Chapter

4

Chebyshev Filters

Chebyshev filters are designed to have an amplitude response characteristic that has a relatively sharp transition from the pass band to the stop band. This sharpness is accomplished at the expense of ripples that are introduced into the response. Specifically, Chebyshev filters are obtained as an equiripple approximation to the pass band of an ideal lowpass filter. This results in a filter characteristic for which

$$|H(j\omega)|^2 = \frac{1}{1 + \epsilon^2 T_n^2(\omega)} \tag{4.1}$$

where $\epsilon^2 = 10^{r/10} - 1$

$T_n(\omega)$ = Chebyshev polynomial of order n

r = passband ripple, dB

Chebyshev polynomials are listed in Table 4.1.

TABLE 4.1 Chebyshev Polynomials

n	$T_n(\omega)$
0	1
1	ω
2	$2\omega^2 - 1$
3	$4\omega^3 - 3\omega$
4	$8\omega^4 - 8\omega^2 + 1$
5	$16\omega^5 - 20\omega^3 + 5\omega$
6	$32\omega^6 - 48\omega^4 + 18\omega^2 - 1$
7	$64\omega^7 - 112\omega^5 + 56\omega^3 - 7\omega$
8	$128\omega^8 - 256\omega^6 + 160\omega^4 - 32\omega^2 + 1$
9	$256\omega^9 - 576\omega^7 + 432\omega^5 - 120\omega^3 + 9\omega$
10	$512\omega^{10} - 1280\omega^8 + 1120\omega^6 - 400\omega^4 + 50\omega^2 + 1$

4.1 Transfer Function

The general shape of the Chebyshev magnitude response will be as shown in Fig. 4.1. This response can be normalized as in Fig. 4.2 so that the ripple bandwidth ω_r is equal to 1, or the response can be normalized as in Fig. 4.3 so that the 3-dB frequency ω_0 is equal to 1. Normalization based on the ripple bandwidth involves simpler calculations, but normalization based on the 3-dB point makes it easier to compare Chebyshev responses to those of other filter types.

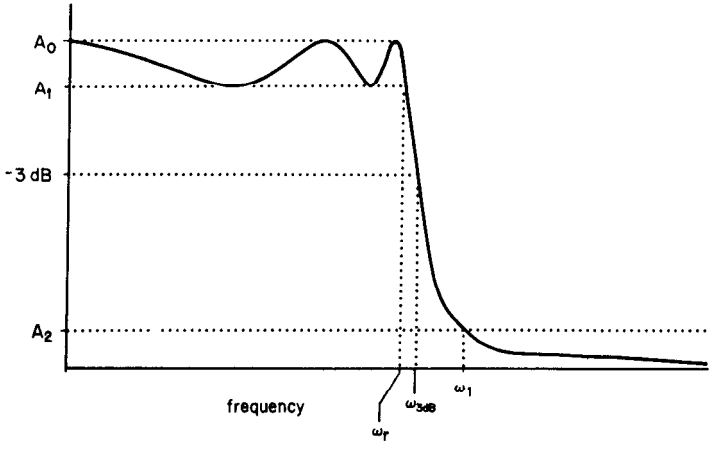

Figure 4.1 Magnitude response of a typical lowpass Chebyshev filter.

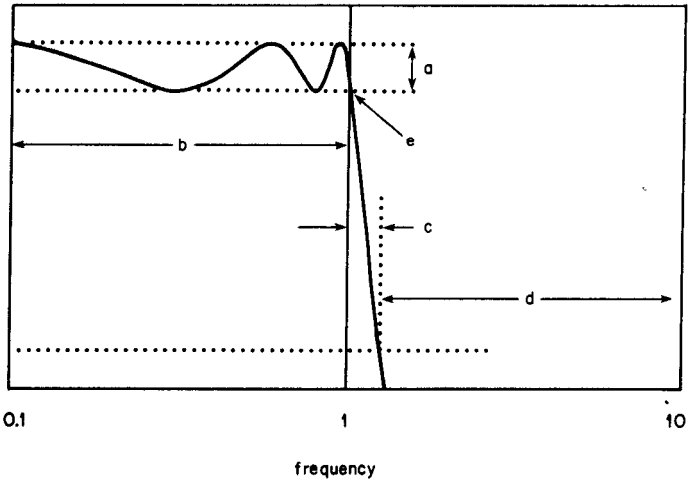

Figure 4.2 Chebyshev response normalized to have pass-band end at $\omega = 1$ rad/s. Features are: (*a*) ripple limits, (*b*) pass band, (*c*) transition band, (*d*) stop band, and (*e*) intersection of response and lower ripple limit at $\omega = 1$.

