EE438 - Laboratory 5: Digital Filter Design

October 14, 1996

1 Introduction

In most of the digital signal processing applications, it is desired to change the relative amplitudes of the frequency components of a signal or even eliminate some frequency components completely. This process is called *filtering*. In linear time invariant systems, filtering can be carried out by multiplying the spectrum of the input by the frequency response of the filter. Common examples of linear time invariant filters are the systems in audio equipment that allow the listener to adjust the bass (low-freuqency energy) and the treble (high frequency energy).

Even though noncausal filters can also be implemented, in this laboratory experiment we will dwell on the design of causal filters. The design procedure of a digital causal filter has three stages:

1. The specification of the desired properties of the filter
2. The approximation of the specifications using a rational tranfer function
3. The implementation of the transfer function using a software and hardware technology of the filter

In this laboratory experiment, we will study all of these 3 stages with main emphasis being on the second stage. The first stage depends on the application and its output of the first stage is a set of tolerance specifications for the frequency behaviour of the filter (Figure 1).

The tolerance parameters for the frequency response of a filter $H(e^{j\omega})$ are the lower and upper passband edge frequencies (ω_{pl} , ω_{pu}), the lower and upper stopband edge frequencies (ω_{sl} , ω_{su}) the passband ripple (δ_1) and the lowpass ripple (δ_2). is the lowpass ripple.

Goto: **Lab home page** - **Matlab help** - **lab1** - **lab2** - **lab3** - **lab4** - **lab5**

Questions or comments concerning this laboratory should be directed to Prof. Charles A. Bouman, School of Electrical and Computer Engineering, Purdue University, West Lafayette IN 47907; (317) 494-0340; bouman@ecn.purdue.edu

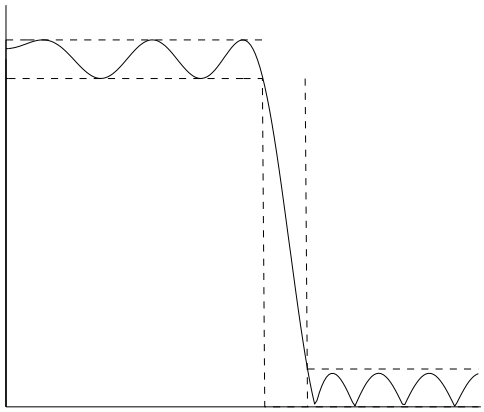

Figure 1: Tolerance specifications for the frequency response of a filter.

Within the passband, the magnitude of the frequency response of the filter must approximate 1 with an error of $\pm\delta_1$ i.e.

$$(1-\delta_1) \leq |H(e^{j\omega})| \leq (1+\delta_1) \text{ for } \omega_{pl} \leq \omega \leq \omega_{pu}.$$

At the passband, the magnitude of the frequency response of the filter must approximate 0 with an error of $\pm\delta_2$ i.e.

$$|H(e^{jw})| \leq \delta_2 \text{ for } \omega_{sl} \leq \omega \leq \omega_{su}.$$

The region between the passband and the stopband is called the transition region and has a nonzero width for nonideal filters.

1.1 Design of a Simple FIR Filter

Down load **nspeech1.mat**
Down load **Matlab audio utilities**
How to **load and play audio signals**
Down load **DTFT.m**

To illustriate the use of zeros in filter design, you will design a simple second order FIR filter with the two zeros on the unit circle shown in Fig. 2. In order for the filter's impulse response to be real, the two zeros must be complex conjugates of one another so the zeros may be written as

$$z_1 = e^{j\theta} \quad z_2 = e^{-j\theta}$$

where θ is the angle relative to the real axis. We will latter see that $\theta \in [-\pi, \pi]$ may be interpretated as the frequency location of the zero.

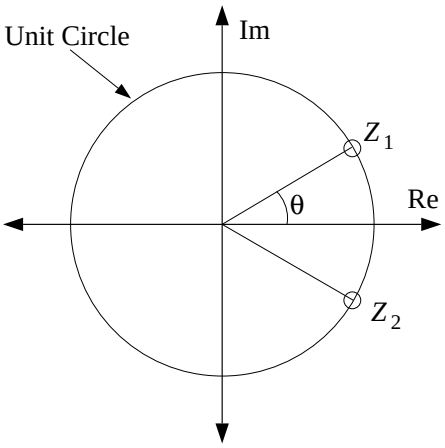

Figure 2: Location of two zeros for simple a FIR filter.

