfrac{\omega}{\omega_{trap}} \right)^2} \tag{155}$$

If the series impedance is an inductance

$$\frac{\partial Z}{\partial L_{series}} = j\omega \tag{156}$$

In the case where the series impedance is a series LC-trap (parallel L and C),

$$Z = \frac{1}{sC_{trap} + \dfrac{1}{sL_{trap}}} = \frac{j\omega L_{trap}}{1 - \left(\dfrac{\omega}{\omega_{trap}} \right)^2} \tag{157}$$

which leads to

$$\frac{\partial Z}{\partial L_{trap}} = \frac{j\omega}{1 - \left(\dfrac{\omega}{\omega_{trap}} \right)^2} \tag{158}$$

It is assumed here that the trap resonant frequencies are known a priori as part of the transfer function approximation step. Consequently,

$$\omega_{trap} = \frac{1}{\sqrt{L_{trap} C_{trap}}} \tag{159}$$

and this relationship can be used to eliminate one of the unknowns during the iterative calculations for each trap.

Recapping then, the needed partial derivatives are given by (146) where v_2 is available from (149) as

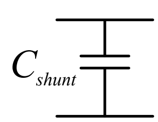
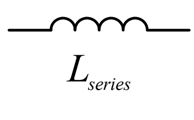
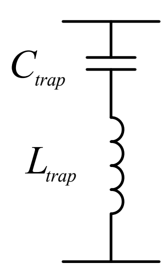
$$v_2 = \frac{1}{R_1 C + \frac{R_1}{R_2} D + A + \frac{B}{R_2}} \tag{160}$$

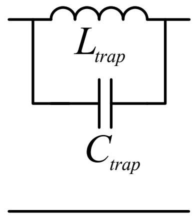
and the partial derivatives with respect to *Y* and *Z* are given by (150) and (152) respectively. The chain rule must then be applied to these in order to translate them into partial derivatives with respect to e_n per (153), (155), (156), or (158) as appropriate.

11.1 Iterative Calculation

Assume now that a power-gain transfer function goal $G_{goal}(f)$ is known and that a filter circuit topology has been chosen which contains the correct number of poles and zeros to realize this transfer function. The iterative calculation consists of using a sufficient number of (fixed) frequency points to enable a least-squares solution to take place using Newton's Method in an iterative manner.

Table 13 Summary of Lowpass Section Types

Type	Lowpass Section Type	Partial Derivative	ABCD
1		$\frac{\partial Y}{\partial C_{shunt}} = j\omega$ (161)	$\begin{bmatrix} 1 & 0 \\ j\omega C_{shunt} & 1 \end{bmatrix}$ (162)
2		$\frac{\partial Z}{\partial L_{series}} = j\omega$ (163)	$\begin{bmatrix} 1 & j\omega L_{series} \\ 0 & 1 \end{bmatrix}$ (164)
3		$\frac{\partial Y}{\partial C_{trap}} = \frac{j\omega}{1 - \left(\frac{\omega}{\omega_{trap}}\right)^2}$ (165)	$\begin{bmatrix} 1 & 0 \\ \frac{j\omega C_{trap}}{1 - \left(\frac{\omega}{\omega_{trap}}\right)^2} & 1 \end{bmatrix}$ (166)

Type	Lowpass Section Type	Partial Derivative	ABCD
4		$$\frac{\partial Z}{\partial L_{trap}} = \frac{j\omega}{1-\left(\frac{\omega}{\omega_{trap}}\right)^2}$$ (167)	$$\begin{bmatrix} 1 & \frac{j\omega L_{trap}}{1-\left(\frac{\omega}{\omega_{trap}}\right)^2} \\ 0 & 1 \end{bmatrix}$$ (168)

Let the set of fixed evaluation frequencies be denoted by f_k and the difference between the goal attenuation values and the ones at iteration-*k* denoted by

$$G_{err}\left(f_k\right) = G_{goal}\left(f_k\right) - G_{dB}\left(f_k\right) \tag{169}$$

The (non-square) matrix of partial derivatives has the form

$$J = \left[\frac{\partial G_{dB}\left(f_k\right)}{\partial U_n} \frac{\partial U_n}{\partial e_n} \right] \tag{170}$$

where the matrix rows correspond to the different evaluation frequencies f_k and the matrix columns correspond to the circuit element values being iterated. The element values after the p^{th} iteration are given by

$$\left[e_n\right]_p = \left[e_n\right]_{p-1} + \gamma \, lms\left(J_{p-1}, G_{err,p}\right) \tag{171}$$

where γ is a numerical gain term having a magnitude < 1 and *lms* designates a least-mean-square solution for the matrix and vector involved.

Table 14 Normalized LC Values for 3rd-Order Unloaded Inverse Chebyshev Lowpass

Stopband, dB	C_1	C_2	C_3	L
25	0.494	0.42	2.495	1.784
30	0.732	0.323	3.000	2.322
35	0.992	0.255	3.631	2.941
40	1.287	0.204	4.39	3.671
50	2.028	0.135	6.472	5.561

Table 15 Normalized LC Values for 5th-Order Unloaded Inverse Chebyshev Lowpass

Stopband, dB	C_1	C_2	C_3	C_4	C_5	L_1	L_2
30	0.0021	0.4960	1.3204	0.6302	1.7103	0.6966	1.4351
40	0.1743	0.3385	1.7652	0.4419	2.1750	1.0205	2.0467
50	0.3533	0.2458	2.3203	0.3271	2.7592	1.4055	2.7647
60	0.5453	0.1852	3.0065	0.2495	3.4924	1.8654	3.6255

11.2 Appendix: MATLAB Script for Unloaded Case ($R_2 = \infty$)

The first portion of the script computes the poles and zeros of the inverse Chebyshev lowpass filter of interest. There are a number of calculations done pertaining to characteristic functions, etc.

```
%================ u22336_inverse_chebyshev_iterated_synthesis.m =================
%
%
%   J.A. Crawford
%   20 March 2014
%
%   Earlier synthesis program appended with iteration-based design of
%   5th order inverse Chebyshev lpf as an unloaded LC network
%   First attempt at using Orchard's iterative design technique and this
%   example shows that it works well.
%
%   A more general script is required if other load impedance values are
%   needed, or if the order needs to be changed.
%
%   Pretty cool, I must say. Anxious to incorporate this method into
%   a general synthesis tool in C#. I've had need for being able to
%   use an arbitrary load resistance on quite a few past occassions.
%
%   Don't get good convergence for stopband attenuations less than about
%   30 dB for some reason. Otherwise, fantastic even up to 110 dB
%   stopband attenuations. Found that the reason is that C1 must be allowed
%   to go negative for these lower stopband attenuations.
%
fil_order= 5;
Astop_dB= 80;
jx= i;

Astop= 10^(0.1*Astop_dB);
epsilon= sqrt(1/(Astop-1));

odd_order= (mod(fil_order,2)==1);       % 1 if odd-order, otherwise 0

%
%   Computes poles and zeros of inverse Chebyshev
%
aa= sinh( (1/fil_order)*asinh(1/epsilon) );
bb= cosh( (1/fil_order)*asinh(1/epsilon));
%
nden= fil_order;
nnum= floor(fil_order/2);

kk=1:nden;
theta= (2*kk-1)*pi/(2*fil_order);
cheby_poles= -aa*sin(theta) + jx*bb*cos(theta)

kk=1:nnum;
inv_cheby_poles=1./cheby_poles;                        % All poles in the left-half plane
inv_cheby_zeros= jx./cos( (2*kk-1)*pi/(2*fil_order) );
inv_cheby_zeros(nnum+kk)= conj(inv_cheby_zeros(kk));    % All zeros on jw axis

gam= 1;
for jk=1:length(inv_cheby_poles)
   gam= gam * inv_cheby_poles(jk);
```

```
end
for jk=1:length(inv_cheby_zeros)
   gam= gam / inv_cheby_zeros(jk);
end

%
%   Sweep filter
%
Npts= 512*4;
fswp= 10.^(-3+5*(1:Npts)/Npts);
Hfil= zeros(1,Npts);
tau= zeros(1,Npts);
for jk=1:Npts
   Hcas= 1;
   taux= 0;
   ss= i*2*pi*fswp(jk);
   for ii=1:length(inv_cheby_zeros)
      Hcas= Hcas * (ss - inv_cheby_zeros(ii));
   end
   for ii=1:length(inv_cheby_poles)
      Hcas= Hcas / (ss - inv_cheby_poles(ii));

      px= -real(inv_cheby_poles(ii) );
      py= imag( inv_cheby_poles(ii) );
      taux= taux + px/(px^2 + (abs(ss)-py)^2);
   end
   Hfil(jk)= 10*log10( abs(gam*Hcas)^2 );
   tau(jk)= taux;
end

figure(1);
clf;
p1= semilogx( fswp, Hfil, 'r' );
set( p1, 'LineWidth', 2 );
grid on
h= gca;
set( h, 'LineWidth', 2 );
xlabel( 'Frequency, Hz', 'FontName', 'Arial', 'FontSize', 12 );
ylabel( 'Gain, dB', 'FontName', 'Arial', 'FontSize', 12 );
title( 'Inverse Chebyshev', 'FontName', 'Arial', 'FontSize', 14 );
axis( [0.001, 100, -80, 10] );

%
%   Look at filter group delay
%
figure(2);
clf;
p1= semilogx( fswp, tau, 'r' );
set( p1, 'LineWidth', 2 );
grid on
h= gca;
set( h, 'LineWidth', 2 );
xlabel( 'Frequency, Hz', 'FontName', 'Arial', 'FontSize', 12 );
ylabel( 'Group Delay', 'FontName', 'Arial', 'FontSize', 12 );
title( 'Inverse Chebyshev', 'FontName', 'Arial', 'FontSize', 14 );
%axis( [0.01, 100, -80, 0] );
%
%=====================================================================
%
%
%
%   Form H(s) polynomial
```

```
%
%
%
H_num= 1;
for ii=1:length(inv_cheby_zeros)
   H_num= conv( [1 -inv_cheby_zeros(ii)], H_num );
end
H_den= 1;
for ii=1:length(inv_cheby_poles)
   H_den= conv( [1 -inv_cheby_poles(ii)], H_den );
end

Hx= gam*polyval( H_num, jx*2*pi*fswp ) ./ polyval( H_den, jx*2*pi*fswp );

figure(3);
clf;
p1= semilogx( fswp, 10*log10( abs(Hx).^2 ), 'r' );
set( p1, 'LineWidth', 2 );
title( 'Check on H(w) Using Poles & Zeros',  'FontName', 'Arial', 'FontSize', 14 );
set( p1, 'LineWidth', 2 );
grid on
h= gca;
set( h, 'LineWidth', 2 );
xlabel( 'Frequency, Hz', 'FontName', 'Arial', 'FontSize', 12 );
ylabel( 'Group Delay', 'FontName', 'Arial', 'FontSize', 12 );
%
%==========================================================================
%==========================================================================
%
%
%   Form Characteristic Function K(s)
%
%
K2= abs(1./Hx).^2 - 1;      % Transducer gain is 1/Hx here
K2_dB= 10*log10( K2 );

figure(4);
clf;
p1= semilogx( fswp, K2_dB, 'r' );
set( p1, 'LineWidth', 2 );
grid on
title( '|K(\omega)|^2',  'FontName', 'Arial', 'FontSize', 14 );
%
%   H(s)=  gam * [H_num] / [H_den]
%
%   |H(s)|^2 = gam*gam * [H_num]*[H_num] / ( [H_den]*[H_den] )
%
%   |T(s)|^2 = |1/H(s)|^2 = 1 + |K(s)|^2
%
%   |T(s)|^2 = [H_den]*[H_den]/(gam^2 * [H_num]*[H_num] )
%           = [p_num]/[p_den]
%
p_den= gam*gam*conv(H_num, H_num)
%
%   H_num only has even-power polynomial coeffs whereas
%   H_den has both, so must take care of conjugation (i.e., negation)
%   of odd-power polynomial coefficients
%
H_den2= H_den;
for ii=length(H_den)-1:-2:1 H_den2(ii)= -H_den2(ii); end
p_num= conv(H_den, H_den2)
%
```

```
%   Check this form for |T(s)|^2
%
if( 0 )
   T2check= polyval(p_num, jx*2*pi*fswp) ./ polyval(p_den, jx*2*pi*fswp);
   hold on
   p1= semilogx( fswp, 10*log10( abs(T2check)), 'ko' );
end
%
%   Form numerator polynomial for |K(s)|^2 = |T(s)|^2 - 1
%
p_wrk= p_num;
lx= length(p_wrk)-length(p_den)+1;
jk= length(p_den);
for ii=length(p_wrk):-1:lx
   p_wrk(ii)= p_wrk(ii) - p_den(jk);
   jk= jk - 1;
end
p_wrk= real(p_wrk)

figure(5);
clf;
K2= polyval( p_wrk, jx*2*pi*fswp ) ./ polyval( p_den, jx*2*pi*fswp );
p1= semilogx( fswp, 10*log10( abs(K2) ), 'k' );
set( p1, 'LineWidth', 2 );
grid on
title( '|K|^2 From Polynomials','FontName', 'Arial', 'FontSize', 14 );
%
%   Factor K^2
%
K2_num_roots= roots( p_wrk );
K2_den_roots= roots( p_den );

figure(6);
clf;
plot( real(K2_num_roots), imag(K2_num_roots), 'ro' );
title( 'Roots of |K|^2 Numerator' );
grid on
%
%   Retain only left-plane roots
%
K_num= 1;
mm=1;
for ii=1:length(K2_num_roots)
   if( real(K2_num_roots(ii)) <= 0.001 )
      K_num= conv( [1 -K2_num_roots(ii)], K_num );
      K_num_lhp_roots(mm)= K2_num_roots(ii);
      mm= mm+1;
   end
end
K_den= H_num
K_num= K_num

figure(7);
clf;
K= (1/gam)*polyval( K_num, jx*2*pi*fswp ) ./ polyval( K_den, jx*2*pi*fswp );
p1= semilogx( fswp, 10*log10( abs(K).^2 ), 'b-' );
set( p1, 'LineWidth', 2 );
grid on
title( 'Final K(\omega)','FontName', 'Arial', 'FontSize', 14 );
h= gca;
set( h, 'LineWidth', 2 );
xlabel( 'Frequency, Hz', 'FontName', 'Arial', 'FontSize', 12 );
```

```
ylabel( 'dB', 'FontName', 'Arial', 'FontSize', 12 );
axis( [0.0001, max(fswp), -60, 80] );

disp( 'Characteristic Function Numerator:' );
disp( num2str( real(K_num), '%5.4e  ' ) );
disp( 'Denominator:' );
disp( num2str( real(K_den), '%5.4e  ' ) );

K_num_lhp_roots
K2_den_roots
%=======================================================================
%
%   Iteratively design filter
%
%   Focus on 5th order filter here
%
C1= 0.5;
C2= 0.25;
C3= 3.0;
C4= 0.25;
C5= 3.0;
wo1= 1.7013;
wo2= 1.0515;
L1= 1/(C2*wo1*wo1);
L2= 1/(C4*wo2*wo2);

freqs= 0.001*10.^(4*(0:50)/50);
Nfreqs=length(freqs);

abcd1= @(s) [ 1 0; s*C1 1 ];
abcd2= @(s) [ 1 1/(s*C2+1/(s*L1)); 0 1];
abcd3= @(s) [ 1 0; s*C3 1 ];
abcd4= @(s) [ 1 1/(s*C4+1/(s*L2)); 0 1];
abcd5= @(s) [ 1 0; s*C5 1 ];

err= 0;
figure(100);
clf;
PD= zeros(Nfreqs,5);    % Partial derivatives
clear g1;
clear g2;
for iter= 1:40
   for ff=1:Nfreqs
      g1(ff)= gam*polyval( H_num, jx*2*pi*freqs(ff) ) ./ polyval( H_den, jx*2*pi*freqs(ff) );
      g1(ff)= 10*log10( abs(g1(ff))^2 );

      ss= jx*2*pi*freqs(ff);

      abcd= abcd1(ss) * abcd2(ss) * abcd3(ss) * ...
           abcd4(ss) * abcd5(ss);
      g2(ff)= 1/(abcd(1,1) + abcd(2,1));
      g2(ff)= 10*log10( abs(g2(ff))^2 );

      dg(ff)= g1(ff) - g2(ff);
      if( abs(dg(ff)) > 10 )
         dg(ff)= 10*sign(dg(ff));
      end

      %
      %   Get partial derivatives for this frequency
      %   and for each element value
      %
```

```
    %   Cap C1
    %
    ss= jx*2*pi*freqs(ff);

    abcd= abcd1(ss)*abcd2(ss)*abcd3(ss)*abcd4(ss)*abcd5(ss);
    apc= (-20/log(10)) / ( abcd(1,1) + abcd(2,1) );

    M1= [ 1 0; 0 1];
    M2= abcd2(ss)*abcd3(ss)*abcd4(ss)*abcd5(ss);
    PD(ff,1)= ( M1(1,2)*M2(1,1) + M2(1,1)*M1(2,2) )*ss*apc;

    %
    %   First LC trap
    %
    M1= abcd1(ss);
    M2= abcd3(ss)*abcd4(ss)*abcd5(ss);
    PD(ff,2)= ( M1(1,1)*M2(2,1) + M1(2,1)*M2(2,1) )*ss/( 1 - abs(ss/wo1)^2 )*apc;

    %
    %   Cap C3
    %
    M1= abcd1(ss)*abcd2(ss);
    M2= abcd4(ss)*abcd5(ss);
    PD(ff,3)= ( M1(1,2)*M2(1,1) + M2(1,1)*M1(2,2) )*ss*apc;

    %
    %   Second LC trap
    %
    M1= abcd1(ss)*abcd2(ss)*abcd3(ss);
    M2= abcd5(ss);
    PD(ff,4)= ( M1(1,1)*M2(2,1) + M1(2,1)*M2(2,1) )*ss/( 1 - abs(ss/wo2)^2 )*apc;

    %
    %   Cap C5
    %
    M1= abcd1(ss)*abcd2(ss)*abcd3(ss)*abcd4(ss);
    M2= [ 1 0; 0 1 ];
    PD(ff,5)= ( M1(1,2)*M2(1,1) + M2(1,1)*M1(2,2) )*ss*apc;
end
%
%   Update element values
%
de= lscov(real((PD)),dg');

gamma= 0.25;

C1= abs(C1 + gamma*de(1));
L1= abs(L1 + gamma*de(2));
C3= abs(C3 + gamma*de(3));
L2= abs(L2 + gamma*de(4));
C5= abs(C5 + gamma*de(5));

C2= abs(1/(wo1^2*L1));
C4= abs(1/(wo2^2*L2));

abcd1= @(s) [ 1 0; s*C1 1 ];
abcd2= @(s) [ 1 1/(s*C2+1/(s*L1)); 0 1];
abcd3= @(s) [ 1 0; s*C3 1 ];
abcd4= @(s) [ 1 1/(s*C4+1/(s*L2)); 0 1];
abcd5= @(s) [ 1 0; s*C5 1 ];

semilogx( freqs, g1, 'r' );
```

```
    hold on
    semilogx( freqs, g2, 'k--' );
    hold on
end

figure(200);
clf;
for ii=1:Npts
    ss= jx*2*pi*fswp(ii);

    abcd= abcd1(ss) * abcd2(ss) * abcd3(ss) * ...
             abcd4(ss) * abcd5(ss);
    gn(ii)= 10*log10( abs(1/(abcd(1,1) + abcd(2,1)))^2 );


    g1(ii)= gam*polyval( H_num, ss ) ./ polyval( H_den, ss );
        g1(ii)= 10*log10( abs(g1(ii))^2 );
end
axes( 'FontName', 'Arial', 'FontSize', 12 );
p1= semilogx( fswp, gn, 'r' );
set( p1, 'LineWidth', 2 );
hold on
p1= semilogx( fswp, g1, 'k--' );
set( p1, 'LineWidth', 2 );
h= gca;
set( h, 'LineWidth', 2 );
grid on
xlabel( 'Normalized Frequency, Hz', 'FontName', 'Arial', 'FontSize', 12 );
ylabel( 'Gain, dB', 'FontName', 'Arial', 'FontSize', 12 );
title( 'Original Versus Modified Filter Gain', 'FontName', 'Arial', 'FontSize', 12 );
legend( 'Iterative Design Result', 'Ideal from Poles & Zeros' );

C1
C2
C3
C4
C5
L1
L2
```