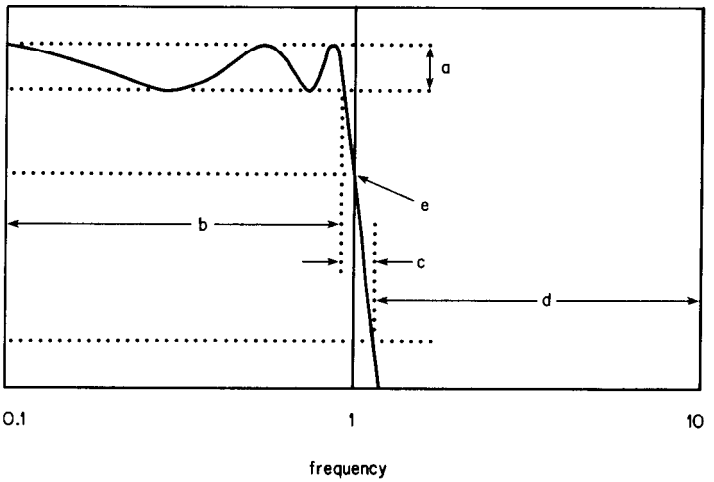

Figure 4.3 Chebyshev response normalized to have 3-dB point at $\hat{\omega} = 1$ rad/s. Features are: (*a*) ripple limits, (*b*) pass band, (*c*) transition band, (*d*) stop band, and (*e*) response that is 3 dB down at $\omega = 1$.

The general expression for the transfer function of an nth-order Chebyshev lowpass filter is given by

$$H(s) = \frac{H_0}{\prod_{i=1}^{n}(s - s_i)} = \frac{H_0}{(s - s_1)(s - s_2)\cdots(s - s_n)} \tag{4.2}$$

$$\text{where } H_0 = \begin{cases} \prod_{i=1}^{n}(-s_i) & n \text{ odd} \\ 10^{r/20}\prod_{i=1}^{n}(-s_i) & n \text{ even} \end{cases} \tag{4.3}$$

$$s_i = \sigma_i + j\omega_i \tag{4.4}$$

$$\sigma_i = \left[\frac{(1/\gamma) - \gamma}{2}\right]\sin\frac{(2i-1)\pi}{2n} \tag{4.5}$$

$$\omega_i = \left[\frac{(1/\gamma) + \gamma}{2}\right]\cos\frac{(2i-1)\pi}{2n} \tag{4.6}$$

$$\gamma = \left(\frac{1 + \sqrt{1 + \epsilon^2}}{\epsilon}\right)^{1/n} \tag{4.7}$$

$$\epsilon = \sqrt{10^{r/10} - 1} \tag{4.8}$$

The pole formulas are somewhat more complicated than for the Butterworth filter examined in Chap. 3, and several parameters—ϵ, γ, and r—must be

determined before the pole values can be calculated. Also, all the poles are involved in the calculation of the numerator H_0.

Algorithm 4.1 Determining poles of a Chebyshev filter

This algorithm computes the poles of an nth-order Chebyshev lowpass filter normalized for a ripple bandwidth of 1 Hz.

Step 1. Determine the maximum amount (in decibels) of ripple that can be permitted in the pass-band magnitude response. Set r equal to or less than this value.

Step 2. Use Eq. (4.8) to compute ϵ.

Step 3. Select an order n for the filter that will ensure adequate performance.

Step 4. Use Eq. (4.7) to compute γ.

Step 5. For $i = 1, 2, \ldots, n$; use Eqs. (4.5) and (4.6) to compute the real part σ_i and imaginary part ω_i of each pole.

Step 6. Use Eq. (4.3) to compute H_0.

Step 7. Substitute the values of H_0 and s_1 through s_n into Eq. (4.2).

Example 4.1 Use Algorithm 4.1 to determine the transfer-function numerator and poles (normalized for ripple bandwidth equal to 1) for a third-order Chebyshev filter with 0.5-dB pass-band ripple.

solution Algorithm 5.1 produces the following results:

$$\epsilon = 0.349311 \qquad \gamma = 1.806477 \qquad s_1 = -0.313228 + 1.021928j$$

$$s_2 = -0.626457 \qquad s_3 = -0.313228 - 1.021928j \qquad H_0 = 0.715695$$

The form of Eq. (4.2) shows that an nth-order Chebyshev filter will always have n poles and no finite zeros. The poles will all lie on the left half of an ellipse in the s plane. The major axis of the ellipse lies on the $j\omega$ axis, and the minor axis lies on the σ axis. The dimensions of the ellipse and the locations of the poles will depend upon the amount of ripple permitted in the pass band. Values of pass-band ripple typically range from 0.1 to 1 dB. The smaller the pass-band ripple, the wider the transition band will be. In fact, for 0-dB ripple, the Chebyshev filter and Butterworth filter have exactly the same transfer-function and response characteristics. Pole locations for third-order Chebyshev filters having different ripple limits are compared in Fig. 4.4. Pole values for ripple limits of 0.1, 0.5, and 1 dB are listed in Tables 4.2, 4.3, and 4.4 for orders 2 through 8.

