Chapter

3

Butterworth Filters

Butterworth lowpass filters (LPF) are designed to have an amplitude response characteristic that is as flat as possible at low frequencies and that is monotonically decreasing with increasing frequency.

3.1 Transfer Function

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

$$H(s) = \frac{1}{\prod_{i=1}^{n}(s - s_i)} = \frac{1}{(s - s_1)(s - s_2)\cdots(s - s_n)} \tag{3.1}$$

where $s_i = e^{j\pi[(2i+n-1)/2n]} = \cos\left(\pi\frac{2i+n-1}{2n}\right) + j\sin\left(\pi\frac{2i+n-1}{2n}\right)$ (3.2)

Example 3.1 Determine the transfer function for a lowpass third-order Butterworth filter.

solution The third-order transfer function will have the form

$$H(s) = \frac{1}{(s - s_1)(s - s_2)(s - s_3)}$$

The values for s_1, s_2, and s_3 are obtained from Eq. (3.2):

$$s_1 = \cos\left(\frac{2\pi}{3}\right) + j\sin\left(\frac{2\pi}{3}\right) = -0.5 + 0.866j$$

$$s_2 = e^{j\pi} = \cos(\pi) + j\sin(\pi) = -1$$

$$s_3 = \cos\left(\frac{4\pi}{3}\right) + j\sin\left(\frac{4\pi}{3}\right) = -0.5 - 0.866j$$

Thus,
$$H(s) = \frac{1}{(s + 0.5 - 0.866j)(s + 1)(s + 0.5 + 0.866j)}$$
$$= \frac{1}{s^3 + 2s^2 + 2s + 1}$$

The form of Eq. (3.1) indicates that an nth-order Butterworth filter will always have n poles and no finite zeros. Also true, but not quite so obvious, is the fact that these poles lie at equally spaced points on the left half of a circle in the s plane. As shown in Fig. 3.1 for the third-order case, any odd-order Butterworth LPF will have one real pole at $s = -1$, and all remaining poles will occur in complex conjugate pairs. As shown in Fig. 3.2 for the fourth-order case, the poles of any even-order Butterworth LPF will all occur in complex conjugate pairs. Pole values for orders 2 through 8 are listed in Table 3.1.

3.2 Frequency Response

A C function, **butterworthFreqResponse( )**, for generating Butterworth frequency response data is provided in Listing 3.1. Figures 3.3 through 3.5

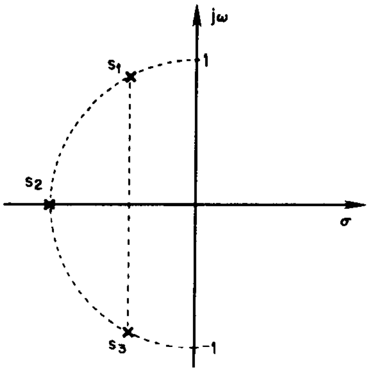

Figure 3.1 Pole locations for a third-order Butterworth LPF.

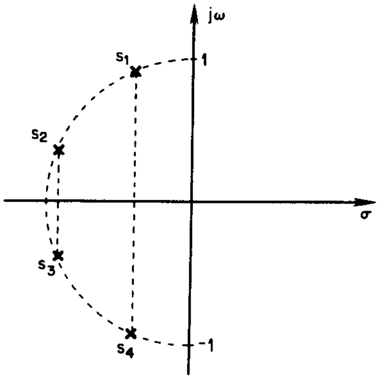

Figure 3.2 Pole locations for a fourth-order Butterworth LPF.

