Amal Banerjee

Automated Electronic Filter Design

EXTRAS ONLINE

Springer

Automated Electronic Filter Design

Amal Banerjee

Automated Electronic Filter Design

Springer

Amal Banerjee
Analog Electronics
Kolkata, India

ISBN 978-3-319-43469-8 ISBN 978-3-319-43470-4 (eBook)
DOI 10.1007/978-3-319-43470-4

Library of Congress Control Number: 2016949738

© Springer International Publishing Switzerland 2017
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG Switzerland

This book is dedicated to:

Late Sivadas Banerjee
Meera Banerjee
Anuradha Datta

A dear friend, mentor and guide
Dr. Andreas Gerstlauer

The two amazing physicists who taught me the basics of radio frequency:
Dr. C. Fred Moore
Dr. M.E.L. Oakes

Contents

Chapter 1
Introduction and Problem Statement

With the worldwide proliferation of wireless communication networks, precision analog signal processing is becoming more and more important each day. A crucial component of analog signal processing is signal filtering, thus the need for designing/implementing electronic filters that accurately satisfy design specifications (cutoff frequency, pass/stop band ripple, bandwidth, group delay/phase shift, etc.). This book elaborates on an automated, efficient, and yet very powerful scheme for designing/implementing and evaluating/fine-tuning performance characteristics of electronic filters. As all digital filters are derived from analog filters, this scheme can be extended to the digital filter domain easily.

In a nutshell, this scheme circumvents some key but complicated, manual (thus error-prone and time-consuming) steps of the traditional electronic filter design process. Easily automated (in this case is an ANSI C language program), the output is in the universally used circuit simulator SPICE input format. The filter designer can then easily evaluate and fine-tune the performance characteristics of the new filter design. A brief overview of the traditional electronic filter design process is presented to explain how this proposed scheme achieves its goal.

Briefly, traditional filter design process consists of the following steps, in that order:

1. Basic loop equations are derived from Kirchhoff's current/voltage laws (KCL/KVL). These loop equations are differential equations, sometimes nonlinear. Typically, these differential equations are converted to more tractable algebraic equations, using Laplace transforms, effectively going from the time to frequency domain.
2. In the filter transfer function $H(s)$, the Laplace transform of the unit impulse response of the filter is obtained by evaluating $H(s)$ at $s = jw$ (in general, $s = a + jw$ and $s = jw$ represent a pure sinusoidal input). For calculation purposes, it is the ratio of the output to the input voltage in the frequency domain. The transfer function is almost always in a denominator-numerator polynomial form. The goal is to determine the roots of these two polynomials in the complex

© Springer International Publishing Switzerland 2017

A. Banerjee, *Automated Electronic Filter Design*,
DOI 10.1007/978-3-319-43470-4_1

s-plane to obtain the poles and zeros. Poles and zeros are, respectively, the roots of the denominator and numerator polynomials. Only poles in the left half of the complex *s*-plane guarantee filter stability and need to be complex conjugates of each other to ensure real-valued coefficients in the differential equations representing the filter. Most importantly, for the generic case, the pole (and zero) values are analytical expressions involving the capacitors, inductors, and resistors to be used in the filter. Both the denominator and numerator polynomials need to be factorized to extract the zeros and poles of the transfer function. Sometimes, instead of using the transfer function, expressions for the filter insertion loss or loss magnitude versus frequency are used [1], but that scheme involves first evaluating the loss expression in terms of the filter component values and then using a judicious combination of heuristics, ladder network, and precalculated table values.

3. The most difficult step is to use the values of the poles and zeros evaluated previously and to determine values of each of the resistive and reactive components (capacitors, inductors) in the filter circuit. In the generic case, using the analytical expressions for the values of poles and zeros in combination with the predefined numerical values of design specifications (cutoff frequency, pass/stop band ripple, etc.) is an extremely complicated, manual, and thus error-prone/time-consuming/trial-and-error process. The designer can also utilize the causality and stability conditions and set resistor values to 1 Ω to aid/simplify this calculation. Clearly, this step is feasible for low-order filters only.

To illustrate the issues involved, consider a third-order low pass filter, consisting only of passive components as capacitors (C_1, C_3), inductors (L_2), and resistors (R_L, R_S), Fig. 1.1.

The generalized or generic transfer function for this filter is

$$H(s) = \frac{R_L}{R_S + R_L}\left(\frac{1}{s^3\left(\frac{R_S R_L L_2 C_1 C_3}{R_L + R_S}\right) + s^2 L_2\left(\frac{R_S C_1 + R_L C_3}{R_S + R_L}\right) + s\left(\frac{R_S R_L (C_1 + C_3)}{R_S + R_L}\right) + 1}\right) \tag{1.1}$$

The generalized transfer function (1.1) is a cubic equation in *s*, with real and imaginary roots. Once the analytical expressions for the transfer function roots and zeros are determined, the numerical values of the capacitors, inductors, and resistors have to be calculated, using predefined numerical values for the cutoff frequency, pass/stop band ripple, etc., and conditions of filter stability. *This is another manual*, *time-consuming*, *and very error-prone calculation step*. The design task is

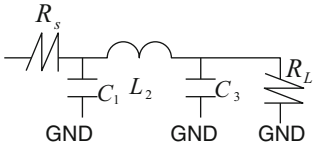

Fig. 1.1 Third-order low pass filter with passive components. "GND" is signal ground

not complete until the filter is simulated with SPICE and it is found that predefined constraints are satisfied accurately. While steps 1 and 2 outlined above can be partially automated with existing mathematical software (Matlab, Mathematica, etc.) tools and computer-aided design (CAD) tools as SPICE and Cadence Spectre, performing step 3 for the generic case is impossible without simplifying assumptions and/or appropriate transformations applied to the raw transfer function $H(s)$ in steps 1 and 2. For example, a fifth-order low pass filter can be implemented with, for example, a first-order filter and two second order filters, connected in series. Thus, any scheme to circumvent these issues must satisfy the following two conditions:

- Eliminate all manual and/or time-consuming/error-prone calculation steps.
- It must be possible to seamlessly verify that the final design satisfies specifications with a proven technique or tool.

The proposed scheme exploits the twin concepts of canonical (*or normalized or prototype*) *filter and "ladder networks" in combination with predefined filter tables* [2]. Ladder networks consist entirely of passive (resistor) and reactive (capacitor, inductor) and avoid the physical constraints (e.g., gain bandwidth product, slew rate, etc.) of active semiconductor-based devices, e.g., operational amplifier. Reactive components have a cutoff frequency (typically in the high MHz range); effective modifications have been developed to tackle this issue. The entire design process can be fully automated and is applicable to the entire frequency range from low (e.g., audio) to microwave (100's of MHz and 10's of GHz). Automation completely removes the manual error-prone step, and when used in combination, the popular circuit simulation tool, SPICE, performance characteristics can be evaluated and fine-tuned quickly. These topics are elaborated on in the subsequent chapters.

References

1. Matthaei, G. L., Young, L., & Jones, E. M. T. (1964). *Microwave filters, impedance-matching networks, and coupling structures*. New York: McGraw-Hill. LCCN 64-7937.
2. Zverev, A. I. Handbook of filter synthesis (Rev. Ed.). ISBN-13: 978-0471749424; ISBN-10: 0471749427.

Chapter 2
Automated Electronic Filter Design Scheme

2.1 The Framework

The proposed automated filter design scheme is explained in detail, here. First, some terminology:

- *Ladder network*. A *ladder network* consists of alternating series and shunt reactive elements, Fig. 2.1a, b. The signal flow is from left to right. The source device is the Thevenin equivalent of the circuit feeding the ladder network, while the load device is the Thevenin equivalent of the circuit that the ladder network is driving. If the source and load impedances have the same value, then the ladder network is *maximally matched*. Absence of resistors in the ladder network minimizes ohmic heating losses. A ladder network can be *even or odd ordered*, depending on the total number of reactive components. A first-order low pass filter consists of an inductor and a capacitor, with the capacitor grounded, and the output is obtained from the common node of the capacitor and inductor. A high-order low pass filter may be constructed by cascading a number of these paired capacitor and inductor segments in series. Ladder networks have been used for a long time and are related to transmission lines.
- *Canonical or normalized or prototype filters*. The simple low pass filter example of Chap. 1 is a *ladder filter*. The generalized transfer function is too difficult to solve and extract the numerical values of the components. This problem is addressed with the concept of *canonical* or *normalized* or *prototype* filter, which is a low pass filter with both the source and load resistances having value of 1 Ω (*maximally matched*), and the cutoff frequency is set to 1 rad/s. With these modifications, tables [1] of capacitance and inductance values are easily calculated. A ladder filter is of even order if it ends with a horizontal L (inductor) branch or odd order if it ends with a vertical C (capacitor) branch, with one capacitor terminal grounded. The generic filter design scheme consists of first designing a normalized filter of the appropriate order and then scaling it to the required cutoff frequency and source/load impedance with simple

© Springer International Publishing Switzerland 2017

A. Banerjee, *Automated Electronic Filter Design*,
DOI 10.1007/978-3-319-43470-4_2

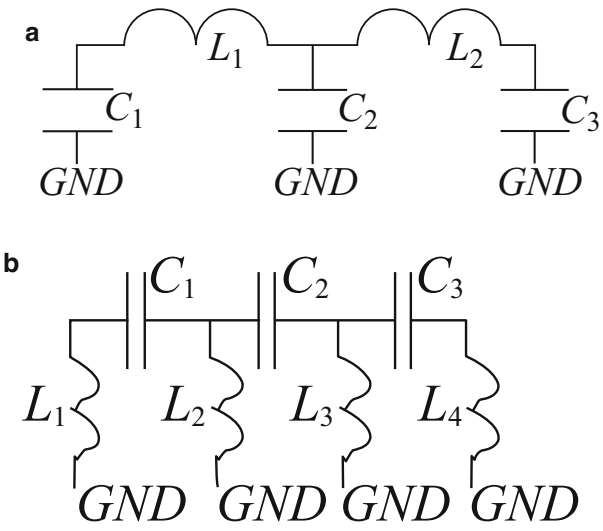

Fig. 2.1 (**a**) Simple ladder network; (**b**) simple ladder network

mathematical transformations. The sequence of steps involved in designing a real-world filter can be automated easily, in this case with a *C* language program, and its output is in the popular *S*imulation *P*rogram with *I*ntegrated *C*ircuit *E*mphasis (SPICE) [2] input format, allowing for quick and easy performance evaluation and fine-tuning.

- *Maximum available source power for AC circuit*. For a maximally matched (source and load resistance identical in value) AC circuit, the maximum available source power is $P_{\mathrm{m}} = \frac{V_{\mathrm{s}}^2}{8R_{\mathrm{s}}}$ where V_{s} is the source voltage and R_{s} is the source/load resistance.
- *Pass/stop band ripple* and *pass band edge/stop band start frequencies*. The frequency response plot of any real-world filter has deviations from a flat line or *ripples* (d_{P}, d_{S}) however small, both in the pass and stop bands. These ripples indicate how much the filter signal response deviates from the ideal signal response. The frequency response plot can be divided into three regions—pass band, transition band, and stop band. The angular frequencies W_{P} and W_{S} indicate the end of the pass band and the start of the stop bands, respectively.
- *Insertion loss*. Signal passing through a real-world filter will always lose a small fraction of its input energy—*insertion loss*. The ideal filter has zero insertion loss. The unit of insertion loss is decibel (dB).
- *Shape factor and rejection*. Shape factor is the sharpness of the filter response, while rejection measures the attenuation of undesired signals.
- *Quality factor*. A parameter to measure the selectivity of a filter. For an unloaded filter, the quality factor is defined as $Q_{\mathrm{unloaded}} = 6.28 f_{\mathrm{c}} \frac{\text{maximum energy stored} \in \text{the filter at } f_{\mathrm{c}}}{\text{filter power loss}}$ where f_{c} is the cutoff (high/low

pass filter) or center frequency (band/notch filter). For a loaded filter, the quality factor is defined as $Q_{\text{loaded}} = 6.28 f_{\text{c}} \frac{\text{maximum energy stored} \in \text{the filter at } f_{\text{c}}}{\text{power loss} \in \text{filter} \wedge \text{external load circuit}}$.

Some popular electronic filters are Bessel, Butterworth, and Chebyshev. The elliptic or Cauer filter, although having several superior characteristics as compared to the other three, is difficult to design and implement and thus used in specific demanding applications. The Butterworth low pass filter, theoretically, has maximally flat pass band, with some negligible ripple in the stop band. The Bessel low pass filter ideally has maximally flat frequency response both in the pass and stop bands. The Chebyshev type I low pass filter has pass band ripple, and the Chebyshev type II low pass filter has stop band ripple, which is mostly ignored. In fact, the Chebyshev filter's pass band ripple allows the designer to convert a Butterworth low pass filter to a Chebyshev low pass filter with appropriate transformations. While this discussion has focused on the low pass filter, others (band, high pass, etc.) can be synthesized from a low pass filter with appropriate transformations, as examined in detail in the subsequent sections.

2.2 Normalized Butterworth Filter

When the Butterworth filter was developed, it was found that as the number of stages in a low pass filter is increased, the frequency response became more and more flat in the pass band. More interestingly, a low pass filter could be designed with cutoff frequency *normalized* to 1 rad/s and whose frequency response (gain) could be expressed as $G(w) = \sqrt{\frac{1}{1+w^{2n}}}$ where w is the angular frequency in radian per second and n is the number of poles or equivalently the number of reactive elements in a passive filter—*n is filter order. If $w = 1$, the amplitude of the frequency response in the pass band is $1/\sqrt{2} \approx 0.707$, which is half power or -3 dB*.

On a logarithmic Bode plot, the response monotonically decreases toward negative infinity. While a first-order filter's frequency response decreases at -6 dB/octave (equivalently -20 dB/decade), that of a second-order filter decreases at -12 dB/octave, a third-order at -18 dB, etc. The transfer function of a third-order *un-normalized* low pass Butterworth filter in the frequency domain s-plane is

$$\frac{V(s)_{\text{o}}}{V(s)_{\text{i}}} = \frac{R}{s^3(C_2L_1L_3) + s^2(C_2L_1\,R) + s(\,L_1 + L_3) + R}. \tag{2.1}$$

Using sample trial values of $C_2 = 1.333$ F, $R = 1\,\Omega$, $L_1 = 1.5$ H, and $L_3 = 0.5$ H and keeping in mind that $s = \sigma + jw$ is the complex frequency, the circuit equations can be manipulated to transform the transfer function to

$$H(s) = \frac{V(s)_{\mathrm{o}}}{V(s)_{\mathrm{i}}} = \frac{1}{s^3 + 2s^2 + 2s + 1}. \tag{2.2}$$

The magnitude of the frequency response or gain is given as

$$G^2(w) = \frac{1}{1 + w^6} \text{ or } G(w) = \frac{1}{\sqrt{1 + w^6}}. \tag{2.3}$$

The group delay, defined as *the distortion in the signal introduced by phase differences for different frequencies*, is measured by taking the derivative of the phase with respect to angular frequency w. For the low pass Butterworth filter, there are no ripples in the gain curve in either the pass band or the stop band. Generalizing, the gain of an nth-order Butterworth filter is

$$G(w)^2 = \left|H(jw)\right|^2 = \frac{G_{\mathrm{dc}}^2}{1 + \left(\frac{w}{w_{\mathrm{c}}}\right)^{2n}}, \tag{2.4}$$

where G_{dc} is the DC or zero frequency gain, w_{c} is the cutoff or -3 dB roll-off frequency, and n is the filter order.

As n approaches infinity, the gain becomes a rectangle function and frequencies below w_{c} will be passed with gain G_{dc}. Theoretically, all frequency components above the cutoff frequency will be *filtered out*. The filtering effect is pronounced with higher values of n.

Using $s = a + jw$ and as the transfer function can be expressed as $H(s)^2 = H(s)H(-s) = \left|H(s)\right|^2$,

$$H(s)H(-s) = \frac{G_{\mathrm{dc}}^2}{1 + \left(\frac{-s^2}{w_{\mathrm{c}}^2}\right)^n}. \tag{2.5}$$

The n poles are located symmetrically (about the imaginary axis in the complex plane) separated by the same angle, on a circle of radius w_{c}. To guarantee filter stability, the transfer function $H(s)$ is structured so that the poles occur only in the negative real half of complex s-plane. Then, the kth pole can be expressed as

$$\frac{-s_k^2}{w_{\mathrm{c}}^2} = (-1)^{\frac{1}{n}} = \mathrm{e}^{\frac{3.14j(2k-1)}{n}} \text{ where } k = 1, 2, 3, \ldots, n. \tag{2.6}$$

In addition, $S_k = w_{\mathrm{c}}\mathrm{e}^{\frac{3.14j(2k+n-1)}{n}}$ where $k = 1, 2, 3, \ldots, n$.

Using the above expressions, the transfer function can be expressed in a product form of the k poles ($k = 1, 2, 3, \ldots, n$) as

$$H(s) = \frac{G_{\mathrm{dc}}}{\mathrm{Product}\left(\frac{3-s_k}{w_c}\right)}. \tag{2.7}$$

Equation (2.7) is the generalized Butterworth polynomial in complex form. In practice, they are usually written with real coefficients by multiplying each pole with its complex conjugate. The polynomials are *normalized* by setting $w_c = 1$ rad/s. The normalized Butterworth polynomials look like

n even:

$$B_n(s) = \mathrm{Product}\left(s^2 - 2s\cos\left(3.14\frac{2k+n-1}{2n}\right) + 1\right) \text{ where } k = 1, \ldots, \frac{n}{2}, \tag{2.8}$$

n odd:

$$B_n(s) = (s+1)\,\mathrm{Product}\left(s^2 - 2s\cos\left(3.14\,\frac{2k+n-1}{2n}\right) + 1\right) \text{ where } k = 1, \ldots, \frac{n-1}{2}. \tag{2.9}$$

Combining these expressions, the factors of the first five normalized Butterworth polynomial $B_n(s)$ are listed in Table 2.1. These can be extended to any order of one's choice.

Finally, combining the concepts of normalized low pass filter and normalized Butterworth polynomial, for a source resistance $R_s = 1$, the normalized susceptances and reactances (often referred to as immittances) can be expressed as

$$a_k = 2\sin\left(\frac{3.14(2k-1)}{2n}\right) \text{ where } k = 1, 2, \ldots, n. \tag{2.10}$$

These immittance values for the first seven normalized Butterworth low pass filters are shown in Table 2.2.

Table 2.1 First five normalized Butterworth polynomials

Order(n)	Normalized and factorized Bessel polynomial
1	$(s+1)$
2	$(s^2 + 1.4142s + 1)$
3	$(s+1)(s^2 + s + 1)$
4	$(s^2 + 0.7654s + 1)(s^2 + 1.8478s + 1)$
5	$(s+1)(s^2 + 0.6180s + 1)(s^2 + 1.6180s + 1)$

Table 2.2 Normalized immittance low pass Butterworth

Order	R_s	C_1 a_1	L_2 a_2	C_3 a_3	L_4 a_4	C_5 a_5	L_6 a_6	C_7 a_7
1	1	1	2					
2	1	1.4142	1.4142					
3	1	1	2	1				
4	1	0.7654	1.8478	1.8478	0.7654			
5	1	0.618	1.618	2	1.618	0.618		
6	1	0.5176	1.4142	1.9319	1.9319	1.4142	0.5176	
7	1	0.445	1.247	1.8010	2	1.8010	1.247	0.445

2.3 Practical Normalized Low Pass Butterworth Filter

In the previous discussion, it is assumed that the filter's pass band frequency is maximally flat, unlike in the real world where both the pass and stop bands have finite (maybe very small) ripples. These ripples may be quantified with the values d_p and d_s (both unit dB) for the pass and stop bands, respectively. Frequency values w_p (*pass band edge/end frequency*) and w_s (*stop band start frequency*) correspond to d_p and d_s, respectively. These quantities are related to each other via the following constraints:

$$\begin{gathered} \left|H\left(w_p\right)\right| = 1 - d_p \text{ and } \left|H\left(w_s\right)\right| = d_s, \\ \left(\frac{w_p}{w_c}\right)^{2n} = \left(\frac{1}{1-d_p}\right)^2 - 1 \text{ and } \left(\frac{w_s}{w_c}\right)^{2n} = \left(\frac{1}{d_s}\right)^2 - 1. \end{gathered} \tag{2.11}$$

Manipulating the above four equations provides the key equation for the order of a normalized low pass Butterworth filter, with predefined permissible pass and stop band ripple values:

$$n = (\text{Integer})\left[\left(\frac{1}{2}\right)\left[\frac{\log\left(\frac{d_p\, d_s^2\left(2-d_p\right)}{\left(1-d_p\right)^2\left(1-d_s\right)^2}\right)}{\log\left(\frac{w_p}{w_s}\right)}\right]\right]. \tag{2.12}$$

In Eq. (2.12), n is the order of the filter, and the integer operation returns the integer value of the computed floating point value.

Based on Eqs. (2.1)–(2.12), a very simple algorithm (algorithm A) can be formulated to design a normalized low pass Butterworth filter:

- Designer specifies the pair (filter order, cutoff frequency) or pass/stop band maximum attenuation in dB, the pass band edge and stop band start frequencies, respectively.

- If the filter order is specified, the normalized filter coefficients may be obtained from a table or use of very simple mathematical formulas. These coefficients are the values of the capacitor and inductor values to be used in the ladder network.
- If the filter order is not specified, its value is calculated using Eq. (2.12) and the cutoff frequency from Eq. (2.11), and then the normalized filter coefficients may be obtained from a table or use of very simple mathematical formulas. The computed cutoff frequency is *not* the normalized cutoff frequency of 1 rad/s. These coefficients are the values of the capacitor and inductor values to be used in the ladder network.

This simple algorithm can be implemented easily as a computer program.

2.4 Normalized Chebyshev Low Pass Filter

The ideal normalized low pass Butterworth filter, discussed in Sect. 2.2, has a maximally flat pass band region and a gradual roll-off to the stop band. Another widely used filter is the Chebyshev filter, which compared to the ideal normalized low pass Butterworth filter, has a sharp roll-off, introducing ripples in its pass band. Chebyshev filters can be of two types, the type I and type II, depending on the type of underlying Chebyshev polynomial. The normalized Chebyshev polynomial of type I order n is the basis for a normalized low pass Chebyshev filter of type I, and a normalized Chebyshev polynomial of type II is the basis for a normalized low pass Chebyshev filter of type II. The Chebyshev type I low pass filter has ripples in the pass band, while the type II filter has ripples in the stop band. The type II Chebyshev filter is called *inverse Chebyshev filter* and is less common than the type I.

Chebyshev filters are based on Chebyshev polynomials defined as

$$T_n(w) = \cos(n\arccos(w)), |w| \leq 1 \text{ and } T_n(w) = \cos h(n\arccos(w)), |w| > 1. \tag{2.13}$$

Given $T_0(w) = 1, T_1(w) = w$, the higher-order Chebyshev polynomials are generated recursively, using the recursion relation:

$$T_{n+1}(w) = 2wT_n(w) - T_{n-1}(w), \quad n \geq 1 \tag{2.14}$$

satisfying the following properties:

$$-1 \leq |T_n(w)| \leq +1 \text{ for } |w| \leq 1$$

$|T_n(w)|$ increases monotonically with w for $|w| > 1$,
$T_n(1) = 1$ for all n,
$T_n(0) = \pm 1$ for n even, and $T_n(0) = 0$ for n odd.
All zero crossings occur in the range $-1 \leq w \leq +1$.

