Chapter

6

Bessel Filters

Bessel filters are designed to have maximally flat group-delay characteristics. As a consequence, there is no ringing in the impulse and step responses.

6.1 Transfer Function

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

$$H(s) = \frac{b_0}{q_n(s)} \tag{6.1}$$

where $q_n(s) = \sum_{k=1}^{n} b_k s^k$

$$b_k = \frac{(2n-k)!}{2^{n-k}\,k!(n-k)!}$$

The following recursion can be used to determine $q_n(s)$ from $q_{n-1}(s)$ and $q_{n-2}(s)$:

$$q_n = (2n-1)q_{n-1} + s^2 q_{n-2}$$

Table 6.1 lists $q_n(s)$ for $n = 2$ through $n = 8$. These values were generated by the C function **besselCoefficients( )** provided in Listing 6.1. This function is used by other Bessel filter routines presented later in this section.

Unlike the transfer function for Butterworth and Chebyshev filters, Eq. (6.1) does not provide an explicit expression for the poles of the Bessel filter. The numerator of (6.1) will be a polynomial in s, upon which numerical root-finding methods (such as Algorithm 2.1) must be used to determine the pole locations for $H(s)$. Table 6.2 lists approximate pole locations for $n = 2$ through $n = 8$.

TABLE 6.1 Denominator Polynomials for Transfer Functions of Bessel Filters Normalized to Have Unit Delay at $\omega = 0$

n	$q_n(s)$
2	$s^2 + 3s + 3$
3	$s^3 + 6s^2 + 15s + 15$
4	$s^4 + 10s^3 + 45s^2 + 105s + 105$
5	$s^5 + 15s^4 + 105s^3 + 420s^2 + 945s + 945$
6	$s^6 + 21s^5 + 210s^4 + 1260s^3 + 4725s^2 + 10{,}395s + 10{,}395$
7	$s^7 + 28s^6 + 378s^5 + 3150s^4 + 17{,}325s^3 + 62{,}370s^2 + 135{,}135s + 135{,}135$
8	$s^8 + 36s^7 + 630s^6 + 6930s^5 + 9450s^4 + 270{,}270s^3 + 945{,}945s^2 + 2{,}027{,}025s + 2{,}027{,}025$

TABLE 6.2 Poles of Bessel Filter Normalized to Have Unit Delay at $\omega = 0$

n	Pole values
2	$-1.5 \pm 0.8660j$
3	-2.3222 $-1.8390 \pm 1.7543j$
4	$-2.1039 \pm 2.6575j$ $-2.8961 \pm 0.8672j$
5	-3.6467 $-2.3247 \pm 3.5710j$ $-3.3520 \pm 1.7427j$
6	$-2.5158 \pm 4.4927j$ $-3.7357 \pm 2.6263j$ $-4.2484 \pm 0.8675j$
7	-4.9716 $-2.6857 \pm 5.4206j$ $-4.0701 \pm 3.5173j$ $-4.7584 \pm 1.7393j$
8	$-5.2049 \pm 2.6162j$ $-4.3683 \pm 4.4146j$ $-2.8388 \pm 6.3540j$ $-5.5878 \pm 0.8676j$

The transfer functions given by (6.1) are for Bessel filters normalized to have unit delay at $\omega = 0$. The poles p_k and denominator coefficients b_k can be renormalized for a 3-dB frequency of $\omega = 1$ using

$$p'_k = Ap_k \qquad b'_k = A^{n-k} b_k$$

where the value of A appropriate for n is selected from Table 6.3. (The values from the table have been incorporated in the **besselCoefficient( )** function.)

TABLE 6.3 Factors for Renormalizing Bessel Filter Poles from Unit Delay at $\omega = 0$ to 3-dB Attenuation at $\omega = 1$

n	A
2	1.35994
3	1.74993
4	2.13011
5	2.42003
6	2.69996
7	2.95000
8	3.17002

6.2 Frequency Response

Figures 6.1 and 6.2 show the magnitude responses for Bessel filters of several different orders. The frequency response data was generated by the C routine **besselFreqResponse( )**, which is provided in Listing 6.2.

6.3 Group Delay

Group delays for lowpass Bessel filters of several different orders are plotted in Fig. 6.3. The data for these plots was generated by the C function **besselGroupDelay( )**, provided in Listing 6.3, which performs numerical differentiation of the phase response to evaluate the group delay.

