ECE438 - Laboratory 5:
Digital Filter Design
(Week 2)
October 6, 2010

1   Introduction

This is the second part of a two week laboratory in digital filter design. The **first week** of the laboratory covered some basic examples of FIR and IIR filters, and then introduced the concepts of filter design. In this week we will cover more systematic methods of designing both FIR and IIR filters.

2   Filter Design Using Standard Windows

Download **DTFT.m**

We can generalize the idea of truncation by using different windowing functions to truncate an ideal filter's impulse response. Note that by simply truncating the ideal filter's impulse response, we are actually multiplying (or "windowing") the impulse response by a shifted $rect()$ function. This particular type of window is called a *rectangular* window. In general, the impulse reponse $h(n)$ of the designed filter is related to the impulse response $h_{ideal}(n)$ of the ideal filter by the relation

$$h(n) = w(n)h_{ideal}(n), \tag{1}$$

where $w(n)$ is an N-point window. We assume that

$$h_{ideal}(n) = \frac{\omega_c}{\pi}\text{sinc}\left(\frac{\omega_c}{\pi}\left(n - \frac{N-1}{2}\right)\right), \tag{2}$$

where ω_c is the cutoff frequency and N is the desired window length.

The rectangular window is defined as

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

The DTFT of $w(n)$ for $N = 21$ is shown in Fig. 1. The rectangular window is usually not preferred because it leads to the large stopband and passband ripple as shown in Fig. 2.
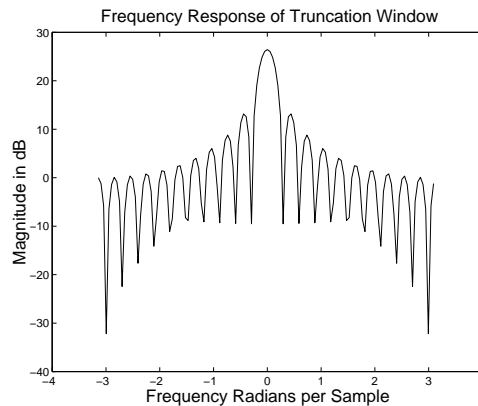

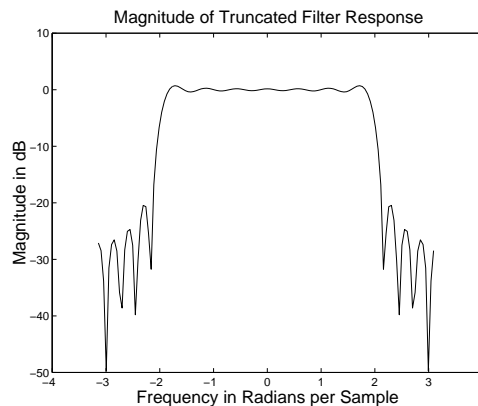Figure 1: DTFT of a rectangular window of length 21.

Figure 2: Frequency response of low-pass filter, designed using the truncation method.

More desirable frequency characteristics can be obtained by making a better selection for the window, $w(n)$. In fact, a variety of raised cosine windows are widely used for this purpose. Some popular windows are listed below.

1. Hanning window (as defined in Matlab, command *hann(N)*):

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

2. Hamming window

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

3. Blackman window

$$w(n) = \begin{cases} 0.42 - 0.5\cos\frac{2\pi n}{N-1} + 0.08\cos\frac{4\pi n}{N-1} & n = 0, 1, \ldots, N-1 \\ 0 & \text{otherwise} \end{cases}$$

In filter design using different truncation windows, there are two key frequency domain effects that are important to the design: the transition band *roll-off*, and the passband and stopband *ripple* (see Fig. 3 below). There are two corresponding parameters in the spectrum of each type of window that influence these filter parameters. The filter's roll-off is related to the *width* of center lobe of the window's magnitude spectrum. The *ripple* is influenced by the ratio of the mainlobe amplitude to the first sidelobe amplitude (or difference if using a dB scale). These two window spectrum parameters are *not* independent, and you should see a trend as you examine the spectra for different windows. The theoretical values for the *mainlobe width* and the *peak-to-sidelobe amplitude* are shown in Table 1.

Window (length N)	Mainlobe width	Peak-to-sidelobe amplitude (dB)
Rectangular	$4\pi/N$	$-13dB$
Hanning	$8\pi/N$	$-32dB$
Hamming	$8\pi/N$	$-43dB$
Blackman	$12\pi/N$	$-58dB$

Table 1: Approximate spectral parameters of truncation windows. See [1].

