AC 2012-3836: ENHANCE YOUR DSP COURSE WITH THESE INTERESTING PROJECTS

Dr. Joseph P. Hoffbeck, University of Portland

Joseph P. Hoffbeck is an Associate Professor of electrical engineering at the University of Portland in Portland, Ore. He has a Ph.D. from Purdue University, West Lafayette, Ind. He previously worked with digital cell phone systems at Lucent Technologies (formerly AT&T Bell Labs) in Whippany, N.J. His technical interests include communication systems, digital signal processing, and remote sensing.

Enhance your DSP Course with these Interesting Projects

Abstract

Students are often more interested learning technical material if they can see useful applications for it, and in digital signal processing (DSP) it is possible to develop homework assignments, projects, or lab exercises to show how the techniques can be used in realistic situations. This paper presents six simple, yet interesting projects that are used in the author's undergraduate digital signal processing course with the objective of motivating the students to learn how to use the Fast Fourier Transform (FFT) and how to design digital filters. Four of the projects are based on the FFT, including a simple voice recognition algorithm that determines if an audio recording contains "yes" or "no", a program to decode dual-tone multi-frequency (DTMF) signals, a project to determine which note is played by a musical instrument and if it is sharp or flat, and a project to check the claim that cars honk in the tone of F. Two of the projects involve designing filters to eliminate noise from audio recordings, including designing a lowpass filter to remove a truck backup beeper from a recording of an owl hooting and designing a highpass filter to remove jet engine noise from a recording of birds chirping. The effectiveness of these projects was assessed using anonymous course evaluations, and the results show that most of the students stated that these projects helped them learn. These projects can be implemented in almost any computer language or DSP environment, and the recordings for these projects are available for free from the author.

Background

The FFT and filter design are two fundamental techniques in DSP. Showing the students some examples of how these techniques can be used in practice can help motivate them to learn the mathematical theory. Some DSP courses incorporate laboratory experiments[1,2,3], some use MATLAB/Simulink projects[4,5,6], and some use web-based environments[7]. The projects described below are used in the author's undergraduate DSP lecture course, which has a course in signals and systems as the prerequisite. Some of the projects are assigned as part of a homework assignment and some of them are standalone projects. The goal of the projects is to increase the students' interest the FFT and filter design techniques so that they will be more motivated to learn the theoretical material. Although these projects are used in a non-real-time environment (MATLAB) in the author's course, they do use real audio signals and require the students to write programs to perform realistic tasks, which make the projects more interesting for students than working with simulated signals. These projects could easily be adapted for use in a real-time environment.

One of the student outcomes for the course is that the students will be able to compute and interpret the FFT of a signal. The four projects based on the FFT achieve this goal by having the students use the FFT to extract useful information about various audio signals (speech, DTMF tones, musical instruments, and car horns). In order to complete the projects, the students need to be able to compute the FFT, convert between the FFT bin number and the corresponding frequency in Hz, and interpret graphs of the magnitude of the FFT.

Another of the student outcomes for the course is that the students will be able to design digital filters for a given application. The two filter design projects require that the students analyze audio signals to identify the frequency range of the signal (bird chirps or an owl hoot) and that of the noise (jet engine or backup beeper). The goal of these experiments, as presented in the author's course, is to design a finite impulse response (FIR) filter to attenuate the noise enough that it is no longer audible. The specifications of the filter are not given, which requires the students to design a filter, test it, and if necessary, design a better filter.

Simple Speech Recognition

In order to illustrate the use of the FFT, a simple speech recognition project was developed. The main goal of the project was not to teach speech recognition techniques, but to show that the spectrum of a signal gives useful information about a signal and to give the students practice in calculating which FFT bin corresponds to a given analog frequency. The file set for this project contains 100 recordings (recorded by the author) of different people saying the word yes and 100 recordings of no. The recordings are organized in two groups. The author gives one group, called the training files, to students for use in designing their speech recognition program, and the other group, called test files, is reserved for testing the students' programs. This arrangement prevents students from using features that are specific only to the training files such as keeping a list of values from the yes and no training files and comparing the unknown file to this list.

