INTRODUCTION

This lab demonstrates the use of digital filters on a DSP (digital signal processor). It consists of two steps, the first being the design of some filters using MATLAB and LabVIEW, and the second, implementing those filters, as well as some others on the DSP system.

The first part involves the design of some fourth order and higher filters using MATLAB and a filter design toolkit within LabVIEW. To help you towards this end, the Appendix provides a list of MATLAB commands useful for filter design. Further details may be found in the MATLAB help pages. Instructions for using the National Instruments LabVIEW Digital Filter Design toolkit are included later in the lab procedure.

The second part involves the implementation and analysis of these filters on the DSP board. Two methods of implementation will be evaluated. In order to gain some familiarity with digital filter design, you are required to solve the following problem prior to the first lab session:

Convert the analog low-pass filter: \( H(s) = \frac{1}{1 + sRC} \) to an equivalent digital low-pass filter. The filter should have a cut-off frequency of 143 rad/sec. Use the impulse invariant transformation with a sampling period \( T \) of 7 msec. Your answer should be in the \( Z \) domain, and should include an associated block diagram.

A single fourth order digital filter may be implemented two different ways mathematically:

\[
H(z) = \frac{A + Bz^{-1} + Cz^{-2} + F + Gz^{-1} + Hz^{-2}}{1 + Dz^{-1} + Ez^{-2} + 1 + Iz^{-1} + Jz^{-2}}
\]

and

\[
H(z) = \frac{A + Bz^{-1} + Cz^{-2} \times F + Gz^{-1} + Hz^{-2}}{1 + Dz^{-1} + Ez^{-2} + 1 + Iz^{-1} + Jz^{-2}}
\]

Block diagrams of the two filter forms are shown in Figure 1 and Figure 2, however only the second form (cascade Type II) is implemented by the filter toolkit. The filters are run on a Speedy-33
DSP under LabVIEW. There are a few items to note here. Only the left channel is wired in the BNC-1/8” mini stereo plug cables in the lab. The A/D input used for the incoming analog signal has a range $1.2V_{p-p}$ and the D/A output has a maximum range of a little over $3.5V_{p-p}$. Although the DSP system has good low frequency response, as an audio system it purposely blocks DC to prevent damage to components. This system will attenuate frequencies below about 0.5Hz. All filters implemented by the LabVIEW tool have a gain of 3 when a unity gain would be expected. This is consistent and should be taken into consideration when analyzing data.

**FIGURE 1.** Parallel Filter (Type I).
BACKGROUND

THE APPROXIMATION PROBLEM

The approximation problem is one of finding a match between the idealized frequency response desired, and the various responses possible.

The ideal low-pass filter response is as shown in Figure 3a. The filter has a gain equal to 1 for $|f| < f_c$, and a gain equal to 0 for $|f| > f_c$. This response is practically and theoretically unrealizable. Consider the inverse Fourier transform of this filter. It is a sinc pulse centered at $t = 0$ (Figure 3b), which is a non-causal output. A time delay can be added to the filter and the response is now as in Figure 3c. For a large enough delay, $h(t)$ will be negligible for $t < 0$, and can be approximated by a realizable filter.

FIGURE 2. Cascade Filter (Type II).
There are three main types of low-pass filter approximations. They are the Butterworth (or maximally flat), the Chebyshev (two versions), and the elliptic approximations.

The Butterworth low-pass filter of order $n$ has an amplitude ratio given by
\[|H(f)| = \left[1 + \left(\frac{f}{B}\right)^{2n}\right]^{-\frac{1}{2}}\]

This filter, whose Bode plot is shown in Figure 4, has a gain which decreases monotonically from unity at \(f = 0\)Hz. As \(n\) (the filter order) is increased, the rate of fall off of the filter at its cutoff frequency is increased. This is not without a penalty, because as the filter degree increases, the phase shift gets worse, and the impulse response does not follow the sinc pulse very closely.

The Chebyshev filter Type I (but not to be confused with parallel Type I implementation) has a ripple in the pass-band, and decreases monotonically in the stop-band. The Type II filter reverses these bands. A typical frequency response is shown in Figure 5. This filter has the advantage of a faster rate of fall off, and a more linear phase shift. In the pass-band, the magnitude of the frequency response fluctuates between 1 and \(1/(1 + e^{2})^{1/2}\). For a larger \(e\), the ripple is larger but the fall off is faster. There is a design trade-off between the ripple size and the fall off for a given filter order.

The elliptic filter allows ripples in both the pass and stop-bands, as shown in Figure 6. This has the fastest fall off rate of the three filter types but has a large phase shift. This filter again has a trade off between ripple size and fall-off rate. For further details on analog filter types see reference [3].

**BAND-PASS FILTER DESIGN**

A band-pass filter with center pass-band frequency \(w_0\) can be derived from a low-pass filter by using the low-pass to band-pass transformation.

A pole-zero pattern and frequency response curve for a typical low-pass filter is shown in Figure 7a. To make a band-pass filter you might try to make the substitution \(s \rightarrow s - jw_0\) to move the poles up to \(jw_0\). This would not work because any circuit built with real elements must have all complex poles and zeros in complex conjugate pairs.

A substitution that does work is the replacement of the Laplace domain variable \(s\) in the low-pass filter \(H(s)\) by

\[
s \rightarrow \frac{(s_b - j\omega_0)(s_b + j\omega_0)}{2s_b} = \frac{s_b^2 + \omega_0^2}{2s_b}
\]

where \(s_b\) is the Laplace variable of the transformed band-pass filter. Then, for frequencies of operation close to the center frequency \(w_0\) (i.e. \(s_b\) is approx equal to \(jw_0\)), the transformed low-pass filter becomes

\[H(s) = H\left(\frac{(s_b - j\omega_0)(s_b + j\omega_0)}{2s_b}\right) = \frac{H(j\Delta)2s_b}{2s_b} = H(j\Delta)\]

where \(D\) is the deviation from \(w_0\). Thus, the shape and bandwidth of the low-pass filter are preserved. This transformation leads to complex conjugate poles and zeros as shown in Figure 7b, and is therefore realizable.

FIGURE 4b. Butterworth Low-Pass Filter (2nd Order).

FIGURE 4c. Butterworth Low-Pass Filter (3rd Order).
FIGURE 5. Chebyshev Type I Low-Pass Filter (pass band ripple).

FIGURE 6. Elliptic Low-Pass Filter (pass and stop band ripple).

FIGURE 7a. Low-Pass Filter.
DISCRETE TIME SYSTEMS

A discrete signal is an ordered sequence of numbers. If you sampled a continuous signal \( x(t) \) every \( T \) seconds, your output would be a function \( (x(kT) : k = 0, 1, 2, \ldots) \), which is a discrete signal (i.e., a series of values occurring every \( T \) seconds). A discrete system is one in which all the variables are discrete signals.