Plot the rectangular, Hamming, Hanning, and Blackman window functions of length 21 on a single figure using the *subplot* command. You may use the Matlab commands *hamming*, *hann*, and *blackman*. Then compute and plot the DTFT magnitude of each of the four windows. Plot the magnitudes on a decibel scale (i.e., plot $20\log_{10}|W(e^{j\omega})|$). Download and use the function **DTFT.m** to compute the DTFT.

Hint: Use at least 512 sample points in computing the DTFT by typing the command `DTFT(window,512)`. Type `help DTFT` for further information on this function.

Measure the null-to-null mainlobe width (in rad/sample) and the peak-to-sidelobe amplitude (in dB) from the logarithmic magnitude response plot for each window type. The Matlab command *zoom* is helpful for this. Make a table with these values *and* the theoretical ones.

Now use a Hamming window to design a lowpass filter $h(n)$ with a cutoff frequency of $\omega_c = 2.0$ and length 21. **Note:** You need to use equations (1) and (2) for this design. In the same figure, plot the filter's impulse response, and the magnitude of the filter's DTFT in decibels.

INLAB REPORT:

1. Submit the figure containing the time domain plots of the four windows.

2. Submit the figure containing the DTFT (in decibels) of the four windows.

3. Submit the table of the measured and theoretical window spectrum parameters. Comment on how close the experimental results matched the ideal values. Also comment on the relation between the width of the mainlobe and the peak-to-sidelobe amplitude.

4. Submit the plots of your designed filter's impulse response and the magnitude of the filter's DTFT.

---

3 Filter Design Using the Kaiser Window
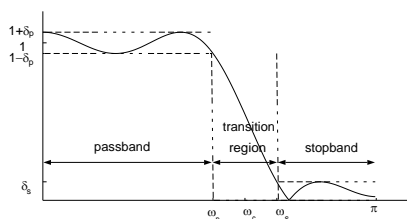
Download **nspeech2.mat**

Figure 3: Tolerance specifications for the frequency response of a filter.

The standard windows of Section 2 are an improvement over simple truncation, but these windows still do not allow for arbitrary choices of transition bandwidth and ripple. In 1964, James Kaiser derived a family of near-optimal windows that can be used to design filters which meet or exceed any filter specification. The Kaiser window depends on two parameters: the window length N, and a parameter β which controls the shape of the window. Large values of β reduce the window sidelobes and therefore result in reduced passband and stopband ripple. The only restriction in the Kaiser filter design method is that the passband and stopband ripple must be equal in magnitude. Therefore, the Kaiser filter must be designed to meet the smaller of the two ripple constraints:

$$\delta = \min\{\delta_p, \delta_s\}$$

The Kaiser window function of length N is given by

$$w(n) = \begin{cases} \dfrac{I_0\left(\beta \dfrac{\sqrt{n(N-1-n)}}{N-1}\right)}{I_0(\beta)} & n = 0, 1, \ldots, N-1 \\ 0 & \text{otherwise} \end{cases}$$

where $I_0(\cdot)$ is the zero'th order modified Bessel function of the first kind, N is the length of the window, and β is the shape parameter.

Kaiser found that values of β and N could be chosen to meet any set of design parameters, $(\delta, \omega_p, \omega_s)$, by defining $A = -20 \log_{10} \delta$ and using the following two equations:

$$\beta = \begin{cases} 0.1102(A - 8.7) & A > 50 \\ 0.5842(A - 21)^{0.4} + 0.07886(A - 21) & 21 \le A \le 50 \\ 0.0 & A < 21 \end{cases} \tag{3}$$

$$N = \left\lceil 1 + \frac{A - 8}{2.285(\omega_s - \omega_p)} \right\rceil \tag{4}$$

where $\lceil \cdot \rceil$ is the *ceiling* function, i.e. $\lceil x \rceil$ is the smallest integer which is greater than or equal to x.

To further investigate the Kaiser window, plot the Kaiser windows and their DTFT magnitudes (in dB) for $N = 21$ and the following values of β:

- $\beta = 0$

- $\beta = 1$

- $\beta = 5$

For each case use at least 512 points in the plot of the DTFT.

Hint: To create the Kaiser windows, use the Matlab command `kaiser(N,beta)` command where `N` is the length of the filter and `beta` is the shape parameter β. To insure at least 512 points in the plot use the command `DTFT(window,512)` when computing the DTFT.