Although speech recognition in general is a very complex problem, even a simple program can distinguish between the two words yes and no. Although any two words could be used for a project like this, yes and no were chosen because there are real systems that do exactly this task. For example, calls to a company and telephone surveys are often handled by automated systems that ask the person questions and attempt to determine the response using speech recognition. In a situation where the question is answered by yes or no, a simple yes/no speech recognition system is very useful.

In order to distinguish yes from no, it is necessary to find a feature that can be computed on an unknown signal whose value is usually small for yes and large for no (or vice versa). The power spectral density, which is based on the FFT, is a plot of the estimated spectrum of a signal. If the

power spectral density of a recording of a person saying yes is compared to that of no, usually the spectrum of yes has more energy in the high frequencies because of the "s" sound in yes (see Figure 1).

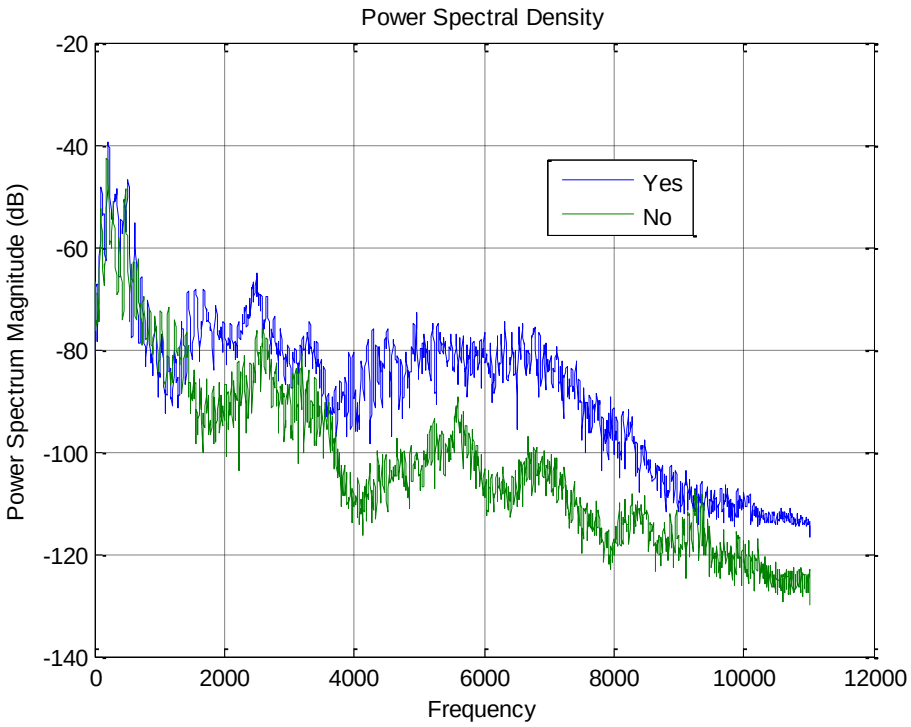

Figure 1: Power Spectral Density for Yes and No

Therefore one possible feature would be to take the sum of the magnitude of the FFT components that correspond to the low frequencies and divide it by the sum of the magnitude of the FFT components that correspond to the high frequencies. So, for example, one possible feature is the sum of magnitude of the FFT values for the frequencies 0 to 5000 Hz divided by sum of magnitude of the FFT values from 5000 to 11025 Hz, which is the highest frequency for the training files because their sampling rate is 22050 Hz. A recording that contains the word no will usually have a higher value than would be found if the recording contained yes. Since this feature is a ratio, its value is not affected by different volume levels. Also, this feature is insensitive to the duration of the recording.

Next it is necessary to find a threshold value that separates the feature value for yes from that of no. One way to find an appropriate threshold is to calculate the feature for all of the training files and to examine the histogram for the yes values and the no values. The students usually compute the feature for all of the training files by modifying the program in the file set (called test_yes_no.m) that loads each of the training files so that it returns the feature value instead of the recognition result. The feature values from the yes files are stored in one vector, the feature values from the no files are stored in a different vector, and then histograms are generated for

each vector. Careful examination of the histograms in Figure 2 reveals that, for this particular choice of feature, a threshold value of 12 separates most of the yes and no values.