TABLE 3.1 Poles of Lowpass Butterworth Filters

n	Pole values
2	$-0.707107 \pm 0.707107j$
3	-1.0
	$-0.5 \pm 0.866025j$
4	$-0.382683 \pm 0.923880j$
	$-0.923880 \pm 0.382683j$
5	-1.0
	$-0.809017 \pm 0.587785j$
	$-0.309017 \pm 0.951057j$
6	$-0.258819 \pm 0.965926j$
	$-0.707107 \pm 0.707107j$
	$-0.965926 \pm 0.258819j$
7	-1.0
	$-0.900969 \pm 0.433884j$
	$-0.623490 \pm 0.781831j$
	$-0.222521 \pm 0.974928j$
8	$-0.195090 \pm 0.980785j$
	$-0.555570 \pm 0.831470j$
	$-0.831470 \pm 0.555570j$
	$-0.980785 \pm 0.195090j$

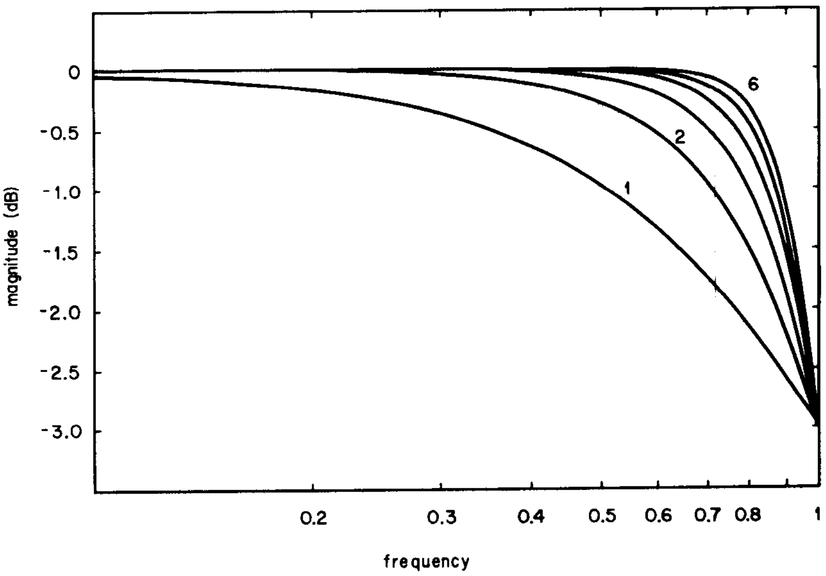

Figure 3.3 Pass-band amplitude response for lowpass Butterworth filters of orders 1 through 6.

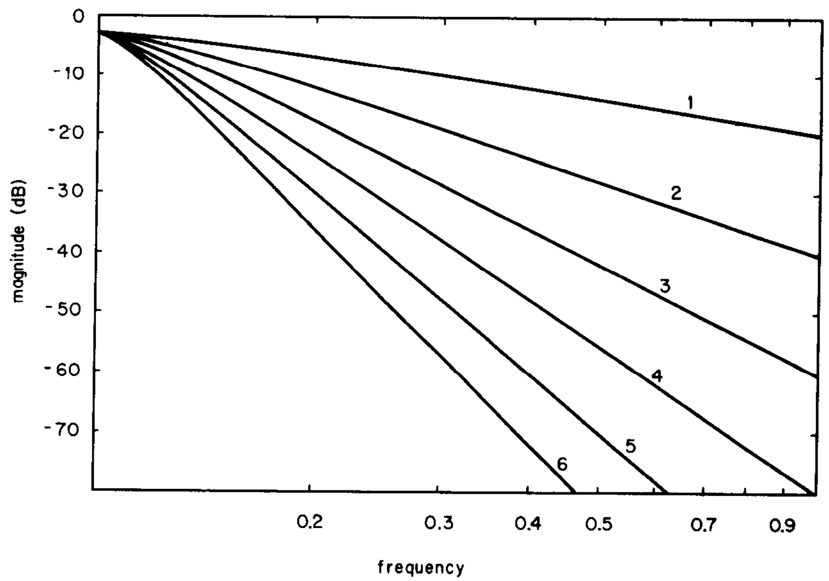

Figure 3.4 Stop-band amplitude response for lowpass Butterworth filters of orders 1 through 6.

show, respectively, the pass-band magnitude response, the stop-band magnitude response, and the phase response for Butterworth filters of various orders. These plots are normalized for a cutoff frequency of 1 Hz. To denormalize them, simply multiply the frequency axis by the desired cutoff frequency f_c.