All the transfer functions and pole values presented so far are for filters normalized to have a ripple bandwidth of 1. Algorithm 4.2 can be used to renormalize the transfer function to have a 3-dB frequency of 1.

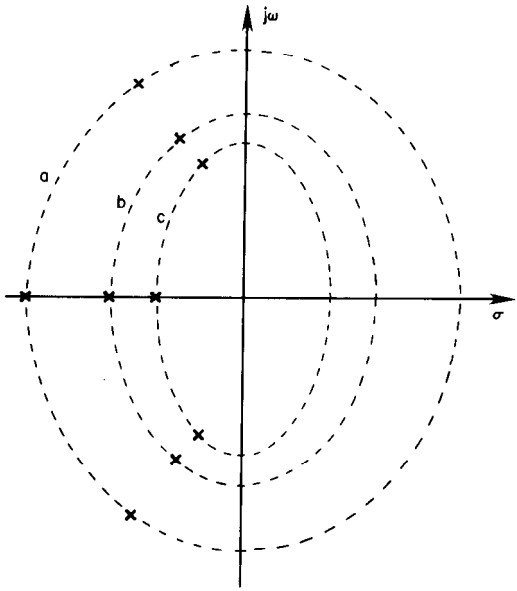

Figure 4.4 Comparison of pole locations for third-order lowpass Chebyshev filters with different amounts of pass-band ripple: (*a*) 0.01 dB, (*b*) 0.1 dB, and (*c*) 0.5 dB.

TABLE 4.2 Pole Values for Lowpass Chebyshev Filters with 0.1-dB Pass-Band Ripple

n	Pole values
2	$-1.186178 \pm 1.380948j$
3	-0.969406 $-0.484703 \pm 1.206155j$
4	$-0.637730 \pm 0.465000j$ $-0.264156 \pm 1.122610j$
5	-0.538914 $-0.435991 \pm 0.667707j$ $-0.166534 \pm 1.080372j$
6	$-0.428041 \pm 0.283093j$ $-0.313348 \pm 0.773426j$ $-0.114693 \pm 1.056519j$
7	-0.376778 $-0.339465 \pm 0.463659j$ $-0.234917 \pm 0.835485j$ $-0.083841 \pm 1.041833j$
8	$-0.321650 \pm 0.205314j$ $-0.272682 \pm 0.584684j$ $-0.182200 \pm 0.875041j$ $-0.063980 \pm 1.032181j$

TABLE 4.3 Pole Values for Lowpass Chebyshev Filters with 0.5-dB Pass-Band Ripple

n	Pole values
2	$-0.712812 \pm 1.00402j$
3	-0.626457 $-0.313228 \pm 1.021928j$
4	$-0.423340 \pm 0.420946j$ $-0.175353 \pm 1.016253j$
5	-0.362320 $-0.293123 \pm 0.625177j$ $-0.111963 \pm 1.011557j$
6	$-0.289794 \pm 0.270216j$ $-0.212144 \pm 0.738245j$ $-0.077650 \pm 1.008461j$
7	-0.256170 $-0.230801 \pm 0.447894j$ $-0.159719 \pm 0.807077j$ $-0.057003 \pm 1.006409j$
8	$-0.219293 \pm 0.199907j$ $-0.185908 \pm 0.569288j$ $-0.124219 \pm 0.852000j$ $-0.043620 \pm 1.005002j$

TABLE 4.4 Pole Values for Lowpass Chebyshev Filters with 1.0-dB Pass-Band Ripple

n	Pole values
2	$-0.548867 \pm 0.895129j$
3	-0.494171
	$-0.247085 \pm 0.965999j$
4	$-0.336870 \pm 0.407329j$
	$-0.139536 \pm 0.983379j$
5	-0.289493
	$-0.234205 \pm 0.611920j$
	$-0.089458 \pm 0.990107j$
6	$-0.232063 \pm 0.266184j$
	$-0.169882 \pm 0.727227j$
	$-0.062181 \pm 0.993411j$
7	-0.205414
	$-0.185072 \pm 0.442943j$
	$-0.128074 \pm 0.798156j$
	$-0.045709 \pm 0.995284j$
8	$-0.175998 \pm 0.198206j$
	$-0.149204 \pm 0.564444j$
	$-0.099695 \pm 0.844751j$
	$-0.035008 \pm 0.996451j$

Algorithm 4.2 Renormalizing Chebyshev LPF transfer functions

This algorithm assumes that ϵ, H_0, and the pole values s_i have been obtained for the transfer function having a ripple bandwidth of 1.