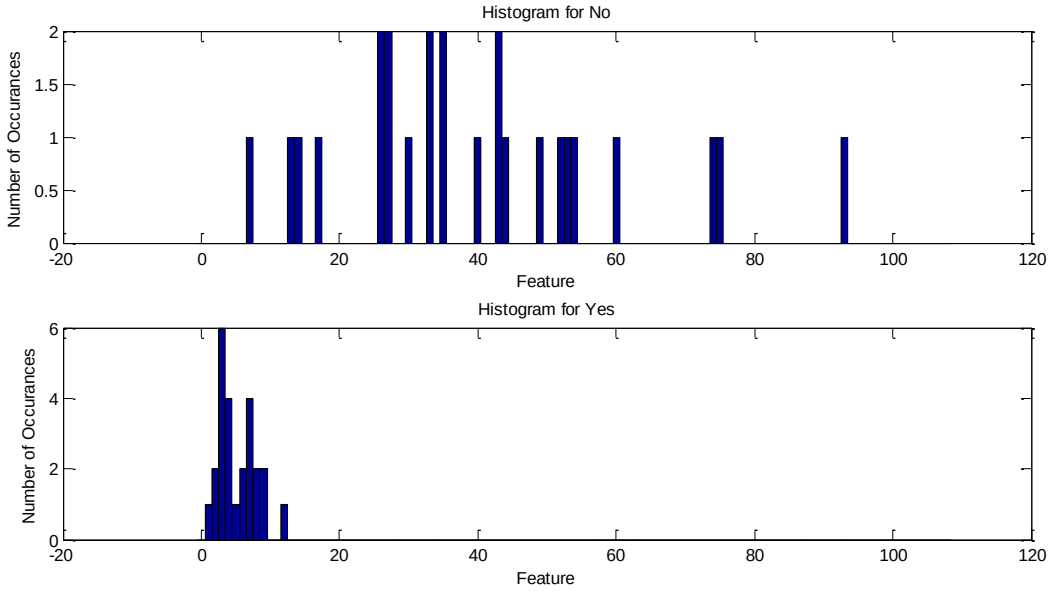

Figure 2: Histogram of Feature Values

Once the feature and threshold values are determined, the contents of a new, unknown recording can be determined by computing the feature value and comparing it to the threshold as shown in the MATLAB program shown below.

```
function output = yes_no(x,fs)
% function output = yes_no(x,fs)
% Simple algorithm for deciding whether the audio signal
% in vector x is the word 'yes' or 'no'.
% x (vector) speech signal
% fs (scalar) sampling frequency in Hz
% output (string) 'yes' or 'no'

threshold = 12;     % threshold value

N = length(x);
k1 = round(N*5000/fs);  % FFT component corresponding to 5000 Hz
k2 = round(N*11025/fs); % FFT component corresponding to 11025 Hz

X = abs(fft(x));
f = sum(X(1:k1))/sum(X(k1:k2)); % calculate feature

if f < threshold,
    output = 'yes'; % if feature is below threshold, return 'yes'
else
    output = 'no';  % if feature is above threshold, return 'no'
end
```

The file set includes a MATLAB program (called test_yes_no.m) that loads each of the training files and runs the recognition program on it. The students can use this program to test how well their algorithm works on the training files. There is also a version of this program (called test_yes_no_for_instructors.m) that the instructor can use to measure how well the students' programs work on both the training and test files.

The simple feature described above achieves an accuracy of 98% in deciding between yes and no in the training and test files, and it illustrates how the spectrum of an audio signal can be useful in determining the contents of the recording.

Recognizing DTMF Tones

DTMF tones are the tones that a telephone makes to indicate to the phone company which button was pressed. This signal makes a good example because students are familiar with it, and it serves a useful purpose. The signal consists of two sinusoids at different frequencies. For example, when button "1" is pressed, the phone generates a 697 Hz tone to indicate the button is in the first row and a 1209 Hz tone to indicate that the button is in the first column (see Table 1).