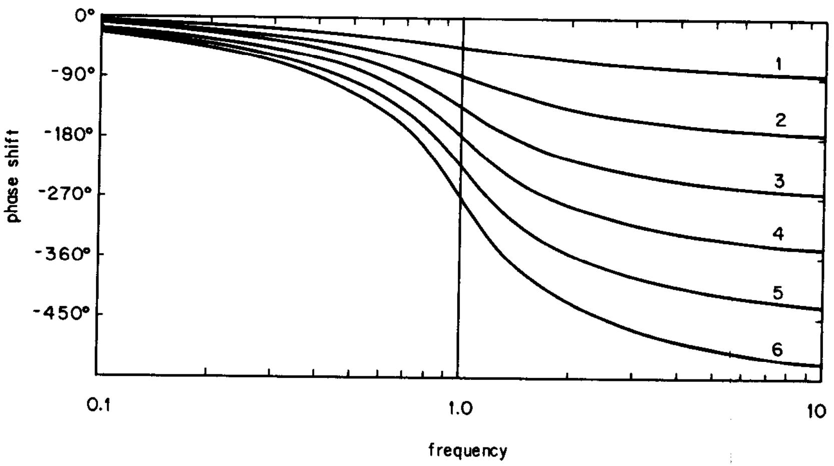

Figure 3.5 Phase response for lowpass Butterworth filters of orders 1 through 6.

Example 3.2 Use Figs. 3.4 and 3.5 to determine the magnitude and phase response at 800 Hz of a sixth-order Butterworth lowpass filter having a cutoff frequency of 400 Hz.

solution By setting $f_c = 400$, the $n = 6$ response of Fig. 3.4 is denormalized to obtain the response shown in Fig. 3.6. This plot shows that the magnitude at 800 Hz is approximately -36 dB. The corresponding response calculated by **butterworthFreqResponse( )** is -36.12466 dB. Likewise, the $n = 6$ response of Fig. 3.5 is denormalized to

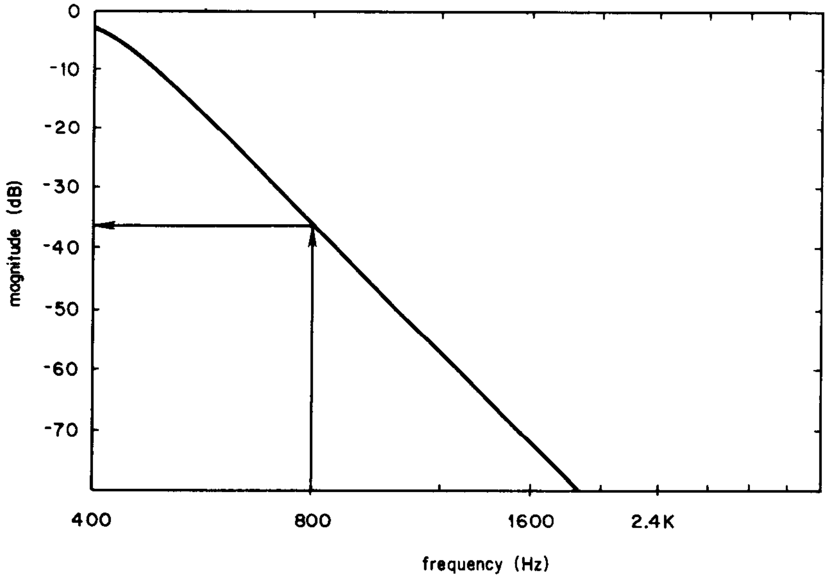

Figure 3.6 Denormalized amplitude response for Example 3.2.