11.3 Appendix: MATLAB Script for Unequally Terminated Case (Arbitrary R_2)

```
%=============== u22345_inverse_chebyshev_iterated_synthesis.m ==================
%
%   Same as u22336_inverse_chebyshev_iterated_synthesis.m except that
%   arbitrary load impedance can be specified
%
%   J.A. Crawford
%   23 March 2014
%
%   Earlier synthesis program appended with iteration-based design of
%   5th order inverse Chebyshev lpf as an unloaded LC network
%   First attempt at using Orchard's iterative design technique and this
%   example shows that it works well.
%
%   A more general script is required if other load impedance values are
%   needed, or if the order needs to be changed.
%
%   Pretty cool, I must say. Anxious to incorporate this method into
%   a general synthesis tool in C#. I've had need for being able to
%   use an arbitrary load resistance on quite a few past occassions.
%
%   Don't get good convergence for stopband attenuations less than about
%   30 dB for some reason. Otherwise, fantastic even up to 110 dB
%   stopband attenuations. Found that the reason is that C1 must be allowed
%   to go negative for these lower stopband attenuations.
%
fil_order= 5;
Astop_dB= 60;
jx= i;

Astop= 10^(0.1*Astop_dB);
epsilon= sqrt(1/(Astop-1));

odd_order= (mod(fil_order,2)==1);       % 1 if odd-order, otherwise 0

%
%   Computes poles and zeros of inverse Chebyshev
%
aa= sinh( (1/fil_order)*asinh(1/epsilon) );
bb= cosh( (1/fil_order)*asinh(1/epsilon));
%
nden= fil_order;
nnum= floor(fil_order/2);

kk=1:nden;
theta= (2*kk-1)*pi/(2*fil_order);
cheby_poles= -aa*sin(theta) + jx*bb*cos(theta)

kk=1:nnum;
inv_cheby_poles=1./cheby_poles;                     % All poles in the left-half plane
inv_cheby_zeros= jx./cos( (2*kk-1)*pi/(2*fil_order) );
inv_cheby_zeros(nnum+kk)= conj(inv_cheby_zeros(kk));    % All zeros on jw axis

gam= 1;
for jk=1:length(inv_cheby_poles)
   gam= gam * inv_cheby_poles(jk);
end
for jk=1:length(inv_cheby_zeros)
   gam= gam / inv_cheby_zeros(jk);
```

```
end

%
%   Sweep filter
%
Npts= 512*4;
fswp= 10.^(-3+5*(1:Npts)/Npts);
Hfil= zeros(1,Npts);
tau= zeros(1,Npts);
for jk=1:Npts
   Hcas= 1;
   taux= 0;
   ss= i*2*pi*fswp(jk);
   for ii=1:length(inv_cheby_zeros)
      Hcas= Hcas * (ss - inv_cheby_zeros(ii));
   end
   for ii=1:length(inv_cheby_poles)
      Hcas= Hcas / (ss - inv_cheby_poles(ii));

      px= -real(inv_cheby_poles(ii) );
      py= imag( inv_cheby_poles(ii) );
      taux= taux + px/(px^2 + (abs(ss)-py)^2);
   end
   Hfil(jk)= 10*log10( abs(gam*Hcas)^2 );
   tau(jk)= taux;
end

figure(1);
clf;
p1= semilogx( fswp, Hfil, 'r' );
set( p1, 'LineWidth', 2 );
grid on
h= gca;
set( h, 'LineWidth', 2 );
xlabel( 'Frequency, Hz', 'FontName', 'Arial', 'FontSize', 12 );
ylabel( 'Gain, dB', 'FontName', 'Arial', 'FontSize', 12 );
title( 'Inverse Chebyshev', 'FontName', 'Arial', 'FontSize', 14 );
axis( [0.001, 100, -80, 10] );

%
%   Look at filter group delay
%
figure(2);
clf;
p1= semilogx( fswp, tau, 'r' );
set( p1, 'LineWidth', 2 );
grid on
h= gca;
set( h, 'LineWidth', 2 );
xlabel( 'Frequency, Hz', 'FontName', 'Arial', 'FontSize', 12 );
ylabel( 'Group Delay', 'FontName', 'Arial', 'FontSize', 12 );
title( 'Inverse Chebyshev', 'FontName', 'Arial', 'FontSize', 14 );
%axis( [0.01, 100, -80, 0] );
%
%======================================================================
%
%
%
%   Form H(s) polynomial
%
%
%
```

```
H_num= 1;
for ii=1:length(inv_cheby_zeros)
   H_num= conv( [1 -inv_cheby_zeros(ii)], H_num );
end
H_den= 1;
for ii=1:length(inv_cheby_poles)
   H_den= conv( [1 -inv_cheby_poles(ii)], H_den );
end

Hx= gam*polyval( H_num, jx*2*pi*fswp ) ./ polyval( H_den, jx*2*pi*fswp );

figure(3);
clf;
p1= semilogx( fswp, 10*log10( abs(Hx).^2 ), 'r' );
set( p1, 'LineWidth', 2 );
title( 'Check on H(w) Using Poles & Zeros',  'FontName', 'Arial', 'FontSize', 14 );
set( p1, 'LineWidth', 2 );
grid on
h= gca;
set( h, 'LineWidth', 2 );
xlabel( 'Frequency, Hz', 'FontName', 'Arial', 'FontSize', 12 );
ylabel( 'Group Delay', 'FontName', 'Arial', 'FontSize', 12 );
%
%===========================================================================
%===========================================================================
%
%
%   Form Characteristic Function K(s)
%
%
K2= abs(1./Hx).^2 - 1;      % Transducer gain is 1/Hx here
K2_dB= 10*log10( K2 );

figure(4);
clf;
p1= semilogx( fswp, K2_dB, 'r' );
set( p1, 'LineWidth', 2 );
grid on
title( '|K(\omega)|^2',  'FontName', 'Arial', 'FontSize', 14 );
%
%   H(s)=  gam * [H_num] / [H_den]
%
%   |H(s)|^2 = gam*gam * [H_num]*[H_num] / ( [H_den]*[H_den] )
%
%   |T(s)|^2 = |1/H(s)|^2 = 1 + |K(s)|^2
%
%   |T(s)|^2 = [H_den]*[H_den]/(gam^2 * [H_num]*[H_num] )
%           = [p_num]/[p_den]
%
p_den= gam*gam*conv(H_num, H_num)
%
%   H_num only has even-power polynomial coeffs whereas
%   H_den has both, so must take care of conjugation (i.e., negation)
%   of odd-power polynomial coefficients
%
H_den2= H_den;
for ii=length(H_den)-1:-2:1 H_den2(ii)= -H_den2(ii); end
p_num= conv(H_den, H_den2)
%
%   Check this form for |T(s)|^2
%
if( 0 )
```

```
    T2check= polyval(p_num, jx*2*pi*fswp) ./ polyval(p_den, jx*2*pi*fswp);
    hold on
    p1= semilogx( fswp, 10*log10( abs(T2check)), 'ko' );
end
%
%   Form numerator polynomial for |K(s)|^2 = |T(s)|^2 - 1
%
p_wrk= p_num;
lx= length(p_wrk)-length(p_den)+1;
jk= length(p_den);
for ii=length(p_wrk):-1:lx
    p_wrk(ii)= p_wrk(ii) - p_den(jk);
    jk= jk - 1;
end
p_wrk= real(p_wrk)

figure(5);
clf;
K2= polyval( p_wrk, jx*2*pi*fswp ) ./ polyval( p_den, jx*2*pi*fswp );
p1= semilogx( fswp, 10*log10( abs(K2) ), 'k' );
set( p1, 'LineWidth', 2 );
grid on
title( '|K|^2 From Polynomials','FontName', 'Arial', 'FontSize', 14 );
%
%   Factor K^2
%
K2_num_roots= roots( p_wrk );
K2_den_roots= roots( p_den );

figure(6);
clf;
plot( real(K2_num_roots), imag(K2_num_roots), 'ro' );
title( 'Roots of |K|^2 Numerator' );
grid on
%
%   Retain only left-plane roots
%
K_num= 1;
mm=1;
for ii=1:length(K2_num_roots)
    if( real(K2_num_roots(ii)) <= 0.001 )
        K_num= conv( [1 -K2_num_roots(ii)], K_num );
        K_num_lhp_roots(mm)= K2_num_roots(ii);
        mm= mm+1;
    end
end
K_den= H_num
K_num= K_num

figure(7);
clf;
K= (1/gam)*polyval( K_num, jx*2*pi*fswp ) ./ polyval( K_den, jx*2*pi*fswp );
p1= semilogx( fswp, 10*log10( abs(K).^2 ), 'b-' );
set( p1, 'LineWidth', 2 );
grid on
title( 'Final K(\omega)','FontName', 'Arial', 'FontSize', 14 );
h= gca;
set( h, 'LineWidth', 2 );
xlabel( 'Frequency, Hz', 'FontName', 'Arial', 'FontSize', 12 );
ylabel( 'dB', 'FontName', 'Arial', 'FontSize', 12 );
axis( [0.0001, max(fswp), -60, 80] );
```

```
disp( 'Characteristic Function Numerator:' );
disp( num2str( real(K_num), '%5.4e  ' ) );
disp( 'Denominator:' );
disp( num2str( real(K_den), '%5.4e  ' ) );

K_num_lhp_roots
K2_den_roots
%==========================================================================
%
%   Iteratively design filter
%
%   Focus on 5th order filter here
%
R1= 1;      % Source impedance
R2= 0.5;      % Load impedance

C1= 0.5;
C2= 0.25;
C3= 3.1;
C4= 0.25;
C5= 3.2;
wo1= 1.7013;
wo2= 1.0515;
L1= 1/(C2*wo1*wo1);
L2= 1/(C4*wo2*wo2);

freqs= 0.01*10.^(3*(0:75)/75);
Nfreqs=length(freqs);

abcd1= @(s) [ 1 0; s*C1 1 ];
abcd2= @(s) [ 1 1/(s*C2+1/(s*L1)); 0 1];
abcd3= @(s) [ 1 0; s*C3 1 ];
abcd4= @(s) [ 1 1/(s*C4+1/(s*L2)); 0 1];
abcd5= @(s) [ 1 0; s*C5 1 ];

err= 0;
figure(100);
clf;
PD= zeros(Nfreqs,5);    % Partial derivatives
clear g1;
clear g2;
for iter= 1:60
   for ff=1:Nfreqs
      g1(ff)= gam*polyval( H_num, jx*2*pi*freqs(ff) ) ./ polyval( H_den, jx*2*pi*freqs(ff) ) * R2/(R1+R2);
      g1(ff)= 10*log10( abs(g1(ff))^2 );

      ss= jx*2*pi*freqs(ff);

      abcd= abcd1(ss) * abcd2(ss) * abcd3(ss) * ...
           abcd4(ss) * abcd5(ss);
      g2(ff)= 1 /(abcd(1,1) + abcd(1,2)/R2 + R1*abcd(2,1) +(R1/R2)*abcd(2,2) );
      g2(ff)= 10*log10( abs(g2(ff))^2 );

      dg(ff)= g1(ff) - g2(ff);
      if( abs(dg(ff)) > 10 )
         dg(ff)= 10*sign(dg(ff));
      end

      %
      %   Get partial derivatives for this frequency
      %   and for each element value
      %
```

```
        %   Cap C1
        %
        ss= jx*2*pi*freqs(ff);

        abcd= abcd1(ss)*abcd2(ss)*abcd3(ss)*abcd4(ss)*abcd5(ss);
        apc= (-20/log(10)) / (abcd(1,1) + abcd(1,2)/R2 + R1*abcd(2,1) +(R1/R2)*abcd(2,2) );

        M1= [ 1 0; 0 1];
        M2= abcd2(ss)*abcd3(ss)*abcd4(ss)*abcd5(ss);
          PD(ff,1)= ( R1*M1(2,2)*M2(1,1) + (R1/R2)*M1(2,2)*M2(1,2) + M1(1,2)*M2(1,1) + (1/R2)*M1(1,2)*M2(1,2)
)*ss*apc;

        %
        %   First LC trap
        %
        M1= abcd1(ss);
        M2= abcd3(ss)*abcd4(ss)*abcd5(ss);
          PD(ff,2)= (R1*M1(2,1)*M2(2,1) + (R1/R2)*M1(2,1)*M2(2,2) + M1(1,1)*M2(2,1) + (1/R2)*M1(1,1)*M2(2,2)
)*ss/( 1 - abs(ss/wo1)^2 )*apc;

        %
        %   Cap C3
        %
        M1= abcd1(ss)*abcd2(ss);
        M2= abcd4(ss)*abcd5(ss);
          PD(ff,3)= ( R1*M1(2,2)*M2(1,1) + (R1/R2)*M1(2,2)*M2(1,2) + M1(1,2)*M2(1,1) + (1/R2)*M1(1,2)*M2(1,2)
)*ss*apc;

        %
        %   Second LC trap
        %
        M1= abcd1(ss)*abcd2(ss)*abcd3(ss);
        M2= abcd5(ss);
         PD(ff,4)= (R1*M1(2,1)*M2(2,1) + (R1/R2)*M1(2,1)*M2(2,2) + M1(1,1)*M2(2,1) + (1/R2)*M1(1,1)*M2(2,2)
)*ss/( 1 - abs(ss/wo2)^2 )*apc;

        %
        %   Cap C5
        %
        M1= abcd1(ss)*abcd2(ss)*abcd3(ss)*abcd4(ss);
        M2= [ 1 0; 0 1 ];
          PD(ff,5)= ( R1*M1(2,2)*M2(1,1) + (R1/R2)*M1(2,2)*M2(1,2) + M1(1,2)*M2(1,1) + (1/R2)*M1(1,2)*M2(1,2)
)*ss*apc;
    end
    %
    %   Update element values
    %
    de= lscov(real((PD)),dg');

    gamma= 0.25;

    C1= (C1 + gamma*de(1));
    L1= (L1 + gamma*de(2));
    C3= (C3 + gamma*de(3));
    L2= (L2 + gamma*de(4));
    C5= (C5 + gamma*de(5));

    C2= abs(1/(wo1^2*L1));
    C4= abs(1/(wo2^2*L2));

    abcd1= @(s) [ 1 0; s*C1 1 ];
    abcd2= @(s) [ 1 1/(s*C2+1/(s*L1)); 0 1];
```

```
    abcd3= @(s) [ 1 0; s*C3 1 ];
    abcd4= @(s) [ 1 1/(s*C4+1/(s*L2)); 0 1];
    abcd5= @(s) [ 1 0; s*C5 1 ];

    semilogx( freqs, g1, 'r' );
    hold on
    semilogx( freqs, g2, 'k--' );
    hold on
end

figure(200);
clf;
for ii=1:Npts
    ss= jx*2*pi*fswp(ii);

    abcd= abcd1(ss) * abcd2(ss) * abcd3(ss) * ...
          abcd4(ss) * abcd5(ss);
    gn(ii)= 10*log10( abs(1/(abcd(1,1) + abcd(1,2)/R2 + R1*abcd(2,1) +(R1/R2)*abcd(2,2) ))^2 );


    g1(ii)= gam*polyval( H_num, ss ) ./ polyval( H_den, ss ) *(R2/(R1+R2));
       g1(ii)= 10*log10( abs(g1(ii))^2 );
end
axes( 'FontName', 'Arial', 'FontSize', 12 );
p1= semilogx( fswp, gn, 'r' );
set( p1, 'LineWidth', 2 );
hold on
p1= semilogx( fswp, g1, 'k--' );
set( p1, 'LineWidth', 2 );
h= gca;
set( h, 'LineWidth', 2 );
grid on
xlabel( 'Normalized Frequency, Hz', 'FontName', 'Arial', 'FontSize', 12 );
ylabel( 'Voltage Gain, dB', 'FontName', 'Arial', 'FontSize', 12 );
title( 'Original Versus Modified Filter Gain', 'FontName', 'Arial', 'FontSize', 12 );
legend( 'Iterative Design Result', 'Ideal from Poles & Zeros' );

C1
C2
C3
C4
C5
L1
L2
```