Table 1: DTMF Tones

	1209 Hz	1336 Hz	1477 Hz
697 Hz	1	2	3
770 Hz	4	5	6
852 Hz	7	8	9
941 Hz	*	0	#

The spectrum of the DTMF tone generated by an actual commercial device is shown in Figure 3.

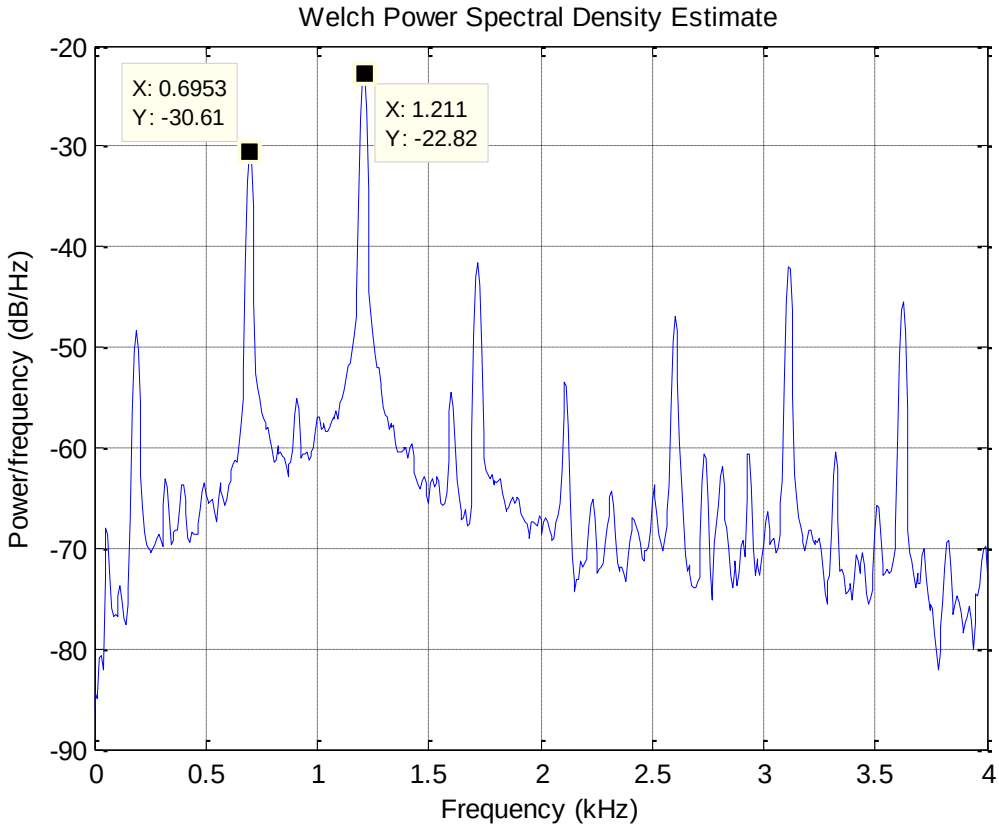

Figure 3: Spectrum of DTMF Tone for Button 1

The FFT can be used to determine the row of the button by determining which of the four FFT bins that correspond to the row frequencies has the biggest magnitude. Likewise, the column of the button can be found by determining which of the three FFT bins that correspond to the column frequencies has the largest magnitude. The number of the button can then be determined by a table lookup as shown in the MATLAB program below, which works on 100% of the training and test files in the file set.

```
function output = dtmf(x,fs)
% function output = dtmf(x,fs)
% Algorithm for deciding which DTMF tone is in the audio signal in vector x.
% x (vector) input audio signal
% fs (scalar) sampling frequency in Hz
% output (scalar) integer value between 1 and 12 that corresponds
%  to the button as follows:
%   output | Button
%   -------|--------
%     1    |    1
%     2    |    2
%     3    |    3
%     4    |    4
%     5    |    5
%     6    |    6
%     7    |    7
%     8    |    8
%     9    |    9
%    10    |    *
%    11    |    0
%    12    |    #

% list of the outputs for each button
Button = [
    1,2,3;      % values for buttons 1,2,3
    4,5,6;      % values for buttons 4,5,6
    7,8,9;      % values for buttons 7,8,9
    10,11,12];  % values for buttons *,0,#

N = 512;    % length of DFT
row_k_values = round([697 770 852 941]*N/fs);   % DTMF row frequencies
col_k_values = round([1209 1336 1477]*N/fs);    % DTMF col frequencies
start = round(length(x)/2-N/2); % start DFT in center of recording
X = fft(x(start:start+N-1));    % take length N DFT
[tmp,r] = max(abs(X(row_k_values)));    % Find row with highest mag
[tmp,c] = max(abs(X(col_k_values)));    % Find col with highest mag
output = Button(r,c);  % Lookup button number in table
```