The transfer function for the filter is given by

$$\begin{aligned} H_f(z) &= (1 - z_1 z^{-1})(1 - z_2 z^{-1}) \\ &= (1 - e^{j\theta} z^{-1})(1 - e^{-j\theta} z^{-1}) \\ &= 1 - 2\cos\theta z^{-1} + z^{-2} \ . \end{aligned}$$

Use this transfer function to determine the difference equation for this filter. Then draw the corresponding flow diagram and compute the filter's impulse response $h(n)$.

This filter is called a finite impulse response (FIR) filter because it has a impulse response $h(n)$ of finite length. Any filter with only zeros and no poles other than those at 0 and $\pm\infty$ is an FIR filter. Zeros in the transfer function represent frequencies that are not passed through the filter. This can be useful for removing unwanted frequencies in a signal. Since the filter $H_f(z)$ has zeros at $e^{\pm j\theta}$, then $H_f(e^{\pm j\theta} e^{\pm j\theta}) = 0$ which means that the filter will not pass pure sine waves at frequency of $\omega = \theta$.

Compute the magnitude of the filter's frequency response $|H_f(e^{j\omega})|$ on $|\omega| < \pi$ for the following three values of θ.

i) $\theta = \pi/6$

ii) $\theta = \pi/3$

ii) $\theta = \pi/2$

Put all three plots on the same figure using the *subplot* command.

INLAB REPORT:
Submit the difference equation, impulse response and flow diagram for the filter $H_f(z)$. Also submit the plot of the magnitude response for the three values of θ. Explain how the value of θ affects the magnitude response of the filter.

In following experiment, you will use the filter $H_f(z)$ to remove an undersirable sinusiodal interferance from a speech signal. To run the experiment, first down load the audio signal *nspeech.mat*, the Matlab audio utilities, and the .m-file *DTFT.m* . Load *nspeech.mat* into matlab using the command `load nspeech`. This will load the signal *nspeech1* into your workspace. Play *nspeech1* using the command *auread*, and then plot 100 samples of the signal for the time indices (100:200).

We will next use the *DTFT* command to compute samples of the DTFT of the audio signal. The *DTFT* command has the syntax

```
[X,w]=DTFT(x,M)
```

where `x` is a causal signal which is assumed to start at time $n = 0$, and the number of output points in always greater than `M`. The outputs of the command are then `X`, the samples of the DTFT, and `w` the corresponding frequecies of the samples. Normally, the command `[X,w]=DTFT(x,0)` will generate sufficient numbers of points to get a smooth plot of the DTFT, but if the number of points in not sufficient, it may be increased by specifying `M`.

Compute the magnitude of the DTFT of the 1000 samples of the audio signal for the time indices (100:1100). Plot the magnitude of the DTFT samples versus radial frequecy for $|\omega| < \pi$. Notice that there are two large peaks corresponding to the sinusiodal interference signal. Use the Matlab *find* and *max* commands to determine the exact radial frequency of the peaks. This will be the value of θ that you will use for filtering with $H_f(z)$.

Write a Matlab function `fir(x)` that implements the filter $H_f(z)$ with the measured value of θ and outputs the filtered signal. Apply your new command *fir* to the *nspeech1* to attenuate the sinusiodal interference. Play back the filtered signal, plot 100 samples of the signal for the time indices (100:200), and plot the magnitude of the DTFT of the 1000 samples of the filtered signal for the time indices (100:1100).

INLAB REPORT:
For both the original audio signal and the filtered output, hand in the following:

- The time domain plot of 100 samples.
- The plot of the magnitude of the DTFT for 1000 samples.

Comment on how the frequency content of the signal has changed after filtering. Is the filter you used a lowpass, a highpass, a bandstop or a bandpass filter? Also, comment on how filtering changed the quality of the audio signal.