12 Candidate Network Circuit Topologies

poles at infinity, poles at zero, finite poles, zeros, etc. and LC networks

13 References

1. Ellis, Michael G., *Electronic Filter Analysis and Synthesis*, Chapter 7, Artech House.
2. Lam, Harry Y-F., *Analog and Digital Filters, Design and Realization*, Prentice-Hall, 1979.
3. Cuthbert, Thomas R., *Circuit Design Using Personal Computers*, John Wiley & Sons, 1983.
4. Williams, Arthur B., *Electronic Filter Design Handbook*, McGraw-Hill Book, 1981.
5. Green, E., "Synthesis of Ladder Networks to Give Butterworth or Chebyshev Response in the Passband," *Proc. of the IEE*, Part III: Radio and Communication Engineering, March 1954.
6. Orchard, H.J., "Formulae for Ladder Filters," *Wireless Engineer*, Jan. 1953.
7. Darlington, Sidney, "Simple Algorithms for Elliptic Filters and Generalizations Thereof," *IEEE Trans. Circuits and Systems*, Dec. 1978.
8. Antoniou, Andreas, *Digital Filters Analysis and Design*, McGraw-Hill Book, 1979.
9. Korn, G.A., and T.M. Korn, *Mathematical Handbook for Scientists and Engineers*, 2nd ed., McGraw-Hill Book, 1968.
10. Daniels, Richard W., *Approximation Methods for Electronic Filter Design*, McGraw-Hill Book, 1974.
11. Amstutz, Pierre, "Elliptic Approximation and Elliptic Filter Design on Small Computers," *IEEE Trans. Circuits and Systems*, Dec. 1978.
12. Safar, F.G., F.W. Stephenson, R.W. Steer, "A PC-Based Program for the Interactive Design of Cauer Filters," *30th Symposium on Circuits and Systems*, 1988.
13. Orfanidis, S.J., "Lecture Notes on Elliptic Filter Design," Nov. 2006, www.ece.rutgers.edu/~orfanidi/ece521.
14. Temes, G.C., and J.W. LaPatra, *Circuit Synthesis and Design*, McGraw-Hill Book, 1977.
15. Temes, G.C., and S. K. Mitra, *Modern Filter Theory and Design*, John Wiley & Sons, 1973.
16. Christian, E., *LC-Filters, Design, Testing, and Manufacturing*, John Wiley & Sons, 1983.
17. Skwirzynski, J.K., "On Synthesis of Filters," *IEEE Trans Circuit Theory*, Jan. 1971.
18. Saal, R., and E. Ulbrich, "On the Design of Filters by Synthesis," *IRE Trans. Circuit Theory*, Dec. 1958.
19. Crawford, J.A., *Advanced Phase-Lock Techniques*, Artech House, 2008.
20. Orchard, H.J., "Filter Design by Iterated Analysis," *IEEE Trans. Circuits and Systems*, Nov. 1985.
21. Hank Zumbahlen, Analog Devices, "Mini Tutorial MT-204, The Bessel Response," U22163.
22. I.M. Filanovsky, "Bessel-Butterworth Transitional Filters," IEEE, 2014, U22370.