Included in the file set are recordings of four different commercially available devices generating the 12 different DTMF tones. Again the recordings are organized into a training set which is given to the students and a test set that the instructor can use to test the students' programs for a total of 96 files. Also included are a MATLAB program (test_dtmf.m) that the students can use to test their programs on the training files and a program (test_dtmf_for_instructors.m) that the instructor can use to test the students' programs on both the training files and the test files.

Musical Instrument Tuning

The FFT of the recording of a musical instrument can be used to determine which note is being played and whether the instrument is in tune, sharp, or flat. For example, the spectrum of an oboe playing a single note is shown in Figure 4.

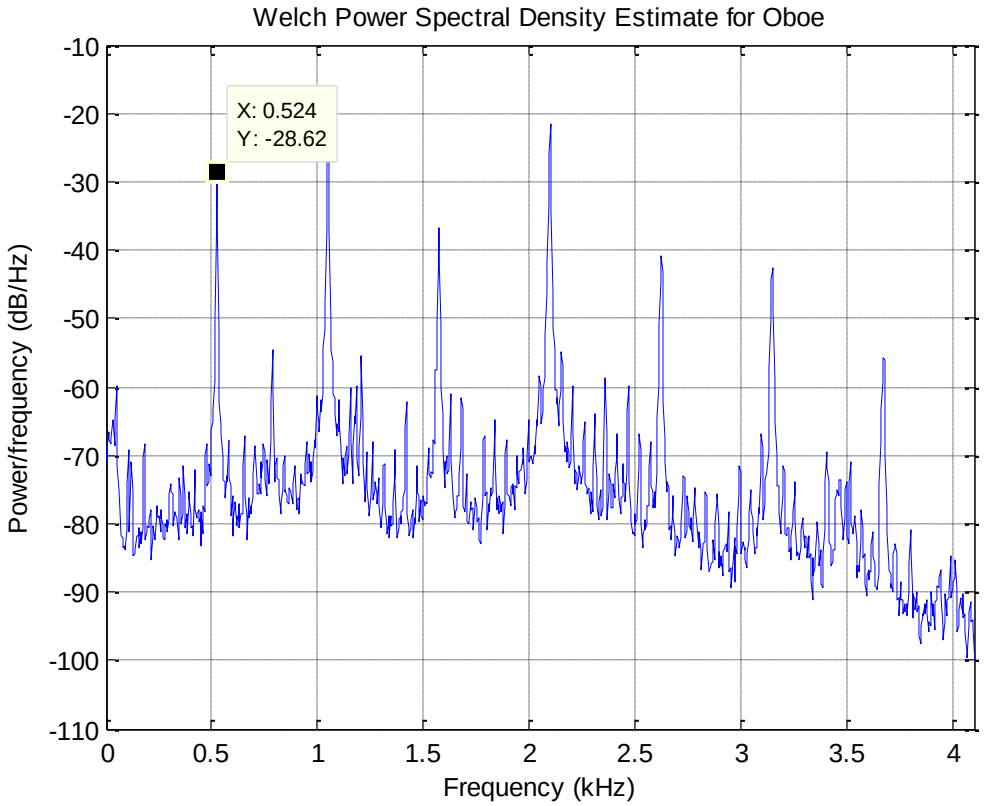

Figure 4: Spectrum of an Oboe Playing a Single Note

The note that an instrument played can be found by determining the fundamental frequency of the signal and comparing it to a table that lists the frequency associated with each note (see Table 2). A more complete table is available in Wikipedia[8]. It can be seen in Figure 4 that the fundamental frequency of the oboe is 524 Hz, which is closest to the note C. Since the oboe's frequency is higher than the frequency listed in the table, the oboe is slightly sharp. The FFT can also be used to compare the harmonic structure of different instruments. Although no recordings of instruments are included in the file set, many recordings can be obtained for free[9].