1.2 Design of A Simple IIR Filter

Down load **pcm.mat**

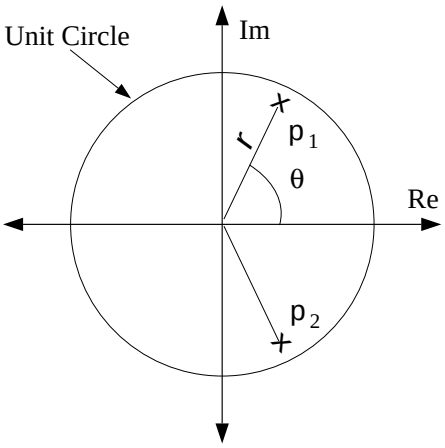

Figure 3: Location of two poles for a simple IIR filter.

While zeros attenuate a filtered signal, poles amplify signals that are near their frequency. In this section, we will illustrate how poles can be used to separate a narrow-band signal from adjacent noise. Such filters are commonly used to separated modulated signals from background noise in applications such as radio-frequency demodulation.

Figure 3 show how the poles are arranged for a simple second order IIR filter with complex poles. For this case, the poles have the form

$$p_1 = re^{j\theta} \quad p_2 = re^{-j\theta}$$

where r is the distance from the origin, and θ is the angle relative to the real axis. The transfer function for this system $H_i(z)$ is given by

$$\begin{aligned} H_i(z) &= \frac{1}{(1 - re^{j\theta}z^{-1})(1 - re^{-j\theta}z^{-1})} \\ &= \frac{1}{1 - 2rcos(\theta)z^{-1} + r^2z^{-2}} \end{aligned}$$

From the theory of Z-transforms, we know that a causal filter is stable if and only if its poles are located within the unit circle. That implys that this filter is stable if and only if $|r| < 1$. However, we will see that by locating the poles close to the unit circle, the filter may be made very selective to radial frequencies near θ.

This two poles system is an example of an infinite impulse response (IIR) filter because its impulse response has infinite length. Any filter with nontrivial (not 0 or $\pm\infty$) poles are infinite impulse response (IIR) unless the poles are cancelled by zeros.

Calculate the magnitude of the filter's frequency response $|H_i(e^{jw})|$ on $|\omega| < \pi$ for the following three values of θ and r.

- $\theta = \pi/3$ and $r = 0.99$
- $\theta = \pi/3$ and $r = 0.9$

- $\theta = \pi/3$ and $r = 0.7$

Put all three plots on the same figure using the *subplot* command.

INLAB REPORT:
Submit the difference equation, impulse response and flow diagram for the filter $H_i(z)$. Also submit the plot of the magnitude response for the three values of θ and r. Explain how the value of r affects the magnitude response of the filter.

Set $r = 0.99$ and do the following:

1. Write a Matlab function `iir(x,theta)` that implements this filter, filters the input signal `x` and outputs the filtered signal. You can assume that `y(n)` is equal to 0 for negative values of n.
2. Down load the pulse code modulated (PCM) signal (a periodic rectangular waveform multiplied by a cosine function) **pcm.u**.
3. Read the signal contained in *pcm.u* using *auread*, plot it and then play it back with *auplay*.
4. Take the DTFT of the signal contained in *pcm.u* and plot the magnitude of the DTFT.
5. Create a white noise signal `n` of the same size as your pulse code modulated signal `x` by typing the command `randn(size(x))`.
6. Add `0.05*n` to the original signal.
7. Plot the noisy signal and play it back with *auplay*.
8. Take the DTFT of the noisy signal and plot the magnitude of the DTFT.
9. The signal in **pcm.u** is a periodic rectangular waveform multiplied by a cosine function with a frequency of 3kHz. Find the *theta* value of the IIR filter that will filter out the noise and yet keep as much of the PCM signal as possible. Assume that the PCM signal is sampled at 8kHz.
10. Draw a zero-pole plot for the filter with 2 complex conjugate poles at the radius $r = 0.99$ and the θ value that you computed.
11. Filter the signal using your *iir* function with this θ value, take the DTFT of the filtered signal and plot the magnitude of the DTFT.
12. Plot the filtered signal, play it back with *auplay* and listen to it carefully.

INLAB REPORT:
Submit the following:

1. Your flow diagram
2. The prinout of the code for your *iir* function
3. The plot of the magnitude of the DTFT of the original PCM signal
4. The plot of the magnitude of the DTFT of the noisy signal
5. The plot of the magnitude of the DTFT of the filtered noisy signal
6. The time domain plots for the original PCM and the filtered noisy PCM signals

Were you able to filter out all the noise? If not, why? Comment on the difference between the original PCM signal and the filtered noisy PCM signal. Explain the reason behind this difference. Comment on how the frequency content of the signal has changed after filtering.