A discrete system is analogous to a continuous system in many ways. The output of the system at any future time is known if you know the system’s present state and the input.

A state variable equation can be written as \( y(kT) = S[q_0 : x(kT)] \) \( k \geq k_0 \), where \( x(kT) \) is the input, \( q_0 \) is the initial state at \( k = k_0 \), and \( y(kT) \) is the output. A fixed, linear discrete system will obey the principles of decomposability, superposition, and time invariance (see reference [2]). Discrete systems are described by difference equations in the same way that continuous systems are described by differential equations. The block diagram elements of a discrete system are unit delay elements and scalars.

All systems involving digital computers for signal processing are discrete time systems. To work on a signal, it must first be coded into some binary representation. This analog to digital conversion takes some finite amount of time. Therefore, there is some maximum sampling frequency. If the signal is to be processed in real time, the amount of time taken to perform calculations and output results must be added to this conversion time. This reduces the maximum possible sampling frequency. To simplify the manipulation of continuous systems, the Laplace transform is used. An
analogous tool for the discrete system is the $\mathcal{Z}$-transform. The $\mathcal{Z}$-transform of $v(k)$ is defined as the infinite summation

$$V(z) = \sum_{k=0}^{\infty} v(k) z^{-k}$$

which is sometimes expressed as $V(z) = z[v(k)]$, or by the transform pair $v(k) \leftrightarrow V(z)$. The $\mathcal{Z}$-transform converts a difficult to solve finite difference equation into an easy to solve algebraic equation in $z$.

There are many techniques for designing digital filters. The method used in this lab is to design an analog filter for the desired response, and then to transform the $H(s)$ into an $H(z)$ by one of three methods.

The first two methods used are impulse and step invariance. These two techniques set the response of the digital filter to a particular input to be equal to the response of the analog filter to the same input.

To get the impulse invariant filter, it is necessary to obtain the time domain impulse response $h(t)$ of the desired analog filter. This is then sampled giving the values $h(0), h(1),...$ etc. The corresponding $\mathcal{Z}$-transform of the impulse invariant digital filter is thus

$$H(z) = h(z) + \frac{h(1)}{z} + \frac{h(2)}{z^2} + ...$$

Alternately, if $H(s)$ is the transfer function of the analog filter, then

$$H(z) = \sum \text{residues} \left[ \frac{H(s)}{1 - e^{sT} z^{-1}} \right]$$

At this point it should be noted that the DC or zero frequency gains of $H(s)$ and $H(z)$ will not be the same. Thus a scaling factor is needed for $H(z)$,

$$DC \text{ gain } = \sum_{i=0}^{\infty} h(i)$$

For $H(s)$,

$$DC \text{ gain } = \int_{0}^{\infty} h(t) \, dt$$

However,

$$\int_{0}^{\infty} h(t) \, dt = T \sum_{i=0}^{\infty} h(i)$$

Therefore, $H(z)$ must be multiplied by $T$ prior to its implementation.

With the step invariant transformation, the output of the digital filter is to be equal to the sampled outputs of the corresponding analog filter. An example of this is the step invariant Butterworth second order low-pass filter shown in Figure 9. As can be seen, the digital response is identical to the analog response every $T$ seconds. This technique guarantees the output for a step input, but in turn says nothing about the impulse response of the digital filter.

To get the step invariant filter, it is necessary to get the time domain response of the analog filter to a step input. Then $t$ is converted to $kT$ and the $\mathcal{Z}$-transform of this is obtained. This is
divided by \( z/(z-1) \), the Z-transform of the unit step, to get the desired step invariant filter. No correction factor is needed.

**FIGURE 9.** Step Response of a Butterworth Low-Pass Filter.

The other technique used for the experiment is the bilinear transformation. This is accomplished by replacing \( s \) by \((z-1)/(z+1)\) in the transform of the analog filter \( H_A(s) \). Rearranging the new transfer function gives the desired \( H_D(z) \). The magnitude and the phase plots of \( H_D(z) \) obtained are guaranteed to have the same general shape as those corresponding to \( H_A(s) \), but with distorted frequency scales. For example, consider the response of the filter to a sinusoidal input of radian frequency \( \omega_0 \). The transfer function is given by

\[
H_D(z) = \left|\frac{s-1}{s+1}\right|_{z = e^{j\omega_0 T}}
\]

\[
j\omega_A = \frac{z-1}{z+1}_{z = e^{j\omega_0 T}} = \frac{e^{j\omega_0 T} - 1}{e^{j\omega_0 T} + 1} = j \tan\left(\frac{\omega_0 T}{2}\right)
\]

Hence,

\[
\omega_A = \tan\left(\frac{\omega_0 T}{2}\right)
\]

This warping is illustrated for a band-pass filter in Figure 10.

An alternative approach is to use the scaled bilinear transform, \( s = \frac{2(z-1)}{T(z+1)} \). This transformation or the warped bilinear is used mostly where you have stop and pass-bands and has less frequency distortion. Its equation relating specific analog frequencies to digital frequencies is

\[
\omega_A = \frac{2}{T} \tan\left(\frac{\omega_0 T}{2}\right)
\]

The advantage of the bilinear transformation is related to fold-over or alias problems. Fold-over is the situation which occurs if the frequency response is not band-limited to half the sampling frequency. Then, because the frequency response repeats every \( f_s \), the characteristic has an overlap, i.e., an aliasing problem. The bilinear transformation maps the entire \( s \)-plane into a strip bounded by \( s = +jn f_s \) and \( s = -jn f_s \); then mapping this into the \( z \)-plane results in no aliasing fold-over problem since there are no frequency components past \( s = jn f_s \).
FIGURE 10. Warping of the Frequency axis caused by the unscaled Bilinear Transformation.