The transfer function of a type I low pass Chebyshev filter of order n is

$$|H_n(jw)|^2 = \frac{1}{1 + \text{epsilon}^2 T^2_{n,\text{ prime}}}, \tag{2.15}$$

where $T_{n,\text{prime}} = T_n\left(\frac{w}{w_\text{p}}\right)$, w_p, is the pass band edge frequency (radians/second) and *epsilon* is the maximum allowable pass band ripple parameter computed using filter designer-supplied maximum pass band attenuation (A_p unit dB). For a type I Chebyshev filter, ripples increase monotonically with filter order. The peak-to-peak pass band ripple is expressed as $\frac{1}{\sqrt{1+\text{epsilon}^2}}$ or $10\log\left(1 + \text{epsilon}^2\right)$. The stop band start frequency w_s is defined in relation to the stop band attenuation (A_s unit dB) such that at the stop band start frequency, the magnitude of the frequency response (Eq. 2.15) satisfies $|H_n(jw)| < \frac{1}{A_\text{s}}$ where $w_\text{p} < w_\text{s} < w$ in the stop band. In the frequency range $w > w_\text{s}$, the output signal attenuation is $20\log(A_\text{s})$ dB as compared to its pass band value.

To design a normalized type I Chebyshev low pass filter, the designer supplies four quantities w_p, w_s, pass band attenuation, and stop band attenuation, to compute *epsilon*, the filter order n, the cutoff frequency, and then the poles. The ripple factor *epsilon* is computed as

$$\text{epsilon} = \sqrt{10^{\frac{A_\text{p}}{10}} - 1}. \tag{2.16}$$

To compute the filter order n, one starts with condition that in the stop band, the magnitude of the transfer function is less than or equal to inverse of the stop band attenuation, resulting in the following inequality:

$$n \geq \frac{\text{arccos}h\left(\sqrt{\frac{10^{0.1A_\text{s}}-1}{10^{0.1A_\text{p}}-1}}\right)}{\text{arccos}h\left(\frac{w_\text{s}}{w_\text{p}}\right)}. \tag{2.17}$$

The cutoff frequency for this filter is calculated from the input values of pass band edge frequency, stop band start frequency, pass band attenuation, and stop band attenuation and is given by

$$w_\text{c} = w_\text{p} \cos h\left(\frac{\text{arccos}h\left(\frac{1}{\sqrt{10^{0.1A_\text{p}}-1}}\right)}{n}\right). \tag{2.18}$$

Equation (2.18) assumes that the cutoff frequency is larger than the pass band edge frequency, the common case. In special cases, the cutoff frequency is less than the

pass band edge frequency, and then Eq. (2.18) is modified by replacing the hyperbolic cosine and inverse hyperbolic cosines with the normal cosine.

A simple, but powerful alternative approach to directly computing the poles starts with the computation steps of Eqs. (2.16) and (2.17) and then exploiting one intermediate quantity such that the immittance values c_i of a type I Chebyshev low pass filter of order n can be computed recursively from the corresponding immittance values a_i of a low pass Butterworth filter of order n. This quantity is

$$\text{beta} = \sin h\left(\frac{\text{arctan}h\left(\frac{1}{\sqrt{1+\text{epsilon}}}\right)}{n}\right). \tag{2.19}$$

Then, the Chebyshev immittance values c_i are computed recursively as

$$c_1 = \frac{a_1}{\text{beta}} \quad \text{and} \quad c_i = \frac{a_i a_{i-1}}{c_{i-1}\left(\text{beta}^2 + \sin\left(\frac{3.14(i-1)}{n}\right)\sin\left(\frac{3.14(i-1)}{n}\right)\right)}, \tag{2.20}$$

where $i = 2, 3, 4, \ldots, n$. This is an extremely powerful result for the following reasons.

Although pass band ripple and stop band attenuation/ripple are included in the design of the ideal Butterworth low pass filter, theoretically the Butterworth filter has a maximally flat pass band, and a single table of filter coefficients is sufficient to design a low pass filter of any order. This is strictly not true for the ideal Chebyshev type I low pass filter, because pass band ripple is explicitly included in the filter coefficient calculation, and theoretically a new table of coefficients needs to be generated for each value of pass band ripple. Equation (2.19) eliminates this issue related to Chebyshev filters. So, using an example pass band ripple of 0.2 dB, the normalized immittances for a Chebyshev low pass filter for orders 1–7 are listed below (Table 2.3).

In the discussions for the Butterworth and Chebyshev normalized low pass filters, Eqs. (2.12) and (2.17) above provide the *lowest* value of filter order *n* that

Table 2.3 Normalized Chebyshev low pass immittance values for $n = 1$ to $n = 7$ for pass band ripple factor of 0.2 dB

Order	R_s	C_1	L_2	C_3	L_4	C_5	L_6	C_7
		c_1	c_2	c_3	c_4	c_5	c_6	c_7
1	1	0.4342						
2	1	1.0378	0.6746					
3	1	1.2275	1.1525	1.2275				
4	1	1.3028	1.2844	1.9752	0.8468			
5	1	1.3394	1.3370	2.1601	1.3370	1.3394		
6	1	1.3598	1.3632	2.2395	1.4556	2.0974	0.8838	
7	1	1.2723	1.2782	2.2757	1.5002	2.2757	1.3782	1.3723

will satisfy the designer's choice of pass band ripple, stop band attenuation/ripple, pass band edge frequency, and stop band start frequency. A higher value of filter order would generate a filter that satisfies the designer's initial conditions more accurately.

Based on Eqs. (2.14)–(2.19) above, a sequence of simple calculations based on designer input can be used to calculate the filter order and determine the coefficients of the corresponding normalized low pass Chebyshev filter. Each of these calculation steps can be automated in as a computer program. The minor difference between this scheme for the Chebyshev filter and the Butterworth filter is that for the latter, the designer can simply use the filter order and cutoff frequency to design the filter, because the low pass Butterworth filter has a maximally flat pass band frequency response. The interested reader may refer to [1–8] for rigorous derivation of normalized filter equations and polynomial coefficients for each of Bessel, Butterworth, and Chebyshev filters.

2.5 Normalized Inverse Chebyshev Filter

A type I low pass Chebyshev filter discussed above is based on Chebyshev type I polynomial, whereas an inverse Chebyshev low pass filter is based on the Chebyshev type II polynomial. These two types of low pass filters are differentiated by the type I having ripples in the pass band only, while the type II has ripples in the stop band only. Since a low pass filter with ripples in the stop band has a frequency response that looks similar to a Butterworth filter, this filter is not so popular as the type I Chebyshev filter.

2.6 Normalized Bessel Filter

The Bessel filter is linear (just as the Butterworth and Chebyshev) and closely related to the Gaussian filter. It has maximally flat group and phase delay, i.e., maximally linear phase response. In combination, these properties enable the filter to maintain the shape of the filtered signal. It has no overshoot in the step response in the time domain, a unique characteristic.

In general, the transfer function of an n-order low pass Bessel filter is

$$B_n(S) = \frac{B_n(0)}{B_n\left(\frac{s}{s_c}\right)}, \tag{2.21}$$

where B_n is the reverse Bessel polynomial of order n, and w_c is the cutoff frequency, and the low-frequency group delay is $\frac{1}{w_c}$. By definition, $B_n(0)$ is a singularity, but can be removed by taking appropriate limits. A compact representation of the same

Table 2.4 Delay normalized to frequency normalized Bessel polynomial conversion factors for $n = 2$ to $n = 10$

Order	Conversion factor
2	1.3616
3	1.7557
4	2.1139
5	2.4274
6	2.7034
7	2.9517
8	3.1796
9	3.2916
10	3.5910

Table 2.5 Frequency normalized Bessel polynomials for capacitor first configuration, for order $n = 1$ to $n = 9$

n	R_s	C_1	L_2	C_3	L_4	C_5	L_6	C_7	L_8	C_9
1	1	2.00								
2	1	2.1478	0.5755							
3	1	2.2034	0.8705	0.4474						
4	1	2.2040	1.0815	0.6725	0.2334					
5	1	2.2582	1.1110	0.8040	0.5072	0.1743				
6	1	2.2645	1.1126	0.8538	0.6391	0.4001	0.1365			
7	1	2.2659	1.1052	0.8690	0.7020	0.5249	0.3259	0.1105		
8	1	2.2656	1.0855	0.8695	0.7302	0.5936	0.4409	0.2719	0.0919	
9	1	2.2649	1.0863	0.8639	0.7407	0.6396	0.5108	0.3761	0.2313	0.078

Bessel polynomial consists of the sum of product of normalized coefficients and powers of $s(c_j s^j)$, from $j = 0$ to $j = n$. Bessel polynomials are normalized in two ways. The first method, called *delay normalized*, is based on unit delay at $w = 0$. The other scheme based on 3 dB roll-off at 1 rad/s is referred to as *frequency normalized*. These two normalization schemes are used because of the unique properties of the Bessel filter. The focus here is on frequency normalization. The delay normalized Bessel polynomials can be converted to frequency normalized ones, using a set of scaling factors as listed in Table 2.4.

Also, ladder network-based Bessel filters can be configured in two equivalent ways for any filter order n (even or odd) to have an inductor or capacitor as the first reactive element right after the source resistance—the first is called *inductor first* and the second *capacitor first*. Using identical sequence of steps as in the case for Butterworth and Chebyshev filters, expressions for filter order and cutoff frequency may be derived. The frequency normalized Bessel polynomials for capacitor first configuration and source and load resistances of 1 Ω are listed below in Table 2.5.

2.7 Denormalizing Prototype Filters to Real-World Filters

Each of the normalized prototype filters examined so far has a cutoff frequency of 1 rad/s and source/load impedance/resistance of 1 Ω—useless for any real-world filtering operation. In addition, although the design procedures discussed above use input parameters (pass/stop band attenuation, pass band edge, and stop band start frequencies) that are not normalized values, the basic starting point for a given design is the normalized prototype filter tables (Tables 2.2, 2.3, etc.). The question is how would the designer transform a normalized prototype filter to a real-world one that accurately satisfies the designer's specifications. This is achieved by two transformations—*frequency* and *impedance* scaling.

2.7.1 Frequency Scaling

Any filter design can be transformed from one reference frequency to another reference frequency by dividing all reactive elements (capacitors and inductors) by a frequency scaling factor (simple dimensionless number) k_f defined as

$$k_f = \frac{\text{new reference frequency}}{\text{old reference frequency}}. \tag{2.22}$$

A simple second-order filter is analyzed in detail to understand this step. The generalized transfer function for this filter is

$$H(s) = \left(\frac{R_L}{R_L + R_S}\right)\left(\frac{1}{s^2\left(\frac{C_1 L_2 R_S}{R_L + R_S}\right) + s\left(\frac{C_1 R_L R_S + L_2}{R_L + R_S}\right) + 1}\right). \tag{2.23}$$

As the denominator is a polynomial of degree 2, its roots in the left half of the complex plane may be denoted temporarily as A and B, where A and B are expressions involving the capacitor C_1, L_2, and the source/load resistances. The transfer function can then be rewritten as

$$H(s) = \frac{K}{(As+1)(Bs+1)} \quad \text{where} \quad K = \frac{R_L}{R_L + R_S}. \tag{2.24}$$

Now s in Eq. (2.23) is replaced with $\frac{s}{k_f}$ that is old $s = \frac{\text{new } s}{k_f}$. Using $s = jw$, this transformation corresponds to *new* $w = k_f$ *old* w in the frequency domain. Correspondingly, the transfer function is now changed to

$$H\left(\frac{s}{k_f}\right) = \frac{K}{\left(\left(\frac{A}{k_f}\right)s + 1\right)\left(\left(\frac{B}{k_f}\right)s + 1\right)}. \tag{2.25}$$

Clearly, the new transfer function $H\left(\frac{s}{k_f}\right)$ is obtained from the original transfer function, by applying the transformation $\left(\frac{A}{k_f}\right)$ and $\left(\frac{B}{k_f}\right)$, and this scheme works for any higher-order filter. In the same way, filter components are transformed as

$$C_{new} = \frac{C_{old}}{k_f} \text{ and } L_{new} = \frac{L_{old}}{k_f}. \tag{2.26}$$

The resistances remain unaffected, as input signal frequency only affects the reactive elements.

2.7.2 Impedance Scaling

Normalized or prototype filters have source/load impedance/resistance of 1 Ω unlike the real world where, for example, RF filters use source/load impedance of 50 Ω and telephone and audio signal processing filters use source/load impedance of 600 Ω. To understand how to transform a filter design to have a general source/load impedance of R_L, the general transfer function for a second-order filter, i.e., Eq. (2.23), is examined in detail.

In Eq. (2.23), the kth coefficient of the denominator has dimensions $\left(\frac{\text{radian}}{\text{second}}\right)^k$ as s (especially when $s = jw$) has dimensions $\left(\frac{\text{radian}}{\text{second}}\right)^k$. As $H(s)$ is dimensionless, so ratio of the source/load impedances (or any linear combination of these, e.g., $\frac{R_L}{R_L+R_S}$) *cannot change* $H(s)$ provided each resistance or impedance is scaled by the identical scaling factor.

This scaling property of resistances or impedances *does not* apply to other combinations as *RC* or *L/R*, etc., that is, these change if only the resistance is scaled, not if *both* are scaled by the same scaling factor. That is, if *L* and *R* are scaled by the identical scale factor, the ratio *L/R* remains constant, and if the capacitance values are scaled by inverse of the scale factor used to scale the resistance, then the product RC remains constant. Then, a new scaling factor may be defined as

$$k_z = \frac{\text{desired load impedance}}{\text{old load impedance}}. \tag{2.27}$$

To scale existing/old capacitance, inductance, and resistance values to new values, the following simple expressions may be used:

$$C_{\text{new}} = \frac{C_{\text{old}}}{k_z}, \quad L_{\text{new}} = k_z(L_{\text{old}}) \text{ and } R_{\text{new}} = k_z(R_{\text{old}}). \tag{2.28}$$

2.8 Filter Transformations

So far the discussion on normalized filters, frequency scaling, and impedance scaling has been based on the low pass filter. There are three other types of filter, high pass, band pass, and band stop, that are equally important. The conversion from a low pass to a high pass and from a low pass to a band pass is examined in detail now. The conversion of a low pass filter to a band stop filter is left as an exercise for the reader.

2.8.1 Low Pass to High Pass Filter

Let the complex frequency variable associated with a low pass filter be s_{L}. To convert the low pass filter to a high pass filter, s_{L} is replaced by $s = \frac{1}{s_{\text{L}}}$ in the low pass filter's transfer function $H(s)$. So, at $s_{\text{L}} = jw_{\text{L}}$ $s = \frac{-1}{w_{\text{L}}}$, the transfer function for the high pass filter is simply the transfer function for the low pass filter evaluated at $s_{\text{L}} = \frac{-j}{w_{\text{L}}}$. For real-valued coefficients, the magnitude of the filter transfer function has even symmetry in w. The reactive elements in the low pass filter transfer function are transformed as

$$\frac{1}{C_{s_{\text{L}}}} = \frac{s}{C} \text{ and } \text{Ls} = \frac{L}{s}. \tag{2.29}$$

Summarizing, a capacitor $C_i^{(\text{LPF})}$ in the low pass filter becomes an inductor $L_i = C_i^{(\text{LPF})}$ in the high pass filter, and an inductor in the low pass filter becomes a capacitor $C_i = L_i^{(\text{LPF})}$ in the high pass filter.

2.8.2 Low Pass Filter to Band Pass Filter

Converting a low pass filter design with a design frequency w_{r} to a band pass filter design with center frequency w_{r} and bandwidth $2w_{\text{r}}$ is achieved in a straightforward manner by replacing s_{L} with $\frac{s^2+w_{\text{c}}^2}{2s}$ in the transfer function for the low pass filter H_{L} (s_{L}). In all the following expressions, the subscript L refers to the low pass filter.

Then, $s^2 - 2s_L s + w_c^2 = 0$, and solving this quadratic equation yields the roots

$$s = s_L \pm \sqrt{s_L^2 - w_c^2} \text{ or } s = s_L \pm j\sqrt{w_c^2 - s_L^2}. \tag{2.30}$$

For each value of s_L, two values of s are generated, one for which the imaginary part is shifted up by $\sqrt{w_c^2 - s_L^2}$ and for the other the imaginary part is shifted down by $\sqrt{w_c^2 - s_L^2}$.

Setting $s = jw_L$ (e.g., resonance)

$$s_L = jw_L \text{ and } s = jw_L \pm j\sqrt{w_c^2 - (jw_L)^2} = j\left(w_L \pm \sqrt{w_c^2 + w_L^2}\right) \tag{2.31}$$

In the special case of $w > 0$, the design frequency $+w_r$ of the low pass filter maps into

$$\pm w_r + \sqrt{w_c^2 + w_r^2}. \tag{2.32}$$

This means that if w_r is the -3 dB frequency of the low pass filter, then the corresponding -3 dB frequency of the corresponding narrow band pass filter ($w_c \gg w_r$) is $2w_r$. Thus, each capacitor C in the low pass filter is converted to a parallel combination of a capacitor and an inductor of values:

$$\frac{C}{2} \text{ and } \frac{2}{w_c^2 C}, \text{ respectively.} \tag{2.33}$$

In a similar manner, each inductor in the low pass filter is converted to a series combination of a capacitor and inductor of values:

$$\frac{2}{w_c^2 L} \text{ and } \frac{L}{2}, \text{respectively.} \tag{2.34}$$

2.9 Automated Filter Design Scheme

Consolidating the information in Sects. 2.1–2.8, an efficient and powerful scheme can be formulated that is easily automated and enables the practicing electronic filter designer to realize, evaluate, and optimize the performance characteristics of any electronic filter to satisfy predefined specifications. The designer can explore the design space and optimize the performance characteristics. As the existing "gold standard" in evaluating electronic circuit performance characteristics is the *S*imulation *P*rogram with *I*ntegrated *C*ircuit *E*mphasis (SPICE), the proposed scheme would output its results (the candidate filter design) in a format that is

acceptable to SPICE, i.e., the text-based SPICE *netlist* format. The designer can immediately evaluate the performance characteristics of this candidate design and fine-tune it. All manual and error-prone steps are eliminated, increasing designer productivity and design accuracy.

For band pass, high pass, and low pass filters, two alternate, but very closely related schemes are presented.

Scheme A

- The designer provides the pass band edge frequency, the stop band start frequency, the pass band attenuation, the stop band attenuation, the source and load resistance, the filter name (Bessel, Butterworth, etc.), and the type (band or high or low pass).
- For a *high pass* filter, the automated scheme calculates the filter order (even or odd) and cutoff frequency for a low pass filter, then looks up or computes the normalized low pass filter coefficients for the computed filter order, and then performs frequency and impedance scaling, followed by a low pass filter to high pass filter transformation. The final results are formatted in SPICE input netlist format.
- For a *low pass* filter, the automated scheme calculates the filter order (even or odd) and cutoff frequency for a low pass filter, then looks up or computes the normalized low pass filter coefficients for the computed filter order, and then performs frequency and impedance scaling. The final results are formatted in SPICE input netlist format.
- A pass band filter can be considered as a series connection of a high pass and a low pass filter or a stand-alone device, and the proposed scheme can tackle both. If the band pass filter is considered to be a series connection of a low pass and a high pass filter, the designer has to provide the pass band edge frequency, the stop band start frequency, the pass band attenuation, the stop band attenuation, and the source and load resistance for **both** the high and low pass filters. If the band pass filter is considered a stand-alone device, the designer provides the filter order, the band pass low end limit, the band pass high end limit, the source resistance, and the load resistance. The automated scheme determines an appropriate low pass normalized filter and then performs frequency and impedance scaling and finally the low pass to band pass transformation.
- In each case above, the output is the SPICE format netlist.

Scheme B

This is a simplified version of scheme A.

- For a high or low pass filter, the designer provides the filter order (even or odd), the cutoff frequency, the pass band ripple factor (applicable to Chebyshev filters), the filter name (Bessel, Butterworth, etc.), and the type (band or high or low).
- For a *high pass* filter, the automated scheme looks up or computes the normalized low pass filter coefficients for the computed filter order and then performs

frequency and impedance scaling, followed by a low pass filter to high pass filter transformation.
- For a *low pass* filter, the automated scheme looks up or computes the normalized low pass filter coefficients for the computed filter order and then performs frequency and impedance scaling.
- The band pass filter is considered to be a stand-alone device, and the designer provides the filter order, the band pass low end limit, the band pass high end limit, the source resistance, and the load resistance. The automated scheme determines an appropriate low pass normalized filter and then performs frequency and impedance scaling and finally the low pass to band pass transformation.
- In each case above, the output is the SPICE format netlist.

Both scheme A and scheme B are easily implemented as C language programs. Developed on the Linux operating system-based computers and compiled with the popular gcc C language compiler, the generated SPICE netlists can be used with any popular SPICE distribution as LTSpice, HSpice, NgSpice, and PSpice. The program can be executed on any Windows operating system machine under the widely used MingW environment or compiled directly with Microsoft Visual Studio C compiler.

2.10 Low Pass to Band Pass Filter Conversion Example

To illustrate the abovementioned seemingly complicated sequence of steps, a simple low pass to band pass filter conversion example is examined in detail here. The design specifications are a sixth Butterworth band pass filter is required with bandwidth 100 kHz and center frequency 1 MHz.

The starting point is a third-order normalized low pass Butterworth filter with the filter coefficients $C_1 = C_3 = 1$ F, $L_2 = 2$ H, and $R_L = R_s = 1\,\Omega$. The frequency scaling factor, as obtained from given specifications, is

$k_f = \frac{6.28 \times 100,000}{2} = 3.14 \times 10^5$, and the impedance scaling factor is $k_z = 50$.

Then, the capacitance and inductance values for this scaled-up low pass filter are $C_1 = C_3 = 63.66$ nF, $L_2 = 318.3$ μH, and $R_L = R_s = 50\,\Omega$.

Using the band pass center frequency of $f_c = 1$ MHz, each capacitor in the scaled-up (or denormalized) low pass filter is replaced by a parallel LC circuit with capacitor and inductor values $\frac{C}{2}$ and $\frac{2}{w_c^2 C}$, and each inductor in the scaled-up low pass filter is replaced by a series LC circuit with capacitor and inductor values $\frac{2}{w_c^2 L}$ and $\frac{L}{2}$, respectively. The finalized capacitor, inductor, and source/load resistance values are listed in the following table. These values, once formatted in the SPICE input netlist format, can be used to determine the frequency and phase response of the filter, i.e., evaluate the performance characteristics.

The ladder network implementation of this band pass filter is shown in Fig. 2.2.