1.3 Ideal low pass filter and effect of trunctation of the impulse response

The frequency response of an ideal low pass filter with cutoff frequency ω_c is

$$H_{idealLPF}(e^{jw}) = \begin{cases} 1 & |\omega| < \omega_c \\ 0 & |\omega_c < |\omega| \leq \pi \end{cases}$$

The impulse response of this ideal LPF is as follows:

$$h_{ideal}(n) = \frac{\omega_c}{\pi} sinc(\frac{\omega_c n}{\pi}) \quad \text{for} - \infty < n < \infty \tag{1}$$

An ideal LPF is a noncausal filter with an infinite length impulse response and; therefore, is unrealizable. If we truncate this impulse response to the $n \in [-M, M]$ by multiplying we get the following impulse response and the length of the filter N is equal to $2M + 1$:

$$h_{trunc}(n) = \begin{cases} \frac{\omega_c}{\pi} sinc(\frac{\omega_c n}{\pi}) & n = -M, \ldots, 0, 1, \ldots, M \\ 0 & \text{otherwise} \end{cases}$$

Figure **??** shows the magnitude response curve of the LPF with cutoff frequency $3\pi/4$ and with impulse response truncated to $n \in [-10, 10]$. Notice the oscillatory behaviour of the magnitude response near the cutoff frequency. This is due to the abrupt truncation of the impulse response which causes an abrupt cutting off of the Fourier series. The filter with a truncated impulse response approximates the ideal LPF better as the filter length $N = 2M+1$ increases. This truncated impulse response is of finite length, yet the corresponding filter is still noncausal. The remedy is to shift this impulse response to the right M units in time

to get an actually realizable causal filter with following the impulse response $h_{actual}(n)$ of length $N = 2M + 1$.

$$h_{actual}(n) = \begin{cases} \frac{\omega_c}{\pi} sinc(\frac{\omega_c(n-M)}{\pi}) & n = 0, 1, \ldots, N-1 \\ 0 & \text{otherwise} \end{cases}$$

A time shift of M units to the right corresponds to multiplying the frequency response by $e^{-j\omega M}$. Therefore; it does not affect the magnitude response of the filter and adds a $-j\omega M$ to the phase response. To examine the effect of filter length on the frequency characteristics of the filter, write a Matlab function `LPFtrunc(N)` that computes the truncated and shifted impulse response of length $N = 2M + 1$ for a LPF with a cutoff frequency of $3\pi/4$. Your function should also compute the frequency response of the filter using the $DTFT$ function that you down loaded and plot the magnitude response. For comparison, also superimpose the magnitude response plot of the ideal LPF with cutoff frequency $3\pi/4$ as dashed lines onto your magnitude plots. **Hints:** Use the Boolean expression `abs(w)<=3*pi/4` to compute the ideal LPF magnitude response. You can superimpose a second graph in the form dashed lines to an existing graph by typing the following commands after creating the graph.

```
hold on
plot(new_x,new_y,'--');
hold off
```

The *hold on* command holds the current plot and all axis properties so that subsequent graphing commands add to the existing graph. The *hold off* returns to the default mode whereby *plot* commands erase the previous plots and reset all axis properties before drawing new plots.

Use your function *LPFtrunc* and the *subplot* command to plot the magnitude responses (along with the magnitude responses of the ideal LPF) of the LPF's with filter length $N = 31$, $N = 61$ and $N = 121$.

INLAB REPORT:
Submit the printout of your *LPFtrunc* function and the printout of the figure with the 3 plots. On each plot, mark the passband, the transition band and the stopband. Explain how filter length affects the transition bandwidth, the passband and the stopband ripple.