14 Appendix I: Group Delay Based Using Hilbert Transforms

The phase response of an all-pole filter (e.g., Butterworth, Chebyshev, Bessel, etc.) can be computed from the amplitude response by way of the Hilbert transform. Given a transfer function $H(\omega)$ of the form

$$H(j\omega) = A(\omega)e^{-j\theta(\omega)} = e^{-\alpha(\omega)}e^{-j\theta(\omega)} \tag{8.1}$$

α () and θ () form a Hilbert transform pair as

$$\begin{aligned}\theta(\omega) &= -\frac{1}{\pi}\int_{-\infty}^{+\infty}\frac{\alpha(\upsilon)}{\omega-\upsilon}d\upsilon \\ \alpha(\omega) &= \frac{1}{\pi}\int_{-\infty}^{+\infty}\frac{\theta(\upsilon)}{\omega-\upsilon}d\upsilon\end{aligned} \tag{8.2}$$

Focusing on the first portion of (8.2) and noting that α ( ) is an even function of ν, this can be re-written as

$$\theta(\omega) = -\frac{2\omega}{\pi}\int_{0}^{+\infty}\frac{\alpha(\upsilon)}{\omega^2-\upsilon^2}d\upsilon \tag{8.3}$$

In order to deal with the denominator singularity at $\upsilon = \omega$, (8.3) can be broken into a left-hand and right-hand side integral as

$$\theta(\omega) \cong -\frac{2\omega}{\pi}\int_{0}^{\omega-\delta\omega}\frac{\alpha(\upsilon)}{\omega^2-\upsilon^2}d\upsilon - \frac{2\omega}{\pi}\int_{\omega+\delta\omega}^{+\infty}\frac{\alpha(\upsilon)}{\omega^2-\upsilon^2}d\upsilon \tag{8.4}$$

where $\delta\omega$ is chosen appropriately small.

The group delay calculation involves the first derivative of θ with respect to time, and while it is tempting to perform this calculation by bringing a differential operator underneath the integrals in (8.4), doing so is illegal in this case because the integration and differentiation operations are not interchangeable. To see this more clearly, consider the portion of (8.4) which has been left out of the integration range in (8.3), namely

$$\delta\theta\Big|_{\omega} = -\frac{2\omega}{\pi}\int_{\omega-\delta\omega}^{\omega+\delta\omega}\frac{\alpha(\upsilon)}{\omega^2-\upsilon^2}d\upsilon = -\frac{2\omega}{\pi}\int_{\omega-\delta\omega}^{\omega+\delta\omega}\frac{\alpha(\upsilon)}{(\omega-\upsilon)(\omega+\upsilon)}d\upsilon \tag{8.5}$$

For $\delta\omega \ll \omega$ and slowly-changing $\alpha(\omega)$, this can be closely approximated by

$$\delta\theta\Big|_{\omega} \cong -\frac{1}{\pi}\int_{\omega-\delta\omega}^{\omega+\delta\omega}\frac{\alpha(\upsilon)}{(\omega-\upsilon)}d\upsilon \rightarrow -\frac{\alpha(\omega)}{\pi}\int_{\omega-\delta\omega}^{\omega+\delta\omega}\frac{d\upsilon}{\omega-\upsilon} \rightarrow -\frac{\alpha(\omega)}{\pi}\int_{-\delta\omega}^{\delta\omega}\frac{du}{u} \tag{8.6}$$

In this form, the singularity is still present, but since the integrand is an odd-function of ν, as $\delta\omega \rightarrow 0$, so does $\delta\theta$.

Temporarily assuming that the order of differentiation and integration can be interchanged in computing the group delay from (8.3), the computation begins as

$$\tau_g(\omega) = -\frac{d}{d\omega}\theta(\omega) = \frac{2}{\pi}\frac{d}{d\omega}\left[\int_0^{+\infty} \frac{\alpha(\upsilon)\omega}{\omega^2 - \upsilon^2} d\upsilon\right] \quad (8.7)$$

Carrying out the differentiation underneath the integral leads to

$$\tau_g(\omega) = \frac{2}{\pi}\int_0^{+\infty} \alpha(\upsilon)\frac{1 - 2\omega^2}{\left(\omega^2 - \upsilon^2\right)^2} d\upsilon \quad (8.8)$$

In this case, the integrand is an even-function of ν and there can be no convergence of the integral near the singularity. Since a group delay function does in fact exist for any given filter, the non-convergence of (8.8) is a restatement that integration and differentiation in (8.7) is not valid in this case.

15 Appendix II: Additional Design Notes

15.1 Butterworth Filter Design Parameters

There are 5 degrees of freedom for Butterworth filter design: (i) filter order *N*, (ii) filter passband (–3 dB) frequency f_{pass}, (iii) maximum passband attenuation A_{pass}, (iv) filter stopband frequency f_{stop}, and (v) filter stopband attenuation A_{stop}. The passband frequency F_{pass} and passband attenuation A_{pass} are assumed to be fixed thereby leaving 3 remaining degrees of freedom. Only 2 of the 3 remaining parameters can be chosen independently. The fundamental design equation is given by (3.8) which can be written as

$$N_{\min} \geq \frac{1}{2} \frac{\log_e\left(\dfrac{10^{A_{stop}/10}-1}{10^{A_{pass}/10}-1}\right)}{\log_e\left(\dfrac{f_{stop}}{f_{pass}}\right)} \tag{9.1}$$

where A_{pass} = 3 dB is assumed and f_{pass} is assumed known.

15.1.1 Butterworth Filter Shape Factor Given A_{stop} and N

$$\frac{f_{stop}}{f_{pass}} = \exp\left[\frac{1}{2N}\log_e\left(\frac{10^{A_{stop}/10}-1}{10^{A_{pass}/10}-1}\right)\right] \tag{9.2}$$

15.1.2 Butterworth Stopband Attenuation Given f_{stop} and N

$$A_{stop} = 10\log_{10}\left\{1+\exp\left[2N\log_e\left(\frac{f_{stop}}{f_{pass}}\right)-\log_e\left(10^{A_{pass}/10}-1\right)\right]\right\} \tag{9.3}$$

15.2 Chebyshev Filter Design Parameters

The Chebyshev filter case has the same degrees of freedom except that f_{pass} is the *passband ripple bandwidth* and A_{pass} is the passband ripple. The key design equation is (4.18) which is rewritten here as

$$N \geq \frac{\cosh^{-1}\left(\sqrt{\dfrac{10^{A_{stop}/10}-1}{10^{A_{pass}/10}-1}}\right)}{\cosh^{-1}\left(\dfrac{f_{stop}}{f_{pass}}\right)} \tag{9.4}$$

15.2.1 Chebyshev Filter Shape Factor Given A_{pass}, A_{stop} and N

$$\frac{f_{stop}}{f_{pass}} = \cosh\left[\frac{1}{N}\cosh^{-1}\left(\sqrt{\frac{10^{A_{stop}/10}-1}{10^{A_{pass}/10}-1}}\right)\right] \tag{9.5}$$

15.2.2 Chebyshev Passband Ripple Given f_{stop}, A_{stop}, and N

$$A_{pass} = 10\log_{10}\left\{1+\left[\frac{\sqrt{10^{A_{stop}/10}-1}}{\cosh\left[N\cosh^{-1}\left(\frac{f_{stop}}{f_{pass}}\right)\right]}\right]^2\right\} \tag{9.6}$$

15.2.3 Chebyshev Stopband Attenuation Given A_{pass}, f_{stop}, and N

$$A_{stop} = 10\log_{10}\left\{1+\left[\sqrt{10^{A_{pass}/10}-1}\;\cosh\left[N\cosh^{-1}\left(\frac{f_{stop}}{f_{pass}}\right)\right]\right]^2\right\} \tag{9.7}$$

16 Appendix III: Detailed Examples

16.1 Odd-Order Elliptical Lowpass Filters

Odd-ordered elliptical lowpass filters are reasonably simple to design because they are symmetric and naturally lead to equal termination impedances. The design examples will consider a 7th-order filter design case where the passband ripple bandwidth (ω_p) is 10 kHz, and θ = 50^{o} corresponding to a stopband frequency of $\omega_s = \omega_p / \sin(\theta) = 1.30541$ kHz. A maximum reflection coefficient magnitude of 20% will be assumed which is equivalent to a passband ripple of 0.177288 dB.

16.1.1 N=7 Elliptic Lowpass with Equal Terminations

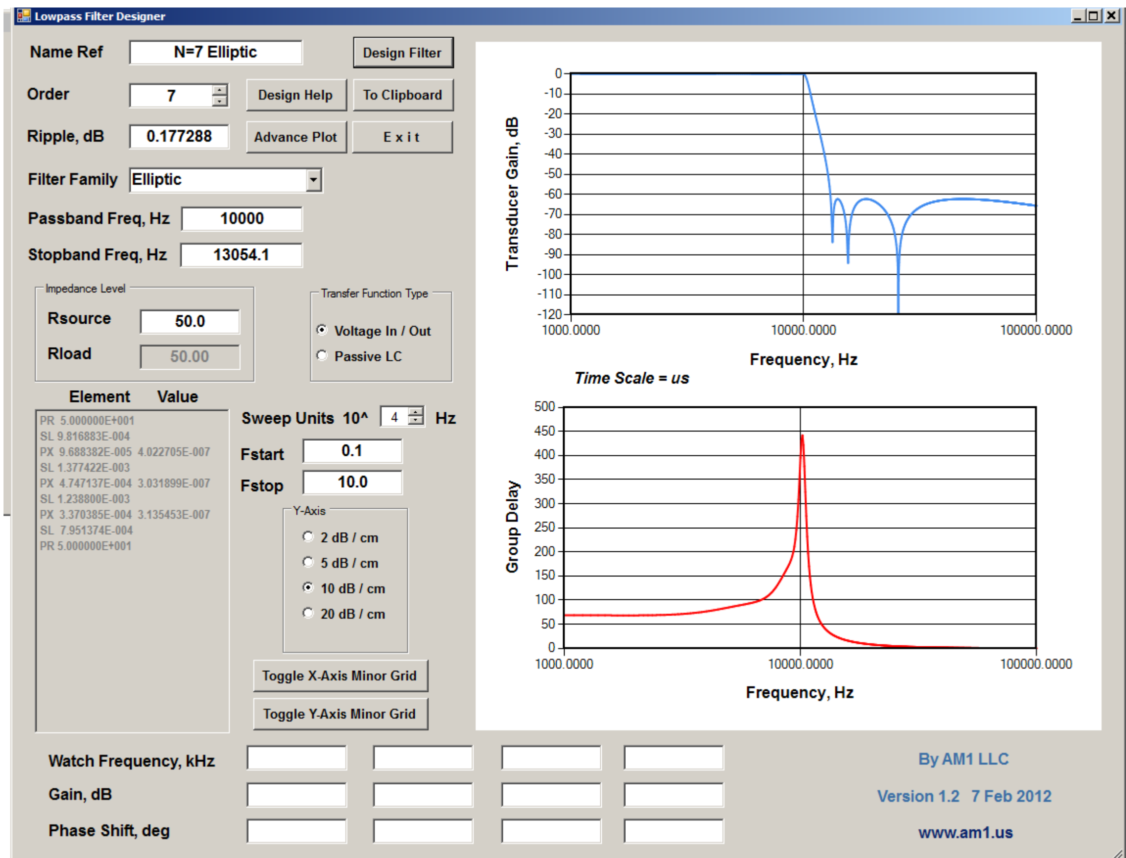


Figure 103 Design parameters for N=7 elliptic lowpass filter with 20% reflection coefficient

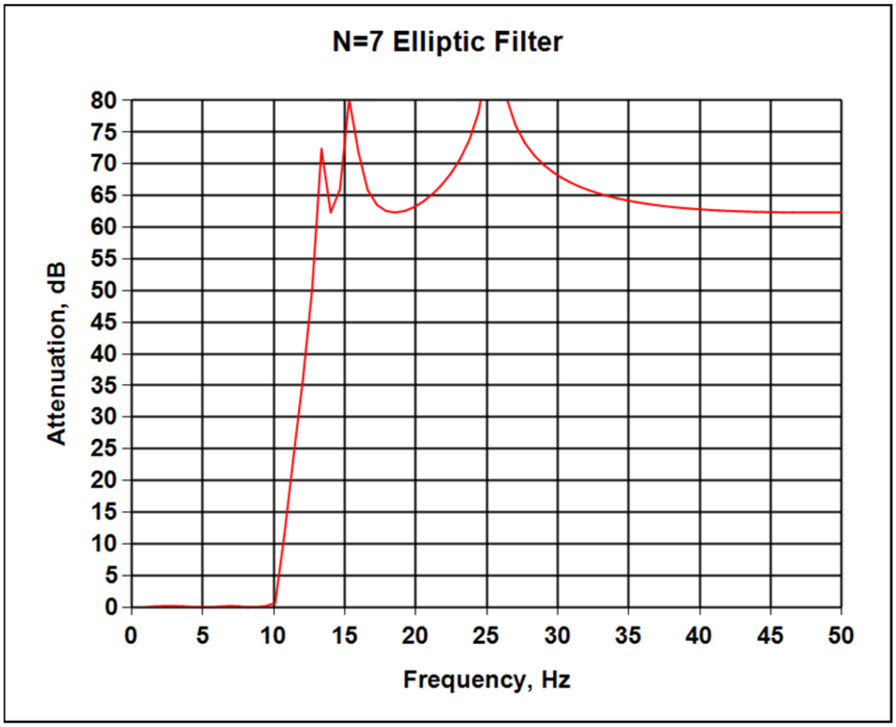


Figure 104 Attenuation sweep of equally-terminated elliptic filter example from Figure 103

Filter Order = 7
Passband, Hz = 10000
Ripple, dB = 0.177288
Stopband, Hz = 13054.1
Passive filter implementation
Rsource = 50
Rload = 50

```
PR  5.000000E+001       (source resistance)
SL 9.816883E-004        (series inductor)
PX  9.688382E-005  4.022705E-007      (shunt series LC section)
SL 1.377422E-003        (series inductor)
PX  4.747137E-004  3.031899E-007      (shunt series LC section)
SL 1.238800E-003        (series inductor)
PX  3.370385E-004  3.135453E-007      (shunt series LC section)
SL  7.951374E-004       (series inductor)
PR 5.000000E+001        (load resistance)
```

16.2 Even-Order Elliptical Lowpass Filters

Even-order elliptical lowpass filters are not immediately realizable in a passive LC-form because they require at least one negative inductor or capacitor. Comments to this effect were made earlier in §10.3. Examples of two different elliptic filter types are given in the following sections.