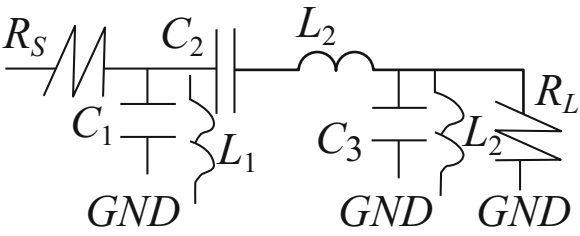

Fig. 2.2 Sixth-order Butterworth band pass filter. Component values are in Table 2.6. "GND" is signal ground

Table 2.6 Finalized capacitor, inductor, and resistor values for band pass filter design example

R source (Ω)	C_1 (nF)	L_1 (nH)	C_2 (pF)	L_2 (μH)	C_3 (nF)	L_3 (nH)	R load (Ω)
50	31.83	795.8	159.16	159.15	31.83	795.8	50

By inspection, the SPICE format netlist for this simple band pass filter is

```
.SUBCKT TESTBP 1 2
* IN
* OUT
C0 3 0 31.83nF
C1 4 2 159.16pF
C2 2 0 31.83nF
L0 3 0 795.8nH
L1 3 4 159.15uH
L2 2 0 795.8nH
R0 1 3 50
R1 2 0 50
.ENDS
```

The input file for the SPICE simulator, with input signal source for the band pass filter and analysis method (.AC or small-signal analysis) specified, is listed below:

```
.SUBCKT TESTBP 1 2
* IN
* OUT
C0 3 0 31.83nF
C1 4 2 159.16pF
C2 2 0 31.83nF
L0 3 0 795.8nH
L1 3 4 159.15uH
L2 2 0 795.8nH
R0 1 3 50
R1 2 0 50
.ENDS
```

```
VS 1 0 DC 0.0001 AC 5
XBP 1 2 TESTBP

.OPTIONS NOPAGE METHOD=GEAR
.AC LIN 20000 500000 2000000
.PRINT AC V(2)
.END
```

Finally, the frequency and phase response of this filter, obtained from the SPICE simulator, is shown in Fig. 2.3a, b, respectively. While the sequence of calculations might appear straightforward, it would be involved for a high-order filter, and it is clear that the entire sequence of steps can be easily automated, especially the generation of the SPICE format netlist.

Exercises

- A Cauer filter is not commonly used and has not been analyzed here. It is characterized by ripple in the pass and stop bands. That is, a Cauer low pass filter frequency response has same ripple in both the pass and stop bands. Derive expressions for the filter order and cutoff frequency of a Cauer low pass filter. Explain why this filter is not used very widely.

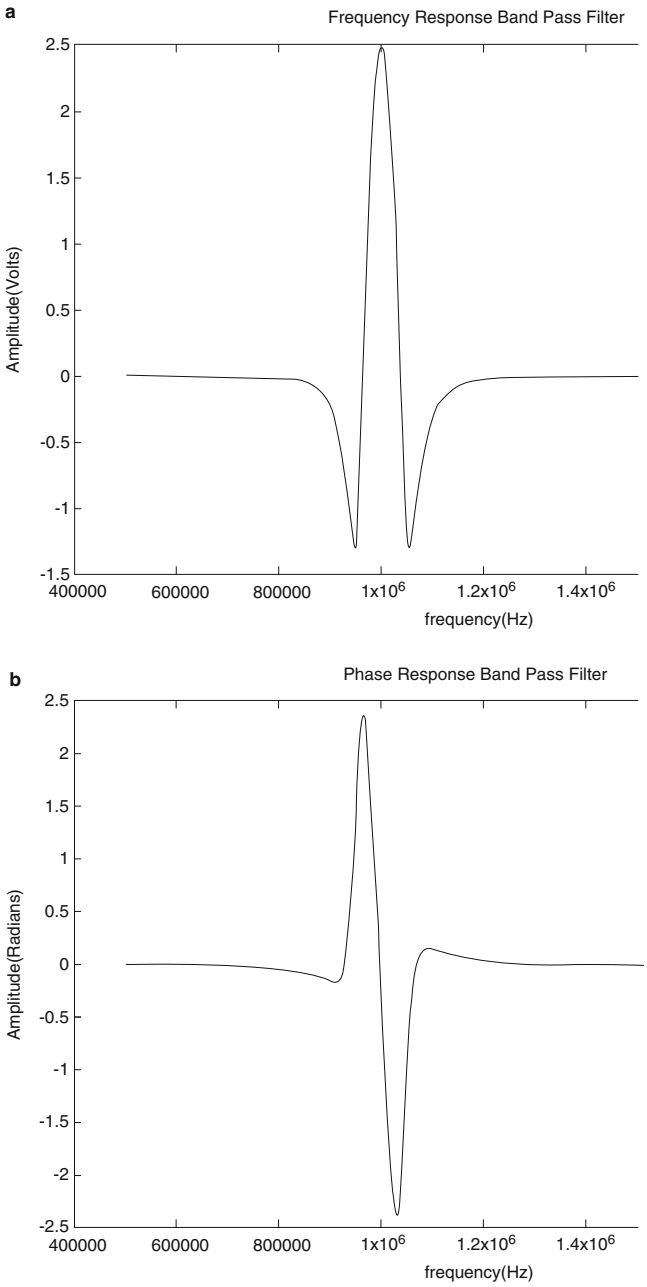

Fig. 2.3 (**a**) Sixth-order band pass filter frequency response; (**b**) sixth-order band pass filter phase response

References

1. Zverev, A. I. Handbook of filter synthesis (Rev. Ed.). ISBN-13: 978-0471749424; ISBN-10: 0471749427.
2. Matthaei, G. L., Young, L., & Jones, E. M. T. (1964). *Microwave filters, impedance-matching networks, and coupling structures*. New York: McGraw-Hill. LCCN 64-7937.
3. Giovanni, B., & Roberto, S. (2007). *Electronic filter simulation & design*. New York: McGraw-Hill. ISBN 978-0-07-149467-0.
4. Daniels, R. W. (1974). *Approximation methods for electronic filter design*. New York: McGraw-Hill. ISBN 0-07-015308-6.
5. Williams, A. B., & Taylors, F. J. (1988). *Electronic filter design handbook*. New York: McGraw-Hill. ISBN 0-07-070434-1.
6. Paarmann, L. D. *Design and analysis of analog filters: A signal processing perspective* (p. 238). Retrieved from http://books.google.com/books?id=l7oC-LJwyegC.
7. Pozar, D. M. (2011). *Microwave engineering* (4th ed.). New York: Wiley. ISBN-10: 0470631554; ISBN-13: 978-0470631553.
8. Retrieved from http://www.matheonics.com/Tutorials/Chebyshev.html.

Chapter 3
Automated Electronic Filter Design Algorithm/Scheme Implementation and Design Examples

3.1 Introduction

Chapter 1 examines in precise detail the steps involved in electronic filter design. To tackle these problems, an algorithm or scheme has been elaborated on in Chap. 2, which makes the design, performance characteristic evaluation, and implementation of these filters more efficient and guarantees that the performance characteristics of the newly designed filter match the initial design specifications. This is because the final filter design is a text file, formatted as a SPICE input netlist. This scheme exploits the properties of the *normalized* or *prototype* filter, which is transformed in a straightforward manner to a filter that satisfies predefined real-world specifications. The simplicity of the intermediate calculation steps allows it to be easily automated as a computer program, e.g., an ANSI C language program. *Automation allows the designer to efficiently explore the design space and fine-tune a design, as well as eliminate manual and error-prone calculation steps*. Specifically, the program output is in the crucial SPICE *netlist* input format, so that the generated netlist can be simulated, with very simple modifications (e.g., addition of signal source, analysis method, etc.) with any common SPICE simulator (HSpice, LTSpice, NgSpice, PSpice, etc.). The scheme is now examined in detail, with exhaustive graded design examples.

Electronic supplementary material: The online version of this chapter (doi:10.1007/978-3-319-43470-4_3) contains supplementary material, which is available to authorized users.

© Springer International Publishing Switzerland 2017

A. Banerjee, *Automated Electronic Filter Design*,
DOI 10.1007/978-3-319-43470-4_3

3.2 Automated Electronic Filter Design Scheme

This scheme exploits the properties of the *normalized* or *prototype* filter and filter transformations to generate the SPICE input format netlist to be simulated to extract the performance characteristics of the filter. As any normalized or prototype filter uses the ladder network topology, which in turn uses *only* passive reactive components (capacitors and inductors), semiconductor device-related issues are eliminated. Though input frequencies in the 100's of MHz to 10's of GHz exceed the cutoff frequencies of almost all discrete passive devices, this scheme can be used to design microstrip filters that work in the microwave sub-band of the frequency spectrum—the fabrication of such filters involves special techniques, to be discussed in Chap. 4. With wireless carrier frequencies increasing each day, these ultrahigh-frequency filters are becoming increasingly important.

As discussed briefly in Sect. 2.9, the scheme can be invoked in one or both of two equivalent ways. Each of Bessel, Butterworth, and Chebyshev filters is *linear*, and the SPICE input netlist format output allows designs to be verified, without much effort. For example, a band pass filter may be conceived of as a stand-alone device, as well as a linear combination of a low pass and a high pass filter—in the case of the latter, the cutoff frequency of the high pass filter is the low end limit of the pass band, and the cutoff frequency of the low pass filter is the high end limit of the pass band. Automation eliminates error-prone manual calculation steps.

A normalized or prototype filter is a low pass filter, and first the filter order and cutoff frequency need to be calculated. These two parameters can be provided directly by the designer or computed from the input values of the pass band edge frequency, the stop band start frequency, the pass, and the stop band attenuation, to be provided by the designer. Once the filter order is known, the corresponding normalized filter coefficients are either obtained from a table or calculated using simple formulas as explained in Chap. 2. For example, for the Butterworth and Bessel normalized filters, there is one table that can supply the filter coefficients for any filter order, but for the Chebyshev type I filter, the coefficients depend on the pass band ripple parameter, so that, e.g., the coefficients for a seventh-order type I normalized Chebyshev filter with pass band ripple parameter of 0.5 dB are different from the coefficients for a seventh-order type I normalized Chebyshev filter with pass band ripple parameter of 0.75 dB. For a Chebyshev type II filter, the ripple is in the stop band, which can be ignored. So for a type I normalized Chebyshev filter of order n and pass band ripple parameter rp, first the coefficients for the normalized Butterworth filter of order n are looked up, and then these values are transformed using simple expressions (discussed in Chap. 2) and the ripple parameter.

Once the normalized filter order, cutoff frequency, and coefficients are known, the values of the cutoff frequency and source and load resistances (also supplied by the designer) are used to *denormalize* the prototype filter, using the two crucial calculation steps—*frequency scaling* and *impedance scaling*. There is no restriction on the order in which these two transformations are to be performed on the normalized filter coefficients. The result is a low pass filter with the computed

cutoff frequency and predefined source and load resistance. Then, depending on predefined requirements, the denormalized low pass filter is transformed to a band pass or high pass or notch filter. There is no restriction on the order in which the denormalization and filter transformation operations are performed. The normalized low pass prototype filter can be first transformed to a band or high pass filter and then frequency and impedance transformed.

The final and possibly most crucial step is to format the finalized filter design in the SPICE input netlist format. This step exploits a set of known heuristics about a ladder network's topology:

- A ladder network-based low pass filter (even or odd order) always has a capacitor connected to ground. Inductors connect two capacitors, but are never grounded. The load resistance is always grounded.
- For a ladder network-based low pass filter of even order, the last reactive element is always an inductor, whereas for an odd order filter, the last reactive element is always a capacitor. The load resistance is always grounded.
- For a ladder network-based high pass filter (even or odd order), a capacitor is never connected to ground. Inductors are always connected to ground. Capacitors interconnect inductors. This is a direct consequence of the transformation used to convert a low pass filter to a high pass filter. The load resistance is always grounded.
- A ladder network-based band pass filter consists of alternating pairs of series and parallel connected capacitors and inductors. Each pair of parallel connected capacitor and inductor has one of its two common nodes grounded. Each series connected pair of capacitor and inductor interconnects two pairs of parallel connected capacitor and inductor.
- For any ladder network, the source resistor may be connected in series to a reactive element or the common node of two reactive elements. The proposed scheme uses the second configuration.

This scheme has been implemented as an ANSI C language computer program, invoked from the command line with appropriate command line arguments. The program has been developed, compiled, and tested on a computer with the Linux operating system (Red Hat Fedora 18) and compiled with the popular *gcc* C/C++ compiler. The executable can be run on any Windows operating system-based computer under the MingW (*M*inimal *G*NU for *W*indows) environment. Alternatively, as the source code is in ANSI C, it can be compiled on any Windows operating system machine with the Visual Studio C compiler and executed as a console application. The final output is a text file containing the SPICE input format netlist. This text file, with minor editing to add signal source and analysis type (SPICE allows DC, small-signal, or .AC, transient, or .TRAN, etc., analysis options), can be simulated on any commonly available SPICE simulator (proprietary or open source) to determine the performance characteristics of the filter. SPICE small-signal or .AC analysis is commonly used (as in the subsequent design examples) to obtain the frequency and phase response curves.

The C language computer program exploits the key features of the C language to enhance execution efficiency. For example, function input/output arguments are passed by pointers, to minimize the overhead of making copies of these. The pointer to each input argument is checked to verify if it is not null—*dereferencing* a null pointer generates an error condition that stops program execution. Once each pointer has been dereferenced, the actual variable values are also checked for validity, e.g., for any low pass filter, the pass band edge frequency must always be less than the stop band start frequency—this needs to be verified before the values are used for any other calculation. Consider the function that computes the filter order and cutoff frequency for a Butterworth filter, using the pass band attenuation, the stop band attenuation, the pass band edge frequency, and the stop band start frequency. The computed cutoff frequency and filter order are stored in variables whose pointers are passed in. The mathematical formulas used have already been discussed in Chap. 2. Documentation/explanations have been inserted as C language-style comments—i.e., between/* and */:

```
void computeBWOrder(double *maxpbdev,
                    double *maxsbdev,
                    double *efreq,
                    double *sfreq,
                    double *bwcfreq,
                    unsigned int *bwfo)
{
 /* Verify that each function input argument
    pointer is not null */
 assert(maxpbdev != NULL);
 assert(maxsbdev != NULL);
 assert(efreq != NULL);
 assert(sfreq != NULL);
 assert(bwcfreq != NULL);
 assert(bwfo != NULL);

 /* Dereference each input argument pointer */
 double lmaxpbdev = *maxpbdev;
 double lmaxsbdev = *maxsbdev;
 double lefreq    = *efreq;
 double lsfreq    = *sfreq;

 /* Declare/define some local variables */
 double delta1 = 0;
 double delta2 = 0;
 double temp0 = 0;
 double temp1 = 0;
 double temp2 = 0;
 unsigned int lfo = 0;
```

```
 /* Verify that input values are valid -
 for example, both pass band edge and
 stop band start frequencies are greater
 than zero, and the pass band edge frequency
 is less than the stop band start frequency */

 assert(lmaxpbdev > 0);
 assert(lmaxsbdev > 0);
 assert(lefreq > 0);
 assert(lsfreq > 0);
 assert(lmaxpbdev < lmaxsbdev);
 assert(lefreq < lsfreq);

 /* Filter order and cut-off frequency calculation */
 delta1 = 1.0 - pow(10.0, -(1.0/lmaxpbdev));
 delta2 = 1.0 - pow(10.0, -(1.0/lmaxsbdev));
 temp0 = log10((delta1*(2.0 - delta2)*delta2*delta2)/
        ((1.0 - delta2*delta2)*(1.0 - delta1*delta1)));
 lfo=(unsigned int)ceil(fabs(temp0/log10(lefreq/lsfreq)));

 /* Store the calculated filter order and cut-off
 frequency by dereferencing their pointers, passed
 in via the function argument list */
 *bwfo = lfo;

 temp1 = 1.0/(2.0*lfo);
 temp2 = (delta2*delta2)/
        (1.0 - delta2*delta2);
 *bwcfreq = lsfreq*pow(temp2, temp1);
 /* print calculated values for inspection
    in console window */
 printf("Cut off %e Hz\tfilter order %d\n",
        *bwcfreq, *bwfo);
}
```

Once the filter order and cutoff frequency are determined, the normalized filter coefficients are either calculated on the fly or looked up from a table. For a Butterworth low pass filter, theoretically the pass band is maximally flat and does not depend on the pass band ripple, and so the normalized filter coefficients can be stored in a table and its contents accessed as required. This is true for Bessel low pass filters as well-normalized Chebyshev low pass filter coefficients however depend on the pass band ripple parameter, and *for a given filter order*, *the normalized filter coefficients are different for different pass band ripple factors. So*, *for normalized Chebyshev low pass filter of order n and a computed pass band ripple parameter*, *the normalized coefficients need to be computed on the fly*, *using the corresponding normalized low pass Butterworth filter coefficients for the same*

order n and a few additional mathematical operations discussed in the previous chapter.

Therefore, to design a Chebyshev filter (band/high/low), the following sequence of steps is executed:

- The designer supplies the pass band edge frequency, the stop band start frequency, and the maximum pass and stop band attenuation values, which the program uses to compute the filter order, the cutoff frequency, and the pass band ripple factor for a low pass Chebyshev filter that will satisfy the designer's input constraints.
- The coefficients for a normalized low pass Butterworth of the same order are extracted from a table and transformed (using the pass band ripple factor) to compute the normalized filter coefficients for a low pass Chebyshev filter.
- The normalized low pass filter coefficients are then frequency and impedance scaled and if required transformed to a band or high pass filter.

The following code snippet computes the intermediate values that are used to transform normalized low pass Butterworth filter (order *n*) coefficients to the corresponding normalized low pass Chebyshev filter coefficients (order *n*). The formulas have been examined in Chap. 2:

```
/* temp0 is local intermediate variable and
  dlfo is the computed filter order */
temp0 = lpbr/10.0;
alpha = pow(10.0, temp0) - 1.0;
beta = sinh(atanh(1.0/(sqrt(1.0 + alpha)))/dlfo);
```

The function *computeCHOrder* computes the filter order for a low pass Chebyshev filter satisfying predefined pass band edge frequency, stop band start frequency, pass band attenuation, and stop band attenuation. The following code snippet calculates the values, using formulas presented in Chap. 2:

```
/* temp0, temp1 and epsilonA are intermediate variables */
/* Epsilon is the pass band ripple factor and chfo
   and chcfreq are respectively the computed
   filter order and cut-off frequency */
temp0 = lmaxpbdev/10.0;
temp1 = lmaxsbdev/10.0;
epsilon = sqrt(pow(10.0, temp0) - 1.0);
epsilonA = sqrt(pow(10.0, temp1) - 1.0);
lfo = (unsigned int)ceil(acosh(epsilonA/epsilon)/
                         acosh(lsfreq/lefreq));
*chfo = lfo;
*chripl = epsilon;
*chcfreq = lefreq*cosh((acosh(1.0/(sqrt(epsilon))))/lfo);
```

If the designer's goal is to design a low pass filter, the normalized low pass filter coefficients generated are frequency and impedance scaled, and the resulting values are those of the reactive elements required to satisfy the design specifications. If the designer wants a high pass filter, then after the normalized filter coefficients have been frequency and impedance scaled, they need to be transformed to generate the reactive element values required to satisfy the design specifications. The function *freqImpScale* calculates the scaled values, and the code snippet shows how this is achieved for a low pass filter:

```
if(lft == 0) /* Low-pass filter */
{
 /* Frequency and Impedance scaling */
 /* Loop over elements of coefficient array */
  for(i = 0; i < MCOL; i++)
  {
   elem = ca[i];
   /* No more filter elements - break loop */
   if(elem == 0) break;

   /* Even indexed coefficient array element is
      a capacitor for a low pass filter */
   if(i % 2 == 0)
   {
    elem /= (PI2*limped*lfreq);
    ca[i] = elem;
 }
   else if(i % 2 !=0)
   {
    elem = ((elem*limped)/(PI2*lfreq));
    ca[i] = elem;
 }
 }
} /* Low pass filter end */
```

For a band pass filter, the computation of the final filter reactive element values is more complicated, because for each reactive element (capacitor, inductor) value of the low pass filter, *a pair* of reactive element values need to be generated to satisfy the design specifications. The function *freqImpScaleBndps* tackles the band pass filter case. The first task is to generate the filter bandwidth and band center frequency as

```
centfreq = PI2*sqrt(lolimf*hilimf);
delta = (PI2*(hilimf - lolimf))/centfreq;
```

The total number of the reactive elements of the band pass filter is double the total number of reactive elements of the low pass filter that is transformed to the

band pass filter. Each shunt capacitor of the low pass filter is transformed to a pair of shunt capacitor and inductor for the band pass filter:

```
elem1 = elem/(delta*centfreq*limped); /*capacitor*/
elem2 = (delta*limped)/(centfreq*elem); /*inductor*/
cabp[j] = elem1;
cabp[j+1] = elem2;
```

Each non-shunt inductor in the low-pass filter is converted to a series pair of capacitor and inductor,

After frequency, impedance scaling and filter transformation (if required) are complete, and the filter is ready, as the values of each of the reactive elements have been determined. The filter may be fabricated, but the designer has no information about the performance characteristics of the filter, specifically frequency and phase response. These performance characteristics can only be determined by a SPICE simulation of the filter, and to do that, the new generated design has to be formatted in the SPICE input netlist format. The process of formatting is the most crucial feature that sets this software package apart from any other available.

The key difference between this tool and others, e.g., Matlab, Mathematica, etc., is that the latter only manipulate mathematical equations, but this tool generates a SPICE input format netlist that any practicing electronics engineer or filter designer is familiar with and most importantly is easily edited/modified to perform the "what-if" type of analysis that is crucial to real-world electronic filter design. By default, SPICE uses ideal reactive element models for simulations, but as desired, these may be replaced by non-ideal capacitors and inductors (thee have parasitic resistors, etc.) for a more realistic simulation—this feature is not available in by any of Matlab, Mathematica, etc.

The functions *genSpiceOutput* and *genSpiceOutputBP* generate text-based SPICE input netlist format for high/low pass and band pass filters, respectively. Both functions exploit a combination of the heuristics (listed above) and features of the topology of ladder networks to achieve this task. For example, the following code snippet shows how the source resistor is added to the SPICE netlist:

```
sprintf(lstr, "%s%d%s%d%s%d%s%f%s",
              "R",0," ",1," ",snode," ",limp,"\n");
strcpy(strArr[3], lstr);
memset(lstr, 0, MCOLE);
```

By convention followed here, the input node of a SPICE format *sub-circuit* (labeled "SUBCKT" in SPICE terminology) is designated the node number 1, and *snode* is the *node to* which *the* source resistor is connected to. In SPICE terminology, any circuit element with a label starting with the letter *R* or *r* is a resistor. The initial value of snode, the resistor value *limp*, is predefined. The above line of code adds the following line to the SPICE netlist file.

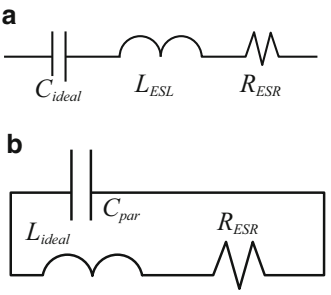

Fig. 3.1 (**a**) Real-world capacitor with parasitic inductor (L_{ESL}) and parasitic resistor (R_{ESR}); (**b**) real-world inductor with parasitic capacitor (C_{par}) and parasitic resistor (R_{ESR})

R0 1 <initial value of snode> <resistor value> with the angle brackets replaced by numerical values.

The formatted character string is stored in an array, and later written to a file, right before the function returns. A hand-generated SPICE input format netlist for a band pass filter has been presented in the design example at the end of the previous chapter.

In all of the previous discussions, the reactive elements used are ideal devices. A real-world capacitor has parasitic resistance and inductance associated with it, as in Fig. 3.1a, b.