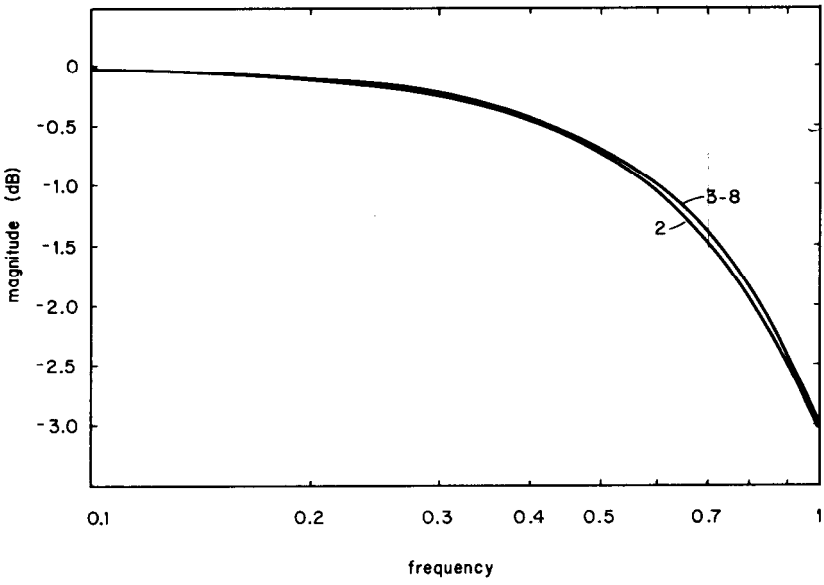

Figure 6.1 Pass-band magnitude response of lowpass Bessel filters.

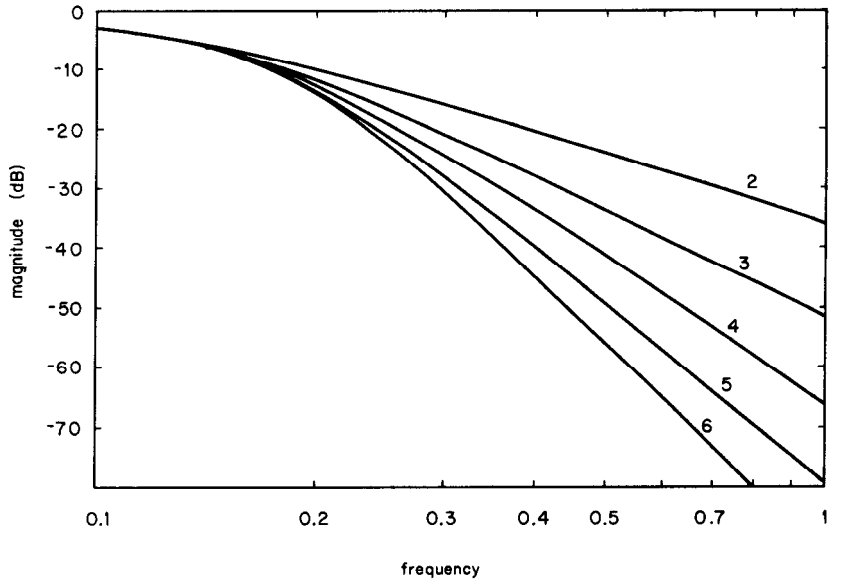

Figure 6.2 Stop-band magnitude response of lowpass Bessel filters.

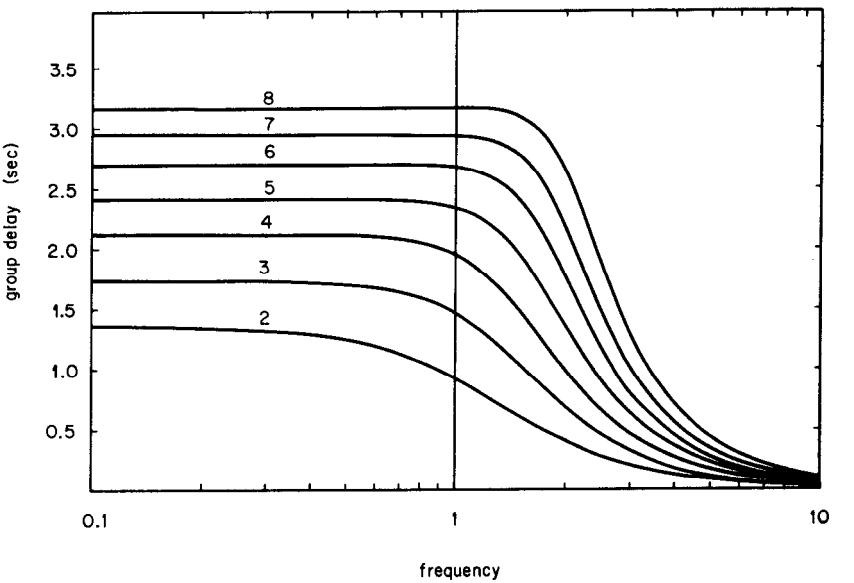

Figure 6.3 Group-delay response of lowpass Bessel filters.