INLAB REPORT: Submit the plots of the 3 Kaiser windows and the magnitude of their DTFT's in decibels. Comment on how the value β affects the shape of the window and the sidelobes of the DTFT.
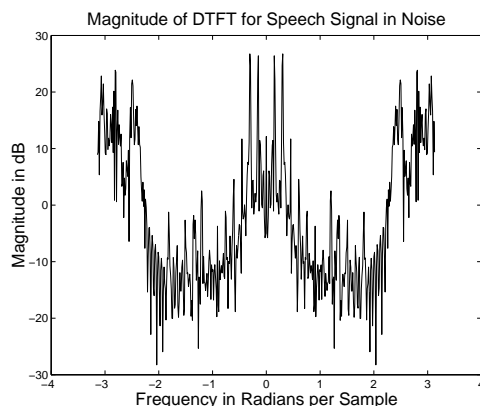

Figure 4: DTFT of a section of noisy speech.

Next use a Kaiser window to design a low pass filter, $h(n)$, to remove the noise from the signal in **nspeech2.mat** using equations (1) and (2). To do this, use equations (3) and (4) to compute the values of β and N that will yield the following design specifications:

$$
\begin{aligned}
\omega_p &= 1.8 \\
\omega_c &= 2.0 \\
\omega_s &= 2.2 \\
\delta_p &= 0.05 \\
\delta_s &= 0.005
\end{aligned}
$$

The low pass filter designed with the Kaiser method will automatically have a cut-off frequency centered between ω_p and ω_s.

$$
\omega_c = \frac{\omega_p + \omega_s}{2}
$$

Plot the magnitude of the DTFT of $h(n)$ for $|\omega| < \pi$. Create three plots in the same figure: one that shows the entire frequency response, and ones that zoom in on the passband and stopband ripple, respectively. Mark ω_p, ω_s, δ_p, and δ_s on these plots where appropriate. **Note:** Since the ripple is measured on a magnitude scale, DO NOT use a decibel scale on this set of plots.

From the Matlab prompt, compute the stopband and passband ripple (do not do this graphically). Record the stopband and passband ripple to three decimal places.

Hint: Find the value of the DTFT at frequencies corresponding to the passband using the command `H(abs(w)<=1.8)` where `H` is the DTFT of $h(n)$ and `w` is the corresponding vector of frequencies. Then use this vector to compute the passband ripple. Use a similar procedure for the stopband ripple.

Filter the noisy speech signal in **nspeech2.mat** using the filter you have designed. Then compute the DTFT of 400 samples of the filtered signal starting at time $n = 20,001$ (i.e. `20001:20400`). Plot the magnitude of the DTFT samples in decibels versus frequency in radians for $|\omega| < \pi$. Compare this with the spectrum of the noisy speech signal shown in Fig. 4. Play the noisy and filtered speech signals back using *sound* and listen to them carefully.

INLAB REPORT: Do the following:

1. Submit the values of β and N that you computed.

2. Submit the three plots of the filter's magnitude response. Make sure the plots are labeled.

3. Submit the values of the passband and stopband ripple. Does this filter meet the design specifications?

4. Submit the magnitude plot of the DTFT in dB for the filtered signal. Compare this plot to the plot of Fig. 4.

5. Comment on how the frequency content and the audio quality of the filtered signal have changed after filtering.

4 FIR Filter Design Using Parks-McClellan Algorithm

Help on Matlab commands for **Parks-McClellan filter design**
Download **nspeech2.mat**

Kaiser windows are versatile since they allow the design of arbitrary filters which meet specific design constraints. However, filters designed with Kaiser windows still have a number of disadvantages. For example,

- Kaiser filters are not guaranteed to be the minimum length filter which meets the design constraints.

- Kaiser filters do not allow passband and stopband ripple to be varied independently.

Minimizing filter length is important because in many applications the length of the filter determines the amount of computation. For example, an FIR filter of length N may be directly implemented in the time domain by evaluating the expression

$$y(n) = \sum_{k=0}^{N-1} x(n-k)h(k) \ . \tag{5}$$

For each output value $y(n)$ this expression requires N multiplies and $N-1$ additions.

Oftentimes $h(n)$ is a symmetric filter so that $h(n) = h(N-1-n)$. If the filter $h(n)$ is symmetric and N is even, then (5) may be more efficiently computed as

$$y(n) = \sum_{k=0}^{N/2-1} \left(x(n-k) + x(n-N+1+k) \right) h(k) \ .$$

This strategy reduces the computation to $N/2$ multiplies and $N - 1$ adds for any value of N[1]. Note that the computational effort is linearly proportional to the length of the filter.