16.3 N=8 Elliptic Lowpass Type-B

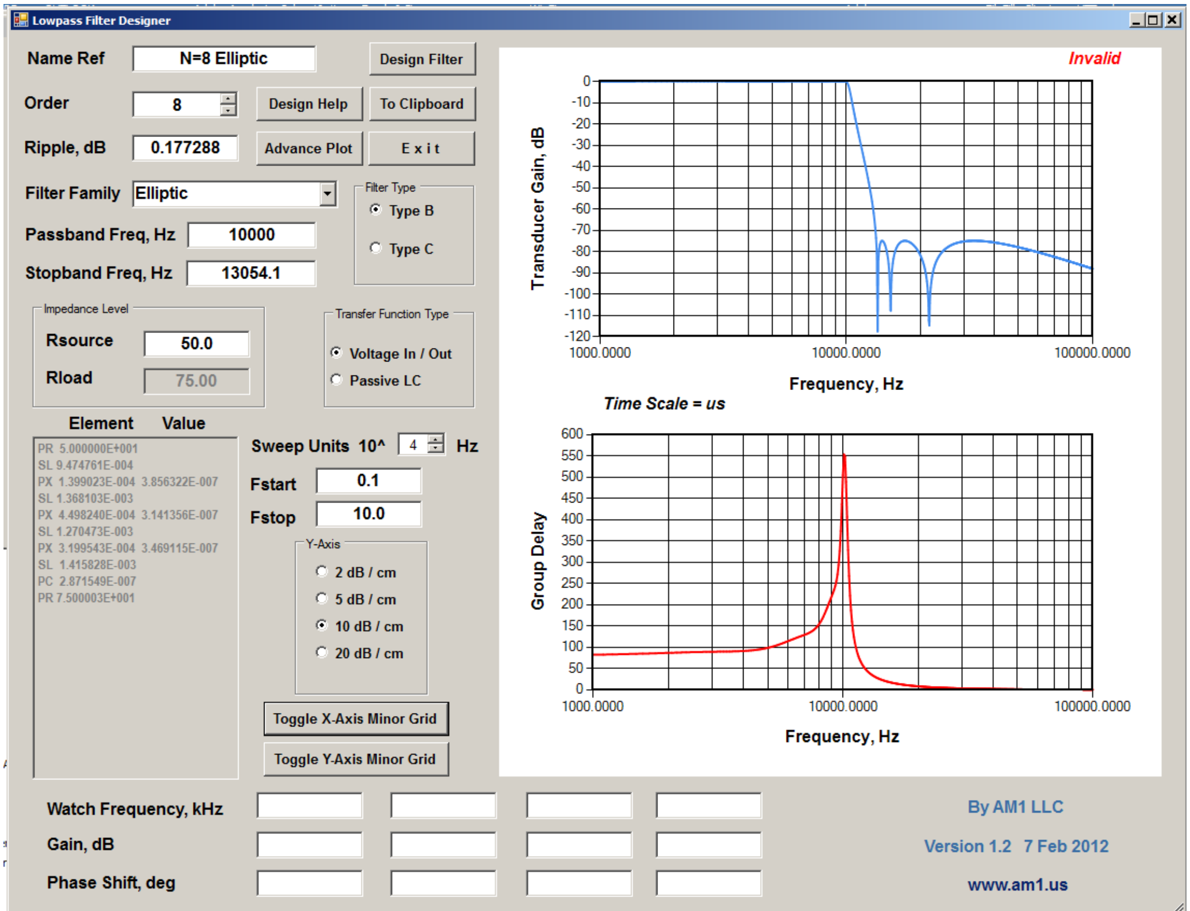


Figure 105 Design parameters for N=8 elliptic lowpass filter with 20% reflection coefficient, type-b filter

Passband, Hz = 10000
Ripple, dB = 0.177288
Stopband, Hz = 13054.1
Voltage In / Out transfer function
Rsource = 50
Rload = 75.00

PR 5.000000E+001
SL 9.474761E-004
PX 1.399023E-004 3.856322E-007
SL 1.368103E-003
PX 4.498240E-004 3.141356E-007
SL 1.270473E-003
PX 3.199543E-004 3.469115E-007
SL 1.415828E-003
PC 2.871549E-007
PR 7.500003E+001

16.4 N=8 Elliptic Lowpass Type-C

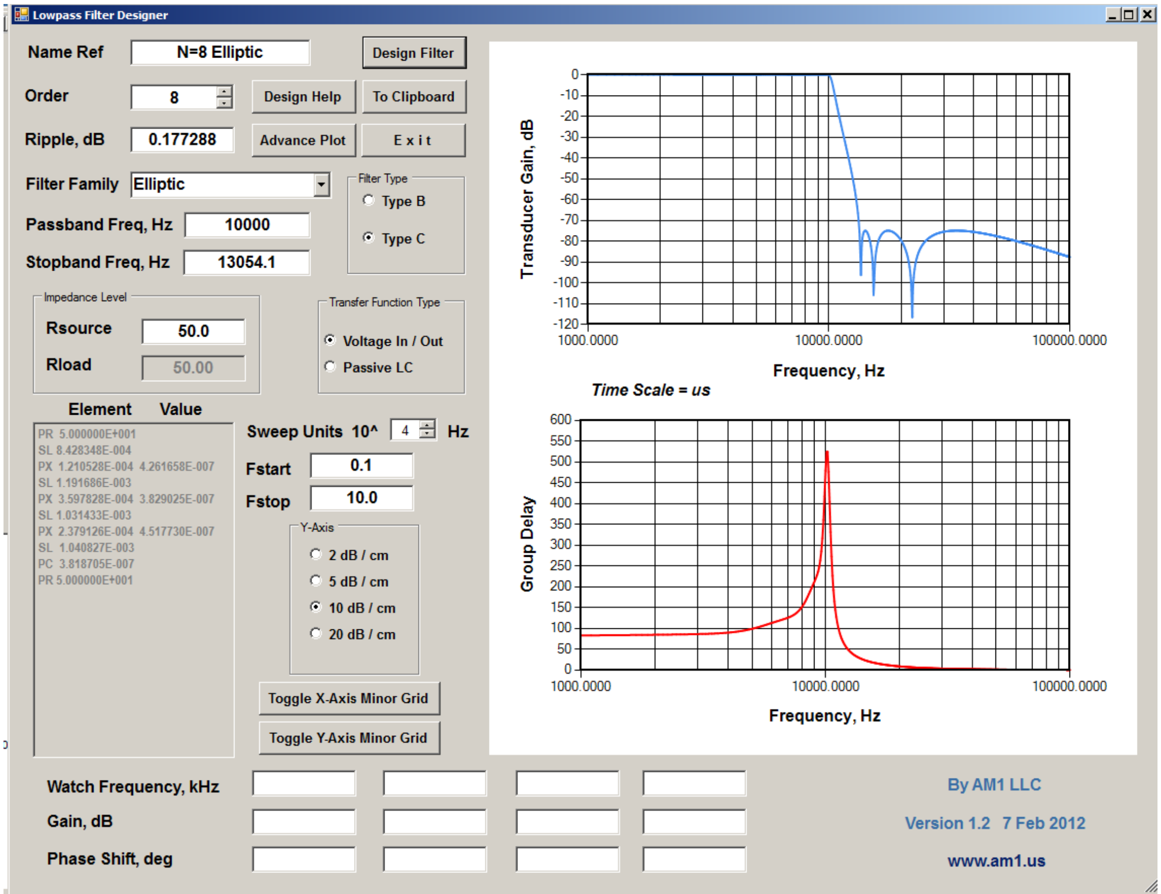


Figure 106 Design parameters for N=8 elliptic lowpass filter with 20% reflection coefficient, type-c filter

Passband, Hz = 10000
Ripple, dB = 0.177288
Stopband, Hz = 13054.1
Voltage In / Out transfer function
Rsource = 50
Rload = 50

PR 5.000000E+001
SL 8.428348E-004
PX 1.210528E-004 4.261658E-007
SL 1.191686E-003
PX 3.597828E-004 3.829025E-007
SL 1.031433E-003
PX 2.379126E-004 4.517730E-007
SL 1.040827E-003
PC 3.818705E-007
PR 5.000000E+001

17 Appendix IV: Amstutz Elliptic Filter Design Programs

Amstutz [11] wrote a now-classic paper about elliptic filter design using small computers in 1978. The paper contains a wealth of knowledge for anyone who wants to understand the inner-workings of elliptic filter design. Aside from originally being written in Fortran with many go-to statements and limited-length variable names, the code contains very few comments and a number of very clever computational *tricks* which make the code very tight and efficient. These same attributes make the code fairly complicated to unravel back to more meaningful high-level equations, however. This appendix exposes many of these details for the antimetric filter design program case. A copy of Amstutz's original paper is assumed available and many references are made to its content herein. Amstutz uses *i* to represent the square-root of –1 in his paper whereas $j = \sqrt{-1}$ is used in the discussions which follow.

The original Fortran code in [11] lacks good clarity due to the very small font used. The translation of this code into *Pet Basic* done by Cuthbert in [3] is far more legible and is consequently adopted here for the discussions which follow.

```
20 PRINT"ANTIMET ELLIP FLTR,CT12/78,1008"
1030 DIM B(16),C(16),D(16),E(30),F(16),R(15),S(15),DB(16),TB(16)
1040 DN=LOG(10)/10:PI=3.1415926
1050 PRINT
1060 PRINT"REJECTION,RIPPLE(DB),1/2-DEG(2-15),TYPE(A,B,OR C):"
1070 INPUT AS,AP,M,T$
1080 IF AS<=AP THEN STOP
1090 N=2*M
2010 ES=EXP(DN*AS)-1
2020 EP=EXP(DN*AP)-1
2030 V=SQR(ES/EP)+SQR(ES/EP-1)
2040 U=PI*PI/(2*LOG(V+V))
2050 V=V/(SQR(ES)+SQR(ES+1))
2060 W=U*LOG(V+SQR(V*V+1))/PI
2070 W=SIN(W)/COS(W):AO=W:W=W*W
2080 Y=EXP(-U):Z=Y:K=M-1
2090 FOR J=1TON
2100 E(J)=1:NEXT J
2110 FORJ=1TO1024
2120 IF K<>M GOTO2150
2130 X=((1-Z)/(1+Z))^2
2140 AO=AO*(W+X)/(1+W*X)
2150 E(K)=E(K)*(1-Z)/(1+Z)
2160 Z=Z*Y:IF Z<.25E-18 GOTO2180
2165 K=K-1
2170 IFK=0 THEN K=N
2175 NEXTJ
2180 E(M)=0:E(N)=E(N)*E(N)
2190 PRINT"U=";U;"AO=";AO;"EP=";E(N)
2200 FOR J=1TOM-1
2210 E(J)=-E(J)*E(N-J)
2220 PRINT"E=";-E(J)
2230 E(N-J)=-E(J):NEXT J
2250 X=SQR(AO*AO+1/(AO*AO)+E(N)*E(N)+1/(E(N)*E(N)))
2260 FOR J=1TOM-1STEP2:K=(J+1)/2
2270 Y=AO*E(J):Y=Y+1/Y
2280 Z=E(N)*E(J)
2290 R(K)=E(M-J)*(1/Z-Z)/Y:S(K)=-X/Y
2300 PRINT"RE=";R(K);"SE=";S(K)
2310 R(M-K+1)=R(K)
2320 S(M-K+1)=-S(K):NEXT J
2330 IF K+K=M GOTO3010
2340 R(K+1)=-AO:S(K+1)=0
2350 PRINT"RE=";-AO
3010 IT=2:IF T$="A" THEN IT=1
3020 E8=-E(1):IF T$="A" THEN E8=E(N)
3030 EO=E(N):IF T$="C" THENEO=-E(1)
3040 FP=SQR((E(N)+EO)/(1+E(N)*E8))
3050 FS=SQR((1+E(N)*EO)/(E(N)+E8))
3060 D(1)=0
3065 FOR J=ITTOM
3070 D(J)=(E(2*J-1)+E8)/(1+E(2*J-1)*EO)
3080 F(J)=SQR(1/D(J)):NEXT J
3100 SR=0:TQ=0:TO=0:B(1)=0:I=1
3110 FOR J=1TOM
3120 W=(AO^2+E(2*J-1)^2)/(1+(AO*E(2*J-1))^2)
3130 X=(1+EO*E8)*S(J)+EO+E8*W
3140 Y=EO^2+2*EO*S(J)+W
3150 Z=1+2*E8*S(J)+E8^2*W
3160 U=SQR(Y/Z):V=X/Z
3170 R(J)=SQR((U-V)/2):S(J)=SQR((U+V)/2)
3180 PRINT"RF=";-R(J)/FP;"SF=";S(J)/FP
3200 SR=SR+R(J)/U
3210 I=-I:W=I*R(J)/S(J)
3220 TQ=(TQ+W)/(1-TQ*W)
3230 IF T$<>"A" GOTO3270
3240 U=(F(2)-S(J))/R(J):V=(F(2)+S(J))/R(J)
3250 W=I*(V-U)/(1+U*V)
3260 TO=(TO+W)/(1-TO*W)
3270 B(1)=B(1)+R(J):NEXT J
4010 IF T$="A" THEN TO=TO/(1+SQR(1+TO*TO))
4020 FOR K=ITTOM
4030 DB(K)=0:TB(K)=TO:I=1
4040 FOR J=1TOM
4050 DB(K)=DB(K)+1/(R(J)+(F(K)-S(J))^2/R(J))+1/(R(J)+(F(K)+S(J))^2/R(J))
4070 I=-I:W=(F(K)-I*S(J))/R(J)
4080 TB(K)=(TB(K)+W)/(1-TB(K)*W):NEXT J:NEXT K
5010 D(M+1)=D(M):F(M+1)=F(M):DB(M+1)=DB(M):TB(M+1)=TB(M):C(1)=0
5020 FOR J=1TOM+1-IT STEP2
5030 TB(M+1-J)=-1/TB(M+1-J):NEXT J
5040 FOR J=ITTOM+1
5050 B(J)=(1+TB(J)^2)*DB(J)/(4*D(J))-TB(J)*F(J)/2
5060 C(J)=TB(J)/F(J):NEXT J
6010 FOR L=1TO2
6020 FOR K=L+2TOM+1 STEP2
6030 FOR J=LTOK-2STEP2
6040 U=C(J)-C(K)
6050 V=1/(U/(B(J)*(D(K)-D(J)))-1)
6060 B(K)=(B(K)-B(J))*V*V-B(J)*(V+V+1)
6070 C(K)=U*V:NEXTJ:NEXTK:NEXTL
7010 W=1:IFT$<>"C" THEN W=((1-TQ*TO)/(TQ+TO))^2
7020 FORJ=1TOM+1 STEP2:B(J)=B(J)*W
7030 C(J)=C(J)*W:NEXT J
7040 PRINT"LD RESIS=";W;"  (";1/W;"  )"
7050 PRINT"         L(C)        C(L)        PEAK"
7070 IF T$<>"A" THENPRINT" 1                     ";FP/B(1)
7085 V=0
7090 FOR J=ITTOM:V=V+C(J)
7100 PRINTJ;FP*C(J);FP/B(J);F(J)/FP:NEXTJ
7110 PRINTM+1;FP*C(M+1);"STPBD EDGE=";FS/FP
7130 PRINT"TESTS";B(M)/B(M+1)-1;(W+1)*SR-V-C(M+1)
7140 STOP
7150 END
```