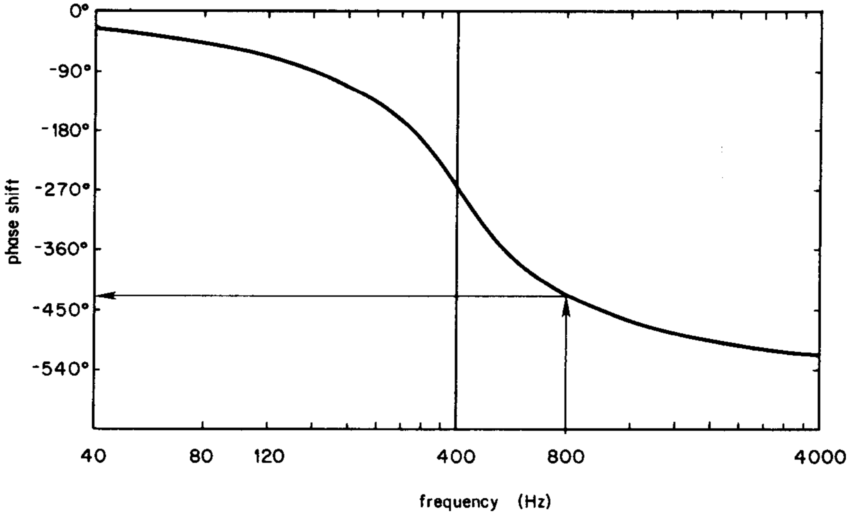

Figure 3.7 Denormalized phase response for Example 3.2.

obtain the response shown in Fig. 3.7. This plot shows that the phase response at 800 Hz is approximately $-425°$. The corresponding value calculated by **butterworthFreqResponse( )** is $-65.474°$, which "unwraps" to $-425.474°$.

3.3 Determination of Minimum Order for Butterworth Filters

Usually in the real world, the order of the desired filter is not given as in Example 3.2, but instead the order must be chosen based on the required performance of the filter. For lowpass Butterworth filters, the minimum order n that will ensure a magnitude of A_1 or lower at all frequencies ω_1 and above can be obtained by using

$$n = \frac{\log(10^{-A_1/10} - 1)}{2 \log(\omega_1/\omega_c)} \tag{3.3}$$

where ω_c = 3-dB frequency

ω_1 = frequency at which the magnitude response first falls below A_1

(*Note*: The value of A_1 is assumed to be in decibels. The value will be negative, thus canceling the minus sign in the numerator exponent.)

3.4 Impulse Response of Butterworth Filters

To obtain the impulse response for an nth-order Butterworth filter, we need to take the inverse Laplace transform of the transfer function. Application of the Heaviside expansion to Eq. (3.1) produces

$$h(t) = \mathscr{L}^{-1}[H(s)] = \sum_{r=1}^{n} K_r \, e^{s_r t} \tag{3.4}$$

$$\text{where } K_r = \left. \frac{(s - s_r)}{(s - s_1)(s - s_2) \cdots (s - s_n)} \right|_{s = s_r}$$

The values of both K_r and s_r are, in general, complex, but for the lowpass Butterworth case all the complex pole values occur in complex conjugate pairs. When the order n is even, this will allow Eq. (3.4) to be put in the form

$$h(t) = \sum_{r=1}^{n/2} [2 \operatorname{Re}(K_r) \, e^{\sigma_r t} \cos(\omega_r t) - 2 \operatorname{Im}(K_r) \, e^{\sigma_r t} \sin(\omega_r t)] \tag{3.5}$$

where $s_r = \sigma_r + j\omega_r$ and the roots s_r are numbered such that for $r = 1, 2, \ldots, n/2$ the s_r lie in the same quadrant of the s plane. [This last restriction prevents two members of the same complex conjugate pair from being used independently in evaluation of (3.5).] When the order n is odd, Eq. (3.4) can be put into the form

$$h(t) = K e^{-t} + \sum_{r=1}^{(n-1)/2} [2 \operatorname{Re}(K_r) \, e^{\sigma_r t} \cos(\omega_r t) - 2 \operatorname{Im}(K_r) \, e^{\sigma_r t} \sin(\omega_r t)] \tag{3.6}$$

where no two of the roots s_r, $r = 1, 2, \ldots, (n-1)/2$ form a complex conjugate pair. [Equations (3.5) and (3.6) form the basis for the C routine **butterworthImpulseResponse( )** provided in Listing 3.2.] This routine was used to generate the impulse responses for the lowpass Butterworth filters shown in Figs. 3.8 and 3.9. These responses are normalized for lowpass filters having a cutoff frequency equal to 1 rad/s. To denormalize the response, divide the time axis by the desired cutoff frequency $\omega_c = 2\pi f_c$ and multiply the time axis by the same factor.