The scheme has been implemented as an ANSI C language computer program, compiled with the benchmark open-source *gcc* compiler. The program source code has been developed, compiled, and tested on a computer running Red Hat Fedora 18 Linux operating system. It has been tested on Windows 7 operating system machine under the MingW environment (Minimal GNU for Windows). As an ANSI C language program, it can be compiled and run in the native Windows environment using Microsoft Visual Studio.

The C language program exists in two marginally different versions, to provide full freedom to the designer as to how it might be used. Both are invoked from the command line with appropriate arguments from inside a shell window. The simplified version accepts as input, e.g., a low pass filter, the filter name (Bessel or Butterworth or Chebyshev), the type (low pass), the cutoff frequency (in hertz), the filter order (maximum 10), and the pass band ripple (only for a Chebyshev type I) and generates as output the text SPICE input format netlist. This text file is edited (to add signal source and type of analysis—small signal or AC for filter frequency/phase response) and then simulated with any SPICE simulator.

The full-blown version of the C program would accept, for the same low pass filter, the pass band edge frequency, the stop band start frequency, the maximum pass band attenuation, and the maximum stop band attenuation. The filter order and cutoff frequency are calculated and used for the remainder of the computations. The designer cannot specify the cutoff frequency or the filter order.

A band pass filter can be considered a stand-alone device or a series combination of a low pass filter and a high pass filter. The low pass filter cutoff frequency is the upper limit of the pass band of target band pass filter, and the high pass filter cutoff frequency is the lower limit of the pass band of the target band pass filter. The full-blown version supports both, the simpler version only the stand-alone case. Extensive help files guide the user.

3.3 Designing Filters with New Scheme

The C language implementation of the scheme exists in two versions. The simplified implementation accepts the filter order and cutoff frequency (high/low pass filter), and the lower/upper pass band frequency limits (band pass filter) and generates the SPICE input format netlist. The full-blown version *does not* accept the filter order or cutoff frequency, but computes the quantities from the input values of pass band edge frequency, the stop band start frequency, and maximum pass/stop band attenuation values. Both C language implementations are executed from the command line, and exhaustive help features guide the user. A GUI-driven version is under development. Although developed on Linux operating system-based computers, both versions can be compiled and executed in the Windows operating system environment. The beginner, trying to understand how to run the simplified implementation simply, has to type in the executable name at the Linux shell window command prompt and hit "Return" as

```
./microwavefilt
```

This immediately prints out the help details in the shell window as

```
Incorrect or insufficient arguments ...
Please check usage ...
For help ...
./microwavefilt -h|-H|-help|-HELP
To run ...
./microwavefilt <filter name(be|bw|ca|ch)>
<filter order(1|2|3|4|5|6|7|8|9|10>
<cut-off frequency(Hz)>
<source/load impedance(Ohms)>
<pass/stop band ripple factor(dB)>
<filter type<h|l|b>
<band pass low limit(Hz)>
<band pass high limit(Hz)>
BESSEL filter inductor first <y/n>
NON-ZERO cut-off frequency ONLY for high or low pass
NON-ZERO pass/stop band ripple ONLY for Cauer and Chebyshev filters
NON-ZERO low/high pass band limit frequency  ONLY for band pass filter
ONLY 0.5dB and 5dB equiripple Cauer filters are supported
```

Units for the various input variables are indicated in parentheses; the user *does not* have to add any units explicitly. In the subsequent discussions of the design examples, all instructions or data typed in by the user at the command prompt is shown in bold font.

3.4 Seventh-Order Low Pass Butterworth Filter: Simplified Scheme Implementation

To design a seventh-order 25 MHz cutoff frequency low pass Butterworth filter with the simplified scheme implementation, the user types in at the command prompt and hits "Return":

```
./microwavefilt bw 7 25000000 50 0 1 0 0 n
```

- The first argument is the filter name Butterworth.
- The second argument is the filter order 7.
- Cutoff frequency is the third argument 25,000,000.
- Source and load resistance is 50 Ω (fourth argument)—maximally matched filter.
- Pass band ripple is zero (frequency response of the Butterworth filter is theoretically maximally flat in the pass band).
- Filter type is low pass ("l").
- The pass band start and end frequencies are both zero (this is a low pass filter).
- Bessel filter "inductor first"; option is set to *n* (NO) as this is a Butterworth filter.

During execution, the program prints out some intermediate values; the useful information is the SPICE netlist input netlist text-based output file name:

```
Filter order 7
SPICE netlist file name bw7O50250000001.cir
```

bw7O50250000000l indicates a Butterworth (bw), seventh order (7O), 50 Ω source/load resistance (50), and 25 MHz cutoff frequency.

(25000000) low pass filter (l). The file name extension (".cir") is a common text-based SPICE netlist file name extension.

The actual SPICE input format netlist generated is

```
.SUBCKT bw7OLP 1 6
* IN
* OUT
R0 1 3 50.00
C0 3 0 9.018117e-12
L1 3 4 6.317748e-08
C2 4 0 3.649804e-11
L3 4 5 1.013272e-07
```

```
C4 5 0 3.649804e-11
L5 5 6 6.317748e-08
C6 6 0 9.018117e-12
R1 6 0 5.000000e+01
.ENDS
```

The SPICE format sub-circuit (.SUBCKT) named bw7OLP to *this netlist has the correct topology of a ladder network-based low pass filter*, *with each capacitor having one terminal connected to ground and inductors interconnecting capacitors.* This text-based SPICE netlist file has to be edited to add both the signal source and analysis details, to generate the finalized SPICE input netlist, which can then be simulated with any SPICE simulator (in this case the latest NgSpice version—NgSpice 2.6):

```
.SUBCKT bw7OLP 1 6
* IN
* OUT
R0 1 3 50.00
C0 3 0 9.018117e-12
L1 3 4 6.317748e-08
C2 4 0 3.649804e-11
L3 4 5 1.013272e-07
C4 5 0 3.649804e-11
L5 5 6 6.317748e-08
C6 6 0 9.018117e-12
R1 6 0 5.000000e+01
.ENDS

* SIGNAL SOURCE WITH DC OFFSET OF 0.0001 VOLTS
* AC SIGNAL AMPLITUDE
VS 1 0 DC 0.0001 AC 5
XBW 1 2 bw7OLP

* ANALYSIS OPTIONS
* NUMERICAL INTEGRATION BY GEAR METHOD
.OPTIONS NOPAGE METHOD = GEAR
* AC DECADE ANALYSIS WITH 20000 DATA POINTS
* START FREQUENCY 10 MHz END FREQUENCY 50 MHz
.AC DEC 20000 10000000 50000000
* PRINT AC OUTPUT AT CIRCUIT NODE 2
.PRINT AC V(2)
.END
```

The NgSpice is invoked from the command line as

```
ngspice -b bw7O50250000001.cir > out
```

It is recommended that the reader refer to his/her own SPICE distribution's user manual to execute the netlist on his/her own SPICE simulator. The text-based output is "piped" to a text file out, whose contents look like

```
No. of Data Rows : 20000
AC Analysis Sat Apr 2 15:48:26 2016
--------------------------------------------------------------------
Index  frequency    v(2)
--------------------------------------------------------------------
0    1.000000e+07  2.398317e+00,  -7.05744e-01
1    1.000200e+07  2.398277e+00,  -7.05881e-01
2    1.000400e+07  2.398236e+00,  -7.06019e-01
3    1.000600e+07  2.398196e+00,  -7.06156e-01
4    1.000800e+07  2.398155e+00,  -7.06294e-01
5    1.001000e+07  2.398115e+00,  -7.06431e-01
6    1.001200e+07  2.398074e+00,  -7.06568e-01
7    1.001400e+07  2.398034e+00,  -7.06706e-01
8    1.001600e+07  2.397993e+00,  -7.06843e-01
9    1.001800e+07  2.397953e+00,  -7.06981e-01
10   1.002000e+07  2.397912e+00,  -7.07118e-01
11   1.002200e+07  2.397872e+00,  -7.07255e-01
12   1.002400e+07  2.397831e+00,  -7.07393e-01
.........
.........
```

Here, the third column contains the frequency amplitude and the fourth the phase.

The frequency and phase response plots for the seventh-order low pass Butterworth filter are shown in Fig. 3.2a, b.

3.5 Seventh-Order Low Pass Chebyshev Filter: Simplified Scheme Implementation

To demonstrate the sharp roll-off characteristics of the Chebyshev filter (as compared to the Butterworth filter), the seventh-order low pass filter is now examined as a Chebyshev type I filter with nonzero pass band ripple parameter set at 0.75 dB. The user types in the following command line arguments at the command prompt and hits "Return":

```
./microwavefilt ch 7 25000000 50 0.75 l 0 0 n
```

Clearly, the pass band ripple parameter is not zero, as in the case of Butterworth. The program generates the SPICE netlist input format text file as

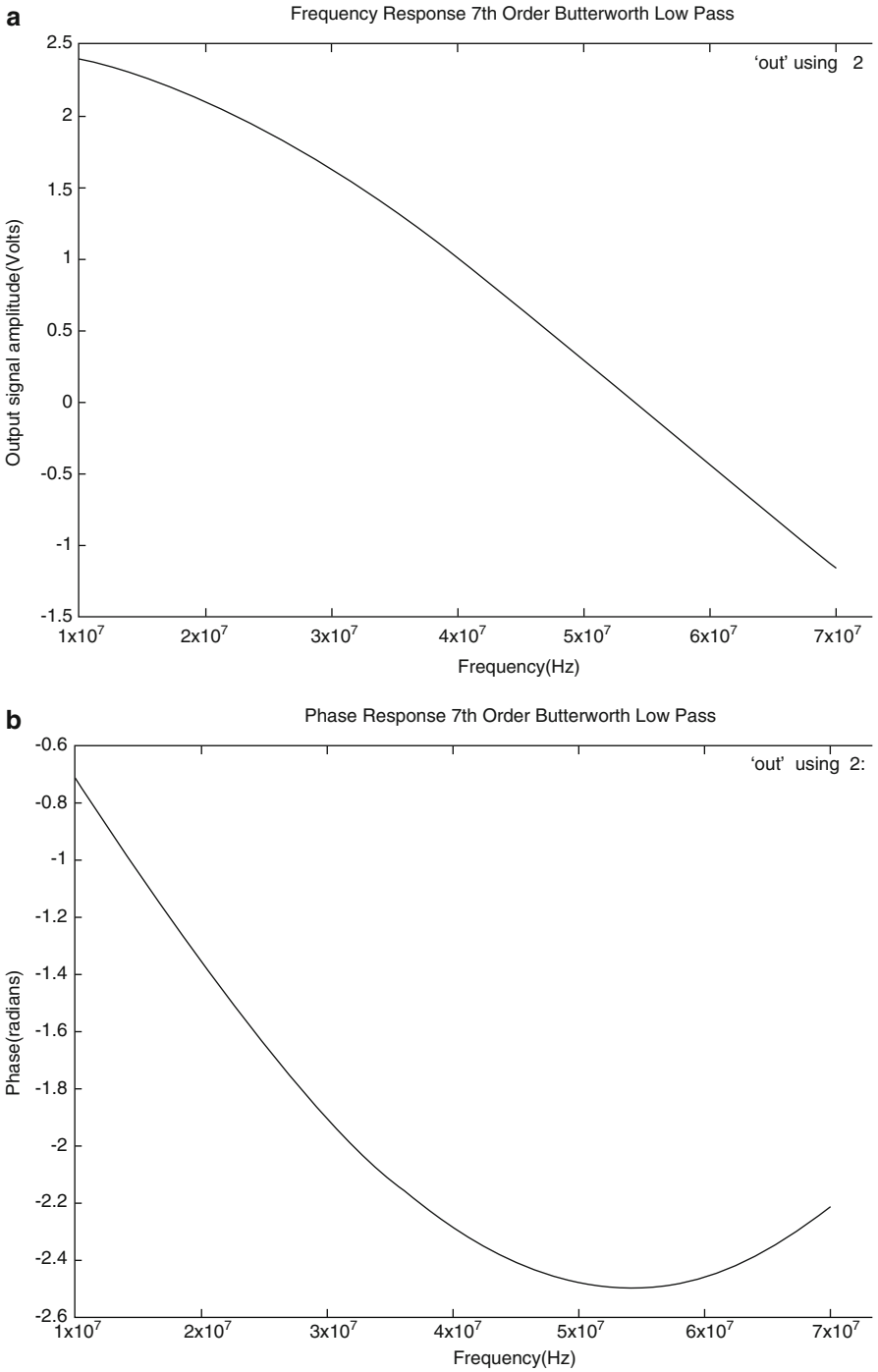

Fig. 3.2 (**a**) Seventh-order Butterworth low pass filter frequency response; (**b**) seventh-order low pass Butterworth filter phase response

```
Filter order 7
SPICE netlist file name ch7O50250000001.cir
```

The netlist file name indicates that it is a seventh-order low pass Chebyshev filter with cutoff frequency of 25 MHz. *Clearly this netlist is perfect with the ladder network structure of a low pass filter, with each capacitor having one terminal connected to ground and inductors interconnecting capacitors*. The SPICE input format netlist is

```
.SUBCKT ch7OLP 1 6
* IN
* OUT
R0 1 3 50.00
C0 3 0 3.983240e-11
L1 3 4 2.790506e-07
C2 4 0 3.452574e-11
L3 4 5 1.617878e-07
C4 5 0 2.282561e-11
L5 5 6 1.008051e-07
C6 6 0 8.516703e-12
R1 6 0 5.000000e+01
.ENDS
```

This text-based netlist file is edited to add the signal source and analysis type as

```
.SUBCKT ch7OLP 1 6
* IN
* OUT
R0 1 3 50.00
C0 3 0 3.983240e-11
L1 3 4 2.790506e-07
C2 4 0 3.452574e-11
L3 4 5 1.617878e-07
C4 5 0 2.282561e-11
L5 5 6 1.008051e-07
C6 6 0 8.516703e-12
R1 6 0 5.000000e+01
.ENDS

* SIGNAL SOURCE WITH DC OFFSET OF 0.0001 VOLTS
VS 1 0 DC 0.0001 AC 5
XCH 1 2 ch7OLP

* ANALYSIS OPTIONS GEAR METHOD
.OPTIONS NOPAGE METHOD=GEAR
* AC DECADE ANALYSIS WITH START FREQUENCY OF 10 MHz
* AND END FREQUENCY OF 50 MHz
```

```
.AC DEC 20000 10000000 50000000
* PRINT AC ANALYSIS VOLTAGE, PHASE AT NODE 2
.PRINT AC V(2)
.END
```

The NgSpice simulator is invoked in the *batch* mode (−d option) as follows, and the output is "piped" to another text file named "out":

```
ngspice -b ch7O50250000001.cir > out
```

The frequency and phase response of this filter is shown in Fig. 3.3a, b, respectively.

The sharp roll-off characteristics of the low pass Chebyshev filter, as compared to the low pass Butterworth filter of the same order, are clear from examination of Figs. 3.2a and 3.3a. The price the designer pays is the nonzero pass band ripple in the Chebyshev case.

3.6 Eighth-Order High Pass Bessel Filter: Simplified Scheme Implementation

An eighth-order high pass Bessel filter with a cutoff frequency of 50 MHz is now examined. Theoretically, this filter has a maximally flat pass and stop band. The C program is invoked with the following command line arguments:

```
./microwavefilt be 8 50000000 50 0 h 0 0 n
```

The "inductor first" option for Bessel filter is not used, and the source load resistance is set to 50 Ω. The pass/stop band ripple parameter is set to zero. The final SPICE netlist input format text file is be8O5050000000h.cir:

```
Filter order 8
SPICE netlist file name be8O5050000000h.cir
```

As the output file name indicates, the Bessel filter order is eight, the source and load resistances are both 50 Ω (maximally matched), the cutoff frequency is 50 MHz, and it is a high pass filter. *Clearly this netlist confirms with the ladder network topology of a high pass filter, with each inductor having one terminal connected to ground and capacitors interconnecting inductors.* The generated SPICE netlist is

```
.SUBCKT be8OHP 1 7
* IN
* OUT
```

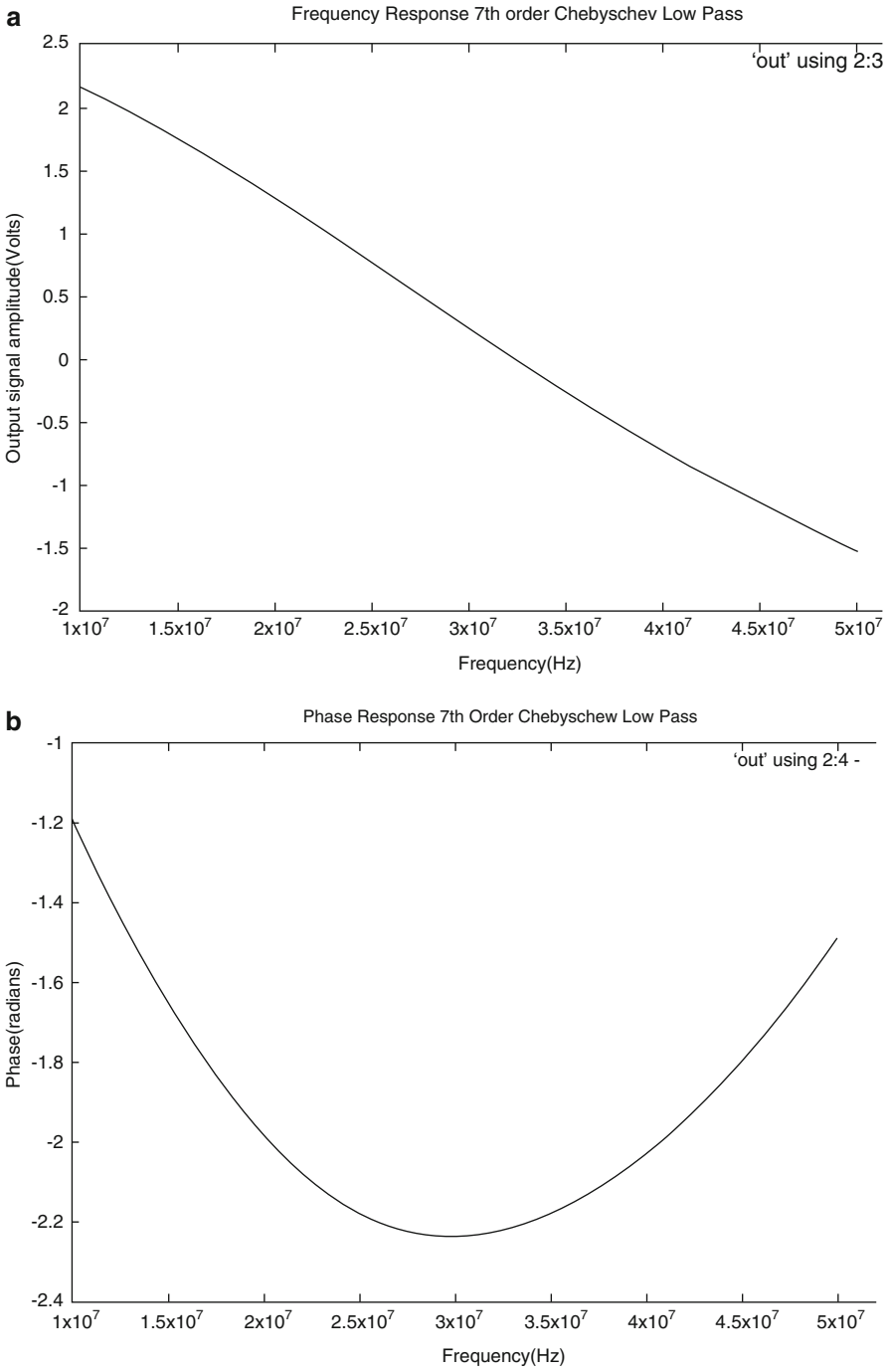

Fig. 3.3 (**a**) Seventh-order low pass Chebyshev filter frequency response; (**b**) seventh-order low pass Chebyshev filter phase response

```
R0 1 3 50.00
L0 3 0 2.756452e-07
C1 3 4 3.726633e-11
L2 4 0 5.745473e-08
C3 4 5 1.706994e-11
L4 5 0 3.426456e-08
C5 5 6 1.165350e-11
L6 6 0 2.312139e-08
C7 6 7 4.472421e-12
R1 7 0 50.000000
.ENDS
```

The netlist file is edited to add signal source and analysis type to yield

```
.SUBCKT be8OHP 1 7
* IN
* OUT
R0 1 3 50.00
L0 3 0 2.756452e-07
C1 3 4 3.726633e-11
L2 4 0 5.745473e-08
C3 4 5 1.706994e-11
L4 5 0 3.426456e-08
C5 5 6 1.165350e-11
L6 6 0 2.312139e-08
C7 6 7 4.472421e-12
R1 7 0 50.000000
.ENDS

* SIGNAL SOURCE HAS A SMALL DC OFFSET OF 0.0001 VOLTS
VS 1 0 DC 0.0001 AC 5
XBE 1 2 be8OHP

* ANALYSIS OPTIONS GEAR METHOD
.OPTIONS NOPAGE METHOD=GEAR
* AC DECADE ANALYSIS WITH START FREQUENCY OF 40 MHz
* AND END FREQUENCY OF 170 MHz
.AC DEC 20000 40000000 170000000
* PRINT AC ANALYSIS VOLTAGE, PHASE AT NODE 2
.PRINT AC V(2)
.END
```

The NgSpice simulator is invoked in the batch mode as

```
ngspice -b be8O5050000000h.cir > out
```

The frequency and phase responses, respectively, of this filter are in Figs. 3.4a and 3.3b.

