FIR Filter Design by Optimization

This lecture describes the design of optimum equiripple FIR filters using the Parks-McClellan algorithm.
After this lecture you should be able to: (1) explain the optimization criteria used to design equiripple FIR filters, (2) specify a filter in terms of transition frequencies, mean passband/stopband levels and relative passband/stopband ripple, (3) explain the importance of checking the results of the Parks-McClellan algorithm and how to proceed if a design fails to converge.
This material is covered is section 6.6 of the text.

Optimality Criteria

The most common design method for FIR filters is to use a computer program to optimize the impulse response, or equivalently, the values of the filter coefficients a_i. The program is given a frequency specification of the desired filter and it then attempts to minimize an error criteria. The design is almost always constrained to have a symmetrical impulse response to ensure linear phase.

The quantity optimized (minimized) most often is the maximum deviation from the desired amplitude response. For FIR filters this produces a design in which the peak deviations from the desired response (the "ripples") are all of equal magnitude. This is known as an *equiripple* design.

Design Procedure

The most widely used optimization algorithm is called "Parks-McClellan" (or "Remez exchange" based on one of the techniques employed by the algorithm). Matlab and many other DSP design tools include a Parks-McClellan FIR design function. This algorithm attempts to equalize and minimize the ripples for a given frequency specification and filter order. If the given filter order is too small the program may fail to find an equiripple solution. If the given filter order is too high the result will be an equiripple filter that exceeds the ripple requirements.

The filter design steps are as follows:

1. The first step is to specify the frequency ranges (typically normalized to half of the sampling rate), the mean gain level (typically in linear units) for each frequency range and the relative weighting of the ripple magnitudes for each of the frequency ranges. This last specification allows filters to be designed with different amounts of ripple in different frequency ranges.

2. The second step is to estimate the filter order that will be required. Simple equations are available to get a first estimate of the order that will be required. This number can be changed on subsequent iterations to try to find the minimum filter order that will meet the specifications.

3. An optimization program is then run to compute the filter coefficients. In the case of Matlab the function `remezord()` helps with the first two steps and the `remez()` function actually designs the filter.

4. It is essential to check the result (typically by plotting the amplitude response) to make sure the program found an equiripple solution that meets the requirements. In Matlab the `freqz()` function can be used to do this.

Design Examples

The design process is best explained by means of examples.

Low-Pass Filter

The first example is a simple low-pass filter. The sampling rate is 16 kHz, the passband is from 0 to 3800 Hz, and the stopband from 4000 Hz up. We want the passband ripple to be less than 0.1dB $- 10^{0.1/20} = 0.0115$ and the minimum stopband attenuation at least -40dB$10^{-40/20} = 0.01$. Note that since the mean stopband level is 0, the ripple is equal to the minimum stopband attenuation.

The first set of Matlab commands uses the `remezord()` function to convert the filter specification given in a (slightly) user-friendly format to the normalized and expanded values required by the `remez()` function. The frequency values specify the transition region frequencies in Hz. The mean levels and the relative ripple for each frequency region are in linear units (in the example below the level/ripple specifications are 1 ± 0.011 and 0 ± 0.01).

The commands:

```
>> F=[ 3800 4000 ];
>> M=[1 0];
>> DEV=[0.011 0.01];
>> Fs=16000;
>> [N Fo Mo W]=remezord(F,M,DEV,Fs)
```

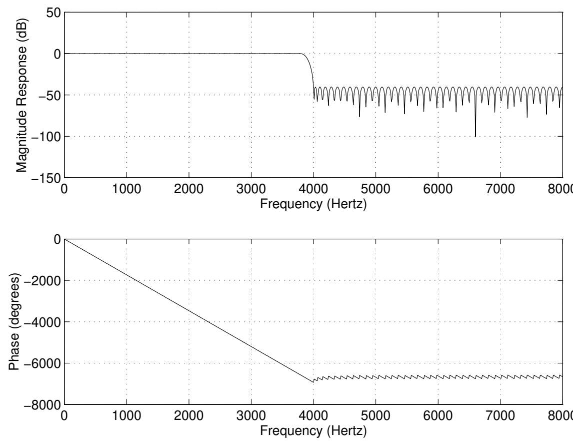
result in an estimated filter order of $N = 154$, expanded and normalized breakpoint frequencies $Fo = \{0, 0.4750, 0.5000, 1.0000\}$, mean passband and stopband values of $Mo = \{1, 1, 0, 0\}$, and relative maximum error (ripple) weightings $W = \{1.0, 1.1\}$.

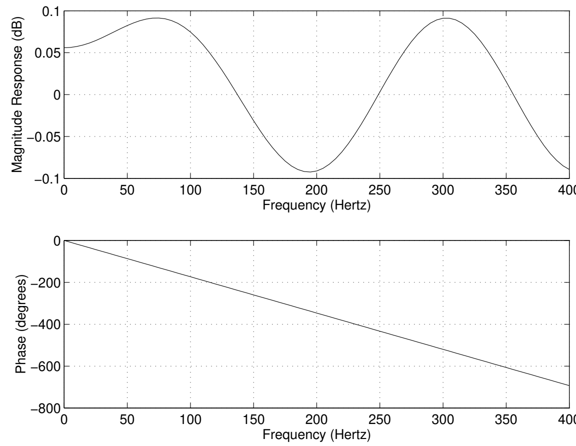
Exercise: What values of F, M, DEV, and Fs, would you use to specify a high-pass filter with a stopband of 0 to 300 Hz, a passband of 400 to 4000 Hz, mean stopband value of -10 dB with 1 dB ripple and mean passband value of 0 dB also with 1 dB ripple?