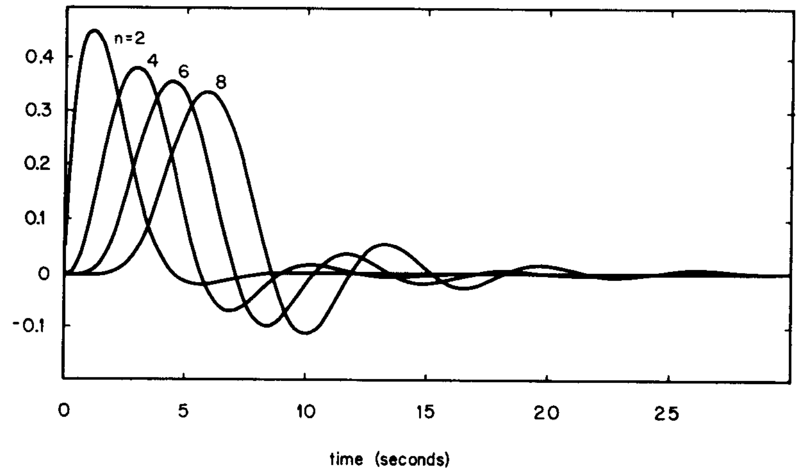

Figure 3.8 Impulse response of even-order Butterworth filters.

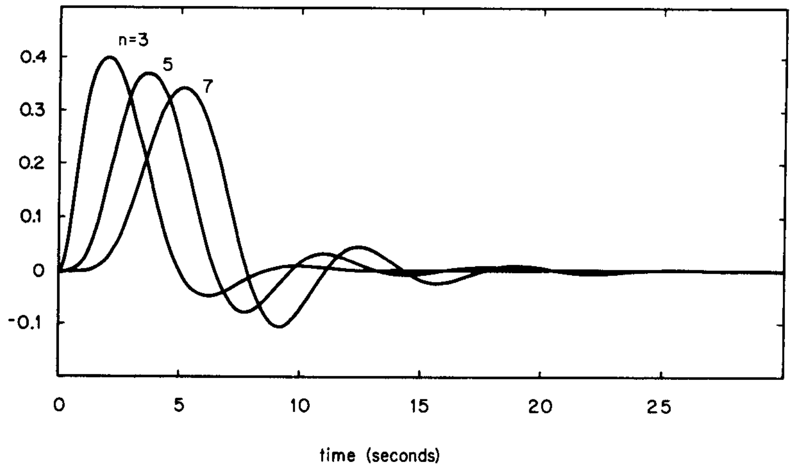

Figure 3.9 Impulse response of odd-order Butterworth filters.

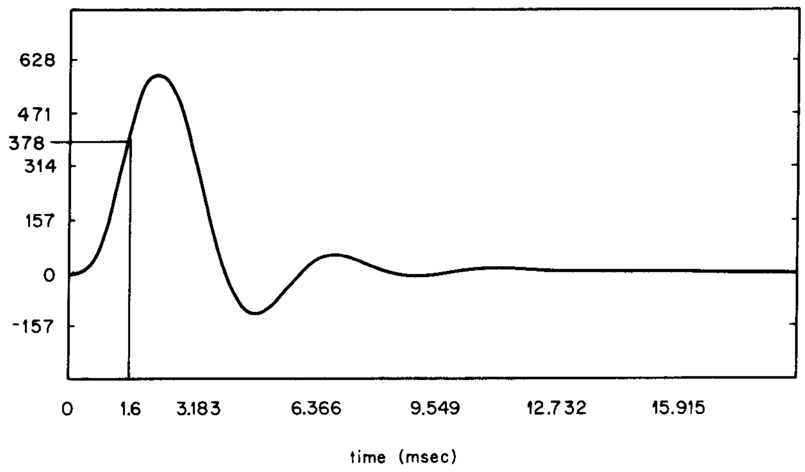

Figure 3.10 Denormalized impulse response for Example 3.3.