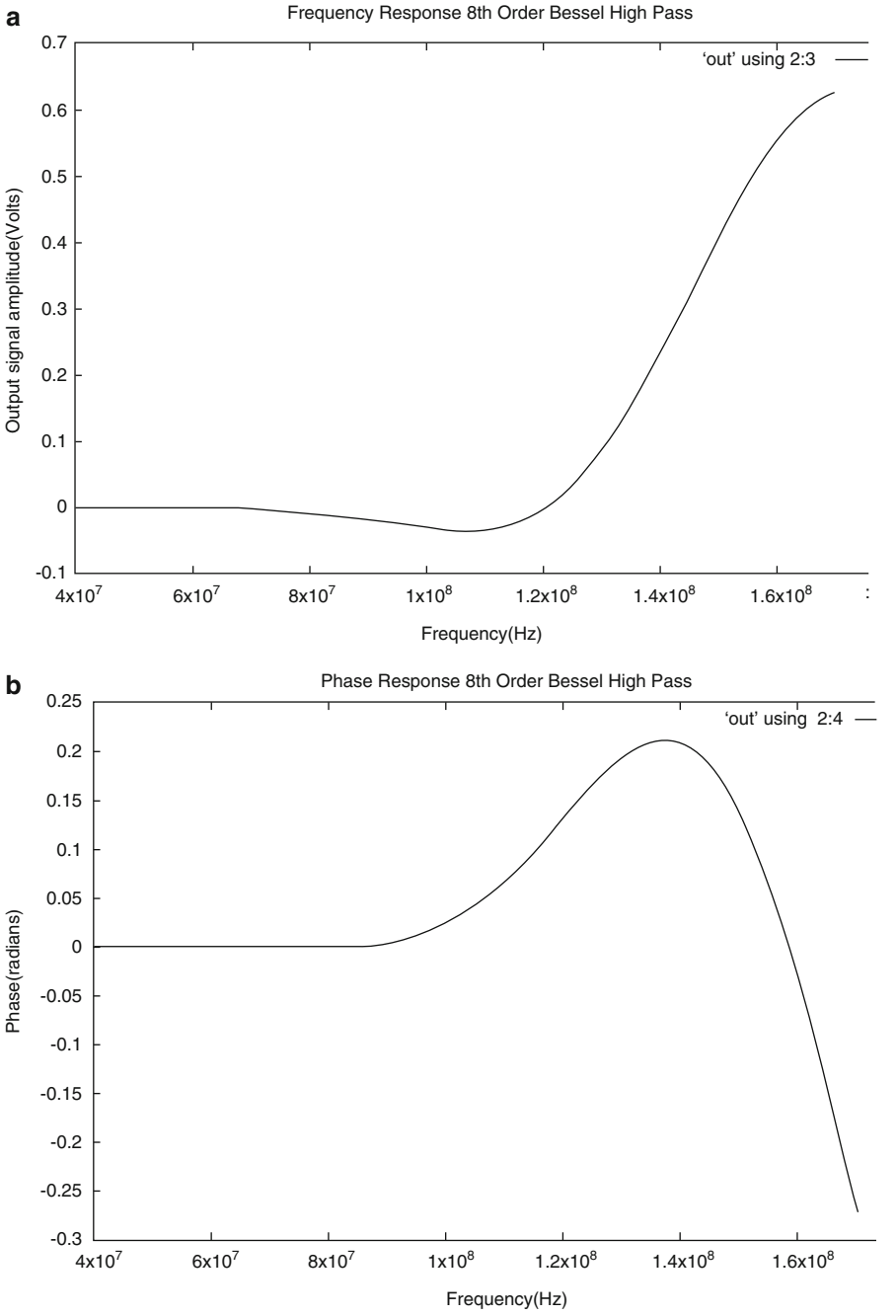

Fig. 3.4 (**a**) Eighth-order high pass Bessel filter frequency response; (**b**) eighth-order high pass Bessel filter phase response

3.7 Eighth-Order Band Pass Chebyshev Filter: Simplified Scheme Implementation

A Chebyshev eighth-order band pass filter is analyzed in detail. The C program implementing the simplified version of this scheme is invoked with the command line arguments:

```
./microwavefilt ch 8 0 50 0.5 b 50000000 60000000 n
```

The filter order is eight, the cutoff frequency is zero, the source and load resistances are both 50 Ω (maximally matched), the filter type is band pass, the lower limit of the pass band is 50 MHz, and the upper limit of the pass band is 60 MHz. The Bessel filter "inductor first" option is irrelevant in this case and is set to "NO":

```
Filter order 8
SPICE netlist file name BP60000000500000050.cir
```

As the text SPICE netlist input file name indicates, the netlist is that of a band pass filter ("BP") with upper pass band limit of 60 MHz, lower pass limit of 50 MHz, and a maximally matched source and load resistance of 50 Ω.

Clearly this netlist is perfect with the ladder network structure of a band pass filter, with alternate shunt pair of capacitor and inductor and series pair of capacitor and inductor. Each series pair interconnects two shunt pairs. The generated SPICE input netlist is

```
.SUBCKT BP344134082 1 7
* IN
* OUT
R0 1 3 50.000000
C0 3 0 2.436416e-10
L1 3 0 3.465718e-08
C2 3 4 5.742311e-12
L3 4 5 1.470476e-06
C4 5 0 5.881904e-10
L5 5 0 1.435578e-08
C6 5 6 1.350986e-11
L7 6 7 6.250199e-07
R1 7 0 50.000000
.ENDS
```

The SPICE input netlist file, after addition of the signal source and analysis technique, appears as

```
.SUBCKT BP344134082 1 7
* IN
* OUT
R0 1 3 50.000000
C0 3 0 2.436416e-10
L1 3 0 3.465718e-08
C2 3 4 5.742311e-12
L3 4 5 1.470476e-06
C4 5 0 5.881904e-10
L5 5 0 1.435578e-08
C6 5 6 1.350986e-11
L7 6 7 6.250199e-07
R1 7 0 50.000000
.ENDS

* THE SIGNAL SOURCE HAS A DC OFFSET OF 0.00001 VOLTS
VS 1 0 DC 0.00001 AC 5
XBP 1 2 BP344134082

* ANALYSIS OPTION METHOD IS GEAR
.OPTIONS NOPAGE METHOD=GEAR
* AC DECADE ANALYSIS, WITH START FREQUENCY OF 40 MHz
* AND END FREQUENCY OF 80 MHz
.AC DEC 20000 40000000 80000000

* PRINT AC ANALYSIS FREQUENCY AND PHASE
* RESPONSE AT NODE 2
.PRINT AC V(2)
.END
```

The frequency and phase response plots for this band pass filter are in Fig. 3.5a, b, respectively.

In the detailed analysis of the filters above, the reactive components are all ideal, that is, a capacitor does not have any extra series resistance or extra series inductance. Identical arguments hold for an ideal inductor, which does not have a parallel parasitic capacitance or an extra series resistance. This issue will be addressed in a future version of the full-blown C language version of the scheme implementation.

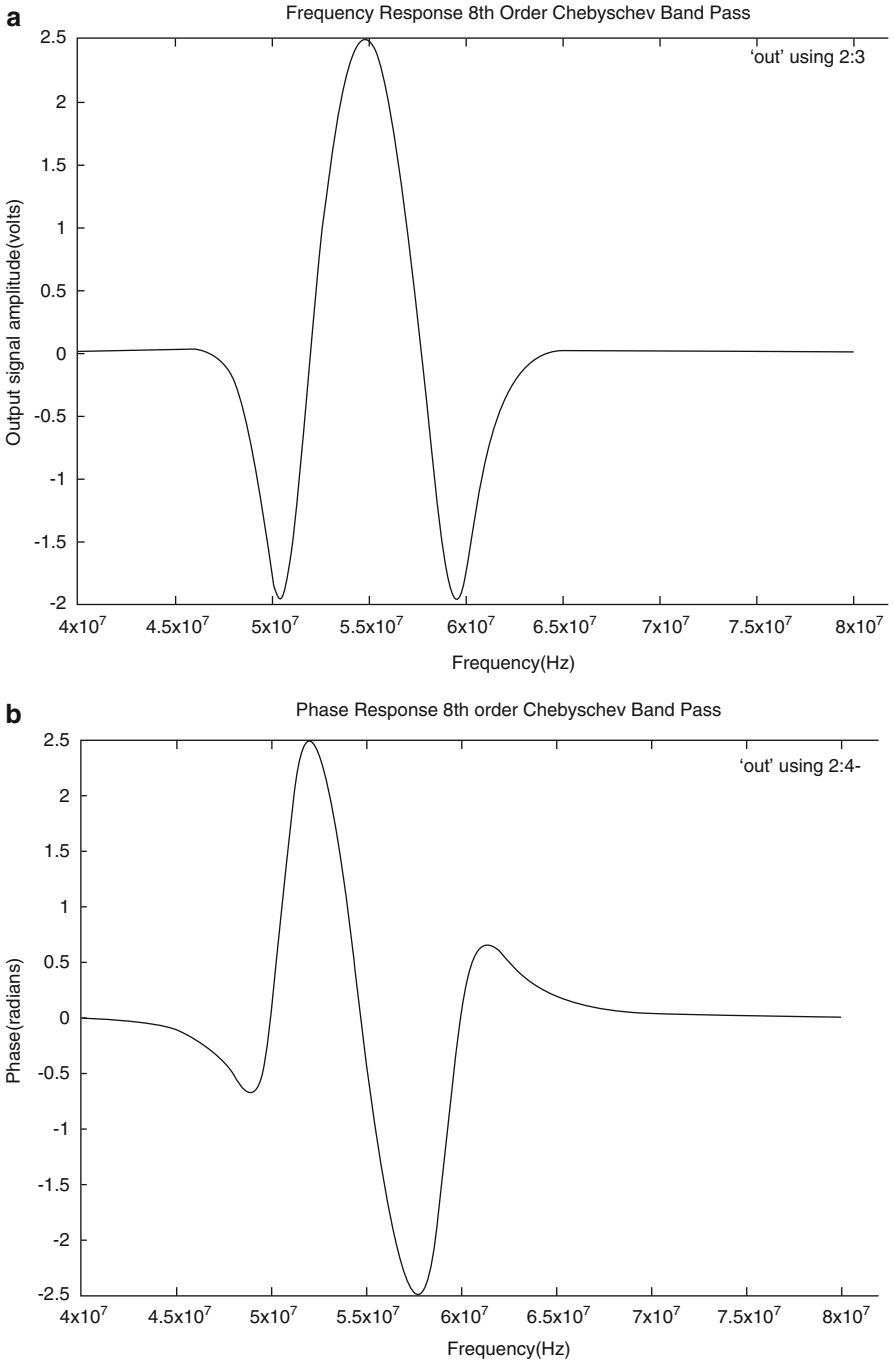

Fig. 3.5 (**a**) Eighth-order Chebyshev band pass filter frequency response; (**b**) eight-order Chebyshev band pass filter phase response

3.8 Designing Filters with New Scheme: Full-Blown Implementation

Although the full-blown C language implementation of the scheme is still under development, enhancement, and testing, it can be used. The main difference between the full-blown and simplified implementations is that now the designer *cannot specify* the filter order or cutoff frequency, as in the case of the simplified version. Now the designer specifies the following four parameters:

- Pass band edge frequency in hertz
- Stop band start frequency in hertz
- Maximum pass band attenuation in dB
- Maximum stop band attenuation in dB

The C language program then calculates the minimum filter order and cutoff frequency for the corresponding low pass filter using formulas discussed in Chap. 2, for each of Bessel, Butterworth, and Chebyshev types. As before, the normalized filter tables and filter transformations are used to generate the SPICE input format netlist that satisfies the designer's specifications. As before, the C language full-blown scheme implementation has been implemented on a Linux operating system-based computer, but the executable can be run on any Windows operating system-based machine, under the MingW environment. As an ANSI C language program, it can be compiled in the native Windows environment using Microsoft Visual Studio. The beginner can start to know how to use the program by typing in at the command line prompt:

```
./microwavefiltnew
```

The following help information is then displayed on the screen:

```
No filter specifications provided
Supported filter types ...
Bessel, Butterworth, Chebyshev band/high/low pass
Butterworth band|high|low pass
./microwavefiltnew bw|BW b|Blh|H|l|L
Bessel band|high|low pass
./microwavefiltnew b|Blh|H|l|L
Chebyshev band|high}low pass
./microwavefiltnew b|Blh|H|l|L
Alternative band pass filters ..
./microwavefiltnew <b|B> <bw|BW|ch|CH>
<max attenuation pass band high pass(dB)>
<max attenuation stop band high pass(dB)>
<max attenuation pass band low pass(dB)>
<max attenuation stop band low pass(dB)>
```

```
>pass band edge frequency high pass (Hz)>
<stop band start frequency high pass(Hz)>
<pass band edge frequency low pass (Hz)>
<stop band start frequency low pass(Hz)>
<source/load load impedance(Ohms)>
```

To get information about how to design a Butterworth high or low pass filter, the user types in at the command prompt:

```
./microwavefiltnew bw l
```

This generates the following help information on the screen:

```
Insufficient arguments - Butterworths low pass filter .. usage

./microwavefiltnew <be|BE|bw|BW|ch|CH> <l|L|h|H>
<max pass band attenuation(dB)>
<max stop band attenuation(dB)>
<pass band edge frequency(Hz)>
<approximate stop band start frequency(Hz)>
<source/load impedance(Ohms)>
<Non-ideal or ideal components(y/n)>
For band pass filters ..
./microwavefiltnew <be|BE|bw|BW|ch|CH> <b|B>
<Filter prder>
<pass band low limit(Hz)>
<pass band hogh limit(Hz)>
<source load impedance(Ohms)>
<pass band ripple parameter(dB)>
pass band ripple is ideally zero for Bessel and Butterworth
```

The alternative full-blown technique (series pair of high and low pass filters) to design a band pass filter will be discussed in detail in a later section.

3.9 Butterworth Low Pass Filter: Calculated Order 10 and Cutoff Frequency 22.7 MHz

In Sect. 3.3, a 25 MHz cutoff frequency low pass Butterworth filter was designed. As now the filter order or cutoff frequency cannot be explicitly specified; using the above help information, the designer specifies a Butterworth low pass filter as

```
./microwavefiltnew bw l 0.5 5 19000000 25000000 50 n
```

The last argument in the list, *n*, indicates non-ideal reactive components are **not** being used. *Here*, *the key assumption is that the cutoff frequency is greater than the pass band edge frequency and smaller than the stop band start frequency*. The maximum pass band attenuation is 0.5 dB, the maximum stop band attenuation is 5 dB, the pass band edge frequency is 19 MHz, and the stop band start frequency is 25 MHz. The program calculates the cutoff frequency and filter order as

```
Cutoff 2.279433e+07 Hz filter order 10
```

Some intermediate results are printed (omitted here) and the text SPICE netlist input format file name is *bw10OLP.cir*. The name indicates that it is a Butterworth tenth-order low pass filter. The SPICE input format netlist is

```
.SUBCKT BW10OLP 1 8
* IN
* OUT
R0 1 3 50.00
C0 3 0 4.369166e-11
L1 3 4 3.169953e-07
C2 4 0 1.974927e-10
L3 4 5 6.221384e-07
C4 5 0 2.758583e-10
L5 5 6 6.896457e-07
C6 6 0 2.488554e-10
L7 6 7 4.937318e-07
C8 7 0 1.267981e-10
L9 7 8 1.092292e-07
R1 8 0 50.000000
.ENDS
```

Clearly, the netlist confirms to the ladder network format low pass filter structure with each capacitor having one terminal grounded, and inductors interconnect capacitors. The file is edited to add the signal source and analysis type:

```
.SUBCKT BW10OLP 1 8
* IN
* OUT
R0 1 3 50.00
C0 3 0 4.369166e-11
L1 3 4 3.169953e-07
C2 4 0 1.974927e-10
L3 4 5 6.221384e-07
C4 5 0 2.758583e-10
L5 5 6 6.896457e-07
C6 6 0 2.488554e-10
L7 6 7 4.937318e-07
```

```
C8 7 0 1.267981e-10
L9 7 8 1.092292e-07
R1 8 0 50.000000
.ENDS

* SIGNAL SOURCE HAS DC OFFSET OF 0.0001 VOLTS
VS 1 0 DC 0.0001 AC 5
XBW 1 2 BW10OLP

* ANALYSIS OPTION - COMPUTATION METHOD IS GEAR
.OPTIONS NOPAGE METHOD=GEAR
* AC DECADE ANALYSIS MODE WITH START FREQUENCY
* 18 MHZ AND END FREQUENCY 30 MHz
.AC DEC 20000 18000000 30000000
* PRINT AC FREQUENCY AND PHASE RESPONSE AT NODE 2
.PRINT AC V(2)
.END
```

The SPICE netlist is simulated with the command line:

```
ngspice -b bw10OLP.cir > out
```

The frequency and phase response plots are in Fig. 3.6a, b, respectively.

3.10 Chebyshev High Pass Filter: Calculated Order 3 Cutoff Frequency 21 MHz Pass Band Ripple 0.45 dB

The values of maximum pass/stop band attenuation, pass band edge, and stop band start frequencies used to the tenth-order Butterworth low pass filter are now used to design a Chebyshev type I high pass filter:

```
./microwavefiltnew ch h 0.8 3 20000000 25000000 50 n
```

The last argument in the list, *n*, indicates non-ideal reactive components are **not** being used. The assumption that the cutoff frequency is higher in value than the pass band edge frequency value, and less than the stop band start frequency value, is also valid here. The C language program computes the filter order, cutoff frequency, and pass band ripple value:

```
Pass band ripple factor 0.449738 filter order 3 Cutoff frequency
21020786.347498Hz
```

The SPICE input format netlist is in the text file:
ch3OHP.cir

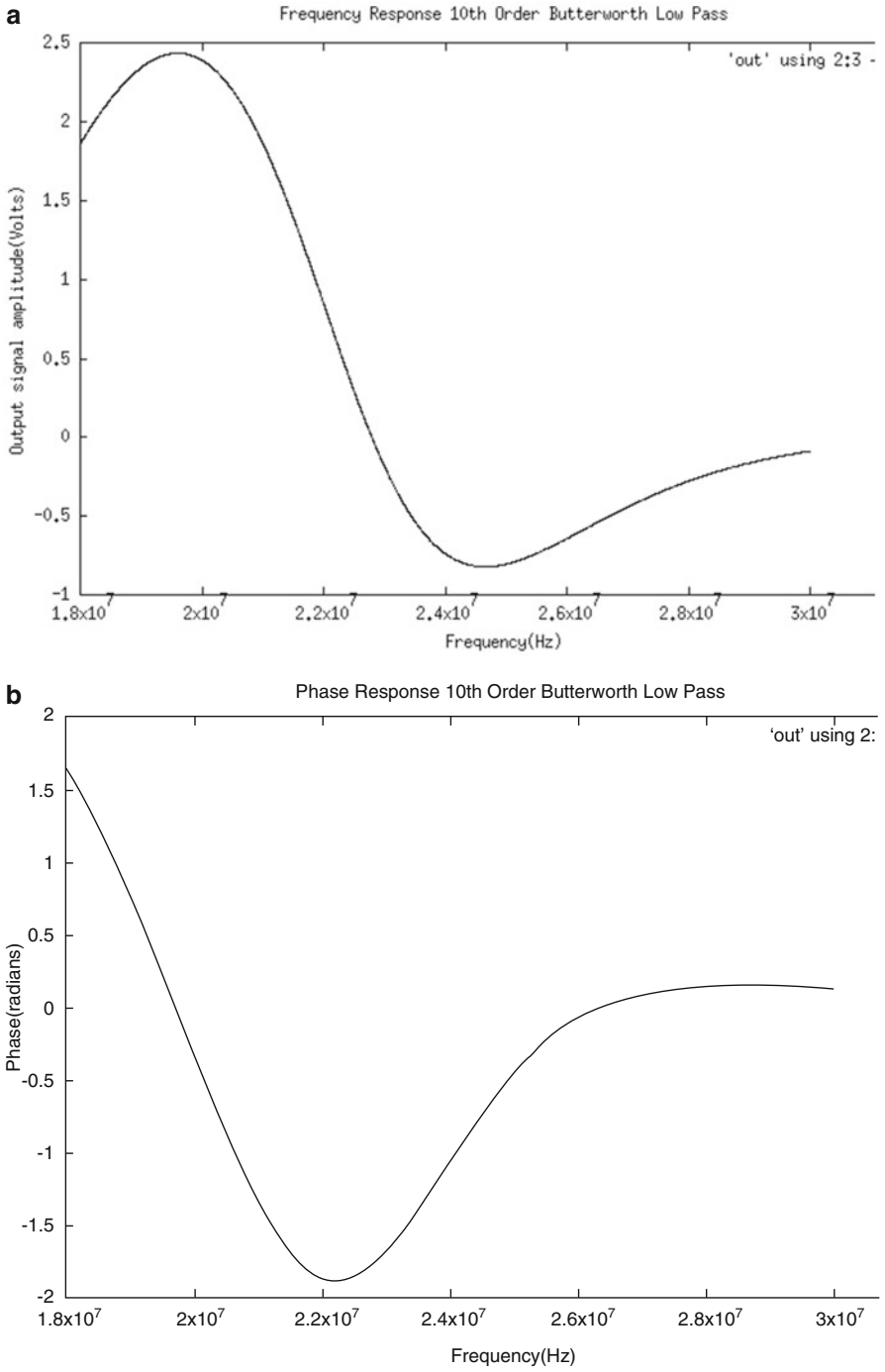

Fig. 3.6 (**a**) Tenth-order low pass Butterworth filter frequency response; (**b**) tenth-order low pass Butterworth filter phase response

The file name indicates that it is a third-order high pass Chebyshev filter. The SPICE input format netlist is

```
.SUBCKT CH3OHP 1 4
* IN
* OUT
R0 1 3 50.00
L0 3 0 3.785768e-07
C1 3 4 7.571536e-11
L2 4 0 3.785768e-07
R1 4 0 50.000000
.ENDS
```

The text netlist file, after editing to add the signal source and type of analysis, appears as

```
.SUBCKT CH3OHP 1 4
* IN
* OUT
R0 1 3 50.00
L0 3 0 3.785768e-07
C1 3 4 7.571536e-11
L2 4 0 3.785768e-07
R1 4 0 50.000000
.ENDS

* SIGNAL SOURCE WITH 0.0001 VOLT DC OFFSET
VS 1 0 DC 0.0001 AC 5

XCH 1 2 CH3OHP

* ANALYSIS OPTION NUMERICAL METHOD IS GEAR
.OPTIONS NOPAGE METHOD=GEAR
* AC LINEAR ANALYSIS WITH 20000 MAXIMUM DATA POINTS
* START FREQUENCY IS 15 MHz, END FREQUENCY IS 40 MHz
.AC LIN 20000 15000000 40000000
* PRINT FREQUENCY AND PHASE RESPONSE AT NODE 2
.PRINT AC V(2)
.END
```

The NgSpice simulator is invoked as

```
ngspice -b ch3OHP.cir > out
```

The frequency and phase response plots for this high pass filter are in Fig. 3.7a, b, respectively.