FILTER TRANSFER FUNCTIONS

A. IMPULSE IN Variant BUTTERWORTH

The impulse invariant Butterworth transfer function is derived from the magnitude squared function \( H(j\omega)^2 = 1/(1 + B(\omega)^{2n}) \), where \( n \) is the order of the filter. To get a cut-off frequency at \( \omega_0 \), it is necessary that \( H(j\omega_0)^2 = 1/2 \) at \( \omega_0 \). Thus, \( 1 + B(\omega_0)^{2n} = 2 \). For example, if \( \omega_0 = 143 \) rad/sec, and \( n = 4 \), then \( B = (143)^4 \). To get \( H(s) \), set
\[
H(s) = \left| H(j\omega) \right|_{\omega=\omega_0}^2
\]
and replace \( \omega^2 \) by \( -s^2 \). All the left-half plane poles are then associated with \( H(s) \) and all the right-half plane poles are associated with \( H(-s) \). Doing this for the above fourth order filter gives

\[
H(s) = \frac{(143)^4}{(s-143\angle112.5^\circ)(s-143\angle247.5^\circ)(s-143\angle157.5^\circ)(s-143\angle202.5^\circ)}
\]

This is then expanded into partial fractions in order to get the impulse invariant Z-transform. A useful formula can be used to facilitate the transformation from a second order pair of complex conjugate \( s \)-plane poles to the \( z \)-plane. If for example,

\[
H(s) = \frac{x+iy}{s-a\angle180^\circ-\theta} + \frac{x-iy}{s-a\angle180^\circ+\theta}
\]
then,

$$H(z) = \mathcal{Z}\{H(s)\} = \frac{x + jy}{1 - z^{-1}e^{j\omega}} + \frac{x - jy}{1 - z^{-1}e^{-j\omega}}$$

where \(b = aT\)

$$H(z) = \frac{2x - z^{-1}(2e^{j\cos(180^\circ)\theta})(x\cos(b\sin(180^\circ - \theta)) + y\sin(b\sin(180^\circ - \theta)))}{1 - z^{-1}(2e^{j\cos(180^\circ)\theta}\cos(b\sin(180^\circ + \theta)) + z^{-2}(e^{2j\cos(180^\circ)\theta}))}$$

The results then for the above fourth order example for \(T = 0.007\) seconds are:

$$H(z) = \frac{132.114 + z^{-1}(-1.37226)}{1 + z^{-1}(-0.736522) + z^{-2}(0.157589)} + \frac{-132.114 + z^{-1}(24.5272)}{1 + z^{-1}(-0.822155) + z^{-2}(0.465161)}$$

The implementation on the computer is such that the above \(H(z)\) is multiplied by \(T\) so that the pass-band response is of the order \(z\) (i.e. the input and output have the same magnitude).

### B. STEP INVARIANT BUTTERWORTH

For the step invariant Butterworth low-pass, it is first necessary to use the low-pass Butterworth transfer function derived in the preceding part. Then, \(H(s)\) is multiplied by \(1/s\) to get the step response of the filter. This response is then converted to a time function \(y(t)\), and the corresponding \(Z\)-transform \(Y(z)\) is taken. This is then divided by the \(Z\)-transform of the unit step function to get the step invariant Butterworth low-pass \(Z\)-transform which yields \(H(z)\).

For the preceding example,

$$Y(z) = \frac{-1.70709 + z^{-1}(0.375603)}{1 + z^{-1}(-0.736522) + z^{-2}(0.157589)} + \frac{0.707108 + z^{-1}(-0.675496)}{1 + z^{-1}(-0.822155) + z^{-2}(0.465161)} + \frac{1}{1 - z^{-1}}$$

and

$$H(z) = \frac{-1.20709 + z^{-1}(1.71443) + z^{-2}(-0.296808)}{1 + z^{-1}(-0.736522) + z^{-2}(0.157589)} + \frac{1.207108 + z^{-1}(-1.79368) + z^{-2}(0.908076)}{1 + z^{-1}(-0.822155) + z^{-2}(0.465161)}$$

### C. ELLIPTIC FILTERS

The elliptic filter is derived from an analog low-pass elliptic filter using the unscaled bilinear transformation \(s = (z-1)/(z+1)\). Use of the bilinear transformation requires that the analog filter be prewarped. Consider a sinusoidal input of radian frequency \(\omega_0\). Then we have

$$s = \frac{z - 1}{z + 1}$$

$$j\omega_{\text{analog}} = \frac{e^{j\omega T} - 1}{e^{j\omega T} + 1} = j\tan\frac{\omega T}{2}$$

$$\omega_{\text{analog}} = \tan\frac{\omega T}{2}$$
The analog elliptic filter used here was designed for a pass-band ripple of 3 dB and stop band attenuation of 23 dB using the theory presented in [4]. For the digital filter to have a cutoff frequency of 143 rad/sec, the analog filter cutoff is given by

$$\omega_{\text{analog}} = \tan \left( \frac{143 \text{ rad/sec}}{2} \right) = \tan \left( \frac{1}{2} \right) = 0.55 \text{ rad/sec}$$

The analog elliptic filter for the warped specifications is given by

$$H(s) = \frac{(.11624)(s-3.07658)(s+.00966 + j.57887)(s+.00966 - j.57887)}{(s-.13674 + j.33993)(s+.13674 - j.33993)(s+.014965 + j.53943)(s+.014965 - j.53943)}$$

$H(s)$ was broken into two additive terms of the form

$$H_i(s) = \frac{As+B}{s^2+Cs+D}$$

Applying the bilinear transformation to $H_i(s)$ gives

$$H_i(z) = \frac{A+B}{1+C+D} + z^{-1} \left( \frac{2B}{1+C+D} \right) + z^{-2} \left( \frac{B-A}{1+C+D} \right)$$

$$1+z^{-1} \left( \frac{2D-2}{1+C+D} \right) + z^{-2} \left( \frac{1-C+D}{1+C+D} \right)$$

Thus, for the above transfer function,

$$H(z) = \frac{0.078985 + z^{-1}(0.183462) + z^{-2}(0.104477)}{1 + z^{-1}(-1.2300) + z^{-2}(0.61146)} + \frac{-0.028022 + z^{-1}(-0.032195) + z^{-2}(-0.0047312)}{1 + z^{-1}(-1.0730) + z^{-2}(0.95469)}$$

### D. BAND-PASS FILTER

The band-pass filter is derived from a low-pass filter by using the low-pass to band-pass transformation followed by the bilinear transformation. When the bilinear transform is applied to an analog filter to produce a digital filter, the bandwidth and center frequency must be warped appropriately using

$$\omega_{\text{analog}} = \tan \left( \frac{\omega_b T}{2} \right)$$

If the desired center frequency of a band-pass filter with bandwidth 14.3 rad/sec is 286 rad/sec and the sampling frequency is 143Hz, then the center frequency of the analog filter must be

$$\omega_{\text{analog}} = \tan \left( \frac{286 \text{ rad/sec}}{2} \right) = \tan(1) = 1.5574 \text{ rad/sec}$$

Bandwidth conversion is done as follows:

$$\Delta \omega_{\text{filter}} = \frac{d \omega_{\text{filter}}}{d \omega} (\Delta \omega) = \frac{1}{\cos^2 \left( \frac{\omega}{2} \right)} (\Delta \omega)$$

$$\Delta \omega_{\text{filter}} = \frac{1}{\cos^2 (1)} (0.007 \text{ sec / 2})(14.3 \text{ rad/sec}) = .171 \text{ rad/sec}$$