1.4 Design of FIR Filters Using Kaiser Windows

In the previuos section, we examined the effect of the truncation of the impulse response of an ideal filter on the frequency characteristics of the filter especially the transition bandwidth, the passband and the stopband ripple. Truncation of the impulse response $h_{ideal}(n)$ to the interval $n \in [-M, M]$ and then shifting it to the right by M can be viewed windowing the shifted impulse response $h_{ideal}(n - M)$ of the filter with a rectangular window.

$$h_{actual}(n) = h_{ideal}(n)w(n) \tag{2}$$

where

$$w(n) = \begin{cases} 1 & n = 0, 1, \ldots, N-1 \\ 0 & \text{otherwise} \end{cases}$$

where $N = 2M + 1$. In the previuos section, we have observed the Gibbs phenomenon which is due to abrupt changing of the rectangular window from 1 to 0. The Gibbs oscillations in the passband and in the stopband can be diminished by using windows that decrease from 1 to 0 gradually. The mathematical expressions for some commonly used such raised cosine windows are:

1. Hamming window

$$w(n) = \begin{cases} 0.54 - 0.46 \cos \frac{2\pi n}{N-1} & n = 0, 1, \ldots, N-1 \\ 0 & \text{otherwise} \end{cases}$$

2. Hanning window

$$w(n) = \begin{cases} 0.5 - 0.5 \cos \frac{2\pi n}{N-1} & n = 0, 1, \ldots, N-1 \\ 0 & \text{otherwise} \end{cases}$$

3. Blackman

$$w(n) = \begin{cases} 0.42 - 0.5 \cos \frac{2\pi n}{N-1} + 0.8 \cos \frac{4\pi n}{N-1} & n = 0, 1, \ldots, N-1 \\ 0 & \text{otherwise} \end{cases}$$

where N is the length of the windowing functions. For this experiment, N will always be chosen as an odd number. Even using these windows, one can not control both the ripple (δ) and the transition bandwidth ($\Delta\omega$) of a filter. In 1964, Kaiser found a near optimal window that controls both of these attributes. The Kaiser window is defined as

$$w(n) = \begin{cases} I_0 \left[\beta \left(1 - [(n-\alpha)/(\alpha)]^2 \right)^{1/2} \right] & n = 0, 1, \ldots, N-1 \\ 0 & \text{otherwise} \end{cases}$$

where $I_0(\cdot)$ is the zeroth order modified Bessel function of the first kind, N is the length of the filter, $\alpha = (N-1)/2$ and β is the shape parameter. After numerical experimentation, Kaiser found two emprical formulas for predicting the values of N and β of the window needed to meet a given set of design tolerance specifications. The value β is given by

$$\beta = \begin{cases} 0.1102(A - 8.7) & A > 50 \\ 0.5842(A-21)^{0.4} + 0.07886(A-21) & 21 \leq A \leq 50 \\ 0.0 & A < 21 \end{cases}$$

where A is the negative of the peak ripple ($delta$) in dB. That is, $A = -20log(\delta)$. The length of the filter N is given by

$$N = 1 + \frac{A - 8}{2.285 \Delta\omega}$$

where $\Delta\omega$ is the transition bandwidth. Note that $\Delta\omega$ is equal to $\omega_s - \omega_c$ and ω_p

Plot the rectangular ,Hamming, Hanning, and Blackman window functions of length 21 in the same figure. Use the *subplot(2,2,x)* command where $x = 1, 2, 3, 4$. Then, compute using the DTFT $W(e^{jw})$ of these windows using the function that you downloaded in before. Also, plot the magnitude response in dB $20log|W(e^{jw})|$ of the 4 windows in a single figure. Note that the *log* command in Matlab computes the natural logarithm. Therefore, use the *log10* command to compute the logarithmic magnitude response.

INLAB REPORT:

1. Submit the printout of the figure with the plots of the 4 windows and the figure with the plots of their magnitude responses.
2. Measure the mainlobe width (in radians) and the peak sidelobe amplitude (in decibels) for the logarithmic magnitude response plot of each window type. Make a table of these values.
3. Comment on the relation between the width of the mainlobe and the peak sidelobe amplitude.

Now, we will implement the LPF with cutoff frequency of $3\pi/4$ using these windows. Window the ideal impulse response

$$h_{ideal}(n) = \frac{3}{4} sinc(\frac{3n}{4}) \quad \text{for} -\infty < n < \infty$$

with the 4 types of windows with length 21 as in equation 2. Then, compute the DTFT of the windowed impulse function and find the magnitude response of the designed filter $|H(e^{jw})|$ for each window type. Plot the 4 magnitude response plots in the same figure using the *subplot* command.