Step 1. Compute A using

$$A = \frac{\cosh^{-1}[(1/\epsilon)]}{n} = \frac{1}{n} \log\left(\frac{1+\sqrt{1-\epsilon^2}}{\epsilon}\right)$$

Step 2. Using the value of A obtained in step 1, compute R as

$$R = \cosh A = \frac{e^A + e^{-A}}{2}$$

(Table 4.5 lists R factors for various orders and ripple limits. If the required combination can be found in this table, steps 1 and 2 can be skipped.)

Step 3. Use R to compute $H_{3\,\mathrm{dB}}(s)$ as

$$H_{3\,\mathrm{dB}}(s) = \frac{H_0/R^n}{\prod_{i=1}^{n}[s-(s_i/R)]}$$

TABLE 4.5 Factors for Renormalizing Chebyshev Transfer Functions

	Order						
Ripple	2	3	4	5	6	7	8
0.1	1.94322	1.38899	1.21310	1.13472	1.09293	1.06800	1.05193
0.2	1.67427	1.28346	1.15635	1.09915	1.06852	1.05019	1.03835
0.3	1.53936	1.22906	1.12680	1.08055	1.05571	1.04083	1.03121
0.4	1.45249	1.19348	1.10736	1.06828	1.04725	1.03464	1.02649
0.5	1.38974	1.16749	1.09310	1.05926	1.04103	1.03009	1.02301
0.6	1.34127	1.14724	1.08196	1.05220	1.03616	1.02652	1.02028
0.7	1.30214	1.13078	1.07288	1.04644	1.03218	1.02361	1.01806
0.8	1.26955	1.11699	1.06526	1.04160	1.02883	1.02116	1.01618
0.9	1.24176	1.10517	1.05872	1.03745	1.02596	1.01905	1.01457
1.0	1.21763	1.09487	1.05300	1.03381	1.02344	1.01721	1.01316
1.1	1.19637	1.08576	1.04794	1.03060	1.02121	1.01557	1.01191
1.2	1.17741	1.07761	1.04341	1.02771	1.01922	1.01411	1.01079
1.3	1.16035	1.07025	1.03931	1.02510	1.01741	1.01278	1.00978
1.4	1.14486	1.06355	1.03558	1.02272	1.01576	1.01157	1.00886
1.5	1.13069	1.05740	1.03216	1.02054	1.01425	1.01046	1.00801

4.2 Frequency Response

Figures 4.5 through 4.8 show the magnitude and phase responses for Chebyshev filters with pass-band ripple limits of 0.5 dB. For comparison purposes, Figs. 4.9 and 4.10 show Chebyshev pass-band responses for ripple limits of 0.1 and 1.0 dB. These plots are normalized for a cutoff frequency of 1 Hz. To

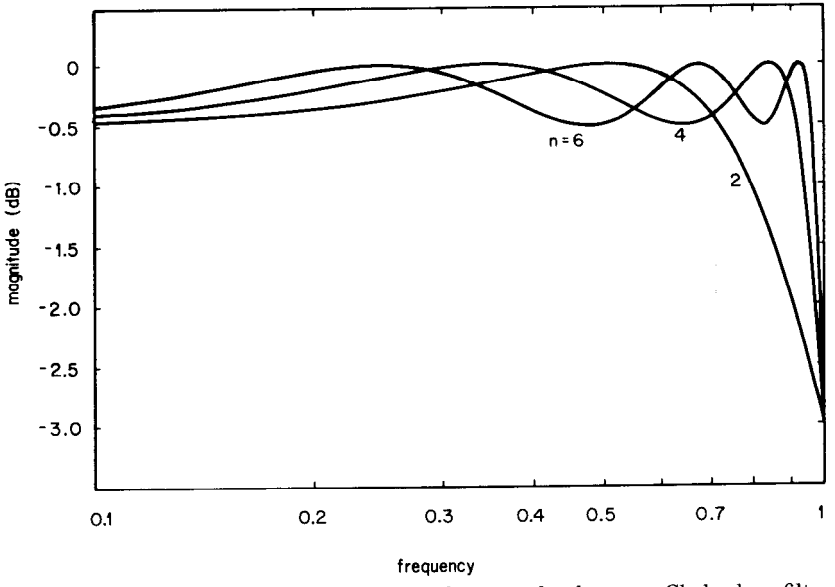

Figure 4.5 Pass-band magnitude response of even-order lowpass Chebyshev filters with 0.5-dB ripple.