Table 2: Fundamental Frequencies of Notes

Note	Frequency (Hz)
G	392.0
G#	415.3
A	440.0
A#	466.2
B	493.9
C	523.3
C#	554.4
D	587.3
D#	622.3
E	659.3
F	698.5
F#	740.0
G	784.0

Do Car Horns Honk in the Tone of F?

Several websites claim that car horns in the U.S. honk in the tone of F, and this claim also appeared in a syndicated comic strip. Is it true? Have your students use the FFT to find out. Included in the file set are recordings of two cars (1995 Ford Explorer and a 2010 Toyota Prius). The spectrum of the Explorer's horn is shown in Figure 5. Careful inspection of the spectrum shows that the horn appears to be a mixture of two tones, one with a fundamental frequency of 406.4 Hz and the other with a fundamental frequency of 481.8 Hz. Neither of these tones is close to F (see Table 2). It would be interesting to have the students record their own car horns, analyze the files, and share the results with the class.

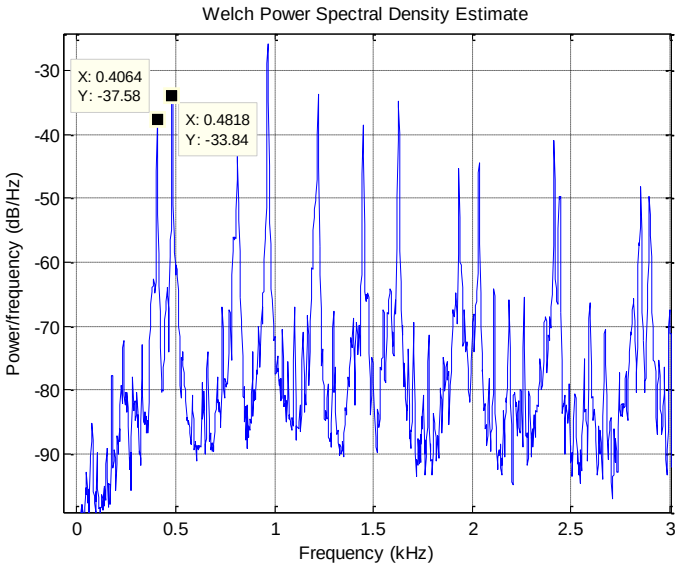

Figure 5: Spectrum of Car Horn (1995 Ford Explorer)

Lowpass Filter Design

Filters are used in many different applications, but one of the applications that students can easily relate to is audio processing. Included in the file set is a recording of a great horned owl[10] mixed with the annoying sound of a backup beeper of a utility van (recorded by the author). The spectrogram, which is based on the FFT, can be used to determine the frequency of the owl hoots and the backup beeper. Figure 6 shows the spectrogram that results using the MATLAB command "spectrogram(x,512,[ ],512,fs)".