INLAB REPORT:

1. Submit the printout of the figure with the 4 magnitude response plots.
2. For each window type, measure the peak ripple amplitude and the the transition bandwidth (the distance between the last ripple peak in the passband and the first ripple peak in the stopband). Append the table you made above, the one with the two columns
   (a) the mainlobe width of the window
   (b) the peak sidelobe amplitude of the window

   to include the following 2 columns:
   (a) the transition bandwidth of the resulting filter
   (b) the peak ripple amplitude of the resulting filter
3. Comment on the relation between the columns of the table. What is the design tradeoff that you observe?
4. For what type applications will you prefer to use a rectangular window to a Blackman window, and vice versa? Why?

1.5 Optimal FIR Filter Design

Help on **Parks-McClellan design of FIR filters using Matlab**
Down load **noisyspeech.u**

Even though the Kaiser window has certain near-optimal properties, the filters designed using it are not optimal in the minimum mean-squared error sense or the maximum error sense.

It can be shown that the filters that minimize integral mean-squared frequency error E given by

$$E_{mse} = \tag{3}$$

are nothing but the truncation of the ideal impulse response i.e the windowing of the ideal impulse response using a rectangular window. Yet, in the sections 1.3 and 1.4 we have seen several drawbacks of the rectangular window. The peak ripple value can not be decreased even if the filter length is increased. The amplitude of the ripple is largest on either side of the transition region and the error decreases for frequencies away from the transition region. Another drawback of rectangular or any of the common windows discussed in section

1.4 is that they result in almost equal peak passband and stopband ripple amplitudes. If the approximation error (ripple) can be spread out uniformly and if the passband and the stopband ripples can be adjusted separately, a given design specification can be met with a lower order filter. In 1972, Parks and McClellan devised an algorithm that minimizes the maximum error between the desired frequency response and the actual frequency response by spreading the approximation error uniformly and adjusting the passband and the stopband ripples separately. Their algorithm makes use of the Remez exchange algorithm and the Chebyshev approximation theory. Filters designed using their algorithm exhibit equiripple behaviour in the passband and in the stopband and are therefore sometimes called equiripple filters.

In Matlab, designing a filter with Parks-McClellan algorithm is a two step process. Given the design specifications, you first need to calculate the order of the filter using the *remezord* command. Then, you need to use the *remezord* command to actually design the filter. Read the help document on **Parks-McClellan design of FIR filters using Matlab**.

You are provided a signal speech signal contaminated with high frequency noise **noisyspeech.u**. Down load this file, read the signal contained in *pcm.u* using *auread*, then play it back with *auplay*. Then, take the DTFT of the signal contained in *pcm.u* and plot the magnitude of the DTFT, your plot should look like Figure **??**. Also plot the magnitude of the DTFT in dB (Recall x in dB means 20log(x)). In this signal, the speech is bandlimited to the frequencies $-1.9 < \omega < 1.9$ and the noise is bandlimited to the frequencies $|\omega| < 2.2$. Calculate the attenuation in dB in the stopband required to decrease the magnitude of the DTFT of the filtered signal to less than 0.001 in the stopband. Then, do the following:

1. Write a *.m* script file that will design an FIR low-pass filter $h(n)$ with a passband ripple of 1 dB and a stopband attenuation that you calculated above using the *remezord* and *remez* commands.

2. Take the DTFT of the impulse response of the filter $h(n)$ and plot the magnitude response of the filter in dB. Measure the passband and the stopband ripple and check if you met the design specifications.

3. Filter the signal contained in **noisyspeech.u** by convolving it with the filter. Use the *conv* command.

4. Take the DTFT of the filtered signal and plot the magnitude of the DTFT of the filtered signal.

5. Play the filtered signal back with *auplay*, listen to it carefully and compare it with the unfiltered noisy signal.

INLAB REPORT:
Submit the printout of the magnitude response plot of the filter. Mark the passband and the stopband ripple on this printout. Also, submit the printout of the magnitude plot of the DTFT of the filtered signal. Mark the maximum magnitude of the DTFT of the filtered signal in the stopband. Is it within your design tolerance specifications? Comment on how the frequency content of the signal and the audio quality of the signal has changed after filtering.