The second order Butterworth low-pass filter for these warped specifications is
\[ H(s) = \frac{2a^2}{s^2 - 2as + 2a^2} = \frac{.0599}{s^2 - .346s + .0599} \]

where \( a \) is the bandwidth. Applying the band-pass transformation,

\[ s \rightarrow \frac{s^2 + \omega^2}{2s} \]

yields

\[ H(s) = \frac{s(.01907) - 0.32548}{s + s(0.2254) + 2.0719} + \frac{s(-0.01907) + 0.4459}{s^2 + s(0.2638) + 2.8386} \]

Applying the unscaled bilinear transformation, we get

\[ H(z) = \frac{(-0.092927) + z^{-1}(-0.19742) + z^{-2}(-0.10449)}{1 + z^{-1}(0.65018) + z^{-2}(0.86326)} + \frac{(0.10404) + z^{-1}(0.21739) + z^{-2}(0.11334)}{1 + z^{-1}(0.89635) + z^{-2}(0.87136)} \]

**EXPERIMENTAL PROCEDURE**

**DIGITAL FILTERING USING LabVIEW DIGITAL FILTER DESIGN & SPEEDY-33 BOARD**

The process of digital filtering will be carried out in two steps. First, the filter file for the filter will be created by one LabVIEW VI. After creating the filter and printing out the coefficients, a second VI will execute the filter on the Speedy-33 DSP board.

**Using Create Filter.vi**

Filters will be designed with passband and stopband parameters as defined in Figure 11.

![Figure 11. Typical Filter Response.](image-url)
The Create_Filter.vi file is run under LabVIEW 8 (8.5). Filters are designed through the “Filter Design Block for the Filter Design GUI” in the block diagram for Create_Filter.vi. The Expected Plots section below explains what the tool does and how it works. The label on the block changes as you change the Filter design method. Close the block diagram and run the Front Panel when the design specifications have been completed. This should prompt a dialog box in which you choose the path and filename for your filter. Note the path and filter name.

Any filter may be designed by the tool and simulated in LabVIEW but filters to be run on the Speedy-33 board are restricted to 5 specified sampling frequencies. Only the following sampling frequencies are available for those designs: 8kHz, 18kHz, 24kHz, 36kHz, or 48kHz. For audio applications it is recommended that sampling frequencies greater than 8kHz be used.

**Expected Plots**

When running the Create_Filter.vi the filter design GUI will be used to design the filter with specified inputs. The screen will look like this:

![Digital Filter Design Toolkit GUI](image)

**FIGURE 12. Digital Filter Design Toolkit GUI.**

On the left side we see the Filter inputs, and on the right we can see the Magnitude and Pole-Zero plots. This is a great way to verify that your filter has been designed properly, and that your filter is stable. A checkbox switches between displaying magnitude as a ratio or in dB (as in Figure 12).

This GUI enables the user to choose the **Filter Type** from the following:
- Lowpass
- Highpass
- Bandpass
- Bandstop

Each one of the Filter Types requires different specification parameters. The Band filters require both low and high settings for Passband and Stopband Edge Frequencies while Highpass and Lowpass filters only take one Passband and Stopband Edge Frequency Parameter. Design Method parameter can be selected from the following:
- Butterworth
- Chebyshev (Type I)
- Inverse Chebyshev (Type II)
- Elliptic
- Kaiser Window
- Dolph-Chebyshev Window
- Equi-Ripple FIR

If there is a problem with you Filter Design you will see an Error Message similar to the following:

![Figure 13. Filter Design Error Message - Input Parameter cannot be met with chosen Filter Design Method.](image)

There are plenty of possible causes for the error messages. The most common are due to the Passband and Stopband Edge Frequencies. For Passband Filters the Passband Edge Frequencies must be between the user specified Stopband Edge frequencies. Below is the Error Response you would expect in this case.

![Figure 14. Filter Design Error Message: Problem with Filter Edge frequencies.](image)

For the Stopband case the Stopband Edge Frequencies must be between the user specified Passband Edge Frequencies. If these limitation have been met, try to switch your Filter Design Method, Passband ripple, and Stopband Attenuation parameters until no error is reported.

### Creating Filters with LabVIEW 8 and Create_Filter.vi

1. After starting the PC with WindowsXP, start LabVIEW 8 from the `Start→All Programs` menu.

2. Download the Create_Filter.vi from the CStudio website (http://www.ecse.rpi.edu/courses/CStudio/RTA_lab/Digital_Filter/) if it isn’t already in the Digital Filter folder (F:\CStudio\New_Cal_lab\Digital_Filter), and open it in the open instance of LabVIEW. Go to Window→Show Block Diagram to see the block diagram. Double click on the DFD Filter block (it has a text box over it with “Click Here for Filter Design GUI”) and design your filter. Choose the filter type, and enter all the filter specifications. Ensure the Sampling Frequency is one of the following: 18kHz, 24kHz, 36kHz, or 48kHz. Hit “OK” when done.

3. Run the VI by clicking the run arrow button on the main menu. This action will open a dialog box prompting for a file name in which to save the text and FDS file that LabVIEW needs to run your filter. Create the files, and note their location on the hard drive.
4. If any issues occur, reference the Possible Problems section below. Note any parameters, coefficients, and response plots on the screen (save the screen), then close the Create_Filter.vi without saving anything. This leaves the Create_Filter.vi file unaltered for other groups to use. The saved files contain all the information needed to implement the filters on the DSP system.

5. This concludes the design portion of the digital filter.

**Speedy-33 Implementation via LabVIEW:**

Implementing filters on the Speedy-33 board via LabVIEW directly is all file manipulation. The hardware should be set up as follows:

![FIGURE 15a. Equipment Setup for Filtering with the DSP and Signal Generator.](image)

![FIGURE 15b. Equipment Setup for Filtering with the DSP and Portable Audio Device.](image)
1. Plug the Speedy-33 board into the USB port.

2. In the Digital Filter folder, if Execute_Filter.lvproj, DFD_Implementation_plots.vi, and DFD_Implementation_no_plots.vi aren’t already there, download the latest versions from the CStudio website for the most up-to-date RTA programs (http://www.ecse.rpi.edu/courses/CStudio/RTA_lab/Digital_Filter/). Double-click on Execute_Filter.lvproj. A window will open with a hierarchial file structure. Click on the ‘+’ next to the Speedy-33 item under Project: Execute_Filter.lvproj to see the DFD Implementation options. Double-click on either DFD_Implementation_no_plots.vi or DFD_Implementation_plots.vi. The ‘plots’ version displays time and frequency plots of the input and output signals, but due to processor and USB speed limitations, can only be used with the slower sampling rates (<24kHz). The ‘no_plots’ version works at any sampling frequency, but doesn’t display the time and frequency plots. View the block diagram (Window→Show Block Diagram) and verify that no broken wires exist.