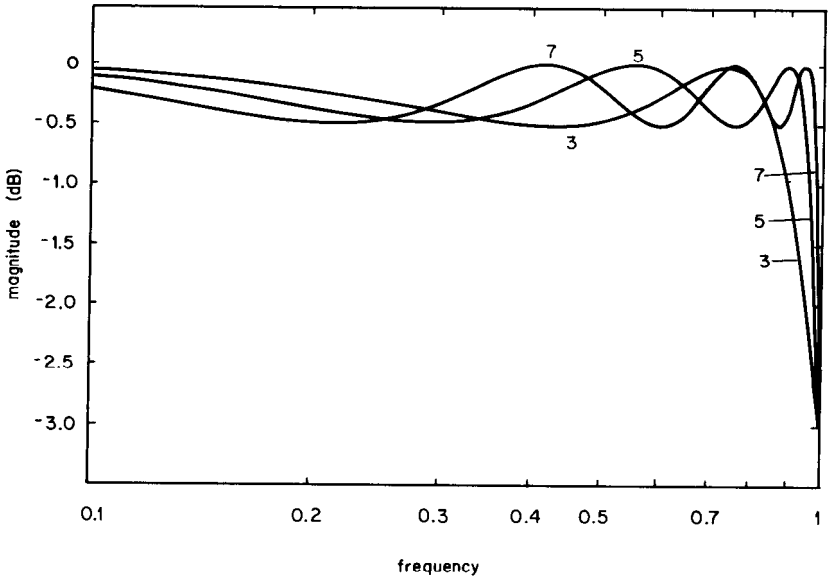

Figure 4.6 Pass-band magnitude response of odd-order lowpass Chebyshev filters with 0.5-dB ripple.

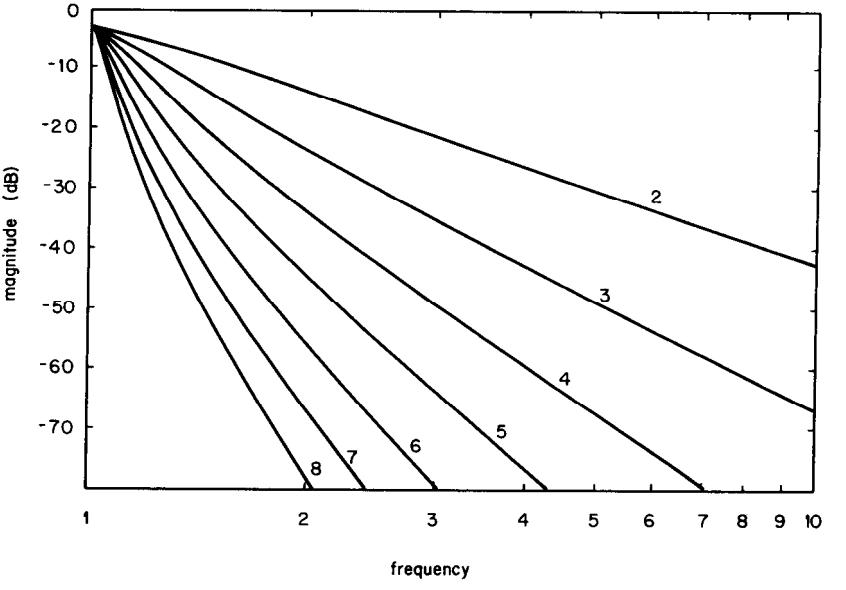

Figure 4.7 Stop-band magnitude response of lowpass Chebyshev filters with 0.5-dB ripple.

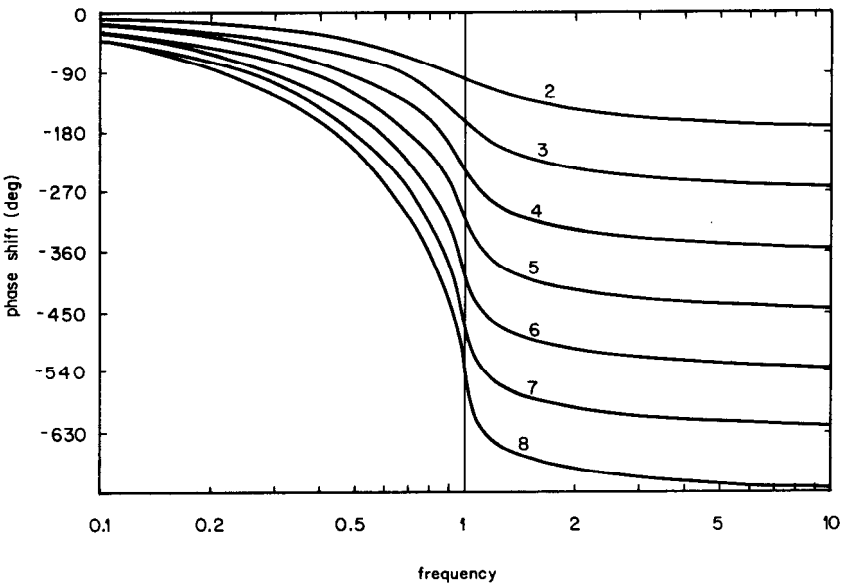

Figure 4.8 Phase response of lowpass Chebyshev filters with 0.5-dB pass-band ripple.