1.6 Design of Discrete-Time IIR Filters from Continuous-Time Filters

One of the approaches to the design of IIR filters is to first design a continuous-time filter to meet the desired specifications and then to transform it into a discrete time filter. It is sometimes advantageous to design the filter in continuous-time because the art of continuous-time IIR design is highly advanced and many continuous-time IIR design methods have simple closed form formulae. There are two techniques of transforming a continuous-time filter into a discrete-time filter. The first technique is called *impulse invariance* which involves the sampling of the impulse response of a continuous-time system. The second technique is called *bilinear transformation* which is an algebraic transformation between the variables s and z. For brevity, we will only do an example with bilinear transformation in this section. Bilinear transformation is transformation that maps the entire $j\Omega$ axis in the s -plane to one revolution of the unit circle in the z -plane. If $H_c(s)$ denotes the continuous-time system function, $H(z)$ denotes the discrete-time system function and T_s denotes the sampling period, the bilinear transformation is equivalent to substituing

$$s = \frac{2}{T_s}\left(\frac{1-z^{-1}}{1+z^{-1}}\right) \tag{4}$$

for s in $H_c(s)$ to get $H(z)$. An analysis of equation 4 shows that the continuous-time frequency Ω is related to the discrete time frequency ω by the following expression:

$$\omega = 2\arctan \Omega T_s/2 \tag{5}$$

Consider the problem of designing a discrete-time IIR filter from continuous-time filter using the bilinear transformation. You are given values for the lower and upper passband edge frequencies (ω_{pl} , ω_{pu}), the lower and upper stopband edge frequencies (ω_{sl} , ω_{su}) the passband ripple (δ_1) and the lowpass ripple (δ_2). Your design specifications are

$$(1-\delta_1) \leq |H(e^{j\omega})| \leq (1+\delta_1) \text{ for } \omega_{pl} \leq \omega \leq \omega_{pu}. \tag{6}$$

$$|H(e^{jw})| \leq \delta_2 \text{ for } \omega_{sl} \leq \omega \leq \omega_{su}. \tag{7}$$

First you need to find the corresponding design specifications to those in the continuous-time domain. To do this, replace $|H(e^{j\omega})|$ by $|H_c(j\Omega)|$ and change the passband and stopband

frequencies from discrete-time values to continuous-time values by substituting equation 5 for ω. Now that you have the specifications for a continuous-time filter, you need to first choose one of following common family of continuous-time filters: Butterworth, Chebyshev, elliptic. These family of continuous-time filters have closed form magnitude response formulae. For example, N-th order Butterworth continuous-time filter has magnitude squared function

$$|H_c(j\Omega)|^2 = \frac{1}{1 + (\Omega/\Omega_c)^{2N}} \tag{8}$$

where Ω_c is the continuous-time cutoff frequency, the frequency at which $|H_c(j\Omega)|$ drops to -3dB i.e $1/\sqrt{2}$ of its maximum value. Using these closed form expressions, you can find the minimum order N and the cutoff frequency Ω_c that will meet your design specifications. N and Ω_c give the location of the poles of the filter. For example, Butterworth filters have poles uniformly distributed in angle on a circle of radius Ω_c. That is, an N-th order Butterworth filter with a cutoff frequency Ω_c has N poles located at

$$p_k = \Omega_c e^{j\frac{\pi}{2}} e^{j\frac{\pi(2k+1)\pi}{2N}} \text{ for } k = 0, 1, \ldots, N-1. \tag{9}$$

After the poles are found, you can write the expression for $|H_c(s)|$.

$$|H_c(s)| = \frac{1}{\prod_{k=0}^{N-1}(s - p_k)} \tag{10}$$

This concludes the design of your continuous-time filter. Finally, you need to obtain the system function $H(z)$ for your discrete-time filter by applying the bilinear transformation (4) to $H_c(s)$.

Assume that you have completed the first stage of the procedure above and that you have calculated the order N of the Butterworth filter that will meet your design specifications to be 1 and its cutoff frequency Ω_c to be 1000Hz. Plot $|H_c(j\Omega)|$ (equation 8) for $0 \leq \Omega < \infty$. Check that $|H_c(j\Omega_c)|$ equals $1/\sqrt{2}$ of its maximum value 1. Find the location of the two poles using equation 9. Find the expression for $H_c(s)$ using equation 10. Then, apply the bilinear transformation (4) with $T_s = 1$ to $H_c(s)$, and obtain $H(z)$. Plot $|H(e^{jw})|$ for $-\pi \leq \omega \leq \pi$. Measure the discrete-time cutoff frequency w_c.