Example 3.3 Determine the instantaneous amplitude of the output 1.6 ms after a unit impulse is applied to the input of a fifth-order Butterworth LPF having $f_c = 250$ Hz.

solution The $n = 5$ response of Fig. 3.9 is denormalized as shown in Fig. 3.10. This plot shows that the response amplitude at $t = 1.6$ ms is approximately 378.

3.5 Step Response of Butterworth Filters

The step response can be obtained by integrating the impulse response. Step responses for lowpass Butterworth filters are shown in Figs. 3.11 and 3.12.

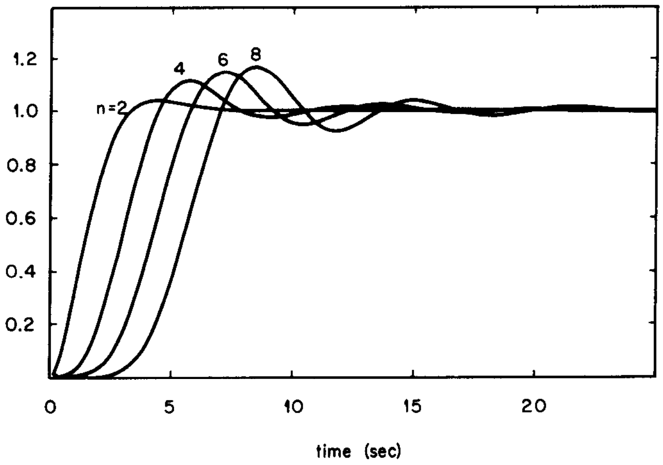

Figure 3.11 Step response of even-order lowpass Butterworth filters.

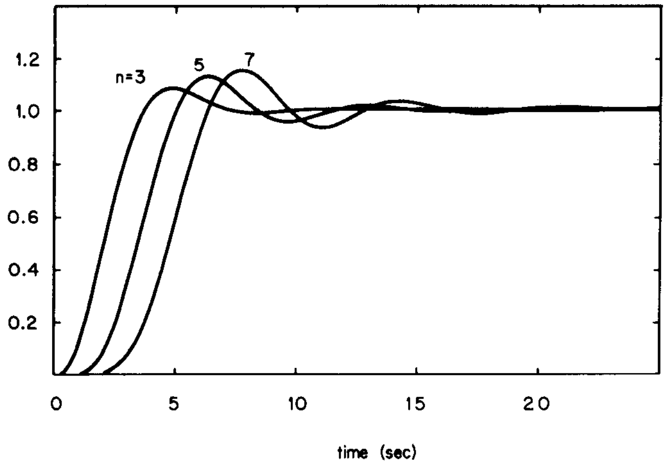

Figure 3.12 Step response of odd-order lowpass Butterworth filters.

These responses are normalized for lowpass filters having a cutoff frequency equal to 1 rad/s. To denormalize the response, divide the time axis by the desired cutoff frequency $\omega_c = 2\pi f_c$.

Example 3.4 Determine how long it will take for the step response of a third-order Butterworth LPF ($f_c = 4$ kHz) to first reach 100 percent of its final value.

solution By setting $\omega_c = 2\pi f_c = 8000\pi = 25{,}132.7$, the $n = 3$ response of Fig. 3.12 is denormalized to obtain the response shown in Fig. 3.13. This plot indicates that the step response first reaches a value of 1 in approximately 150 μs.