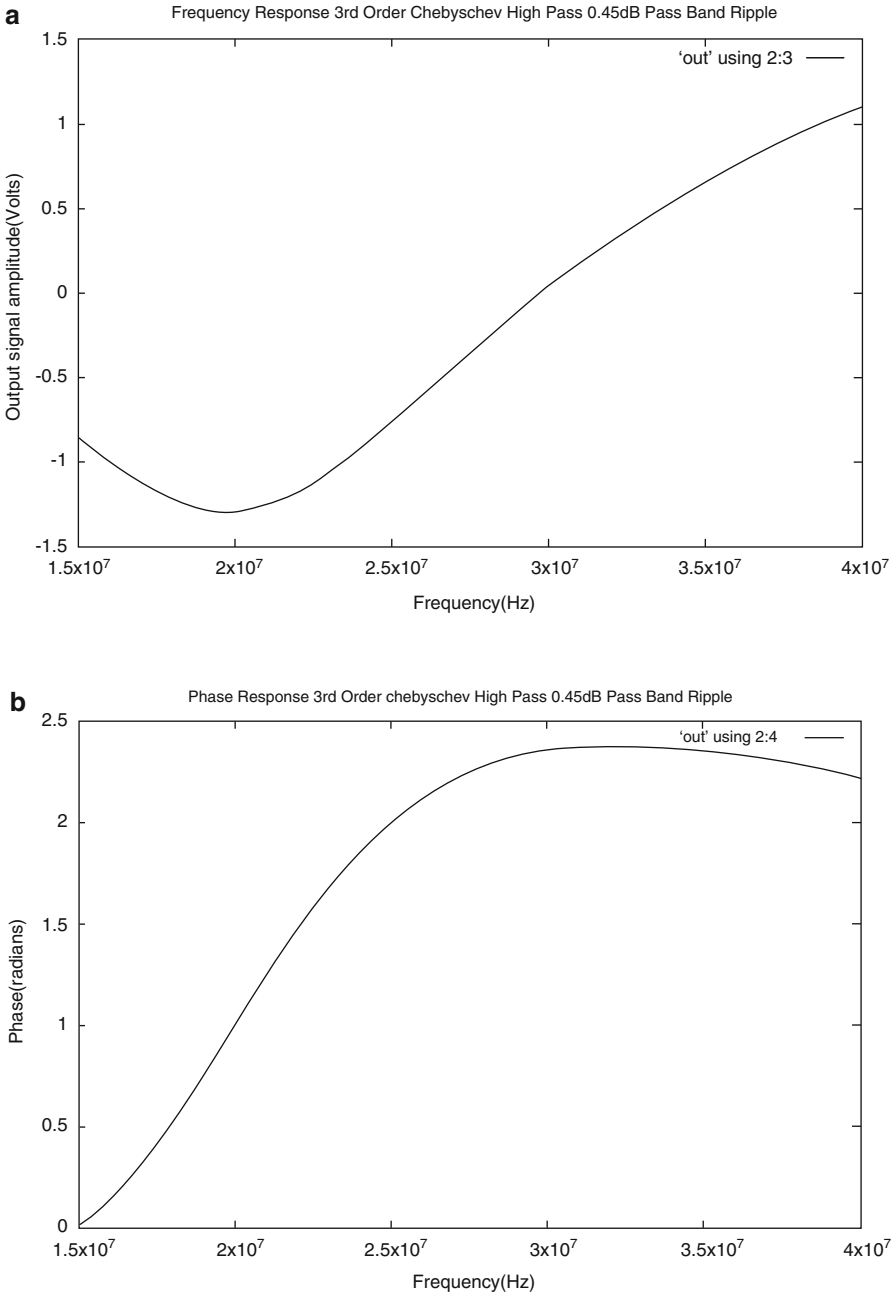

Fig. 3.7 (**a**) Third-order high pass Chebyshev filter frequency response; (**b**) third-order high pass Chebyshev filter phase response

3.11 Chebyshev Band Pass Filter: Series Connection of High Pass and Low Pass Filters

In Sect. 3.8, the general help options for the full-blown scheme implementation state that the command line arguments to be used to design a band pass filter as a series connection of a high pass and a low pass filter are

```
Alternative band pass filters ..
./microwavefiltnew <b|B> <bw|BW|ch|CH>
<max attenuation pass band high pass(dB)>
<max attenuation stop band high pass(dB)>
<max attenuation pass band low pass(dB)>
<max attenuation stop band low pass(dB)>
<pass band edge frequency high pass (Hz)>
<stop band start frequency high pass(Hz)>
<pass band edge frequency low pass (Hz)>
<stop band start frequency low pass(Hz)>
<source/load load impedance(Ohms)>
```

The existing development version of the full-blown implementation of the scheme supports this option for designing only Butterworth and Chebyshev (type I only) band pass filters. So, the following command line arguments are used at the command prompt:

```
./microwavefiltnew b ch 0.6 5 0.65 5 50000000 60000000 75000000 85000000 50
```

Chebyshev type I filters are selected for analysis, as pass band ripple parameter is crucial for this type, unlike the Butterworth filter. The C language program generates a fourth-order low pass and a fourth-order high pass filter for the series combination:

```
Pass band ripple factor 0.384907 filter order 4 Cutoff frequency
51753847.324408Hz
Pass band ripple factor 0.401807 filter order 4 Cutoff frequency
77494435.373705Hz
```

The two SPICE input format netlists are in the text file:
chBP.cir
The SPICE input format netlists for the two filters are

```
.SUBCKT CH4OHP 1 5
* IN
* OUT
R0 1 3 50.00
```

```
L0 3 0 9.932826e-08
C1 3 4 1.645759e-11
L2 4 0 1.252309e-07
C3 4 5 6.475941e-11
R1 5 0 50.000000
.ENDS

.SUBCKT CH4OLP 1 5
* IN
* OUT
R0 1 3 50.00
C0 3 0 6.436754e-11
L1 3 4 3.884843e-07
C2 4 0 5.022269e-11
L3 4 5 9.841200e-08
R1 5 0 50.000000
.ENDS
```

This SPICE input format netlist text file, after adding the signal source and analysis details, appears as

```
.SUBCKT CH4OHP 1 5
* IN
* OUT
R0 1 3 50.00
L0 3 0 9.932826e-08
C1 3 4 1.645759e-11
L2 4 0 1.252309e-07
C3 4 5 6.475941e-11
R1 5 0 50.000000
.ENDS

.SUBCKT CH4OLP 1 5
* IN
* OUT
R0 1 3 50.00
C0 3 0 6.436754e-11
L1 3 4 3.884843e-07
C2 4 0 5.022269e-11
L3 4 5 9.841200e-08
R1 5 0 50.000000
.ENDS

* SIBNAL SOURCE WITH 0.0001 VOLT DC OFFSET
VS 1 0 DC 0.0001 AC 5
* LOW - HIGH PASS FILTER CONNECTION ORDER
```

```
* DOES NOT MATTER. CAN BE HIGH-LOW OR
* LOW-HIGH

XCHHP 1 2 CH4OHP
XCHLP 2 3 CH4OLP

* ANALYSIS OPTION METHOD IS GEAR
.OPTIONS NOPAGE METHOD=GEAR
* AC DECADE ANALYSIS START FREQUENCY
* 30 MHz - END FREQUENCY 90 MHz
.AC DEC 20000 30000000 90000000
* PRINT FREQUENCY AND PHASE RESPONSE AT NODE 3
.PRINT AC V(3)
.END
```

The NgSpice simulator is invoked as

```
ngspice -b chBP.cir > out
```

The frequency and phase response plots are in Fig. 3.8a, b, respectively.

3.12 Effect of Non-ideal Reactive Elements on Filter Behavior and Performance and Design Space Exploration

Although the full-blown implementation of the scheme is under development, enhancement, and testing and consequently *does not* currently include non-ideal reactive element models, the effect of such non-ideal capacitors and inductors can be studied easily by manually editing and modifying an existing SPICE netlist. The simple band pass filter examined at the end of Chap. 2 is selected for this purpose. The SPICE input format netlist with only ideal capacitors and inductors is

```
.SUBCKT TESTBP 1 2
* IN
* OUT
C0 3 0 31.83nF
C1 4 2 159.16pF
C2 2 0 31.83nF
L0 3 0 795.8nH
L1 3 4 159.15uH
L2 2 0 795.8nH
R0 1 3 50
R1 2 0 50
.ENDS
```

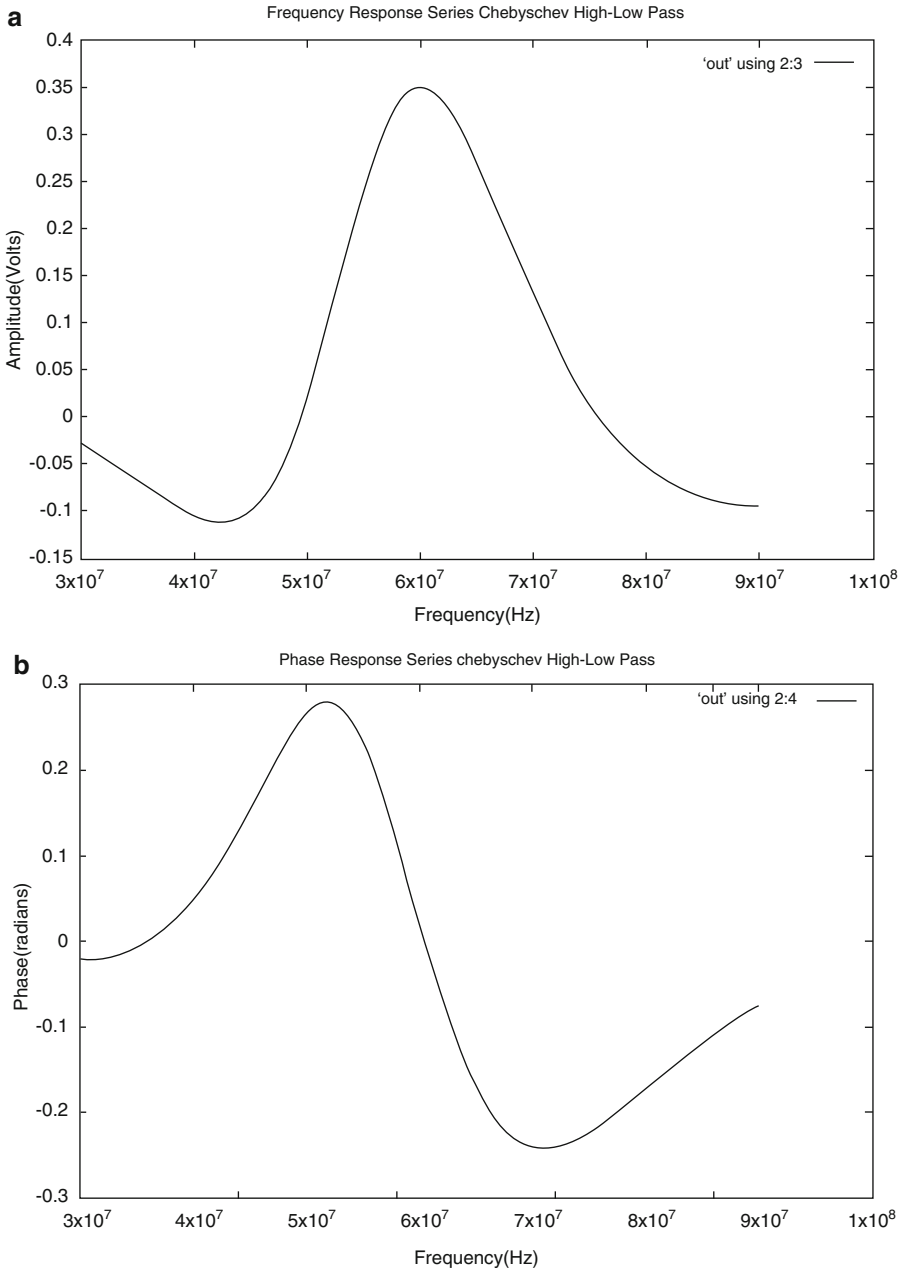

Fig. 3.8 (**a**) Frequency response of Chebyshev band pass filter consisting of a Chebyshev low pass and a Chebyshev high pass filter; (**b**) phase response of Chebyshev band pass filter consisting of a Chebyshev low pass and a Chebyshev high pass filter

Now, each ideal capacitor is replaced by a non-ideal device, using the model presented earlier in this chapter. Likewise for all inductors, the non-ideal capacitor and inductor SPICE models are

```
.SUBCKT NIC0 1 2
* 1 IN
* 2 OUT
C0 1 3 31.83n
L_ESL 3 4 5.0n
R_ESR 4 2 5m
.ENDS

.SUBCKT NIC1 1 2
* 1 IN
* 2 OUT
C0 1 3 159.16p
L_ESL 3 4 5.0n
R_ESR 4 2 5m
.ENDS

.SUBCKT NIL0 1 2
* 1 IN
* 2 OUT
C0 1 2 0.75n
L0 1 3 795.8n
R0 3 2 1.5u
.ENDS

.SUBCKT NIL1 1 2
* 1 IN
* 2 OUT
C0 1 2 0.75n
L0 1 3 159.15u
R0 3 2 1.5u
.ENDS
```

Now the band pass filter SPICE netlist with ideal capacitors and inductors is modified to include the non-ideal devices:

```
.SUBCKT TESTBPNL 1 2
* IN
* OUT
XC0 3 0 NIC0
XC1 4 2 NIC1
XC2 2 0 NIC0
XL0 3 0 NIL0
```

```
XL1 3 4 NIL1
XL2 2 0 NIL0
R0 1 3 50
R1 2 0 50
.ENDS

VS 1 0 DC 0.0001 AC 5
XBP 1 2 TESTBPNL

.OPTIONS NOPAGE METHOD=GEAR
.AC LIN 20000 500000 2000000
.PRINT AC V(2)
.END
```

Initially, when using ideal reactive elements, the band center frequency was set at 1 MHz, and the bandwidth was set at 100 kHz. In the current analysis using non-ideal reactive elements, the SPICE AC scan start and end frequencies were kept unchanged. The frequency and phase responses are completely different from that of a band pass filter with ideal reactive elements. The frequency and phase response plots are in Fig. 3.9a, b, respectively.

At this stage, the designer can exploit the fact that the SPICE input netlist can be easily edited manually—*design space exploration or what -if type of analysis*. The designer of course relies on his/her domain knowledge to achieve the desired goal. The parasitic capacitor values used in the non-ideal inductor model are changed as follows, i.e., from nanoFarad to picoFarad:

```
.SUBCKT NIL0 1 2
* 1 IN
* 2 OUT
C0 1 2 0.75p; WAS 0.75n
L0 1 3 795.8n
R0 3 2 1.5u
.ENDS

.SUBCKT NIL1 1 2
* 1 IN
* 2 OUT
C0 1 2 0.75p; WAS 0.75n
L0 1 3 159.15u
R0 3 2 1.5u
.ENDS
```

With the modifications as above, using the same SPICE AC scan frequency range, the frequency and phase responses are in Fig. 3.10a, b, respectively.

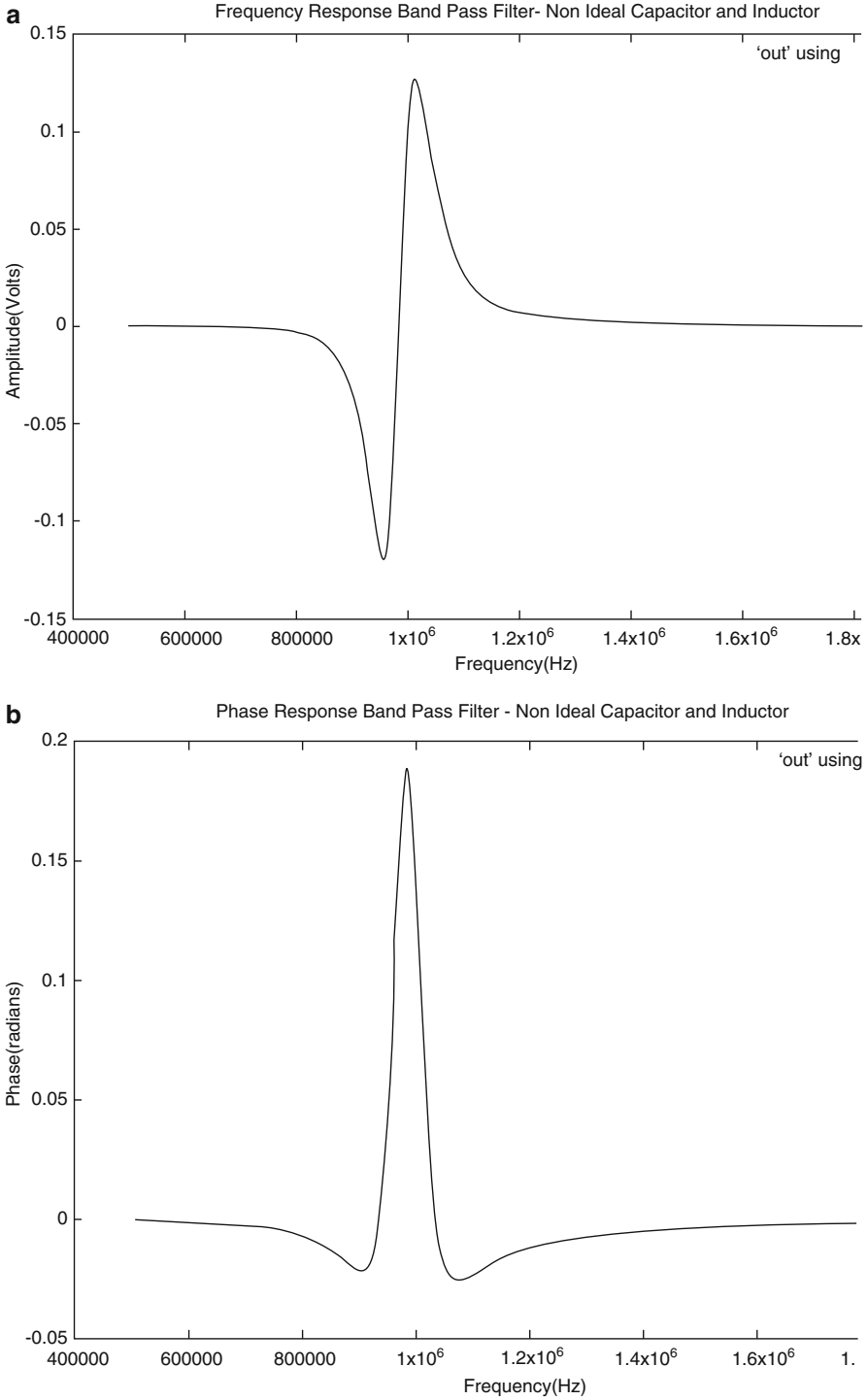

Fig. 3.9 (**a**) Frequency response when ladder network configuration band pass filter's reactive elements are replaced by non-ideal reactive elements. SPICE AC analysis frequency range is identical to ideal reactive element case; (**b**) phase response when ladder network configuration band pass filter's reactive elements are replaced by non-ideal reactive elements. SPICE AC analysis frequency range is identical to ideal reactive element case

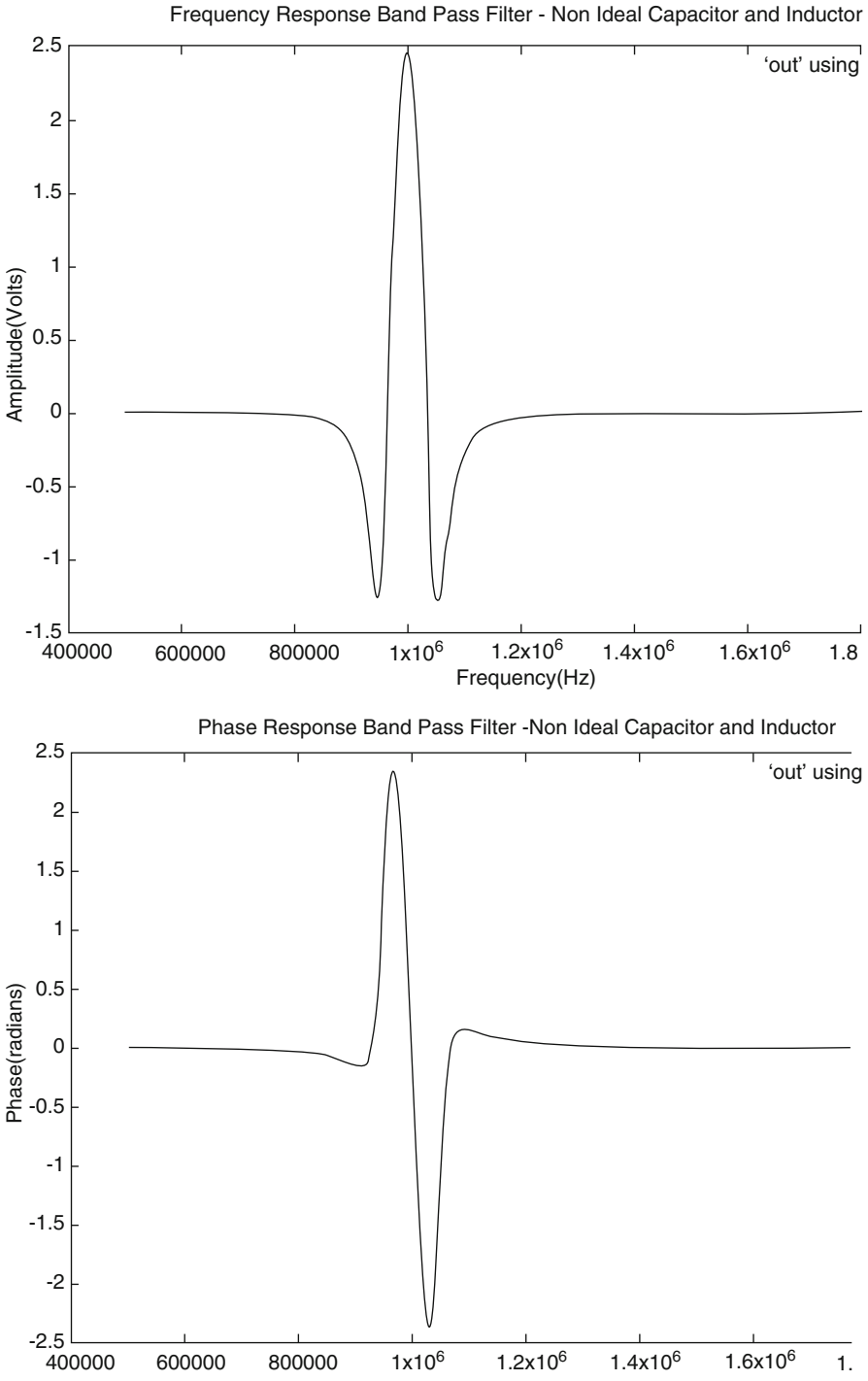

Fig. 3.10 (**a**) Non-ideal reactive element band pass filter frequency response; (**b**) non-ideal reactive element band pass filter phase response

3.13 SPICE: Electronic Circuit Performance Evaluation Gold Standard

SPICE or *S*imulation *P*rogram with *I*ntegrated *C*ircuit *E*mphasis was originally developed 45 years ago at the University of California, Berkeley, by Dr. Lawrence Nagel as SPICE 1 has emerged as the gold standard in evaluating the performance characteristics of any electronic circuit. It is a general-purpose, open-source analog electronic circuit simulator. It is also used in integrated circuit and board-level design to check the integrity of circuit designs and to predict circuit behavior. The original SPICE engine has been rewritten and optimized by various research groups, computer-aided design) tool vendors, etc., so that now any user can choose from a variety of open-source and proprietary distributions, e.g., HSpice, LTSpice, NgSpice, PSpice, etc., each with its detailed user manual. The current official version of SPICE in use is Spice 3. In addition, a number of universities worldwide have put up excellent online tutorial and user/reference manuals that provide detailed information about how to effectively use this tool.

Exercises

- A common filter type not examined here is the notch filter, which stops all signals in predefined frequency band. Derive the expressions that would transform a low pass filter to a notch filter.
- Create the pseudocode that would read normalized filter coefficients from a table, perform the necessary frequency and impedance transformation, and generate the SPICE input format netlist for a notch filter.
- A band pass filter can be created by connecting a low pass and a high pass filter in series, as examined here in detail. What about a notch filter—is there any such combination? Explain your answer.

References

1. Kernighan, B. W., & Ritchie, D. (1990). *The C programming language* (ANSI C Version).
2. *Hgspice Users' Manual*. Retrieved from http://ngspice.sourceforge.net/docs/ngspice-manual.pdf.
3. Zverev, A. I. *Handbook of filter synthesis* (Rev. ed.). ISBN-13: 978-0471749424; ISBN-10: 0471749427.
4. Matthaei, G. L., Young, L., & Jones, E. M. T. (1964). *Microwave filters, impedance-matching networks, and coupling structures*. New York: McGraw-Hill. LCCN 64-7937.
5. Bianchi, G., & Sorrentino, R. (2007). *Electronic filter simulation & design*. New York: McGraw-Hill. ISBN 978-0-07-149467-0.
6. Daniels, R. W. (1974). *Approximation methods for electronic filter design*. New York: McGraw-Hill. ISBN 0-07-015308-6.

7. Williams, A. B., & Taylors, F. J. (1988). *Electronic filter design handbook*. New York: McGraw-Hill. ISBN 0-07-070434-1.
8. Paarmann, L. D. *Design and analysis of analog filters: A signal processing perspective* (p. 238). Retrieved from http://books.google.com/books?id=l7oC-LJwyegC.
9. Pozar, D. M. (2011). *Microwave engineering* (4th ed.). New York: Wiley. ISBN-10: 0470631554; ISBN-13: 978-0470631553.