3. You will notice there are two DFD Filter blocks, one each for the left and right channels. You can choose to investigate this flexibility to test two different filters simultaneously. The left and right channels can be loaded separately but the left channel is the one wired to the BNC-1/8”mini stereo plug cables in the lab. Double-click the DFD_Filter block. Again ignore error messages about files not found or invalid filters. Use the dialog box to choose the appropriate filter file created in the previous filter design procedure, by use of the Path box. Click on the open folder icon to the right of the Path window and move to the directory where the file was saved. Select the desired .fds file. In this window you will see the Frequency and Time Impulse responses in separate bins.

4. Next, right-click in the Analog Input block on the I_Left channel, select the “Properties” menu item, and select your Sampling Frequency. This parameter should match that of your designed filter. Make sure Framesize is set to 512 and Gain is 1. Repeat this with the Analog Output block.

5. You will now be able to run the VI by clicking the white run arrow on the menu bar. If any errors occur, check the USB connection to the Speedy-33 board, and try again. It is very likely something may have been added to the VI which caused it to break. Read the errors and attempt to debug the VI.

6. As the filter runs, check the front panel of the VI. Ensure that the inputs to the board are not overloading the dynamic range of the board. This can easily be seen as clipping or 2’s compliment overflow in the Left Channel Input window. If values are overflowing or clipping; reduce the input amplitude. If a portable audio device is used for an input signal, the labs attenuators or separate antialiasing filters and amplifiers may be needed at the input and output of the board respectively. In the “plots” mode there are two windows on the front panel displaying the FFT spectrum of the filter’s input and output signal. Note that the default y-axes scales are fixed for large signals. They may be changed to automatically adjust themselves to the signal’s amplitude, but then the output signal may appear as large as the input if the ordinate scale is not noted. Toggle autoscale by right-clicking in the spectrum window and checking or unchecking AutoScale Y. Depending on where you click in the window, different menus appear and it may be necessary to go through a Y Scale menu item. There is also an option for choosing between a linear and logarithmic scale. Two different sets of menus appear, depending on whether the VI is stopped or executing. FFT plots are a little different from what first-time users expect. The x-axis goes from 0 to fs/2 in the middle and then displays the negative portion of the x-axis from ~fs/2 to 0. Observers should think of the right half of the x-axis as being part of the left side, representing
negative frequencies. Also note that with amplitude in dB (log scale), pure sine waves appear as pulses with wider bases rather than the expected ideal extremely narrow impulses.

Inputs and outputs may be configured as in Figure 15. In addition to line-in signals, the Speedy-33 supports input from 2 on-board microphones in the corners. The board autodetects inputs to the line-in plug, so this must be unplugged to use the microphones. There is increased amplification available for the microphones that is activated by moving both jumpers labeled **Input Level** from the **Line** to the **Mic** side. Just make sure you restore them both to the **Line** positions before reconnecting an input signal to the line-in plug or the board may be damaged. The board output should remain unchanged.

7. Also check the Input/Output Frequency plots and observe sine waves on the scope to ensure the proper frequencies are being blocked by the filters.

8. After analyzing a given filter and obtaining performance data, stop the VI and quit LabVIEW, again without saving any changes to either the .lvproj or .vi files so as to leave them unaltered for other users.

**Possible Problems**

Ensure all cables are connected properly. If any issues occur while attempting to load the VI to the board, retry a few times. If problems still arise, pay attention to the board memory bar on the Speedy-33 pop up menu, ensure that it is less than 100%. Also ensure there are no broken wires present in the block diagram screen, it is possible that in specifying the FDS file some connections were broken.

**PART I**

Note that the following designs will all be Type I cascade filters (as in Figure 2). For each case use LabVIEW to design and generate filters, and use MATLAB to verify the coefficients. Also note that for all Speedy-33 implementations, the antialiasing filter on the DSP board is fixed at 24kHz. Filters designed with sample rates below 48kHz will result in aliasing when the input frequency exceeds fs/2. Separate antialiasing filters with lower cut-off frequencies must be provided by the users.

**A. BUTTERWORTH LOW-PASS FILTER**

Build an 8th order Butterworth filter with the following parameters:
- fs = 24kHz
- Cutoff Frequency = 2kHz
- Passband Ripple = 1.5dB
- Stopband Attenuation = 30dB

1) Implement this filter on the Speedy-33.
2) Take data to plot the frequency response of this filter from f = 5Hz to f = 24kHz. This can be done simply by varying the frequency of the sine wave generator and comparing the amplitude of the input signal and the amplitude of the steady state output signal.
3) Measure the rise time and overshoot for a square wave input of 500Hz and explain why it occurs.
4) Note the filter order and phase characteristics from the spectrum and the pole/zero locations.
5) Is the frequency response different from what you would expect from the analog filter? How? Why?
B. ELLIPTIC LOW-PASS FILTER

Build a 3rd order Elliptic filter with the following parameters:
- \( f_s = 24 \text{kHz} \)
- Cutoff Frequency = 2kHz
- Passband Ripple = 1.5dB
- Stopband Attenuation = 30dB

1) Implement this filter on the Speedy-33.
2) Take data to plot the frequency response of this filter up to about 24kHz. Is it different from that of the Butterworth filter? How?
3) Note the filter order and phase characteristics from the spectrum and the pole/zero locations.
4) Measure the rise time and overshoot for a square wave input of 500Hz. How does this compare to that of the Butterworth filter? Is there more or less distortion of the wave shape? Why?

C. Chebyshev LOW-PASS FILTERS – BOTH TYPE I & II

Build 4th order Chebyshev filters with the following parameters:
- \( f_s = 24 \text{kHz} \)
- Cutoff Frequency = 2kHz
- Passband Ripple = 1.5dB
- Stopband Attenuation = 30dB

1) Implement this filter on the Speedy-33.
2) Take data to plot the frequency response of this filter up to about 24kHz. How does this compare with that of the Butterworth and elliptic filters?
3) Note the filter order and phase characteristics from the spectrum and the pole/zero locations for both the type I and II filters.
4) Measure the rise time and overshoot for a square wave input of 500Hz. How does this compare to that of the Butterworth filter? Which has the best response? Why?
5) Compare the magnitude & phase response and order to those of the Butterworth and elliptic filters.