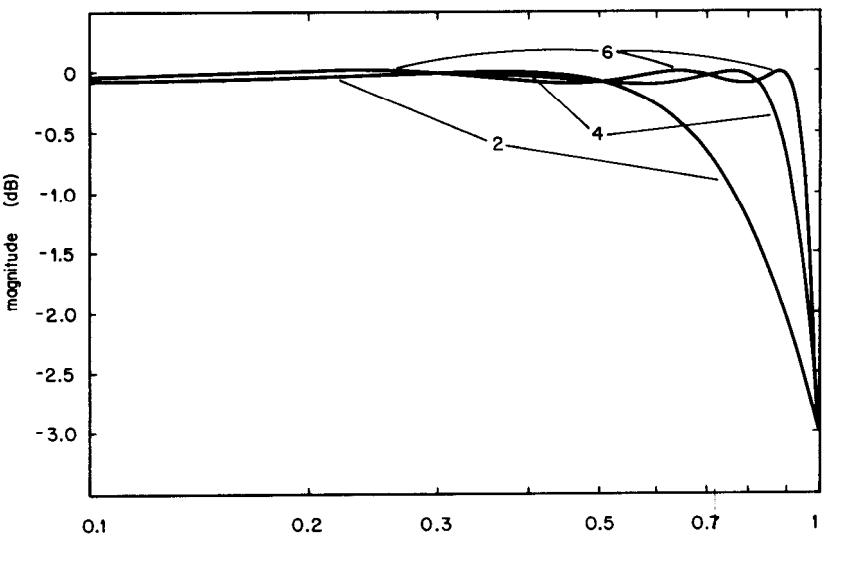

Figure 4.9 Pass-band magnitude response of even-order lowpass Chebyshev filters with 0.1-dB ripple.

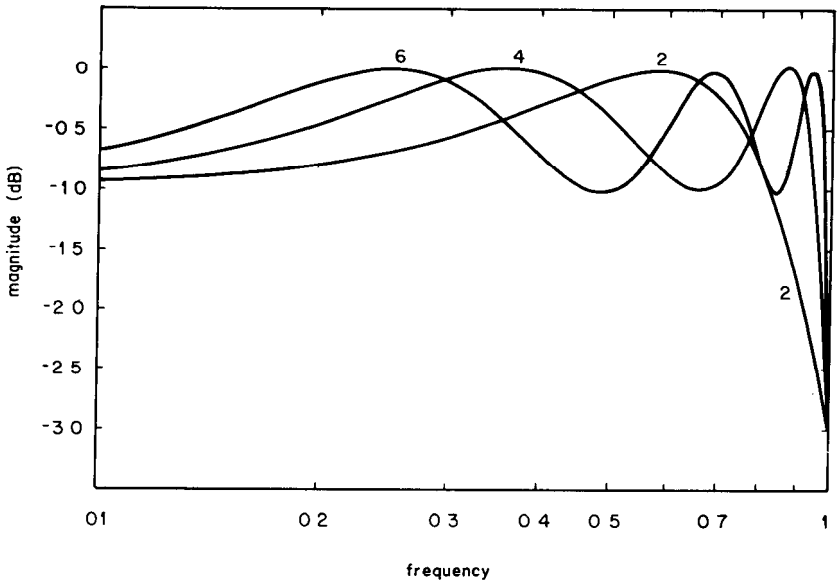

Figure 4.10 Pass-band magnitude response of even-order lowpass Chebyshev filters with 1.0-dB ripple.

denormalize them, simply multiply the frequency axis by the desired cutoff frequency f_c. The C function **chebyshevFreqResponse( )**, used to generate the Chebyshev frequency response data, is provided in Listing 4.1. Note that this function incorporates Algorithms 4.1 and 4.2.

4.3 Impulse Response

Impulse responses for lowpass Chebyshev filters with 0.5-dB ripple are shown in Fig. 4.11. The C routine **chebyshevImpulseResponse( )**, used to generate the data for these plots, is provided in Listing 4.2. These responses are normalized for lowpass filters having a 3-dB frequency of 1 Hz. To denormalize the response, divide the time axis by the desired cutoff frequency f_c and multiply the amplitude axis by the same factor.

4.4 Step Response

The step response can be obtained by integrating the impulse response. Step responses for lowpass Chebyshev filters with 0.5-dB ripple are shown in Fig. 4.12. These responses are normalized for lowpass filters having a cutoff frequency equal to 1 Hz. To denormalize the response, divide the time axis by the desired cutoff frequency f_c.