Chapter 4
Higher Frequencies (100's of MHz to 10's of GHz): Physical Constraints and Distributed Filters

4.1 Terminology

Some terminology is presented first. These terms are explained in greater detail in standard radio-frequency/microwave engineering textbooks.

Electrical permittivity: The physical constant e_0, called vacuum permittivity or permittivity of free space, is a baseline physical constant, representing the *capability of vacuum to permit electric field lines to spread through it*, and has a numerical value of 8.854×10^{-12} F/m.

The dielectric constant appears in the defining equation for the displacement current, where E is the electric field and P the polarization density, $D = e_0 E + P$.

The generalized form of the displacement current is

$$D(r\,,t) = \int dt' \int d^3 r'\, e(r\,,t,r',t') E(r',t') \;\text{ where }\; -\infty \le t \le t. \tag{4.1}$$

The quantity of interest is the relative or static permittivity for any dielectric medium that is the ratio of the permittivity of a given dielectric material to that of the permittivity of free space. From Eq. (4.1) above, the permittivity of any dielectric material varies in time, with changes in the applied electric field.

Skin depth: Skin effect is the result of an alternating time-varying electric field to become distributed within a conductor. So the current density is maximum near the surface of the conductor and decreases with depth in the conductor body. Skin effect arises due to opposing eddy currents induced by the changing magnetic field resulting from the alternating current. The electric current flows on the surface of the conductor, hence *skin effect*, and is bounded inside the body of the conductor at

Electronic supplementary material: The online version of this chapter (doi:10.1007/978-3-319-43470-4_4) contains supplementary material, which is available to authorized users.

© Springer International Publishing Switzerland 2017

A. Banerjee, *Automated Electronic Filter Design*,
DOI 10.1007/978-3-319-43470-4_4

the *skin depth*. Given that resistance is defined as $R = \frac{\text{rho} \cdot \text{l}}{\text{area of cross-section}}$ where rho is the material resistivity, the skin effect increases the effective resistance of the conductor with increasing frequency. Several common techniques exist to control skin depth effects.

Transmission line and characteristic impedance: A transmission line is a specialized structure designed to carry alternating current of radio-frequency signals. *The term transmission line is applicable only if the alternating current signal transmitted by it is such that the signal wavelength is comparable to the physical dimensions of the structure carrying such signals, e.g., a transmission line connects radio transmitters and receivers with their antennas, as inside a cell phone*. Transmission lines are analyzed as a ladder network with a very large number of shunt capacitors and series inductors—Fig. 4.1—it is a linear (complex voltage at any port is proportional to the complex current flowing into that port in the absence of any signal reflection) two-port network with interchangeable ports. The behavior of a transmission that is *uniform along its length* is controlled by a property called the *characteristic impedance* Z_0. C*haracteristic impedance is the ratio of the complex voltage of a given alternating current signal to its complex current at any point on the transmission line*. Typical values of Z_0 are 50 or 75 Ω for a coaxial cable, 600 Ω for a telephone cable, and so on. The *maximal matching* condition arises from the requirement that when power is sent down a transmission line, all of it must be absorbed by the load with no signal reflections. This is guaranteed by making the load impedance equal to Z_0. In the real world, power losses occur when the signal is being fed into the transmission line—*insertion loss*. Ohmic heating losses occur as all real-world transmission lines have finite but small resistance. At high frequencies (100's of MHz to 10's of GHz), several physical effects dominate—proximity effect, skin effect, and dielectric loss or dielectric heating loss. Dielectric loss occurs when the insulating material inside the transmission line absorbs energy from the alternating electric field and converts it to heat. The most common transmission line model consists of alternating series and shunt pairs. A series pair consists of a resistance (R) and inductance (L), and a shunt pair is a parallel combination of a capacitance (C) and conductance (G). Capacitor(s) and inductor (s) are ideally reactive; the resistance and conductance contribute to the loss in a transmission line. The total frequency-dependent power loss of power in a transmission line is specified in decibels per meter (dB/m). Coaxial cable manufacturers, e.g., supply a chart showing the loss in dB/m at a range of frequencies. A loss of 3 dB corresponds approximately to a halving of the power.

Filling factor: When a high-frequency electromagnetic field is being transmitted via a microstrip line, a percentage of the electric field penetrates the dielectric substrate material—this percentage is the filling factor. Filling factor for any dielectric material depends on the effective relative dielectric constant of the material, which is itself frequency dependent. This leads to dielectric loss.

Dielectric loss tangent: If the electric field of a high-frequency signal through a microstrip line is a plane wave, i.e., of the form $E = E_0 e^{jwt}$, then Maxwell's curl equation may be expressed as

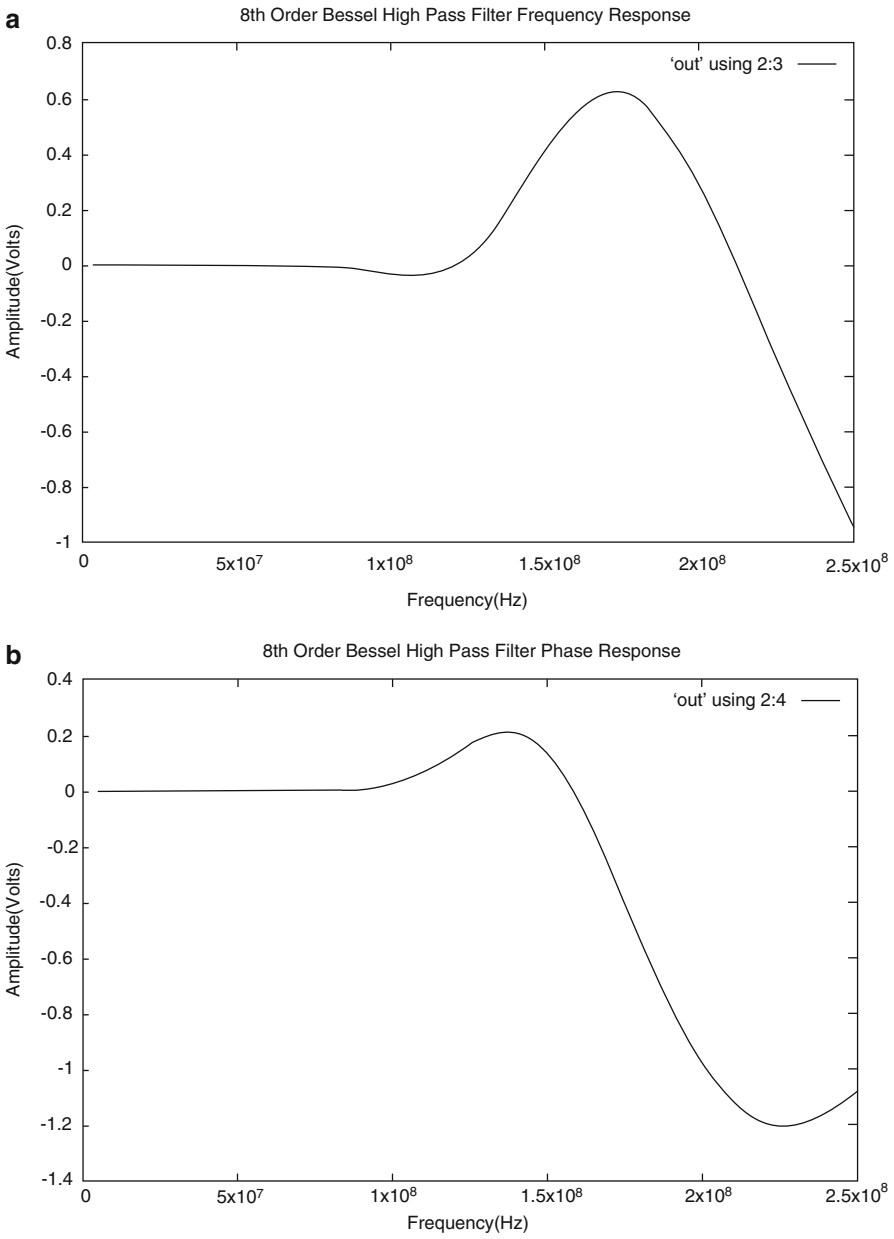

Fig. 4.1 (**a**) Sharp cutoff in frequency response of eighth-order Bessel high pass filter. This occurs when input signal frequency exceeds the discrete reactive elements' cutoff frequency; (**b**) phase response for eighth-order Bessel high pass filter, whose frequency response is in (**a**); and (**c**) infinitesimally small section of transmission line of length del_z

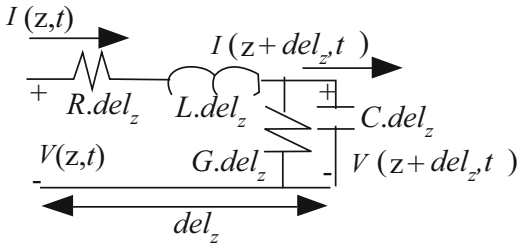

Fig. 4.1 (continued)

$$\nabla xH = jw \cdot \text{epsilon}' E + \left(w \cdot \text{epsilon}'' + \text{sigma}\right)E,$$

where epsilon′ is the imaginary component of permittivity due to bound charge and dipole relaxation effects, both of which contribute to energy loss. This energy loss however cannot be identified separately from energy loss due to free charge conduction, quantified by sigma. The component, epsilon″, represents the lossless permittivity. The loss tangent is then defined as the ratio (or angle in a complex plane) of the lossy reaction to the electric field E in the curl equation to the lossless reaction.

Unit element: While using transmission line sections to build electronic filters, often elements are separated by special transmission line segments called unit elements, whose electrical length is $\text{beta} \cdot \text{len} = 0.785\left(\frac{f}{f_c}\right)$. The details will be elaborated on subsequently.

4.2 High-Frequency Issues with Discrete Element Electronic Filter Fabrication and Transmission Line Fundamentals

The Bessel high pass filter of Chap. 3, Sect. 3.5, will be examined in detail once again, to understand the problems associated with fabricating electronic filters for high-frequency (100's of MHz to 10's of GHz) applications with discrete reactive elements. In the original SPICE simulation, the small-signal (.AC) analysis frequency range was set with a start frequency of 40 MHz and end frequency of 170 MHz:

```
* ANALYSIS OPTIONS GEAR METHOD
.OPTIONS NOPAGE METHOD=GEAR
* AC DECADE ANALYSIS WITH START FREQUENCY OF 40 MHz
* AND END FREQUENCY OF 170 MHz
.AC DEC 20000 40000000 170000000
```

```
* PRINT AC ANALYSIS VOLTAGE, PHASE AT NODE 2
.PRINT AC V(2)
.END
```

The end frequency is now increased to 250 MHz, and new SPICE.AC analysis provides the frequency and phase responses as in Fig. 4.1a, b respectively. The modified SPICE netlist listing is below:

```
.SUBCKT be8OHP 1 7
* IN
* OUT
R0 1 3 50.00
L0 3 0 2.756452e-07
C1 3 4 3.726633e-11
L2 4 0 5.745473e-08
C3 4 5 1.706994e-11
L4 5 0 3.426456e-08
C5 5 6 1.165350e-11
L6 6 0 2.312139e-08
C7 6 7 4.472421e-12
R1 7 0 50.000000
.ENDS
VSIG 1 0 DC 0.0001 AC 5
XBE 1 2 be8OHP

* ANALYSIS OPTIONS GEAR METHOD
.OPTIONS NOPAGE METHOD=GEAR
* AC DECADE ANALYSIS WITH START FREQUENCY OF 40 MHz
* AND END FREQUENCY OF 250 MHz
.AC LIN 20000 5000000 250000000
* PRINT AC ANALYSIS VOLTAGE, PHASE AT NODE 2
.PRINT AC V(2)
.END
```

To circumvent these issues and realizing that as frequency increases the wavelength of any signal must decrease, ultimately becoming comparable to dimensions of the physical structure transmitting the signal, designers have exploited the properties of transmission lines to fabricate filters that will operate in the 100's of MHz to 10's of GHz frequency range. A finite length of transmission line is a series connection of a very large number of infinitesimally small sections made of discrete/lumped elements as shown in Fig. 4.1c, where the length of each infinitesimally small section is denoted as del_z.

In Fig. 4.1c

- C is the shunt capacitance per unit length in Farad/meter and is due to close proximity of the two conductors.

- G is the conductance per unit length in Siemens/meter and occurs due to energy loss in the dielectric material between the conductors.
- L is the inductance per unit length in Henry/meter and is the total self-inductance of the conductors.
- R is the resistance per unit length in Ohms/meter and represents the very small but finite resistance of the conductors.

The traveling current and voltage waves in a transmission line are

$$I(z) = I_o^l \mathrm{e}^{-\mathrm{gamma}\cdot z} + I_o^r \mathrm{e}^{-\mathrm{gamma}\cdot z} \text{ and } V(z) = V_o^l \mathrm{e}^{-\mathrm{gamma}\cdot z} + V_o^r \mathrm{e}^{-\mathrm{gamma}\cdot z}, \quad (4.2)$$

where gamma is the complex propagation constant, l indicates propagation from left to right, r represents propagation from right to left, and z is the propagation direction. The propagation constant is

$$\mathrm{gamma} = \mathrm{alpha} + j \cdot \mathrm{beta} = \sqrt{(R + jwL)(G + JwC)}, \quad (4.3)$$

where alpha is the attenuation constant, beta the wavenumber, and w the frequency in radian/second. The complex characteristic impedance is defined as

$$Z_0 = \frac{V_o^l}{I_o^l} = \frac{V_o^r}{I_o^r} = \sqrt{\frac{R + jwL}{G + jwC}}. \quad (4.4)$$

From electromagnetic theory, the phase velocity of a wave is

$$v_{\mathrm{phase}} = \frac{1}{\sqrt{\mathrm{mu}_0 \cdot \mathrm{epsilon}_0 \cdot \mathrm{epsilon}_\mathrm{r}}}, \quad (4.5)$$

where mu_0 is the magnetic permeability of free space, $\mathrm{epsilon}_0$ is the electrical permittivity of free space, and $\mathrm{epsilon}_r$ is the relative permeability or dielectric constant of the dielectric material between the conductors:

$$\text{The wavelength is lambda} = \frac{v_{\mathrm{phase}}}{\text{frequency}} \text{ unit meter}. \quad (4.6)$$

The wavenumber, which by definition is the total number of waves per unit length, is

$$\text{wavenumber} = \frac{6.28}{\text{wavelength}} \text{ unit radians}. \quad (4.7)$$

4.2.1 Lossless Transmission Lines

In some comparatively low-frequency practical cases, the attenuation parameter alpha is zero, in which case the propagation parameter gamma and wavenumber beta become gamma $= jw \cdot \sqrt{\mathrm{LC}}$ and beta $= w \cdot$ sqrt LC, respectively, where w is the angular frequency in radian/second.

4.2.2 Lossless Terminated Transmission Line

An arbitrary load (*not matched to source impedance*) is attached to a lossless transmission line. The input impedance and characteristic impedance of the transformation line are Z and Z_0, respectively. When a traveling wave signal is injected into the transmission line, the load and characteristic impedance mismatch generates a reflected wave traveling in the opposite direction, from load end to the source end. The ratio of the amplitude of the reflected voltage wave to that of the incident voltage wave is the voltage reflection coefficient:

$$\text{voltage reflection coefficient} = \frac{Z_\text{L} - Z_0}{Z_\text{L} + Z_0}. \tag{4.8}$$

Under mismatch conditions, not all the incident power is absorbed by the load, and the reflected power is called the return loss (RL) and is given by

$$\begin{aligned} \text{RL} &= 20\log(|\text{voltage reflection coefficient}|) \text{ dB and} \\ \text{SWR} &= \frac{1 + |\text{voltage reflection coefficient}|}{1 = |\text{voltage reflection coefficient}|}. \end{aligned} \tag{4.9}$$

The input impedance is

$$Z = Z_0\left(\frac{Z_\text{L} + jZ_0 \tan(\text{beta} \cdot \text{len})}{Z_0 + jZ_\text{L} \tan(\text{beta} \cdot \text{len})}\right), \tag{4.10}$$

where beta is the wavenumber and len is any arbitrary length of the transmission line. For a short-circuited transmission line, $Z_\text{L} = 0$ and $Z = jZ_0 \tan(\text{beta} \cdot \text{len})$. In case of an open circuit, $Z = jZ_0 \cot(\text{beta} \cdot \text{len})$. If the length of the transmission is one-half of a wavelength, then it is the maximally matched condition with $Z = Z_\text{L}$, and in case the length of the transmission line is a quarter of a wavelength (*quarter-wave plate*), $Z = \frac{Z_0^2}{Z_\text{L}}$. Another popular transmission line type is the *lambda-over-8*, i.e., the length of the transmission line is half of that of a quarter-wave plate. The *electrical length* of a transmission line of any arbitrary physical length len is the product:

$$\text{beta} \cdot \text{len} = \left(\frac{6.28}{\text{lambda}}\right)\text{len} = \left(\frac{6.28 \text{ frequency}}{v_\text{phase}}\right)\text{len}, \tag{4.11}$$

where beta is the wavenumber, lambda the wavelength, and v_phase the phase velocity. *The electrical length is the key parameter that allows the designer to convert discrete/lumped capacitor and inductor values to transmission line*

parameters, so that the corresponding microstrip filter can be fabricated using the same techniques as printed circuit boards. A microstrip filter does not use any discrete reactive or resistive components. To successfully transition from discrete/lumped element filter design to a transmission line segment-based filter, the impedance of a section of short or short-circuited transmission line ($Z = jZ_0$ tan(beta $\times$ len)) or open-circuited transmission line ($Z = jZ_0$ cot(beta $\times$ len)) must equal the corresponding discrete/lumped element it replaces, i.e., inductive reactance must equal the impedance of a short-circuited transmission line segment that replaces the corresponding discrete inductor, and capacitive reactance must equal the impedance of an open-circuited transmission line segment that replaces the corresponding discrete capacitor.

4.2.3 Stub Synthesis: Key Equations

Sections of microstrip transmission line are often referred to as *stubs*. A transmission line section of a predefined length with zero load impedance is a *short-circuited stub*, and a transmission line section with infinite load impedance is an *open-circuited stub*. The substitution of discrete element components in a filter with stubs is possible only if the impedance of the discrete/lumped element device equals the impedance of the stub.

From above, the input impedance of a short-circuited stub is $Z = jZ_0$ tan (beta $\times$ len) and that of an open-circuited stub is $Z = jZ_0$ cot(beta $\times$ len). Rewriting these two expressions in terms of admittance, impedance, and scaled (denormalized) values, for a short-circuited stub, the impedance is $Z = jZ_0'$ tan $(\text{beta} \cdot \text{len})$ and the admittance of an open-circuited stub is $Z = jY_0'$ tan $(\text{beta} \cdot \text{len})$, where both Z_0' and Y_0' are scaled values. As $\tan(1.57) = \infty$ if the product beta $\times$ len < 1.57, then for a short-circuited stub, Z is a positive number and is the reactance of an inductor. As admittance is the reciprocal of impedance, when the same constraints are imposed on the expression, the admittance of an open-circuited stub gives the input impedance of the open-circuited stub as $Z = \frac{1}{jY_0'(\text{beta}\cdot\text{len})}$, which is the reactance of a capacitor. As beta $= \frac{6.28}{\text{lambda}}$, where lambda is the wavelength, both the capacitor and inductor reactances are frequency dependent. If the additional constraint len $= \frac{\text{lambda}}{8}$ is imposed on the length of a microstrip transmission line stub, then $Z = jZ_0'$, and as the reactance of an inductor is $Z = jwL$, equating the two expression yields

$$Z_0' = w_c L, \tag{4.12}$$

where w_c is the angular cutoff frequency of the inductor. Summarizing, a short-circuited transmission line stub of length $\frac{\text{lambda}}{8}$ has the same scaled input

impedance as that of an inductor. By identical reasoning, an open-circuited transmission line of length $\frac{\text{lambda}}{8}$ has the same scaled input impedance as that of a capacitor.

4.3 Richard's Transformation and Kuroda's Identities

The following transformation $\text{Omega} = \tan(\text{beta} \cdot \text{len}) = \tan\left(\frac{w \cdot \text{len}}{v_{\text{phase}}}\right)$ maps the w-plane to the Omega plane, where w is the angular frequency in radian/second, len is the physical length of a transmission line stub, and v_{phase} is the signal phase velocity in the same transmission line stub. It is a very powerful technique to synthesize LC networks from transmission line stubs. When the angular velocity w is substituted by Omega, the acceptance of a capacitor and the reactance of an inductor can be rewritten, respectively, as

$$\begin{aligned} jB_{\text{capacitor}} &= j \cdot \text{Omega} \cdot C = j \cdot C \cdot \text{beta} \cdot \text{len} \quad \text{and} \quad jX_{\text{inductor}} = j \cdot \text{Omega} \cdot L \\ &= j \cdot L \cdot \text{beta} \cdot \text{len}. \end{aligned} \tag{4.13}$$

Clearly, a capacitor can be replaced by an open-circuited transmission line stub of electrical length geta.len and characteristic impedance 1/C, and similarly, an inductor can be replaced by a short-circuited transmission line stub of electrical length geta.len and characteristic impedance L. A filter impedance of 1 Ω is assumed.

For a normalized or prototype filter, the cutoff frequency of 1 rad/s corresponds to:

$\text{Omega} = 1 = \tan(\text{beta} \cdot \text{len})$ which means that the transmission line stub has an electrical length of lambda/8 where lambda is the wavelength of the cutoff frequency w_{cutoff} of the signal passing through the transmission line stub. The frequency response is periodic with frequency $4w_{\text{cutoff}}$. At $2w_{\text{cutoff}}$ an attenuation pole occurs, and the electrical length of the transmission line stub will be lambda/4. These transmission line stubs are often called *commensurate* lines—Fig. 4.2a, b.

Kuroda's four identities involve redundant transmission line stubs to realize practical real-world filters based on the following rules:

- Transmission line stubs must be physically separated.
- Transform series stubs to shunt stubs or vice versa.
- Change impractical characteristic impedance stubs to physically realizable ones.

The special transmission line segments separating two transmission line segments are called *unit elements* and have an electrical length of lambda/8 at the cutoff frequency. Each of Kuroda's identities utilizes the following simple formula $k = n_2 = 1 + \frac{Z_1}{Z_2}$ whose significance is clear from Fig. 4.2c–f.