Listing 6.1 besselCoefficients( )

```
/***********************************/
/*                                 */
/*   Listing 6.1                   */
/*                                 */
/*   besselCoefficients()          */
/*                                 */
/***********************************/
#include <math.h>
#include "globDefs.h"

void besselCoefficients(  int order,
                          char typeOfNormalization,
                          real coef[])
{
int i, N, index, indexM1, indexM2;
real B[3][MAXORDER];
real A, renorm[MAXORDER];

renorm[2] = 0.72675;
renorm[3] = 0.57145;
renorm[4] = 0.46946;
renorm[5] = 0.41322;
renorm[6] = 0.37038;
renorm[7] = 0.33898;
renorm[8] = 0.31546;
A = renorm[order];

index = 1;
indexM1 = 0;
indexM2 = 2;

for( i=0; i<(3*MAXORDER); i++) B[0][i] = 0;
B[0][0] = 1.0;
B[1][0] = 1.0;
B[1][1] = 1.0;
for( N=2; N<=order; N++)
    {
    index = (index+1)%3;
    indexM1 = (indexM1 + 1)%3;
    indexM2 = (indexM2 + 1)%3;

    for( i=0; i<N; i++)
        {
        B[index][i] = (2*N-1) * B[indexM1][i];
        }
    for( i=2; i<=N; i++)
```

```
        {
        B[index][i] = B[index][i] + B[indexM2][i-2];
        }
    }
if(typeOfNormalization == 'D')
    {
    for( i=0; i<=order; i++) coef[i] = B[index][i];
    }
else
    {
    for( i=0; i<=order; i++)
        {
        coef[i] = B[index][i] * pow(A, (order - i) );
        }
    }
return;
}
```

Listing 6.2 besselFreqResponse( )

```
/**********************************/
/*                                */
/*   Listing 6.2                  */
/*                                */
/*   besselFreqResponse()         */
/*                                */
/**********************************/
#include <math.h>
#include "globDefs.h"
#include "protos.h"


void besselFreqResponse(   int order,
                           real coef[],
                           real frequency,
                           real *magnitude,
                           real *phase)
{
struct complex numer, omega, denom, transferFunction;
int i;

numer = cmplx( coef[0], 0.0);
omega = cmplx( 0.0, frequency);
denom = cmplx( coef[order], 0.0);

for( i=order-1; i>=0; i--)
    {
```

```
    denom = cMult(omega,denom);
    denom.Re = denom.Re + coef[i];
    }
transferFunction = cDiv( numer, denom);

*magnitude = 20.0 * log10(cAbs(transferFunction));
*phase = 180.0 * arg(transferFunction) / PI;
return;
}
```

Listing 6.3 besselGroupDelay( )

```
/**********************************/
/*                                */
/*   Listing 6.3                  */
/*                                */
/*   besselGroupDelay()           */
/*                                */
/**********************************/

void besselGroupDelay(    int order,
                          real coef[],
                          real frequency,
                          real delta,
                          real *groupDelay)
{
struct complex numer, omega, omegaPlus, denom, transferFunction;
int i;
real phase, phase2;

numer = cmplx( coef[0], 0.0);
denom = cmplx( coef[order], 0.0);
omega = cmplx( 0.0, frequency);

for( i=order-1; i>=0; i--) {
    denom = cMult(omega,denom);
    denom.Re = denom.Re + coef[i];
    }
transferFunction = cDiv( numer, denom);
phase = arg(transferFunction);

denom = cmplx( coef[order], 0.0);
omegaPlus = cmplx(0.0, frequency + delta);

for( i=order-1; i>=0; i--) {
    denom = cMult(omegaPlus,denom);
```

```
    denom.Re = denom.Re + coef[i];
    }
transferFunction = cDiv( numer, denom);
phase2 = arg(transferFunction);
*groupDelay = (phase2 - phase)/delta;
return;
}
```