D. BAND-PASS FILTER (do only either D or E)

Build a 12th order Butterworth filter with the following parameters:
- \( f_s = 24 \text{kHz} \)
- Passband Frequency Range = 500Hz – 2kHz
- Passband Ripple = 2dB
- Stopband Attenuation = 30dB

1) Implement this filter on the Speedy-33.
2) Sketch the frequency response of this filter up to about 12kHz. How is this response different from that of an analog band-pass filter? Why is it different?
3) Note the filter order and phase characteristics from the spectrum and the pole/zero locations.

E. HIGH-PASS FILTER (do only either D or E)

Build an 8th order Butterworth filter with the following parameters:
- \( f_s = 24 \text{kHz} \)
- Passband Frequency Range = 1kHz
- Passband Ripple = 1.5dB
• Stopband Attenuation = 30dB

1) Implement this filter on the Speedy-33.
2) Sketch the frequency response of this filter up to about 16kHz. How is this response different from that of an analog high-pass filter? Why is it different?

F. ALIASING AND NOISE

1) Set up a Butterworth low-pass filter on the Speedy-33. To see aliasing, design the filter with \( f_s = 24 \text{kHz} \) and \( f_{\text{cutoff}} \approx 10\text{kHz} \)
2) Use an analog low-pass filter as an antialiasing filter, input square waves whose frequencies lie between 4 and 7kHz and connect this to the DSP A/D input. Observe the output and input with and without the analog low-pass filter on the D/A output. (Set the antialiasing filter for \( f_c \approx 5 \) to 10kHz).
3) Implement a 4th order Butterworth LPF with \( f_{\text{cutoff}} \approx 4\text{kHz} \) and \( f_s = 36\text{kHz} \).
4) Feed in a triangular wave at 3kHz with a noise source attached. Observe the output waveform. By changing the cutoff frequency of your digital filter, attempt to get the best output waveshape possible, with the least distortion. Note your final choice of parameters and what cutoff frequency this represents. Give an explanation as to why this cut-off frequency worked best.

G. AUDIO INPUT NOTCH FILTER

1) Set up 4 Notch Filters with the following parameters:
   • \( f_s = 24\text{kHz} \)
   • Center Stopband Frequency Range: vary in 400Hz increments from 200Hz to 2kHz
   • Passband Ripple = 1.5dB
   • Stopband Attenuation = 30dB
2) Follow the instructions for a portable music device. Either set up a portable device or use the PC’s CD playing capabilities and its stereo output as an input to the DSP board.
3) Run each of the notch filters and notice the blocked tones, in each interaction. Describe the effects qualitatively.

PART II

For this section, you will be required to use LabVIEW and MATLAB for the filter design calculations. MATLAB and LabVIEW are on the RTA PCs. Some functions to aid you in designing the filters in MATLAB are listed in the Appendix of this handout. Both programs have built-in help manuals and on-line guides. The MATLAB designs can be simulated while LabVIEW filters may be simulated or run directly on the Speedy-33 DSP board linked to the PC. LabVIEW implementations are restricted to cascade forms (no other options are available), but extremely high order filters are realizable. A MATLAB M-file (write_fds.m), discussed in APPENDIX C, is provided that will permit the execution of MATLAB-designed filters on the DSP, if the filter coefficients are in the format of second order blocks.

For each of the following questions, be sure to include the filter coefficients in your report along with a z-plane analysis. Do not include any MATLAB diary files, m-files, etc. Computer generated frequency response plots, however, are needed since they can be compared to your lab results.

1) Pick a sampling frequency for a filter to be designed. For the Speedy-33 board filter designs, only the following sampling frequencies are available: 8kHz, 18kHz, 24kHz, 36kHz, or 48kHz. For audio applications it is recommended to use a sampling frequency greater than 8kHz.
2) Using the sampling frequency from step 1, design several filters using MATLAB or LabVIEW. A few suggestions follow, with the first being required:
- A low-pass filter with cutoff of 500Hz.
- A band-stop notch filter at 1kHz.
- A high-pass filter with cutoff at 3kHz.
- A band-pass filter from 1.0kHz to 2.5kHz and $f_s = 36kHz$. (Use the Speedy-33 and analyze with your own voice & music using a microphone & a portable audio player.)
- At least two more filters of your own choice.

Gather as much information about the filters as is required for the write-up. Remember to mix the approximation types and designs with both LabVIEW and MATLAB. MATLAB allows the design of digital filters with other analog to digital transformations (step invariant, impulse invariant, bilinear). You may want to investigate how these affect the time domain output of square waves. Cabling is available in the lab that permits a side-by-side comparison of 2 filters, different ones running on the left and right channels simultaneously.

3) Make a thorough report of your results, comparing the approximation types, and showing why you would choose one type over another. Plot the filter on the $z$-plane and analyze the filter using the $z$-plane plot.

4) Record the coefficients generated by MATLAB, or LabVIEW for use in the implementation.

5) Implement the filters on the Speedy-33 board, and compare with the results obtained from MATLAB simulations of the same filters. Explain any differences.

REFERENCES


FACULTY RESOURCE

J. W. Woods, W. Pearlman
APPENDIX A

This appendix describes some of the functions provided by MATLAB for designing and analyzing filters. Note that these functions are described in the MATLAB manual, a copy of which is available on-line. The HELP command in MATLAB should also provide sufficient detail.

ANGLE, UNWRAP

angle(h) returns the phase angle in radians, of the elements of the complex matrix. These angles will lie between $+\pi$ and $-\pi$. unwrap(p) corrects the phase angles in vector by adding multiples of $+2\pi$ or $-2\pi$, to smooth the transitions across branch cuts. The phase must be in radians.

BILINEAR

$\left[ z_d, p_d, k_d \right] = \text{bilinear}(z, p, k, f_s)$ converts the $s$-domain transfer function specified by zeros, poles and gain into a discrete equivalent. Inputs $z$ and $p$ are column vectors containing the zeros and poles, and $k$ is a scalar gain factor. $f_s$ is the sample frequency in Hz. The discrete equivalent is returned in column vectors $z_d$, $p_d$ and scalar $k_d$.

$\left[ \text{num}_d, \text{den}_d \right] = \text{bilinear}(\text{num}, \text{den}, f_s)$ converts an $s$-domain transfer function to a discrete equivalent. The function is

\[
\frac{\text{num}(s)}{\text{den}(s)} = \frac{\text{num}(1)s^{\text{num}} + \ldots + \text{num}(\text{num} + 1)}{\text{den}(1)s^{\text{num}} + \ldots + \text{num}(\text{num} + 1)}
\]

$f_s$ is the sample frequency in Hz. The discrete equivalent is returned in row vectors $\text{num}_d$ and $\text{den}_d$ in descending powers of $z$.