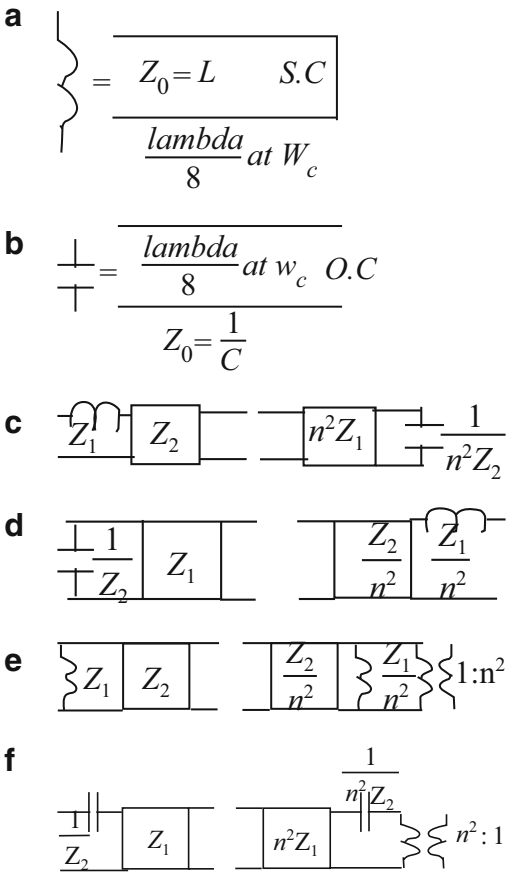

Fig. 4.2 (**a**) Richard's transformation of discrete/lumped element inductor to a short-circuited transmission line of electrical length 1/8 the cutoff wavelength; (**b**) Richard's transformation of discrete/lumped element capacitor to an open-circuited transmission line of electrical length 1/8 the cutoff wavelength; (**c**) Kuroda's first identity; (**d**) Kuroda's second identity; (**e**) Kuroda's third identity; and (**f**) Kuroda's fourth identity

4.4 Distributed Electronic Filter Design Scheme

Integrating the information presented in the previous sections, a simple scheme for designing and fabricating distributed element (as opposed to discrete element) electronic filters is given here. As mentioned earlier, this module is currently in the development, modification, and testing phase and will be fully integrated in the near future with the full-blown C language implementation of the proposed scheme elaborated on in the previous chapters. So, the algorithm presented here is partly automated and partly manual:

- The designer specifies the pass band edge frequency, the stop band start frequency, the maximum pass, the stop band attenuations, and the filter name (Bessel, Butterworth.. etc.). The automated C language program computes the filter order and the cutoff frequency and computes or looks up the corresponding normalized filter coefficients for that filter order.
- Discrete element capacitances and inductances are replaced by appropriate short-circuited or open-circuited *lambda-over-eight* length transmission line stubs. The wavelength lambda corresponds to the cutoff frequency in the case of a low pass or high pass filter and to the named center frequency in the case of a band pass filter.
- Each series stub is converted to shunt stub using Kuroda's identities. "Unit element" stubs are added to the signal entry and exit nodes of the normalized filter.
- The normalized filter is now frequency and impedance scaled; specifically the sizes of the equivalent transmission line stubs are computed.

The first step is fully automated at present, but the last three are partially manual. A number of commercial CAD tools automate the process, but rely on user-supplied heuristics to generate the results, in addition to proprietary techniques.

4.5 Transmission Line Losses

The analysis so far is based on the notion of lossless transmission line model, which does not reflect reality. Signal attenuation in a microstrip line is a combination of conductor or ohmic losses, dielectric or substrate losses, as well as radiation and propagation losses and higher-order modes. The two biggest contributors to the total loss are conductor and dielectric losses and have been examined in detail by researchers who have proposed quasi-static models for characteristic impedance [1–3], quasi-static models for effective dielectric constant [2–4], and the effect of finite thickness of the microstrip traces [2, 4]. Dielectric dispersion at very high signal frequencies has been examined and analyzed in great detail in [5–11].

For example, Jensen et al. [5] propose that the surface roughness of the dielectric substrate is necessary to account for an asymptotic increase seen in the apparent surface resistance with decreasing skin depth. This effect has to be countered with a correction factor. The current distribution factor is a very good approximation provided that the strip thickness exceeds three skin depths.

Dispersion loss arises chiefly due to the effects of finite *loss tangent*. The loss magnitude is proportional to the operating frequency. For common microwave substrate materials like ceramics, a loss tangent less than the dielectric losses can be neglected compared to the conductor losses.

At present, a stand-alone C language module that incorporates the high-frequency transmission line loss [1–11] mechanisms and computes the effective characteristic impedance and effective relative dielectric constant at a predefined

frequency (GHz) has been developed. It will be incorporated into the full-blown implementation of the automated filter design scheme in the near future. Preliminary test results with the stand-alone C language module have been very encouraging.

4.6 Distributed Electronic Filter Design Example Stepped Impedance Low Pass Filter

A microstrip-based low pass Butterworth filter is designed using both the full-blown implementation of the automated electronic filter design scheme and the stand-alone C language program that tackles the high-frequency transmission line losses. The design incorporates the common *stepped impedance* technique that uses a serial connection of high characteristic impedance and low characteristic impedance transaction line (in this case microstrip) segments to create a low pass filter. The filter specifications are:

- Pass band edge frequency is 2 GHz.
- Stop band start frequency is 5 GHz.
- Pass band maximum attenuation is 0.5 dB.
- Stop band maximum attenuation is 5 dB.
- Source/load impedance is 50 Ω.
- Printed circuit board dielectric material thickness is 1.2 mm.
- Printed circuit board material dielectric constant is 4.3.

The full-blown automated filter design tool returns the following values for the filter order, cutoff frequency, the normalized, scaled filter coefficients, and SPICE input format netlist file name:

```
./microwavefiltnew bw l 0.5 5 2000000000 5000000000 50 n

Cutoff 3.675017e+09 Hz  filter order 3
1.000000e+00
2.000000e+00
1.000000e+00
0.000000e+00
0.000000e+00
0.000000e+00
0.000000e+00
0.000000e+00
0.000000e+00
0.000000e+00
******
0, 8.661709e-13
```

```
1, 4.330854e-09
2, 8.661709e-13
bw3OLP.cir
```

So this is a Butterworth third-order low pass filter with cutoff frequency 3.67 GHz and normalized filter coefficients 1.00, 2.00, and 1.00. The SPICE input format netlist is in the text file *bw3OLP.cir*.

Now the key design step is to evaluate the *electrical lengths* corresponding to the capacitor and inductor values. As this is a *stepped impedance* filter, the high characteristic impedance transmission line segment corresponds to an inductor and the low characteristic impedance transmission line segment corresponds to a capacitor. The electrical lengths corresponding to a capacitor and an inductor are

$$\text{beta} \cdot \text{len} = \frac{CZ_{\text{low}}}{R_{\text{o}}} - \text{capacitor}, \tag{4.14a}$$

$$\text{beta} \cdot \text{len} = \frac{LR_{\text{o}}}{Z_{\text{max}}} - \text{inductor}, \tag{4.14b}$$

where beta is the wavenumber, len is the physical length of a transmission line segment, R_{o} is the filter impedance (50 Ω), Z_{min} is the minimum characteristic impedance, and Z_{max} is the maximum characteristic impedance. *It is interesting to note how adroitly frequency and impedance scaling have been included into Eqs. (4.14a) and (4.14b).*

A key assumption, based on fabrication constraints of the physical filter, is that the high characteristic impedance line segment width is 0.75 mm and the low characteristic impedance line segment width is 5 mm.

To compute the physical lengths, first the high and low characteristic impedances need to be computed, as well as the effective signal wavelength at the cutoff frequency, taking into account the loss mechanisms. These computations are done by the stand-alone C language module mentioned earlier. The command line help for this program is as follows:

```
./microstripnew -h
insufficient arguments ... check usage
./microstripnew conductor thickness(mm)
substrate thickness(mm) trace width(mm)
dirlectric permittivity frequency(GHz)
```

This programs requires five input arguments—conductor thickness in mm, substrate thickness in mm, the trace width in mm, and the dielectric constant. For the high characteristic impedance transmission line segment,

```
./microstripnew 0.001 1.2 0.5 4.3 3.67
```

Effective relative permittivity 2.67525 characteristic impedance 114.9641 Ohm frequency 3.670000 GHz

For the low characteristic impedance transmission line segment,

```
./microstripnew 0.001 1.2 5 4.3 3.67
```

Effective relative permittivity 2.79714 characteristic impedance 35.2976 Ohm frequency 3.670000 GHz

The electrical length for the single microstrip inductor is (from Eq. (4.14b))

$$\text{beta} \cdot \text{len}_{\text{capacitor}} = \frac{2.00 \times 50.0}{114.9641} = 0.8598 \text{ radians}, \tag{4.15a}$$

where the physical length of the inductor microstrip is $\text{len}_{\text{inductor}}$, the wavenumber is beta, the normalized inductance is 3.0 H, the characteristic impedance of the microstrip segment is 114.9641 Ω, and the filter impedance is 50.0 Ω.

In a similar fashion, the electrical length for each of the two capacitor microstrip segments is

$$\text{beta} \cdot \text{len}_{\text{capacitor}} = \frac{1.0 \times 33.2978}{50.0} = 0.6649 \text{ radians}. \tag{4.15b}$$

As mentioned previously, at high frequencies, wavelength of the signal changes due to polarization of dielectric (displacement current, filling factor, etc.). The stand-alone C language module that computes the modified characteristic impedance also computes the effective dielectric constant at that frequency. Thus from above, the effective dielectric constant for the inductor microstrip segment is 2.67525 and the corresponding value for the capacitor microstrip segment is 2.79714. The phase velocities ($v_{\text{phase}} = \frac{\text{vacuum velocity of light}}{\sqrt{\text{effective dielectric constant}}}$) in the capacitor and inductor microstrip segments are, respectively,

$$179.2315 \text{ and } 183.2989 \text{ mm/ns}. \tag{4.16}$$

Using results of Eq. (4.16) and the low pass filter cutoff frequency of 3.67 GHz, the modified wavelengths for the capacitor and inductor microstrip segments are, respectively,

$$48.8369 \text{ and } 49.9452 \text{ mm}. \tag{4.17}$$

Combining the results of Eqs. (4.15a), (4.15b), (4.16), and (4.17), the physical lengths of the microstrip capacitor and inductor segments can be calculated. For the single inductor microstrip segment,

$$\text{len}_{\text{capacitor}} = \frac{0.8598 \times 49.9452}{6.28} = 6.838 \text{ mm}. \tag{4.18a}$$

And for each of the two microstrip capacitor segments,

$$\text{len}_{\text{capacitor}} = \frac{06649 \times 48.8369}{6.28} = 5.171 \text{ mm}. \tag{4.18b}$$

Summarizing, the dimensions of the capacitor and inductor microstrip segments are as follows:

$$\text{Capacitor}: \ 5.171 \times 5 \text{ mm}, \tag{4.19a}$$

$$\text{Inductor}: \ 6.838 \times 0.5 \ \text{ mm}. \tag{4.19b}$$

In addition to these, the dimensions of the 50 Ω source and load microstrip transmission line segments need to be computed. These correspond to the source and load impedances. Also, the dimensions of the pads for the two SMA connectors that would connect the filter to external devices need to be computed. This calculation is left as an exercise to the reader.

There are other types of distributed filters as coupled line filters, coupled resonator filters, capacitively coupled filters, interdigitated filters, etc., whose fabrication/implementation involves a number of special techniques, such as those for stepped impedance filter examined here. These special techniques will be examined in detail in future.

Exercises

- For the high-impedance low-impedance low pass filter design, calculate the physical dimensions for the source/load 50 Ω impedances and the pad sizes for the SMA connectors for signal entry/exit into/from the filter.

References

1. Wheeler, H. A. (1965). Transmission-line properties of parallel strips separated by a dielectric sheet. *IEEE Transactions on Microwave Theory and Techniques, 13*(2), 172–185.
2. Schneider, M. V. (1969). Microstrip lines for microwave integrated circuits. *Bell System Technical Journal, 48*, 1421–1444.
3. Hammerstad, E., & Jensen, O. (1980). *Accurate models for microstrip computer-aided design, symposium on microwave theory and techniques* (pp. 407–409). Washington, DC: IEEE.
4. Wheeler, H. A. (1977). Transmission-line properties of a strip on a dielectric sheet on a plane. *IEEE Transactions on Microwave Theory and Techniques, 25*(8), 631–647.
5. Kirschning, M., & Jansen, R. H. (1982). Accurate model for effective dielectric constant of microstrip with validity up to millimeter-wave frequencies. *Electronics Letters, 8*(6), 272–273.

6. Yamashita, E., Atsuki, K., & Ueda, T. (1979). An approximate dispersion formula of microstrip lines for computer aided design of microwave integrated circuits. *IEEE Transactions on Microwave Theory and Techniques, 27*, 1036–1038.
7. Kobayashi, M. (1988). A dispersion formula satisfying recent requirements in microstrip CAD. *IEEE Transactions on Microwave Theory and Techniques, 36*(8), 1246–1250.
8. Getsinger, W. J. (1973). Microstrip dispersion mod. *IEEE Transactions on Microwave Theory and Techniques, 21*(1), 34–39.
9. Edwards, C., & Owens, R. P. (1976). 2–18 GHz dispersion measurements on 10–100 Ohm microstrip lines on sapphire. *IEEE Transactions on Microwave Theory and Techniques, 24*(8), 506–513.
10. Pramanick, P., & Bhartia, P. (1983). An accurate description of dispersion in microstrip. *Microwave Journal, 26*, 89–96.
11. Schneider, M. V. (1972). Microstrip dispersion. *Proceedings of the IEEE Letters, 60*, 144–146.

Chapter 5
Summary and Conclusion

This book elaborates on an automated scheme to efficiently design, verify performance characteristics of, and fabricate electronic filters that accurately satisfy design specifications. To guarantee an accurate design, all error-prone and/or manual steps in the design process are eliminated, freeing the designer to explore the design space and perform *what-if* type of analysis, as required. This guarantee can be enforced if the design process is automated with some designer-supplied inputs and accurately measure the performance characteristics of the final design with a universally acceptable technique or tool. As the gold standard in electronic circuit performance analysis is the SPICE (*S*imulation *P*rogram with *I*ntegrated *C*ircuit *E*mphasis) simulation tool, the designer must be able to seamlessly analyze a design with SPICE. Thus, the automated filter design scheme *must* generate its final output in the SPICE input (in SPICE terminology, the *netlist*) format. Thus, the automated filter design scheme must guarantee that it:

- Eliminates all manual and/or error-prone/time-consuming computation steps.
- In order that the designer verify the performance characteristics of a design, the final output must be in a format that is acceptable to any commonly used electronic circuit performance characterization tool. Then, the designer can seamlessly verify that the final design satisfies specifications as well as explores design space.

Traditional electronic filter design scheme is discussed in detail in Chap. 1. However, most of the key steps involve complicated, manual calculations and are therefore error-prone and time-consuming. In fact, calculations for a high-order filter become so unwieldy that designers often are forced to replace such a filter with a series connection of low-order filters. In the general case, even if performance characteristics (cutoff frequency, pass/stop band ripple, quality factor, etc.) have been specified, determining the numerical values of the reactive components (capacitor, inductor) or resistor becomes an exercise in trial and error. In fact, for any filter of order greater than two, determining the analytical expression for the filter transfer function is difficult, followed by an equally difficult task of

© Springer International Publishing Switzerland 2017

A. Banerjee, *Automated Electronic Filter Design*,
DOI 10.1007/978-3-319-43470-4_5

factorizing this transfer function to calculate its poles and zeros, taking into account the key stability condition that all poles must be in the left half of the complex plane and the poles must be complex conjugates of each other to ensure that the coefficients in the transfer function are real-valued. To circumvent these difficulties and guarantee electronic filter designs that accurately satisfy design specifications, four core concepts are introduced. These are:

- Normalized or prototype filters and their corresponding polynomials
- Filter tables, whose contents are the coefficients of the normalized filter polynomials
- Filter transformations that allow any low pass filter to be converted to a band pass, high pass, or band reject filter
- The ladder network concept

These four concepts in combination allow the filter design process to be automated as in a C language computer program, whose input is the specification of a real-world electronic filter, and the output is a text file containing the numerical values of the components (capacitors, inductors and resistors) *formatted in the SPICE input netlist format*. The designer can analyze the performance characteristics (frequency/phase response, etc.) of the filter using the universally popular SPICE simulator.

The theoretical framework underlying the automated electronic filter design scheme is examined in detail in Chap. 2. For reference purposes, a number of filter design-specific concepts/terms (e.g., filter insertion loss, ladder network, normalized/prototype filter, maximum available source power, pass/stop band ripple, etc.) are examined at start. Three common filters Bessel, Butterworth, and Chebyshev are analyzed in detail in context of the twin concepts of normalized/prototype filter and ladder network. A normalized or prototype low pass filter is one whose source/load impedance is both $1\,\Omega$; if that filter's topology is that of a ladder network, a polynomial can be derived whose numerical coefficients are the values of the capacitors and inductors connected in the ladder network topology. Consequently, tables of polynomial coefficients for each of Bessel, Butterworth, and Chebyshev filters can be tabulated. The basis of discussion is the low pass filter, and simple mathematical transformations can be used to derive the corresponding ladder topology band pass/stop and high pass filters. Finally, simple scaling rules allow the normalized filter coefficients and source load impedance values to be transformed, so that the resulting filter will satisfy designer's real-world specifications. *These calculations can be automated easily, in this case an ANSI C language program. Since for real-world scenario, the designer must also verify that the generated design satisfies performance specifications, the C language program returns the final design formatted in the* ***S****imulation* ***P****rogram with* ***I****ntegrated* ***C****ircuit* ***E****mphasis (SPICE) text-based input netlist format*. The designer can then use any available SPICE simulator (HSpice, LTSpice, HgSpice, PSpice, etc.) to analyze the performance characteristics as well as explore the design space. For any filter, the two key design parameters are the filter order and the cutoff frequency (high/low pass filter) or band center frequency (band pass/stop filter). Both these

values can be computed from four parameters (pass band edge frequency, stop band start frequency, and maximum pass/stop band attenuation) (as in the full-blown automated filter design scheme implementation) or be supplied by the designer (as in the simplified automated filter design scheme implementation).

Building upon the theoretical framework presented in Chap. 2, Chap. 3 examines in detail the implementation of both the simplified and full-blown versions of the automated electronic filter design scheme using the C computer language. Design examples are presented to illustrate how to use both the simplified and full-blown implementations to design and test the performance characteristics of any filter, as per given specifications. *As the final filter design is in SPICE text-based input netlist format, the frequency and phase responses can be determined easily with available SPICE simulator—this feature to generate the SPICE netlist as output is unique to this electronic filter design tool*. The tool exploits each C language feature to ensure robustness and accuracy of calculated values. For example, both scheme implementations make extensive use of C language pointers, and each pointer passed to any function to ensure that it is not null—null pointers cannot be dereferenced or manipulated in any other way. C language code snippets illustrate how the main steps in the overall calculation are executed, e.g., how the filter order for a Chebyshev low pass filter is calculated, followed by evaluation of the normalized filter coefficients, or how normalized filter coefficients are scaled in frequency and source/load impedance for the filter to satisfy design specifications. Design examples using the simplified implementation include an eighth-order Bessel high pass filter, a seventh-order Chebyshev low pass filter, and an eighth-order Chebyshev band pass filter. A band pass filter can be designed as stand-alone or series connection of a high and a low pass filter; the full-blown scheme implementation is used to design a Chebyshev band pass filter to illustrate this. The full-blown implementation computes both the filter order and cutoff/band center frequencies internally and *is not supplied by the designer*. These values are calculated from input values of pass band edge frequency, stop band start frequency, and maximum pass/stop band attenuation. Real-world discrete components are non-ideal—the SPICE text-based output netlist feature allows the designer to explore the design space—an ideal capacitor may be replaced by a non-ideal capacitor (with series parasitic inductance and resistance) by manual editing.

The automated electronic filter design scheme exploits properties of the normalized filter, ladder network, and filter transformations. A ladder network contains only reactive (capacitor, inductor) elements and input signal frequencies in the 100's of MHz to 10's of GHz which exceed the cutoff frequencies of these reactive components—in Chap. 4, Sect. 4.2, the design example Bessel high pass filter in Chap. 3, Sect. 3.5, has been analyzed once again with the NgSpice 2.6 simulator, now with the upper limit of the .AC (small-signal analysis) raised to 250.0 MHz, from 170 MHz. *The sudden roll-off in the frequency response as the input signal frequency exceeds the cutoff frequency is because the cutoff frequency of the reactive elements has been exceeded*. By default, SPICE uses ideal device models, so the effect will be more pronounced with real-world reactive elements. *Thus, although any filter can be designed (to operate in the 100's of MHz to 10's of GHz)*

with the automated scheme, *its physical fabrication cannot use any discrete components*. To address filter fabrication issue, designers have exploited properties of the transmission line, specifically those of the microstrip line. In the 100's of MHz plus frequency range, the effective wavelength becomes comparable to transmission line dimensions, and wave propagation properties can be exploited to compute dimensions of the capacitors or inductors. These capacitors and inductors are then fabricated using printed circuit board manufacturing technology. Two vital concepts are *electrical length* and *characteristic impedance* of a transmission line, which enables the designer to answer a fundamental question—*how to use computed values of reactive elements to synthesize microstrip line segments that will replace those reactive elements*. This is possible because of two key microstrip line properties:

- An open-circuited microstrip line (infinite load impedance) acts as a capacitor
- A short-circuited microstrip (zero load impedance) line acts as an inductor

The microstrip segments are called *stubs*, and key equations to compute their dimensions are provided. In this regard, a powerful technique to synthesize LC networks from microstrip line (more generally transmission line) segments is Richard's rule or transformation. Kuroda's four identities build upon these rules and involve redundant transmission line stubs to realize practical filters. Since discrete reactive elements are replaced by microstrip stubs, filters designed for 100's of MHz and above are called *distributed filters*.

In addition using transmission line properties, distributed filter design must account for losses arising mainly from two mechanisms—*skin effect* and *dielectric loss*. At 100's of MHz plus frequencies, signal attenuation in a microstrip/transmission line is a combination of conductor or ohmic loss (skin effect), dielectric or substrate loss (dielectric loss), and to a lesser extent radiation loss, propagation losses, and high-order mode loss. A number of researchers have analyzed/examined these loss mechanisms in detail and have derived analytical expressions to measure these losses (literature references are provided at the end of Chap. 3). The design of a low pass distributed filter, which uses high and low characteristic impedance microstrip stubs and also accounts for the two loss mechanisms, has been discussed at the end of Chap. 4. The full-blown implementation of the automated filter design scheme is currently being upgraded so that a designer can use it to design microstrip-based electronic filters that account for losses in the calculation steps. This upgraded full-blown implementation of the automated electronic filter design tool will be available in the near future.

Appendix A: Using the Automated Filter Design Tool

The implementation of the simplified version of the automated electronic filter design scheme is available as a pre-compiled executable for both Linux and Microsoft Windows operating systems. Each of these operating system-specific executables takes filter designer input from the command line, and the output in each case is a text file containing the filter design in SPICE input netlist format. Each of these text-based SPICE input netlist format files needs to be edited to add signal sources and specify analysis method to be used for a given filter design. Typically, the analysis method is small signal (.AC) that generates the frequency and phase responses, but other analysis techniques as transient (.TRAN) can be used—this is the designer's choice. A graphical user interface (GUI)-based version for each operating system will be provided in the near future, after testing. As it is, the C language code of the simplified version of the filter design scheme is being modified and upgraded regularly, and newer executable versions for both operating systems will be available in the near future.

The full-blown implementation of the automated filter design scheme is currently being upgraded with modules that will enable the designer to design and analyze filters specific to the ultrahigh-frequency ranges (100's of MHz to 10's of GHz). This full-blown version will be available in the near future, after exhaustive testing.

A typical command line input to be used to design a filter (e.g., a Chebyshev low pass) has been illustrated in Chap. 3. Extensive help instructions guide the user.

© Springer International Publishing Switzerland 2017

A. Banerjee, *Automated Electronic Filter Design*,
DOI 10.1007/978-3-319-43470-4