Figure 107 Antimetric (even-ordered) elliptic filter design program from [3] translated from [11]

```
10 PRINT"SYMMETRICAL ELLIPTIC FLTR,C&S12/78,1009"
1010 DIM B(16),C(16),D(16),E(15),F(30)
1020 DN=LOG(10)/10:PI=3.1415926
2010 PRINT"STBND EDGE (KHZ)=";:INPUT FS
2020 PRINT"PSBND EDGE (KHZ)=";:INPUT FP
2030 IF ABS(FS-FP)<=0 GOTO2010
2040 PRINT"NUMBER OF PEAKS(1-15)=";:INPUT N
2050 IF N<=0 GOTO2010
2060 M=2*N+1
2080 FC=SQR(FS*FP)
2090 R=FC+FC
2100 FOR K=1TO2
2110 S=FS+FP
2120 FOR J=1TO6
2130 P=SQR(S*R)
2140 S=(S+R)/2
2150 IF1E8*(S-P)<S GOTO2170
2160 R=P:NEXT J
2170 IF K>=2 GOTO2200
2180 Q=M/S
2190 R=ABS(FS-FP):NEXT K
2200 Q=Q*S
2210 S=EXP(-PI/Q)
2220 Y=S
2230 PRINT"CRITICAL Q=";Q/(4*(1-S)*S^N)
2250 PRINT"STBND REJECTION (DB)=";:INPUT S
2260 IF S<=0 GOTO2010
2270 S=EXP(S*DN/2)
2280 R=EXP(PI*Q)
2290 P=(LOG(1+(S*S-1)/(R/4+1/R)^2))/DN
2300 PRINT"PSBND RIPPLE (DB)=";P
2310 R=R/(2*(S+SQR(S*S-1)))
2320 R=LOG(R+SQR(R*R+1))/(2*Q)
2330 R=SIN(R)/COS(R)
2340 W=R
2350 PRINT"3 DB (KHZ) ABOUT =";FP+(FS-FP)/(1+FC/(FP*R*R))
2360 PRINT"NOMINAL OHMS RESISTANCE=";:INPUTR
2370 IF R<=0 GOTO2040
2390 Z=Y:E(N)=W:W=W*W
2400 FOR J=1TOM-1
2410 F(J)=1:NEXT J
2420 K=1
2430 FOR J=1TO1024
2440 F(K)=F(K)*(1-Z)/(1+Z)
2450 IF K<M-1 GOTO2500
2460 Z=Z*Y
2470 X=((1-Z)/(1+Z))^2
2480 E(N)=E(N)*(W+X)/(1+W*X)
2490 K=0
2500 Z=Z*Y
2510 IF Z<.25E-18 GOTO2530
2520 K=K+1:NEXT J
2530 FOR J=1TON
2540 F(J)=F(J)*F(M-J)
2550 F(M-J)=F(J):NEXT J
3010 FOR J=1TON
3020 D(J)=F(2*J)*(1-F(J)^4)/F(J)
3030 B(J)=E(N)*F(J):NEXT J
3040 C(1)=1/B(N)
3050 FOR J=1TON-1
3060 C(J+1)=(C(J)-B(N-J))/(1+C(J)*B(N-J))
3070 E(N-J)=E(N+1-J)+E(N)*D(J)/(1+B(J)*B(J)):NEXT J
4010 FOR J=1TON
4020 B(J)=((1+C(J)*C(J))*E(J)/D(J)-C(J)/F(J))/2
4030 C(J)=C(J)*F(J)
4040 D(J)=F(J)*F(J):NEXT J
4050 B(N+1)=B(N):C(N+1)=C(N):D(N+1)=D(N)
5010 IF N=1 GOTO6020
5020 L=1
5030 FOR K=L+2TON+1 STEP2
5040 FOR J=LTOK-2 STEP2
5050 Y=C(J)-C(K)
5060 Z=1/(Y/(B(J)*(D(K)-D(J)))-1)
5070 B(K)=(B(K)-B(J))*Z*Z-B(J)*(1+Z+Z)
5080 C(K)=Y*Z:NEXTJ
5081 NEXT K
5082 IFL=2GOTO6010
5083 L=2:GOTO5030
6010 S=B(N)/B(N+1)-1
6020 Q=.0005/(PI*FC)
6030 P=Q*R:Q=Q/R
6040 IF FS<FP GOTO6150
6060 PRINT"      ** LOW-PASS FILTER **"
6070 FOR J=1TON
6080 C(J)=Q*C(J)
6090 D(J)=Q*B(J)*D(J)
6100 B(J)=P/B(J)
6110 F(J)=FC/F(J):NEXT J
6120 C(N+1)=Q*C(N+1)
6130 GOTO6230
6150 PRINT"      ** HIGH-PASS FILTER **"
6160 FOR J=1TON
6170 C(J)=Q/C(J)
6180 D(J)=Q/(B(J)*D(J))
6190 B(J)=P*B(J)
6200 F(J)=FC*F(J):NEXT J
6210 C(N+1)=Q/C(N+1)
6230 PRINT"   KHZ            FARAD            HENRY"
6240 FOR J=1TONSTEP2
6250 PRINTTAB(12);C(J)
6260 PRINTF(J);J;D(J);B(J):NEXT J
6270 PRINTTAB(12);C(N+1)
6280 IF N=1 THEN STOP
6290 L=(INT((N+1)/2))*2:K=M-1-L
6300 FOR J=L+2TOM-1STEP2
6310 PRINTF(K);K;D(K);B(K)
6320 PRINTTAB(12);C(K)
6330 K=K-2:NEXT J
6340 PRINT"PRECISION TEST:";S
7020 STOP
```

Figure 108 Symmetric (odd-ordered) elliptic filter design program from [3] translated from [11]

17.1 Antimetric Program Details

Most of the program steps are carried out assuming a passband frequency edge of ω_p and stopband frequency edge ω_s such that

$$\omega_s = \frac{1}{\omega_p} \tag{9.8}$$

Once the pertinent results have been computed, they are finally output based upon a passband edge of 1 rad/sec.

The input parameters for the program are (i) passband ripple A_p (dB), (ii) minimum stopband attenuation A_s (dB), (iii) filter-order (N) divided by 2, and (iv) filter type *a, b,* or *c*. The filter order must be an even integer for the antimetric case.

17.1.1 Program Variables

Variable Name	Definition / Meaning
ω_s	Stopband radian frequency, initially such that (9.8) applies
ω_p	Passband radian frequency edge, initially such that (9.8) applies
A_s	Minimum stopband attenuation, dB
A_p	Maximum passband ripple, dB
M	Filter order (N) divided by 2. Also equal to the number or resonator-sections in the filter
N	Filter order, must be even, $N = 2M$
E_p	$E_p = 10^{(A_p/10)} - 1$ Passband ripple
E_s	$E_s = 10^{(A_s/10)} - 1$ Pertains to the stopband level
w	z-plane solution in the *g*( )-plane per (9.13)
a_0	Mapping of the *z*-plane solution to the *s*-plane domain per (9.42)
E_r	Represent different quantities in the program. Initially, the frequency-domain solutions for an *N* / 2 filter, later transformed to the natural frequencies for an N^{th} order filter, and finally transformed for an N^{th}-order type *a, b,* or *c* elliptic filter.
u	Real part of Amstutz's elliptic function period, related to the complete elliptic integral *K* through (9.49), and closely estimated by (9.11)

Program lines 20 through 2020 take care of the input parameters to the program. The first real computation takes place in the next two lines where

$$v = \sqrt{\frac{E_s}{E_p}} + \sqrt{\frac{E_s}{E_p} - 1} \tag{9.9}$$

$$u = \frac{\pi^2}{2\log_e(2v)} = \frac{\pi^2}{\log_e(4v^2)} \tag{9.10}$$

Parameter *u* is one of the more important parameters in that 2*u* is the real-period of Amstutz's *elliptic sine* function *Sn*( ) as discussed shortly. Comparing this to the theory developed earlier in §10.7.3, the real-period of the Jacobi elliptic sine function *sn*( ) is 4*K* where *K* is the associated complete elliptic integral. In Amstutz (4.32), he defines

$$u = \frac{\pi^2}{\log_e\left\{16\left[\dfrac{\exp(2a_s)-1}{\exp(2a_p)-1}\right]\right\}} = \frac{\pi^2}{\log_e\left\{16\left[\dfrac{10^{A_s/10}-1}{10^{A_p/10}-1}\right]\right\}} = \frac{\pi^2}{\log_e\left(16\dfrac{E_s}{E_p}\right)} \tag{9.11}$$

which is consistent with Amstutz (4.30) and (4.31) but not exactly equivalent to (9.10). The difference compared to (9.10) is completely negligible for all practical cases, however. The parameter *u* versus stopband attenuation A_{stop} is plotted in Figure 109 assuming a passband ripple of 0.1 dB. Looking ahead to the discussion involving (9.43), Amstutz apparently realized that the slight modification in (9.9) compared to (9.11) was a simple but effective improvement in the approximation and this improvement is included in his program although not mentioned in his paper.

