Enhancing Speech by Removing Noise

Overview

One area in which discrete-time signal processing has enjoyed widespread use is in the field of speech processing. I*n this project, we consider IIR and FIR filter design in the context of enhancing speech corrupted by additive noise.* While this relies somewhat on material Chapters 6 and 7, some aspects can be started earlier.

This project is an excellent introduction to the issues discussed in *Hardware Implementation Considerations for Removing Noise*.

Historically, DSP courses have put considerable emphasis on design techniques for both IIR and FIR filters, as discussed in Chapter 7 of the course text. A variety of filter design algorithms are now implemented in common software packages such as MATLAB. This makes it unnecessary for most practitioners to learn the details of the design algorithms; however, ceding the design function to a software package makes it more important to understand the properties of different types of optimal filters, the meanings of the design parameters, and some of the trade-offs between filter classes.

In the context of IIR filter design, we suggest that you read Sections 7.0 and 7.1 including Example 7.1. Also, read the examples in Section 7.3.1, but focus on what Butterworth and Chebyshev filters are rather than on the details of the bilinear transformation. For the discussion of FIR filter design, the suggested reading is Sections 7.5, 7.6, and 7.7 through 7.7.1. We will not be going into the details of the issues raised in Section 7.7.1, but we would recommend at least reading that section to get a sense of what some of those issues are.

Playing Sound in MATLAB

Throughout these projects it will be necessary to play audio files which you will have manipulated in MATLAB. The functions `sound` and `soundsc` work well for doing this on most hardware platforms.

For this project, the file `projIB.mat` will need to be first loaded into MATLAB. To access the project zip file, click on the link below where you found this file on the web site. After downloading the zip file, copy its contents into your MATLAB working directory and type `load projIB`.

Matlab's filter commands

In this part of the project, it may be helpful to use MATLAB's order estimation functions (e.g. `buttord`, `cheb1ord`, ...). A caveat in the use of these functions is that MATLAB's definition of 'ripple' differs

between the IIR and FIR filter design functions. Do the *Matlab Ripple* project before proceeding with this project.

Filtering Noisy Speech

This project is concerned with designing a low-pass filter for the removal of high-pass noise from a speech signal. The noisy signal is stored in the variable `noisy` and was sampled at 44100 (`fs`) Hz. It consists of a summation of a speech signal which was band-limited at 4 kHz using a low-pass filter with a very narrow transition band, and a noise signal which was filtered at 4 kHz using a high-pass filter with a very narrow transition band. To filter the noise we must design a discrete-time filter with the following parameters:

1. Passband edge: 2500 Hz.
2. Stopband edge: 4000 Hz.
3. Maximum gain in the passband $G_{pb_{max}}$: 40 dB.
4. Minimum gain in the passband $G_{pb_{min}}$: 37 dB.
5. Maximum gain in the stopband $G_{sb_{min}}$: -55 dB.

Because it may take a few iterations to get each filter right, we suggest you write a `.m` MATLAB script for each filter.

Note that in this part of the project, you may end up with very high order filters since the specifications are rather severe. You can try to implement the IIR filters according to the specs using MATLAB's built in tools (such as the `fdatool` or the command line tools `butter`, `cheby1`, `cheby2`, and `ellip`—type "`help signal`" for more information). If you do so, you will notice that the resulting systems might not be stable. There are several reasons for this, the most important of which being coefficient quantization—even with floating point precision. This issue arises because the specifications are very tight and some of the filter types have all of their poles concentrated near $z = 1$.

Fortunately, there is a workaround. What you need to do is to group the poles and zeros for the desired filter in pairs (conjugate or not) to create smaller stable second order filters of the form *N(z)/D(z)* where *N(z)* and *D(z)* are at most second order polynomials in *z*. A cascade of such filters will produce the desired system, and MATLAB will be able to analyze it. The drawback is that you will have to implement your own methods to generate the plots and simulate the system.

Design a DT filter of each of the following types based on the specifications given above:

Butterworth, Chebyshev Type I, Chebyshev Type II, Elliptic, Parks-McClellan, Kaiser.

For each of the designs,

(a) Determine the order of the filter.

(b) Determine the number of multiplication operations per input sample required to implement the filter. Be sure to consider the structure you assume.
(c) Plot the magnitude response (in dB) from $\omega = 0$ to $\omega = \pi$ using `freqz`. Plot a detail of the magnitude response, focusing on the passband ripple (linear scale). Plot the group delay (in samples) using `grpdelay`. (Use `subplot` to fit the three plots on the same page for each filter.)
(d) Plot the pole-zero diagram.
(e) Plot the impulse response using `filter` and `stem` for 100 samples. (Use `subplot` to fit the pole-zero diagram and the impulse response on the same page.)

Filter `noisy` using your de-noising filter. Listen to the filtered and original files. How do they compare?