The following commands design the optimum filter and plot the result over the range 0 to 4 kHz and then over 0 to 400 Hz.

```
>> B=remez(N,Fo,Mo,W);
>> freqz(B,[1],[0:10:8000],Fs);
>> print -deps example1.eps
>> freqz(B,[1],[0:5:400],Fs);
>> print -deps example1b.eps
```

The plots show that with the number of coefficients suggested by the `remezord()` (154) the specifications are met and the ripple is close to the maximum allowed:





1000 Hz Band-Reject Filter

The second example is a 1000 Hz band-reject filter with a 40 Hz bandwidth (980 to 1020 Hz) and sampling rate of 8 kHz. The desired passband ripple is less than 1 dB and the desired stopband attenuation is more than 60 dB. The following commands compute the values required by `remez()`:

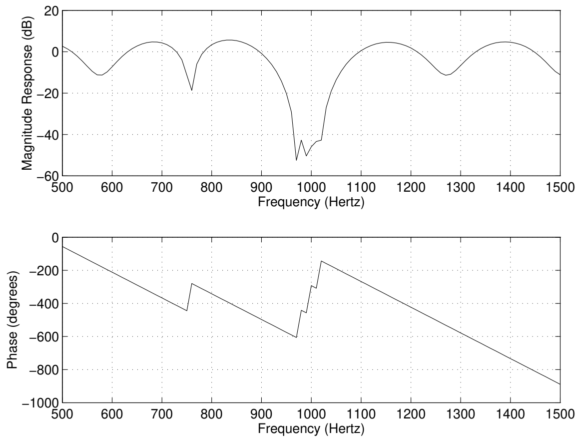
```
>> F=[ 750 980 1020 1250 ];
>> M=[1 0 1];
>> DEV=[0.1 0.001 0.1];
>> Fs=8000;
>> [N Fo Mo W]=remezord(F,M,DEV,Fs)
```

This results in an estimated required order of $N = 69$, and $Fo = \{0, 0.1875, 0.2450, 0.2550, 0.3125, 1.0000\}$, $Mo = \{1, 1, 0, 0, 1, 1\}$, and $W = \{1, 100, 1\}$.

The filter is then designed using `remez()` and the response plotted over the range 500 to 1500 Hz:

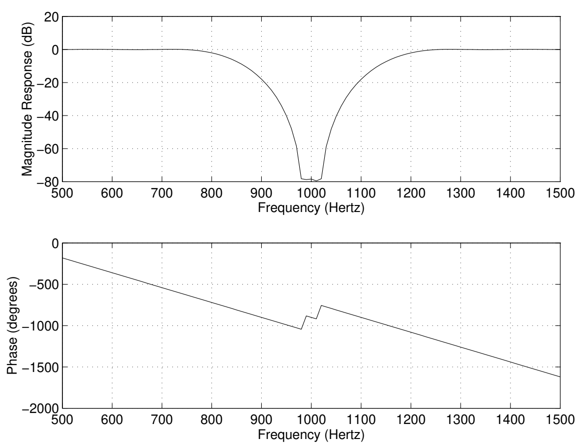
```
>> B=remez(N,Fo,Mo,W);
```

```
>> freqz(B,[1],[500:10:1500],Fs);
>> print -deps example2.eps
```



The plot shows the the initial design with the estimated value of $N = 69$ failed to converge to an equiripple solution. When the filter order is increased to 80 the program finds an equiripple solution that meets the requirements:

```
>> B=remez(80,Fo,Mo,W);
>> freqz(B,[1],[500:10:1500],Fs);
>> print -deps example2b.eps
```



Other Types of FIR Filters

Hilbert Transformers

Another common use for FIR filters is to compute the Hilbert Transform. The Hilbert Transform changes the phase of a signal by 90 degrees at all frequencies. This is used for creating two signals where all the frequency components are in quadrature.

A Hilbert transformer can be designed using the same approach as above by specifying a flat passband and constraining the filter coefficients to have *odd* symmetry.

Exercise: What does odd symmetry in the coefficients imply about the impulse response and the transfer function?

Other Optimum Design Techniques

The equiripple criteria, where the filter is designed to minimize the maximum deviation from the desired response (the *minimax* criteria), is not the only possible design criteria. Another common criteria is to minimize the average of the square of the deviation from the desired response (the *least squares* criteria). The matlab function `firls()` does this type of design.

It's also possible to use optimization methods other than Parks-McClellan for equiripple filter design. For example, the filter design problem can be cast as a linear programming problem and solved using the simplex algorithm. The METEOR program does this.