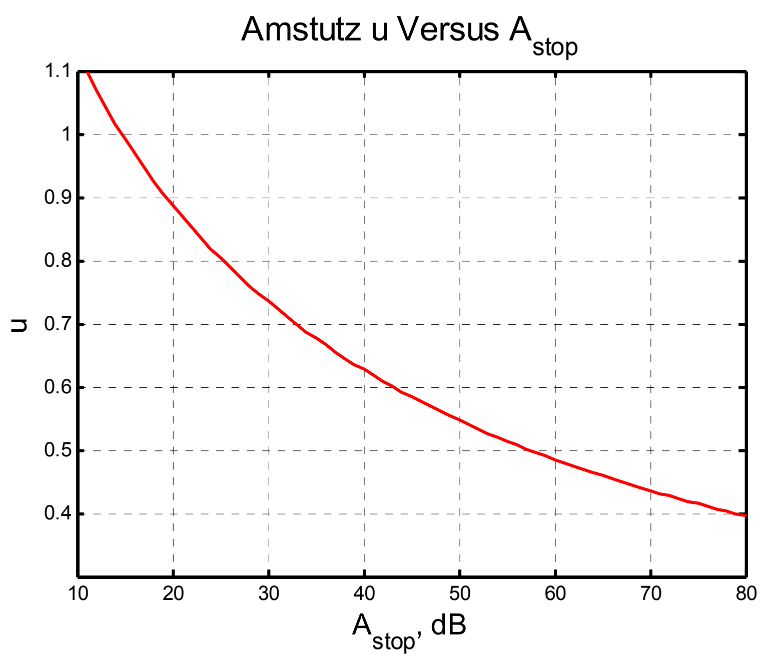


Figure 109 Astutz's *u* parameter versus stopband attenuation assuming A_p = 0.1 dB

Following the calculation in (9.10), the program calculates a new value for *v* in line 2050 as

$$v = \frac{v}{\sqrt{E_s}+\sqrt{E_s+1}} = \frac{\sqrt{\dfrac{E_s}{E_p}}+\sqrt{\dfrac{E_s}{E_p}-1}}{\sqrt{E_s}+\sqrt{E_s+1}} \tag{9.12}$$

Here again, Amstutz apparently uses this form for *v* to calculate *w* in line 2060 of his program[78] with improved accuracy as

$$w = \frac{v}{\pi}\log_e\left(v+\sqrt{v^2+1}\right) \tag{9.13}$$

[78] because $E_s >> E_p$ and the formula in his paper differ.

which can be also be rewritten as

$$w = \frac{v}{\pi} \sinh^{-1}\left(v\right) \tag{9.14}$$

The exact calculation for *w* is taken up in §17.4. In order to see the underlying details more clearly, some time must first be spent with the Amstutz elliptic sine function.

17.2 Amstutz Elliptic Sine Function Sn( )

Amstutz cleverly devised *his own* elliptic function invention which is admittedly more convenient and computationally efficient than using the Jacobi elliptic functions, but this does complicate matters when this theory must be compared with the more traditional literature. As noted in §10.7.3, the Jacobi elliptic sine function *sn*(*z, k*) has a real-period of 4*K* and an imaginary period of 2*K′* where *K* and *K′* are the complete elliptic integral and complimentary complete elliptic integral respectively. The elliptic sine function used by Amstutz also has a real and imaginary period, but they are somewhat more convenient in that the real period is 2*u* and the imaginary period is j π. The Amstutz elliptic sine function is given as[79]

$$Sn\left(u,z\right) = \tanh\left(z\right)\prod_{r=1}^{\infty}\left[\tanh\left(ru-z\right)\tanh\left(ru+z\right)\right] \tag{9.15}$$

The construction of this function is worth looking at more closely. Note that the zeros for this function occur for

$$\begin{aligned} \tanh\left(ru-z\right) &= 0 \\ \tanh\left(ru+z\right) &= 0 \end{aligned} \tag{9.16}$$

or in other words,

$$z_{zero} = ru + jn\pi \quad \text{for all integers } r,n \tag{9.17}$$

The poles occur for

$$\begin{aligned} \cosh\left(ru+z\right) &= 0 \\ \cosh\left(ru-z\right) &= 0 \end{aligned} \tag{9.18}$$

Taking the top equation of the two,

$$\cosh\left(ru+z\right) = \frac{\exp\left(ru+z\right)+\exp\left(-ru-z\right)}{2} = 0 \tag{9.19}$$

from which follows

$$-1 = \exp\left[\pm j\left(2n-1\right)\pi\right] = \exp\left[-2\left(ru+z\right)\right] \tag{9.20}$$

Taking natural logs of both sides and collecting terms reveals the poles given by

$$z_{pole} = -ru \pm j\left(\frac{2n-1}{2}\right)\pi \tag{9.21}$$

for arbitrary integers *r* and *n*. The poles and zeros periodicities are consequently as stated earlier.

79 Amstutz equation (4.1).

A second condition on (9.15) for it to be an acceptable *elliptic function* is for it to have the correct value when *z* = *u*/2 + j π / 4. This is a special value of *z* in that

$$Sn\left(u, \frac{u}{2} \pm j\frac{\pi}{4}\right) = \pm 1 \tag{9.22}$$

To see this more clearly, it is best to view (9.15) in terms of magnitude and phase. Note that

$$\begin{aligned}\left|\tanh\left(v + j\frac{\pi}{4}\right)\right|^2 &= \left|\frac{\sinh(v)\cos\left(\frac{\pi}{4}\right) + j\cosh(v)\sin\left(\frac{\pi}{4}\right)}{\cosh(v)\cos\left(\frac{\pi}{4}\right) + j\sinh(v)\sin\left(\frac{\pi}{4}\right)}\right|^2 \\ &= \frac{\sinh^2(v) + \cosh^2(v)}{\cosh^2(v) + \sinh^2(v)} \equiv 1\end{aligned} \tag{9.23}$$

for any real value of *v*. Consequently, the magnitude of every *tanh*( ) term in (9.15) is unity for this special value of *z*.

The angular argument for each *tanh*( ) term for this special value of *z* is more complicated. To begin with, note that

$$\tanh(a + jb) = \frac{e^a e^{jb} - e^{-a}e^{-jb}}{e^a e^{jb} - e^{-a}e^{-jb}} = \frac{1 - \exp(-2a - j2b)}{1 + \exp(-2a - j2b)} \tag{9.24}$$

In the special case where *b* = π / 4, (9.24) becomes

$$\tanh\left(a + j\frac{\pi}{4}\right) = \frac{1 + j\exp(-2a)}{1 - j\exp(-2a)} \tag{9.25}$$

and the phase argument for this quantity is given by

$$\measuredangle \tanh\left(a + j\frac{\pi}{4}\right) = 2\tan^{-1}\left[\exp(-2a)\right] \tag{9.26}$$

This result can be used to compute the phase argument for the product terms in (9.15) as follows. For a specific value of *r* and the special value case of *z*

$$\begin{aligned} ru + z &= \left(r + \frac{1}{2}\right)u + j\frac{\pi}{4} \\ ru - z &= \left(r - \frac{1}{2}\right)u + j\frac{\pi}{4}\end{aligned} \tag{9.27}$$

Combining this result with (9.26) and inserting into (9.15) produces

$$\measuredangle\prod_{r=1}^{\infty}\left[\tanh\left(ru+z\right)\tanh\left(ru-z\right)\right]=2\sum_{r=1}^{\infty}\tan^{-1}\left\{\exp\left[-(2r+1)u\right]\right\}-\dots \\ 2\sum_{r=1}^{\infty}\tan^{-1}\left\{\exp\left[-(2r-1)u\right]\right\} \tag{9.28}$$

At first glance, this results is still fairly complicated, but writing out the first few terms gives

$$\begin{aligned} &=2\begin{Bmatrix} \tan^{-1}\left(e^{-3u}\right)+\tan^{-1}\left(e^{-5u}\right)+\tan^{-1}\left(e^{-7u}\right)+\dots \\ -\tan^{-1}\left(e^{-u}\right)-\tan^{-1}\left(e^{-3u}\right)-\tan^{-1}\left(e^{-5u}\right)-\dots \end{Bmatrix} \\ &=-2\tan^{-1}\left(e^{-u}\right) \end{aligned} \tag{9.29}$$

This is precisely the negative of (9.26) when $a=u/2$ thereby proving the zero-phase assertion given by (9.22) for this special value for *z*. Further as given by Amstutz (4.2), *Sn*( ) mirrors other characteristics of the Jacobi elliptic sine function *sn*( ) as

$$\begin{aligned} Sn\left(u,z+u\right)&=-Sn(u,z) \\ Sn\left(u,z+j\frac{\pi}{2}\right)&=\frac{1}{Sn\left(u,z\right)} \end{aligned} \tag{9.30}$$

$$\begin{aligned} Sn\left(u,\frac{u}{2}\pm j\frac{\pi}{4}\right)&=\pm 1 \\ Sn\left(u,\pm j\frac{\pi}{4}\right)&=\pm j \end{aligned} \tag{9.31}$$

As pointed out here and elaborated in [12], there is a direct relationship between the Jacobi elliptic sine function *sn*(*z, k*) and the Amstutz elliptic sine function *Sn*(*u, z*). The filter's natural frequencies which are given by Amstutz (4.19) are given by

$$p_n=Sn\left(Nu,\frac{n}{N}\frac{Nu}{2}\right)=\sqrt{k}\,sn\left(\frac{nK}{N},k\right)\quad\text{for } n=1,\dots,N \tag{9.32}$$

17.3 Amstutz Transducer Gain Function and Exact Value for u

Throughout the Amstutz paper, the frequency variable ω is normalized so that the passband frequency edge ω_p and stopband frequency edge ω_s are related as $\omega_p\ \omega_s = 1$. He writes the attenuation characteristic as

$$\exp\left[2a\left(\omega\right)\right]=1+\frac{1}{\sigma^2}g^2\left(\omega\right) \tag{9.33}$$

which precisely parallels the Feldtkeller equation given earlier by (2.15). In the elliptic filter case,

$$\begin{aligned} g(\omega) &= Sn(u,z) = \sin(z) \\ \omega &= Sn(mu,z) = \sin\left(\frac{z}{m}\right) \end{aligned} \tag{9.34}$$

where *m* is the order of the elliptic filter being considered. The first equation is the mapping between the *g*-plane and the *z*-plane whereas the second corresponds to the mapping between the *s*-plane and the *z*-plane. The maximum passband attenuation a_p (nats) corresponds to $z = u / 2$ thereby leading to

$$\exp(2a_p) = 1 + \frac{1}{\sigma^2} g^2(\omega_p) \tag{9.35}$$

Similarly, the minimum stopband attenuation a_s (nats) occurs for $z = u / 2 + j\,\pi / 2$ such that

$$\exp(2a_s) = 1 + \frac{1}{\sigma^2} g^2(\omega_s) \tag{9.36}$$

From (9.35) and (9.36),

$$\begin{aligned} \frac{\exp(2a_p)-1}{\exp(2a_s)-1} &= \frac{g^2(\omega_p)}{g^2(\omega_s)} = \frac{Sn^2\left(\frac{u}{2}\right)}{Sn^2\left(\frac{u}{2}+j\frac{\pi}{2}\right)} \\ &= Sn^4\left(\frac{u}{2}\right) \end{aligned} \tag{9.37}$$

where the last equality makes use of (9.30). To facilitate using this result, Amstutz (4.13) defines

$$\tau = Sn\left(\frac{u}{2}\right) \tag{9.38}$$

Based upon (9.34) and the passband edge corresponding to $z = u / 2$, Amstutz (4.15) gives

$$\omega_p = \frac{1}{\omega_s} = Sn\left(mu, m\frac{u}{2}\right) \tag{9.39}$$

thereby leading to[80]

$$\frac{\omega_p}{\omega_s} = Sn^2\left(mu, m\frac{u}{2}\right) \tag{9.40}$$

It is worthwhile to point out the symmetries between the square-root of (9.37) which applies to the amplitude domain (*g*) and (9.40) which applies to the frequency domain (ω); the only mapping difference in the *z*-domain is the filter order factor *m*. This scaling factor appears repeatedly between the amplitude and frequency domains for elliptic filters.

In solving for the *natural frequencies* of the filter, Amstutz (4.27) and (4.28) are identified as

$$j\sigma = Sn(u, jw) \tag{9.41}$$

$$ja_0 = Sn(mu, jw) \tag{9.42}$$

[80] The Amstutz equation (4.25) is missing the square.

where (9.42) applies to the functional-mapping of *g*( ) to the z-domain and (9.41) applies to the mapping of the *z*-plane to the *s*-plane domain. Amstutz (4.30) uses an approximation (9.10) to compute the filter shape-factor *u* which is quite accurate whereas [12] goes a step further in giving the exact solution as[81]

$$u = \pi \frac{AGM\left(k_1\right)}{AGM\left(\dfrac{1-k_1}{1+k_1}\right)}\left(1+k_1\right)^{-1} \qquad (9.43)$$

where k_1 is given by (6.6) and *AGM* is the arithmetic-geometric mean first introduced in §10.7.1. It is only when this exact result for *u* is compared to Amstutz's approximation used in his program (9.10) and the approximation cited in his paper (9.11) that a complete vindication of (9.10) is possible as shown in Figure 110.

17.4 Calculation of w

Given *u* by way of (9.10), Amstutz (4.33) computes *w* is his program using (9.12) and (9.13) whereas his paper uses the approximation

$$w = \frac{u}{2\pi}\log_e\left[\frac{\exp\left(a_p\right)-1}{\exp\left(a_p\right)+1}\right] \qquad (9.44)$$

This can also be equivalently written as

$$w = \frac{u}{\pi}\sinh^{-1}\left(\sqrt{\exp\left(a_p\right)-1}\right) \qquad (9.45)$$

[81] The original equation (16) in [12] includes an additional factor of ½ which is in error when (9.43) is compared to Amstutz (4.32).