The Kaiser filters do not guarantee the minimum possible filter length. Since the filter has equal passband and stopband ripple, it will usually exceed design requirements in one of the two bands; this results in an unnecessarily long filter. A better design would allow the stopband and passband constraints to be specified separately.

In 1972, Parks and McClellan devised a methodology for designing symmetric filters that minimize filter length for a particular set of design constraints $\{\omega_p, \omega_s, \delta_p, \delta_s\}$. The resulting filters minimize the maximum error between the desired frequency response and the actual frequency response by spreading the approximation error uniformly over each band. The Parks and McClellan algorithm makes use of the Remez exchange algorithm and Chebyshev approximation theory. Such filters that exhibit *equiripple* behavior in both the passband and the stopband, and are sometimes called equiripple filters.

As with Kaiser filters, designing a filter with the Parks and McClellan algorithm is a two step process. First the length (i.e. order) of the filter must be computed based on the design constraints. Then the optimal filter for a specified length can be determined. As with Kaiser windows, the filter length computation is approximate so the resulting filter may exceed or violate the design constraints. This is generally not a problem since the filter can be redesigned for different lengths until the constraints are just met.

The Matlab command for computing the approximate filter length is
```
[n,fo,mo,w] = firpmord(f,m,ripple,2*pi)
```
where the inputs are:

> `f` - vector containing an even number of band edge frequencies. For a simple low pass filter, `f=[wp,ws]`, where `wp` and `ws` are the passband and stopband frequencies, respectively.

> `m` - vector containing the ideal filter magnitudes of the filter in each band. For a simple low pass filter `m=[1,0]`.

> `ripple` - vector containing the allowed ripple in each band. For a simple low pass filter `ripple=[delta_p,delta_s]`, where `delta_p` and `delta_s` are the passband and stopband ripples, respectively.

> `2*pi` - value, in radians, that corresponds to the sampling frequency.

The outputs of the command are `n = filter length - 1`, and the vectors `fo`, `mo`, and `w` which are intermediate filter parameters.

Once the filter length, `n`, is obtained, the Matlab command for designing a Parks-McClellan filter is `b = firpm(n,fo,mo,w)`. The inputs `n`, `fo`, `mo`, and `w` are the corresponding outputs of `firpmord`, and the output b is a vector of FIR filter coefficients such

---

[1]The advantages of using such symmetries varies considerably with the implementation and application. On many modern computing architectures the computational cost of adds and multiplies are similar, and the overhead of control loops may eliminate the advantages of reduced operations.

that

$$H(z) = b(1) + b(2)z^{-1} + \cdots + b(n+1)z^{-n}$$

(What is the impulse response of this filter?)

Read the help document on using Matlab to implement the **Parks-McClellan algorithm** for further information.

Now design a symmetric FIR filter using *firpmord* and *firpm* in Matlab to meet the design specifications given in Section 3. Compute the DTFT of the filter's response for at least 512 points, and use this result to compute the passband and stopband ripple of the filter that was designed. Adjust the filter length until the minimum order which meets the design constraints is found. Plot the magnitude of the DTFT in dB of the final filter design.

INLAB REPORT:
Do the following:

1. Submit the final measured values of filter length, passband ripple, and stopband ripple. How accurate was the filter order computation using Matlab's *firpmord*? How does the length of this filter compare to the filter designed using a Kaiser window?

2. Submit the plot of the filter's DTFT. How does the frequency response of the Parks-McClellan filter compare to the filter designed using the Kaiser window? Comment on the shape of both the passband and stopband.

Use the filter you have designed to remove the noise from the signal **nspeech2.mat** . Play the noisy and filtered speech signals back using *sound* and listen to them carefully. Compute the DTFT of 400 samples of the filtered signal starting at time $n = 20,001$ (i.e. `20001:20400`). Plot the magnitude of the DTFT in decibels versus frequency in radians for $|\omega| < \pi$. Compare this with the spectrum of the noisy speech signal shown in Fig. 4, and also with the magnitude of the DTFT of the Kaiser filtered signal.

INLAB REPORT:
Submit the plot of the DTFT magnitude for the filtered signal. Comment on how the audio quality of the signal changes after filtering. Also comment on any differences in audio quality between the Parks-McClellan filtered speech and the Kaiser filtered speech.

---

5   Design of Discrete-Time IIR Filters Using Numerical Optimization

In this section, we consider the design of discrete-time IIR filters through the direct search of filter parameters that will minimize a specific design criterion. Such "brute force" approaches to filter design have become increasingly more popular due to the wide availability of high speed computers and robust numerical optimization methods.