BUTTAP

$\left[ z, p, k \right] = \text{butterap}(n)$ returns the zeros, poles and gain of an order $n$ normalized Butterworth analog low-pass filter prototype. The poles are returned in length $n$ column vector $p$, the gain in $k$, and $z$ is an empty matrix, as there are no zeros. The transfer function is

\[
H(s) = \frac{z(s)}{p(s)} = \frac{k}{(s - p(1))(s - p(2))\ldots(s - p(n))}
\]

BUTTER

$\left[ b, a \right] = \text{butter}(n, w_n)$ designs an order $n$ low-pass digital Butterworth filter with cutoff frequency $w_n$ and returns the filter coefficients in length $n+1$ row vectors $b$ and $a$.

\[
H(z) = \frac{B(z)}{A(z)} = \frac{b(1) + b(2)z^{-1} + \ldots + b(n + 1)z^{-n}}{1 + a(1)z^{-1} + \ldots + a(n + 1)z^{-n}}
\]

$w_n$ must be between 0 and 1, where 1 corresponds to half the sample frequency. If $w_n = [w_1 \ w_2]$ is a two element vector, an order $2n$ band-pass filter is designed with pass-band $w_1 < w < w_2$. $\left[ b, a \right] = \text{butter}(n, w_n, \text{'high'})$ designs a high-
pass filter with cutoff frequency \( w_n \). \([b,a]=\text{butter}(n,w_n,'stop')\) designs an order 2n band-stop filter if \( w_n \) is a two element vector. The stop-band is \( w_1 < w < w_2 \).

**BUTTERORD**

\([n,w_n]=\text{butterord}(w_p,w_s,r_p,r_s)\) returns the order \( n \) of the lowest order Butterworth filter that loses no more than \( r_p \) dB in the pass-band and has at least \( r_s \) dB of attenuation in the stop-band. The pass-band runs from 0 to \( w_p \) and the stop-band runs from \( w_s \) to 1, the Nyquist frequency. \( w_n \), the natural frequency to use with \text{butter} is also returned.

**CHEB1AP**

\([z,p,k]=\text{cheb1ap}(n,r_p)\) returns the zeros, poles and gain of an order \( n \) normalized Chebyshev type I analog low-pass filter prototype with \( r_p \) decibels of ripple in the pass-band. The poles are returned in length \( n \) column vector \( p \), the gain in scalar \( k \) and \( z \) is an empty matrix.

\[
H(s) = \frac{z(s)}{p(s)} = \frac{k}{(s-p(1))(s-p(2))...(s-p(n))}
\]

The poles are evenly spaced about an ellipse in the left half plane.

**CHEB1ORD**

\([n,w_n]=\text{cheb1ord}(wp,ws,r_p,r_s)\) returns \( n \), the order of the lowest order Chebyshev filter that loses no more than \( r_p \) dB in the pass-band and has at least \( r_s \) dB of attenuation in the stop-band. The pass-band runs from 0 to \( wp \) and the stop-band extends from \( ws \) to 1, the Nyquist frequency. \( w_n \), the Chebyshev normalized frequency to be used with \text{cheby1}, is also returned.

**CHEB2AP**

\([z,p,k]=\text{cheb2ap}(n,r_s)\) returns the zeros, poles and gain of an order \( n \) normalized Chebyshev type II analog low-pass filter prototype with stop-band ripple \( r_s \) dB down from the peak value in the pass-band. The zeros and poles are returned in column vectors \( z \) and \( p \), and the gain is in scalar \( k \):

\[
H(s) = \frac{z(s)}{p(s)} = \frac{(s-z(1))...(s-z(n))}{(s-p(1))...(s-p(n))}
\]

The Chebyshev cutoff frequency \( w_0 \) is set to 1 for a normalized result.

**CHEB2ORD**

\([n,w_n]=\text{cheb2ord}(wp,ws,r_p,r_s)\) returns \( n \), the order of the lowest order Chebyshev filter that loses no more than \( r_p \) dB in the pass-band and has at least \( r_s \) dB of attenuation in the stop-band. The pass-band runs from 0 to \( wp \) and the stop-band extends from \( ws \) to 1, the Nyquist frequency. \( w_n \), the natural frequency to be used with \text{cheby2} is also returned.
CHEBY1 CHEBY2

\[ [b,a] = \text{cheby1}(n,r_p,w_n) \] designs an order \( n \) low-pass digital Chebyshev filter with cutoff frequency \( w_n \) and \( r_p \) dB of ripple in the pass-band. \( n+1 \) long row vectors \( a \) and \( b \) contain the filter coefficients.

\[
H(z) = \frac{B(z)}{A(z)} = \frac{b(1)+b(2)z^{-1}+...+b(n+1)z^{-n}}{1+a(2)z^{-1}+...+a(n+1)z^{-n}}
\]

\( w_n \), the cutoff frequency must be between 0 and 1, where 1 corresponds to half the sample frequency. If \( w_n = [w_1 \, w_2] \) is a two element vector, a band-pass filter of order \( 2n \) is designed with \( w_1 < w < w_2 \).

\[ [b,a] = \text{cheby1}(n,r_p,w_n,'high') \] designs a high-pass filter with cutoff frequency \( w_n \).

\[ [b,a] = \text{cheby1}(n,r_p,w_n,'stop') \] designs an order \( 2n \) stop-band filter if \( w_n \) is a two element vector. \text{cheby2} accepts the same parameters as \text{cheby1}, but designs a type II Chebyshev filter.

ELLIP

\[ [b,a] = \text{ellip}(n,r_p,r_s,w_n) \] designs an order \( n \) low-pass digital elliptic filter with cutoff frequency \( w_n \), \( r_p \) dB of ripple in the pass-band, and a stop-band \( r_s \) dB down from the peak value in the pass-band. The filter coefficients are returned in \( n+1 \) long row vectors \( a \) and \( b \). If \( w_n = [w_1 \, w_2] \) is a two element vector, a band-pass filter of order \( 2n \) is designed with \( w_1 < w < w_2 \).

\[ [b,a] = \text{ellip}(n,r_p,r_s,w_n,'high') \] designs a high-pass filter with cutoff frequency \( w_n \).

\[ [b,a] = \text{ellip}(n,r_p,r_s,w_n,'stop') \] designs an order \( 2n \) stop-band filter if \( w_n \) is a two element vector.

OTHER FUNCTIONS

MATLAB has two toolboxes of useful functions and two demonstrations related to polynomial math and filters. Use the “help” command to get more information on any of these items.