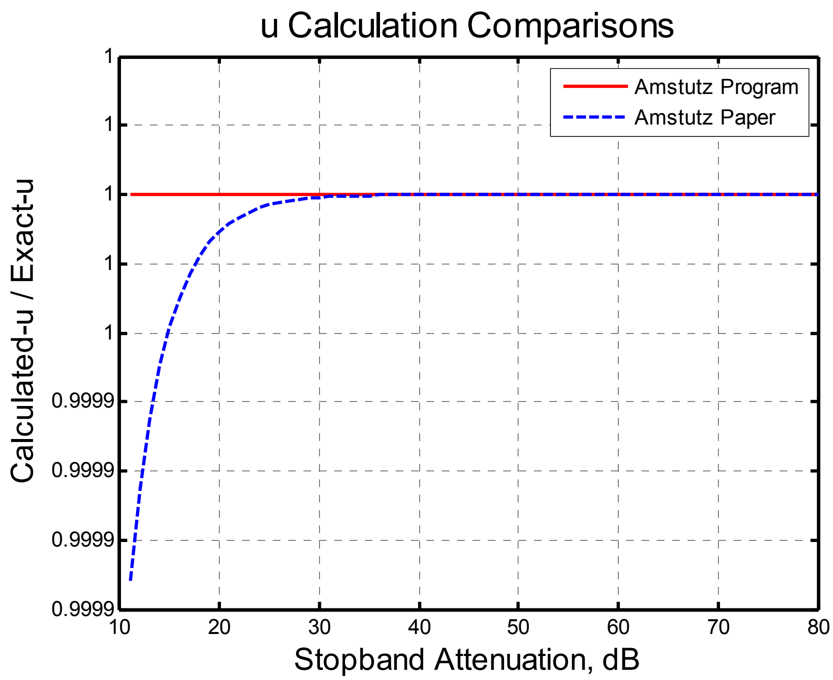


Figure 110 Comparison of u-computation methods[82]. Exact value is given by (9.43), Amstutz program approximation given by (9.10), and Amstutz approximation in the paper given by (9.11). All of the methods give acceptably accurate results.

Reference [12] gives the exact solution for *w* based upon a repeated application of the Landen transformation as follows:

$$Q_0 = \left\{ \left[\exp(a_s) - 1 \right] \left[\exp(a_p) - 1 \right] \right\}^{-1/4}, SN_0 = \sqrt{k_1}$$
$$SN_{n+1} = \frac{SN_n^2}{1 + \sqrt{1 - SN_n^4}} \tag{9.46}$$

$$V_n = \frac{1}{Q_n SN_n}$$
$$Q_{n+1} = \frac{1}{V_n + \sqrt{1 + V_n^2}} \tag{9.47}$$

with

$$w = \frac{u}{\pi} \sinh^{-1}(\alpha)$$
$$= \frac{u}{\pi} \log_e \left(\alpha + \sqrt{\alpha^2 + 1} \right) \tag{9.48}$$

where $\alpha = \lim_{h \to \infty} Q_h / SN_h$. The exact value for *w* and Amstutz approximations given by (9.13) and (9.44) for *w* are compared in Figure 111 showing the excellent behavior of the Amstutz approximation used in his program versus exact.

[82] From u18548_amstutz_equation_checks.m.

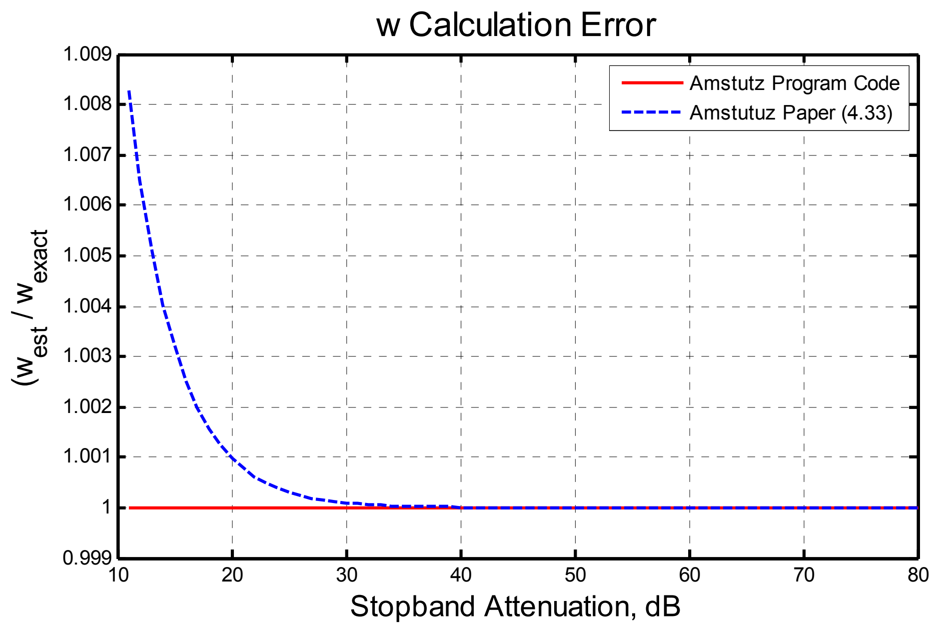


Figure 111 Comparison[83] of estimates for *w* based upon (i) Amstutz program code formula (9.13) and (ii) Amstutz (4.33) repeated here as (9.44). The *exact* value for *w* was computed using (9.46) through (9.48).

From (9.32), there is a direct correlation between the Amstutz *u*-parameter and the classical elliptic sine period given by

$$\frac{u}{2} \Leftrightarrow \frac{K}{N} \tag{9.49}$$

where *K* is the complete elliptic integral associated with modulus *k* and *N* is the filter order. Using this equivalence in (9.44) leads directly to (6.30) aside from a factor of – *j* implying that the approximate relationship used in Amstutz (4.30) is based upon the same reasoning used earlier in (6.28).

The program calculates the filter's natural frequencies in the *z*-domain in lines 2090 – 2230. The correlation between these lines and the Amstutz equations (4.17) and (4.18) is, however elusive for two major reasons. First of all, the Amstutz program makes use of a key statement which appears immediately above Amstutz (4.24A) which reads as follows:

> *It may be interesting to note that a type A characteristic of degree 2m can be deduced in the same way from an elliptic characteristic of degree m by the transformation*

$$f^2 = \frac{\omega - E_{2m}}{1 - E_{2m}\omega} \tag{9.50}$$

In other words, the Amstutz program calculates all of the *z*-plane natural frequencies assuming a *N* / 2 degree filter characteristic, and then translates these E_r values to new E_r values corresponding to a N^{th} order filter using (9.50). The second reason these program lines are difficult to follow in the code stems from the way in which each *tanh*( ) product term is computed in (9.15). In line 2050, the E_k calculation appears to only include the $\tanh(ru+z)$ product term while ignoring the $\tanh(ru-z)$ term in (9.15). This apparent discrepancy is adjusted for by (i) computing the E_r values for *r* = 1,2,…,*N*, and (ii) by exploiting the periodicity of the E_r solutions which comes from the inherent 2*u* periodicity of the *Sn*( ) function in lines 2200 – 2230.

[83] From u18548_amstutz_equation_checks.m.

Program lines 2250 – 2350 translate the z-plane solution given by (9.13) for the natural frequencies into the equivalent s-plane natural frequencies using Amstutz (4.22) and (4.23). These results are adjusted further in program lines 3010 – 3030 depending on the filter type (*a*, *b*, or *c*).

Up until this point in the program, all of the natural frequencies have been calculated for a $m = N / 2$ order filter. Program lines 3010 – 3080 use one of three frequency-transformation formulas (Amstutz (4.24A) through (4.24C)) to simultaneously compute N natural frequencies from the m and adjust these frequencies for a type-*a*, type-*b*, or type-*c* filter. At this point in the program, all of the s-plane natural frequencies have been computed for the N^{th} order filter. The filter passband and stopband frequency edges are computed in lines 3020 – 3050.

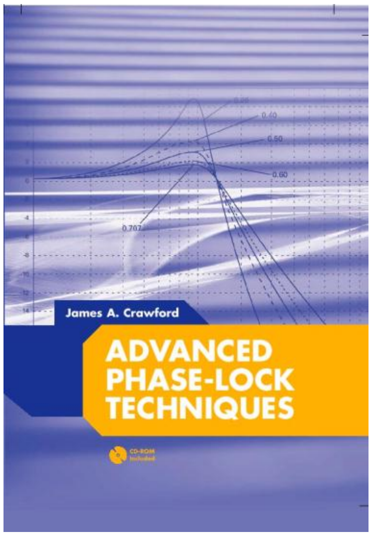
The remaining program computations are still relatively complicated to unravel owing to the extreme tightness of the coding style used. Take for instance, Amstutz (3.4) which gives the input impedance at attenuation pole p_r as

$$\begin{aligned} Z_{in}\left(p_r\right) &= -j\,R_{source}\tan\left\{\sum_n\left[\frac{d_r^n}{2}\right]\right\} \text{ for } \varepsilon = -1 \\ &= -j\,R_{source}\tan\left\{\sum_n\left[\frac{1}{2}\left(\frac{\pi}{2}-\arg\left(p_r - t_n\right)\right)\right]\right\} \end{aligned} \tag{9.51}$$

where the t_n are the transmission zeros from Amstutz (2.6). In program lines 4050 – 4080, this is implemented quite differently by computing a recursive sum of angle arctangents based upon the trigonometric identity

$$\tan\left(\theta_1+\theta_2\right) = \frac{\chi_1+\chi_2}{1-\chi_1\chi_2} \tag{9.52}$$

where $\chi_1 = \tan\left(\theta_1\right)$ and $\chi_2 = \tan\left(\theta_2\right)$. Although this unquestionably leads to better numerical precision and faster computation, it also makes the coding details considerably more difficult to follow with respect to the description given in the paper.



Advanced Phase-Lock Techniques

James A. Crawford

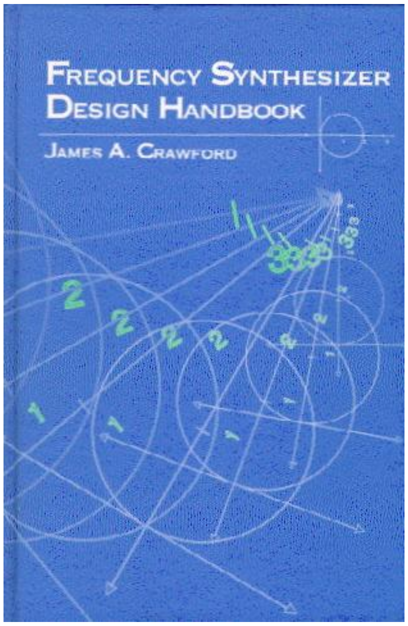
2008

Artech House

510 pages, 480 figures, 1200 equations
CD-ROM with all MATLAB scripts

ISBN-13: 978-1-59693-140-4
ISBN-10: 1-59693-140-X

Chapter	Brief Description	*Pages*
1	*Phase-Locked Systems—A High-Level Perspective* An expansive, multi-disciplined view of the PLL, its history, and its wide application.	26
2	*Design Notes* A compilation of design notes and formulas that are developed in details separately in the text. Includes an exhaustive list of closed-form results for the classic type-2 PLL, many of which have not been published before.	44
3	*Fundamental Limits* A detailed discussion of the many fundamental limits that PLL designers may have to be attentive to or else never achieve their lofty performance objectives, e.g., Paley-Wiener Criterion, Poisson Sum, Time-Bandwidth Product.	38
4	*Noise in PLL-Based Systems* An extensive look at noise, its sources, and its modeling in PLL systems. Includes special attention to 1/*f* noise, and the creation of custom noise sources that exhibit specific power spectral densities.	66
5	*System Performance* A detailed look at phase noise and clock-jitter, and their effects on system performance. Attention given to transmitters, receivers, and specific signaling waveforms like OFDM, M-QAM, M-PSK. Relationships between EVM and image suppression are presented for the first time. The effect of phase noise on channel capacity and channel cutoff rate are also developed.	48
6	*Fundamental Concepts for Continuous-Time Systems* A thorough examination of the classical continuous-time PLL up through 4^{th}-order. The powerful Haggai constant phase-margin architecture is presented along with the type-3 PLL. Pseudo-continuous PLL systems (the most common PLL type in use today) are examined rigorously. Transient response calculation methods, 9 in total, are discussed in detail.	71
7	*Fundamental Concepts for Sampled-Data Control Systems* A thorough discussion of sampling effects in continuous-time systems is developed in terms of the *z*-transform, and closed-form results given through 4^{th}-order.	32
8	*Fractional-N Frequency Synthesizers* A historic look at the fractional-N frequency synthesis method based on the U.S. patent record is first presented, followed by a thorough treatment of the concept based on Δ-Σ methods.	54
9	*Oscillators* An exhaustive look at oscillator fundamentals, configurations, and their use in PLL systems.	62
10	*Clock and Data Recovery* Bit synchronization and clock recovery are developed in rigorous terms and compared to the theoretical performance attainable as dictated by the Cramer-Rao bound.	52



Frequency Synthesizer Design Handbook

James A. Crawford

1994

Artech House

435 pages, 265 figures, 620 equations

ISBN 0-89006-440-7

Chapter	Brief Description	*Pages*
1	*Introduction to Computer-Aided Frequency Synthesizer Design*	2
2	*Building Blocks for Frequency Synthesis Using Phase-Locked Loops*	38
3	*Phase Noise and its Impact on System Performance*	106
4	*Phase-Locked Loop Analysis for Continuous Linear Systems*	55
5	*Frequency Synthesis Using Sampled-Data Control Systems*	43
6	*Fast-Switching Frequency Synthesizer Design Considerations*	40
7	*Hybrid Phase-Locked Loops*	66
8	*MACSET: A Computer Program for the Design and Analysis of Phase-Locked Loop Frequency Synthesizers*	33
9	*Fractional-N Frequency Synthesis*	34