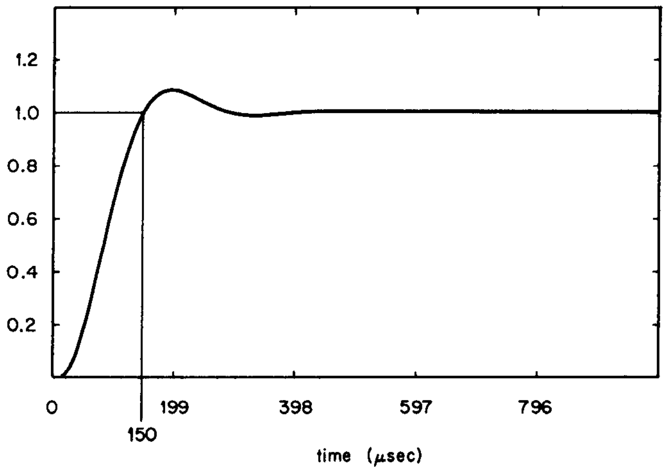

Figure 3.13 Denormalized step response for Example 3.4.

Listing 3.1 butterworthFreqResponse( )

```
/********************************/
/*                              */
/*   Listing 3.1                */
/*                              */
/*   butterworthFreqResponse()  */
/*                              */
/********************************/
#include <math.h>
#include <stdio.h>
#include "globDefs.h"
#include "protos.h"

void butterworthFreqResponse( int order,
                              real frequency,
                              real *magnitude,
                              real *phase)
{
struct complex pole, s, numer, denom, transferFunction;
real x;
int k;
numer = cmplx(1.0,0.0);
denom = cmplx(1.0,0.0);

s = cmplx(0.0, frequency);
for( k=1; k<=order; k++)
    {
    x = PI * ((double)(order + (2*k)-1)) / (double)(2*order);
    pole = cmplx( cos(x), sin(x));
    denom = cMult(denom, cSub(s,pole));
    }
transferFunction = cDiv(numer, denom);
*magnitude = 20.0 * log10(cAbs(transferFunction));
*phase = 180.0 * arg(transferFunction) / PI;
return;
}
```

Listing 3.2 butterworthImpulseResponse( )

```
/*************************************************/
/*                                               */
/*   Listing 3.2                                 */
/*                                               */
/*   butterworthImpulseResponse()                */
/*                                               */
/*************************************************/
#include <math.h>
#include <stdio.h>
#include "globDefs.h"
#include "protos.h"


void butterworthImpulseResponse(  int order,
                                  real delta_t,
                                  int npts,
                                  real yval[])
{
real L, M, x, R, I, LT, MT, cosPart, sinPart, h_of_t;
real K, sigma, omega, t;
int ix, r, ii, iii;
real ymax, ymin;

for( ix=0; ix <= npts; ix++)
    {
    printf("%d/n",ix);
    h_of_t = 0.0;
    t = delta_t * ix;
    for( r=1; r <= (order>>1); r++)
        {
        x = PI * (double)(order + (2*r)-1) / (double)(2*order);
        sigma = cos(x);
        omega = sin(x);


/*  Compute Lr and Mr    */

L = 1.0;
M = 0.0;
for( ii=1; ii<=order; ii++)
    {
    if( ii == r ) continue;
    x = PI * (double)(order + (2*ii)-1) / (double)(2*order);
    R = sigma - cos(x);
    I = omega - sin(x);
```

```
            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;
        if( (real) h_of_t > ymax) ymax = h_of_t;
        if( (real) h_of_t < ymin) ymin = h_of_t;
        continue;
        }
    /* compute the real exponential component for odd-order responses */

        K = 1.0;
        L = 1.0;
        M = 0.0;
        r = (order+1)/2;
        x = PI * (double)(order + (2*r)-1) / (double)(2*order);
        sigma = cos(x);
        omega = sin(x);
        for( iii=1; iii<=order; iii++)
            {
            if( iii == r) continue;
            x = PI * (double)(order + (2*iii)-1) / (double)(2*order);
            R = sigma - cos(x);
            I = omega - sin(x);

            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(-t);
        yval[ix] = h_of_t;
        if( (real) h_of_t > ymax) ymax = h_of_t;
        if( (real) h_of_t < ymin) ymin = h_of_t;
        }
    return;
```