INLAB REPORT:
Submit your expression for $H_c(s)$ and the printout of its magnitude plot. Show how you derived $H(z)$ from $H_c(s)$ and submit the magnitude response plot $|H(e^{jw})|$ of the discrete-time Butterworth filter. Mark the discrete-time cutoff frequency w_c on this second plot and verify that is satisfies the frequency mapping equation (5).

1.7 Design of Discrete-Time IIR Filters Using Numerical Optimization

Another approach to the design of discrete-time IIR filters is direct search of filter parameters to find the filter which minimizes a specific design criterion. Such "brute force" approaches

to filter design have become increasing more popular due to the wide availability of high speed computers and robust numerical optimization methods.

Typically, numerical search approaches to filter design work by first posing a design cost criterion and then minimizing the criterion with respect to the filter parameters. We will perform the required numerical optimization using a function contained in Matlab's *Optimization Toolbox.*

In order to formulate a cost criterion, we must first select a method to parameterize the discrete-time filters of interest. There are many ways of doing this, but we will use the coefficients of the rational transfer function to parameterize the set of second order IIR filters. In this case, the elements of the vector $\theta = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]$ are the coefficients of the transfer function

$$H_\theta(z) = \frac{\theta_1 + \theta_2 z + \theta_3 z^2}{1 + \theta_4 z + \theta_5 z^2} \ . \tag{11}$$

Using this parameterization, we may then define a function $\text{Cost}(\theta)$ which is the "Cost" of using the filter $H_\theta(z)$.

To illustrate this numerical optimization approach, you will design a digital filter that compensates for the roll-off due to the sample-and-hold in an audio CD player. In laboratory 4, you saw that a sample-and-hold of a conventional D/A converter causes the reconstructed signal to be filtered by the function

$$H_{sh}(e^{j\omega}) = \text{sinc}(\omega) \quad \text{for } |\omega| < \pi$$

One method of reducing this distortion is to digitally filter with the inverse transfer function, $1/H_{sh}$. This filter $1/H_{sh}$ predistorts the signal so the reconstructed audio signal has the desired frequency response. For an audio CD player the magnitude of the frequency response is generally considered to be more important than the phase, so we may choose a cost function which computes the total squared error between the magnitudes of the desired correction $1/H_{sh}(e^{j\omega})$ and the second order filter $H_\theta(e^{j\omega})$.

$$\text{Cost}(\theta) = \int_{-\pi}^{\pi} \left(\frac{1}{H_{sh}(e^{j\omega})} - \left| H_\theta(e^{j\omega}) \right| \right)^2 d\omega \tag{12}$$

A more complex approach might also account for the filter phase, but for simplicity we treat the filter magnitude only.

Once the filter is designed, we may compute the deviation of the CD players frequency response from the ideal "flat" response by computing

$$\text{Err}_{dB}(\omega) = 20 \log \left(H_{sh}(e^{j\omega}) H_{\theta^*}(e^{j\omega}) \right) \tag{13}$$

where $H_{\theta^*}(e^{j\omega})$ is the optimized filter design.

Do the following:

- Write a Matlab function `filt(w,theta)` which computes the frequency response $H_{\theta^*}(e^{j\omega})$ of (11) for the vector of input frequecies `w` and the parameter vector `theta`.

- Write a Matlab function `Cost(theta)` which computes the total squared error of equation (12) by sampling the function $H_{\theta^*}(e^{j\omega})$ at points spaced by $\Delta\omega = 0.01$.
- Use the command `fmins` from the Matlab *Optimization Toolbox* to compute the value of the parameter θ which minimizes `Cost(theta)`. **Hint:** The function `fmins` has the syntax `fmins('function_name',initial_value)` where `function_name` is the name of the function being minimized and `initial_value` is the starting value for the unknown parameter. Choose an initial value of θ so that $H_\theta(e^{j\omega}) = 1$.
- Use the *subplot* command to plot the following three functions on the interval $[-\pi, \pi]$.
  - The desired filter response $1/H_{sh}(e^{j\omega})$.
  - The designed IIR filter response $H_{\theta^*}(e^{j\omega})$.
  - The frequency response error in dB of equation 13.

INLAB REPORT:
Submit the printout of your two Matlab functions *filt.m* and *Cost.m* and a printout of the optimized filter parameters θ^*. Also submit the plot of the desired filter response, designed filter response, and error in dB. At which range of frequencies, is the approximation error high? Why?