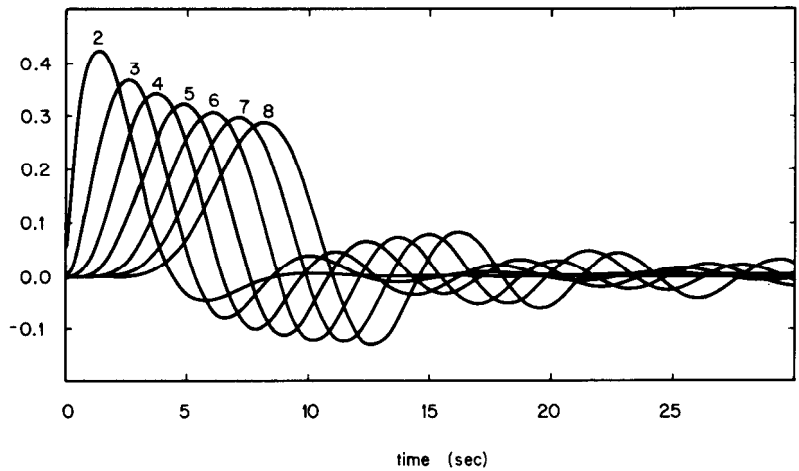

Figure 4.11 Impulse response of lowpass Chebyshev filters with 0.5-dB pass-band ripple.

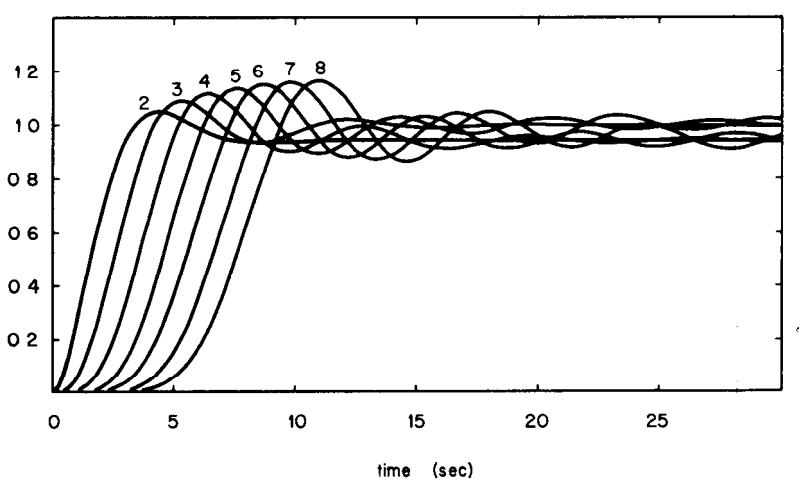

Figure 4.12 Step response of lowpass Chebyshev filters with 0.5-dB pass-band ripple.

Listing 4.1 chebyshevFreqResponse( )

```
/**********************************/
/*                                */
/*   Listing 4.1                  */
/*                                */
/*   chebyshevFreqResponse()      */
/*                                */
/**********************************/
#include <math.h>
#define PI (double) 3.141592653589


void chebyshevFreqResponse(    int order,
                               float ripple,
                               char normalizationType,
                               float frequency,
                               float *magnitude,
                               float *phase)
{
double A, gamma, epsilon, work;
double rp, ip, x, i, r, rpt, ipt;
double normalizedFrequency, hSubZero;
int k, ix;

epsilon = sqrt( -1.0 + pow( (double)10.0, (double)(ripple/10.0) ));
gamma = pow( (( 1.0 + sqrt( 1.0 + epsilon*epsilon))/epsilon),
                (double)(1.0/(float) order) );

if( normalizationType == '3' )
    {
    work = 1.0/epsilon;
    A = ( log( work + sqrt( work*work - 1.0) ) ) / order;
    normalizedFrequency = frequency * ( exp(A) + exp(-A))/2.0;
    }
else
    {
    normalizedFrequency = frequency;
    }

rp = 1.0;
ip = 0.0;

for( k=1; k<=order; k++)
    {
    x = (2*k-1) * PI / (2.0*order);
    i = 0.5 * (gamma + 1.0/gamma) * cos(x);
    r = -0.5 * (gamma - 1.0/gamma) * sin(x);
```

```
        rpt = ip * i - rp * r;
        ipt = -rp * i - r * ip;
        ip = ipt;
        rp = rpt;
        }
    hSubZero = sqrt( ip*ip + rp*rp);
    if( order%2 == 0 )
        {
        hSubZero = hSubZero / sqrt(1.0 + epsilon*epsilon);
        }

    rp = 1.0;
    ip = 0.0;
    for( k=1; k<=order; k++)
        {
        x = (2*k-1)*PI/(2.0*order);
        i = 0.5 * (gamma + 1.0/gamma) * cos(x);
        r = -0.5 * (gamma - 1.0/gamma) * sin(x);
        rpt = ip*(i-normalizedFrequency) - rp*r;
        ipt = rp*(normalizedFrequency-i) - r*ip;
        ip=ipt;
        rp=rpt;
        }
    *magnitude = 20.0 * log10(hSubZero/sqrt(ip*ip+rp*rp));
    *phase = 180.0 * atan2( ip, rp) /PI;
    return;
    }
```