- Toolboxes: polyfun, signal
- Demonstrations: filtdemo, filters (You may wish to copy listings of these M-files to your directory for your own use and modification.)
- Functions: conv, deconv, roots, poly, residue, freqs, pzmap, dimpulse, residuez, freqz (conv is also used for polynomial multiplication.)
APPENDIX B

Create_Filter.vi Front Panel Coefficients

This appendix describes the front panel parameters in Create_Filter.vi. These same parameters are written into the saved text file for the filter to provide the users with a copy of all the pertinent values from the filter design procedure when the VI is run. A digital filter may be represented mathematically in many different forms, but the most common are: 1) factored form or pole-zero form, 2) polynomial form, and 3) cascaded second order systems form (Figure 2). Each one of these is a ratio of polynomials and regardless of the form, they are all equivalent mathematically. The middle list of values in the Create_Filter.vi front panel is the zeros and poles of the filter with the gain shown above them. This would lead to \( H(z) \) written as

\[
H(z) = (gain) \frac{(z - z_1)(z - z_1^*)(z - z_2)(z - z_2^*) \cdots}{(z - p_1)(z - p_1^*)(z - p_2)(z - p_2^*) \cdots}
\]

where the \( z_i \) or \( p_i \) and the * represent the zeros or poles and their complex conjugates and gain is the gain value given in the panel to scale the expression.

If this factored form of \( H(z) \) is multiplied out, the polynomial form is obtained. This is written in standard form as

\[
H(z) = \frac{b_n z^n + b_{n-1} z^{n-1} + b_{n-2} z^{n-2} + \cdots + b_0}{z^n + a_{n-1} z^{n-1} + a_{n-2} z^{n-2} + \cdots + a_0}
\]

Alternatively the complex conjugate pairs of zeros and poles could be combined into second order polynomials and \( H(z) \) could be rewritten in its 3rd standard form of cascaded second order systems

\[
H(z) = \frac{b_{12} z_1^2 + b_{11} z + b_{10}}{z^2 + a_{11} z + a_{10}} \times \frac{b_{22} z_2^2 + b_{21} z + b_{20}}{z^2 + a_{21} z + a_{20}} \cdots
\]

As an example, the default filter for the VI is a 6th order Chebyshev I LPF with passband frequency of 2kHz and stopband frequency 2.6kHz, 1.5dB passband ripple, 30dB stopband attenuation and sample frequency 18kHz. The coefficient text file for this filter is provided on the next page for your convenience (Chby_LPF.2k-18k.txt). The objective of the example is to show how the coefficients in the file are used in the three forms of \( H(z) \) described above.

Note that the coefficients in the polynomials in \( H(z) \) in the following example have been rounded to 4 or 5 places for convenience and to allow the equations to fit on the page in a more readable format.
LabVIEW Filter Type: 11.000000

Gain:
1.000000

Numerator B Coefs (IIR Cascaded Second Order Sections Form II):
0.041396
0.082793
0.041396
0.041396
0.082793
0.041396
0.041396
0.082793
0.041396

Denominator A Coefs:
-1.491118
0.933622
-1.576150
0.820158
-1.700415
0.750390

Transfer Function (Direct Form II):
Numerator (b coefs):
0.000071
0.000426
0.001064
0.001419
0.001064
0.000426
0.000071

Denominator (a coefs):
1.000000
-4.767684
10.070027
-11.974645
8.427061
-3.323950
0.574587

ZPK:
ZPK Gain:
0.000071

Zeros:
-1.000000 + j*0.000000
-1.000000 + j*0.000000
-1.000000 + j*0.000000
-1.000000 + j*0.000000
-1.000000 + j*0.000000
-1.000000 + j*0.000000

Poles:
0.850208 + j*0.165942
0.850208 + j*-0.165942
0.788075 + j*-0.446201
0.788075 + j*0.446201
0.745559 + j*0.614625
0.745559 + j*-0.614625
Since all the zeros in this example are real (at \( z = -1 \)), the ZPK section will yield a factored form

\[
H(z) = \frac{(z + 1.0000)(z + 1.0000)(z + 1.0000)(z + 1.0000)(z + 1.0000)}{(z - 0.8502 + j0.1659)(z - 0.8502 - j0.1659)(z - 0.7881 + j0.4462)(z - 0.7881 - j0.4462)(z - 0.7456 + j0.6146)(z - 0.7456 - j0.6146)}
\]

Multiplying this out to obtain the polynomial form, the coefficients are the Numerator \( b \) and Denominator \( a \) coefficients from the Transfer Function (Direct Form II) section in the file.

\[
H(z) = \frac{0.000071z^6 + 0.000426z^5 + 0.001064z^4 + 0.001419z^3 + 0.0001064z^2 + 0.000426z + 0.000071}{z^6 - 4.7677z^5 + 10.0700z^4 - 11.9746z^3 + 8.4271z^2 - 3.3240z + 0.5746}
\]

Finally, the second order cascaded systems is obtained from the IIR Cascaded Second Order Sections Form II part of the text file. Note that the leading coefficient in the denominator of each section is always 1 and is left out of the list. There are fewer \( A \) coefficients due to this assumption.

\[
H(z) = \frac{0.041396z^2 + 0.082793z + 0.041396}{z^2 - 1.4911z + 0.9336} \times \frac{0.041396z^2 + 0.082793z + 0.041396}{z^2 - 1.5762z + 0.8202} \times \frac{0.041396z^2 + 0.082793z + 0.041396}{z^2 - 1.7004z + 0.75046}
\]

**APPENDIX C**

**write_fds.m Guide**

This appendix describes the MATLAB *.fds* digital filter specification file generator. This script enables students to run MATLAB custom generated \( H(z) \) filter coefficients on the Hyperception Speedy-33 board. It can be found on the course WebCT page or at http://www.ecee.rpi.edu/courses/CStudio/RTA_lab/Digital_Filter/. The inputs are:

1. A Coefficients in IIR Cascaded Second Order Sections Form II format
2. B Coefficients in IIR Cascaded Second Order Sections Form II format
3. Sampling Frequency
4. Type according to Labview’s Filter type enumeration. (11 correlates to IIR Cascaded Second Order Sections Form II, other forms have not been documented)
5. Gain

If the file is in the current MATLAB path, calling "help write_fds" displays the arguments that the function takes: direct form zpk, numerator-denominator, second order system coefficients, dfilt.sos2 object, or a transfer function. The dfilt.sos2 object is what is returned when using the MATLAB filterDesigner (called with "filterDesigner- 2017 or higher" in the command window), which is what can be used for the designs. The other arguments are the ones that are returned from all the functions aside from the analog prototype functions, as well as a few more, so you should be able to just take those outputs and the sample rate into write_fds (for example: write_fds(b,a,Fs) ). Calling the function with the arguments stated here will create a *.fds file, which can be used the same way as the *.fds files created with LabView. After these inputs are specified, and the script is run, a pop up window will appear. Enter a filename and the file <filename>.fds will be generated.