Typically, numerical approaches to filter design have two parts. First, they design a *cost,* or *error* criterion. This criterion is a measure of the difference between the ideal filter response and the response of the computed or "approximate" filter. The goal is to find the approximate filter with the lowest cost (error). Mean square error is a popular cost criterion. The second part is to minimize the cost with respect to the filter parameters. We will perform the required numerical optimization with the *fminsearch* function in Matlab's *Optimization Toolbox.*

In order to formulate a cost criterion, we must first select a model for the discrete-time filter of interest. There are many ways of doing this, but we will use the coefficients of rational transfer function to model (or 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

$$H_\theta(z) = \frac{\theta_1 + \theta_2 z^{-1} + \theta_3 z^{-2}}{1 + \theta_4 z^{-1} + \theta_5 z^{-2}} \ . \tag{6}$$

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, we will design a digital filter that compensates for the roll-off due to the sample-and-hold process in an audio CD player. In lab 4, we saw that the sample-and-hold operation in a conventional D/A converter causes the reconstructed signal to be filtered by the function

$$H_{sh}(e^{j\omega}) = \text{sinc}\left(\frac{\omega}{2\pi}\right) \quad \text{for } |\omega| < \pi$$

One method of reducing this distortion is to digitally "pre-filter" a signal with the inverse transfer function, $1/H_{sh}$. This filter $1/H_{sh}$ pre-distorts the audio signal so the reconstructed signal has the desired frequency response. We would like to approximate the filter $1/H_{sh}$ using the second order filter of equation (6).

For an audio CD player, the magnitude of the frequency response is generally considered to be more important than the phase. This is because we are not perceptually sensitive to phase distortion in sound. Therefore, we may choose a cost function which computes the total squared error between the magnitudes of the desired pre-filter response, $1/H_{sh}(e^{j\omega})$, and the second order filter $H_\theta(e^{j\omega})$:

$$\text{Cost}(\theta) = \int_{-\pi}^{\pi} \left(\left| \frac{1}{H_{sh}(e^{j\omega})} \right| - \left| H_\theta(e^{j\omega}) \right| \right)^2 d\omega \tag{7}$$

The θ parameters that minimize this cost function will be the parameters of our designed filter. A more complex approach might also account for the filter phase, but for simplicity we will only try to match the filter magnitudes.

After the filter is designed, we may compute the difference between the CD player's frequency response in dB and the ideal desired response in dB that the CD player should have:

$$\text{Err}_{dB}(\omega) = 20 \log_{10} \left(H_{sh}(e^{j\omega}) | H_{\theta*}(e^{j\omega}) | \right) \tag{8}$$

where θ^* is the optimized value of θ and $H_{\theta^*}(e^{j\omega})$ is the optimum second-order filter.

Do the following to perform this filter design:

- Write a Matlab function `prefilter(w,theta)` which computes the frequency response $H_\theta(e^{j\omega})$ from equation (6) for the vector of input frequencies `w` and the parameter vector `theta`.

- Write a Matlab function `Cost(theta)` which computes the total squared error of equation (7). Use a sampling interval $\Delta\omega = 0.01$ for the functions $H_\theta(e^{j\omega})$ and $1/H_{sh}(e^{j\omega})$.

- Use the command `fminsearch` from Matlab's *Optimization Toolbox* to compute the value of the parameter θ which minimizes `Cost(theta)`. The function `fminsearch` has the syntax `X = fminsearch('function_name',initial_value)` where `function_name` is the name of the function being minimized (Cost), `initial_value` is the starting value for the unknown parameter, and X is the minimizing parameter vector. Choose an initial value of $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5) = (1, 0, 0, 0, 0)$ 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 magnitude response $1/H_{sh}(e^{j\omega})$.
    - The designed IIR filter magnitude response $|H_{\theta^*}(e^{j\omega})|$.
    - The error in decibels, from equation (8).

INLAB REPORT:
Do the following:

1. Submit the printouts of the code for the two Matlab functions *prefilter.m* and *Cost.m*.

2. Give an analytical expression for the optimized transfer function $H_{\theta^*}(z)$ with the co-efficients that were computed.

3. Submit the three plots. On the error plot, mark the frequency ranges where the approximation error is high.

References

[1] J. G. Proakis, D. G. Manolakis, *Digital Signal Processing, 3rd ed.,* Prentice-Hall, 1996.