Listing 4.2 chebyshevImpulseResponse( )

```
/*************************************************/
/*                                               */
/*   Listing 4.2                                 */
/*                                               */
/*   chebyshevImpulseResponse()                  */
/*                                               */
/*************************************************/
#include <math.h>
#define PI (double) 3.141592653589


void chebyshevImpulseResponse(    int order,
                                  float ripple,
                                  char normalizationType,
                                  float delta_t,
                                  int npts,
                                  float yval[])
```

```
{
double m, p;
double A, gamma, epsilon, work, normFactor;
double rp, ip, x, i, r, rpt, ipt, ss;
double hSub2ero, h_of_t, t, sigma, omega;
double K, L, M, LT, MT, I, R, cosPart, sinPart;
int k, ix, ii, iii, rrr;

epsilon = sqrt( -1.0 + pow( (double)10.0, (double)(ripple/10.0) ));

if( normalizationType == '3')
    {
    work = 1.0/epsilon;
    A = ( log( work + sqrt( work*work - 1.0) ) ) / order;
    normFactor = ( exp(A) + exp(-A))/2.0;
    }
else
    {
    normFactor = 1.0;
    }



gamma = pow( (( 1.0 + sqrt( 1.0 + epsilon*epsilon))/epsilon),
                (double)(1.0/(float) order) );

/*-------------------------------*/
/*  compute H_zero               */
rp = 1.0;
ip = 0.0;

for( k=1; k<=order; k++)
    {
    x = (2*k-1) * PI / (float)(2*order);
    i = 0.5 * (gamma + 1.0/gamma) * cos(x)/normFactor;
    r = -0.5 * (gamma - 1.0/gamma) * sin(x)/normFactor;
    rpt = ip * i - rp * r;
    ipt = -rp * i - r * ip;
    ip = ipt;
    rp = rpt;
    }
hSub2ero = sqrt( ip*ip + rp*rp);
if( order%2 == 0 )
    {
    hSub2ero = hSub2ero / sqrt(1.0 + epsilon*epsilon);
    }
printf("hSub2ero = %f\n",hSub2ero);
/*---------------------------------------*/
```

```
for( ix=0; ix<npts; ix++)
    {
    printf("%d\n",ix);
    h_of_t = 0.0;
    t = delta_t * ix;
    for( rrr=1; rrr <= (order >> 1); rrr++)
        {
        x = (2*rrr-1)*PI/(2.0*order);
        sigma = -0.5 * (gamma - 1.0/gamma) * sin(x)/normFactor;
        omega = 0.5 * (gamma + 1.0/gamma) * cos(x)/normFactor;




    /*  compute Lr and Mr   */

    L = 1;
    M = 0;

    for(ii=1; ii<=order; ii++)
        {
        if( ii == rrr) continue;
        x = (2*ii-1) * PI /(float)(2*order);
        R = sigma -(-0.5*(gamma -1.0/gamma))*sin(x) / normFactor;
        I = omega -(0.5*(gamma +1.0/gamma))*cos(x) / normFactor;

        LT = L * R - M * I;
        MT = L * I + R * M;
        L = LT;
        M = MT;
        }
    L = LT / (LT * LT + MT * MT);
    M = -MT / (LT * LT + MT * MT);

    cosPart = 2.0 * L * exp(sigma*t) * cos(omega*t);
    sinPart = 2.0 * M * exp(sigma*t) * sin(omega*t);

    h_of_t = h_of_t + cosPart - sinPart;
    }
if( (order%2) == 0 )
    {
    yval[ix] = h_of_t * hSubZero;
    }
else
    {
    /*  compute the real exponential component   */
    /*  present in odd-order responses           */
```

```
K = 1;
L = 1;
M = 0;
rrr = (order+1) >> 1;

x = (2*rrr-1) * PI / (float)(2*order);


        sigma = -0.5 * (gamma - 1.0/gamma) * sin(x) / normFactor;
        omega = 0.5 * (gamma + 1.0/gamma) * cos(x) / normFactor;

        for( iii=1; iii<= order; iii++)
            {
            if(iii == rrr) continue;
            x = (2*iii-1) * PI / (float)(2*order);
            R = sigma -(-0.5*(gamma -1.0/gamma))*sin(x) / normFactor;
            I = omega -(0.5*(gamma +1.0/gamma))*cos(x) / normFactor;

            LT = L * R - M * I;
            MT = L * I + R * M;
            L = LT;
            M = MT;
            }
        K = LT / (LT*LT + MT*MT);
        h_of_t = h_of_t + K * exp(sigma*t);
        yval[ix] = h_of_t * hSub2ero;
        }
    }
return;
}
```