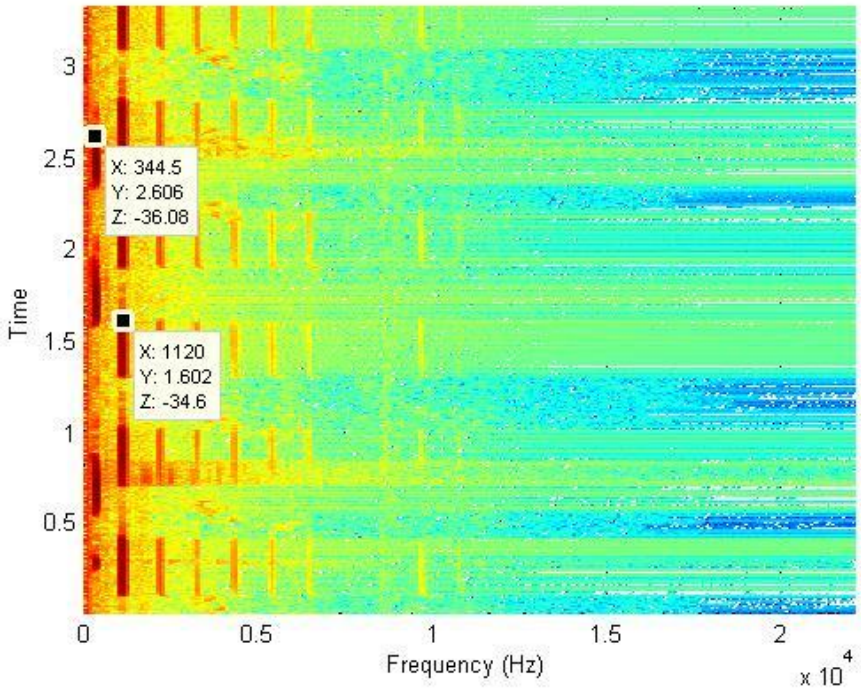

Figure 6: Spectrogram of Owl Hoots and Backup Beeper

In this spectrogram, the owl hoots appear as a series of vertical bands around 300 Hz, and the beeper appears as a series of vertical bands with a fundamental frequency around 1000 Hz and bands for each harmonic. More accurate values for the frequencies can be found by examining the spectrum of the signal as shown in Figure 7, which shows that the owl hoots are at 310.9 Hz and the fundamental frequency of the backup beeper is at 1069 Hz.

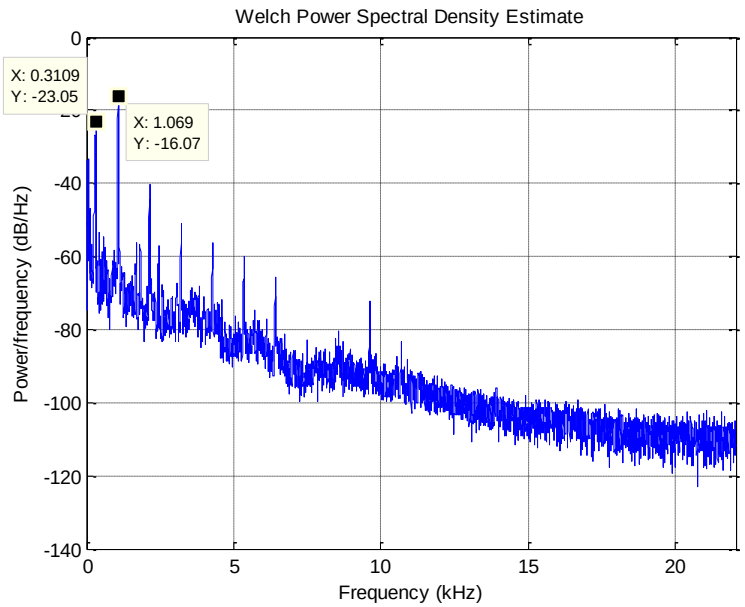

Figure 7: Spectrum of Recording of Owl Hoots and Backup Beeper

Therefore a lowpass filter can be designed to pass the owl hoots and eliminate the backup beeper. Although FIR or IIR filters could be used in this application, the author uses this project while teaching FIR filters. The students are simply asked to design an FIR filter that eliminates the sound of the backup beeper while passing the owl hoots. The MATLAB commands to design and test one possible FIR filter with a cutoff frequency of 730 Hz are shown below.

```
[x,fs] = wavread('owl_beep.wav');   % load audio file
soundsc(x,fs)                       % listen to the input file
b = fir1(230,730/(fs/2));           % design the filter
y = filter(b,1,x);                  % filter the signal
soundsc(y,fs)                       % listen to the output of the filter
```

The filter order of 230 was chosen because that filter's transfer function has a zero near the fundamental frequency of the beeper (1069 Hz). Students are often surprised by how well a filter can remove the unwanted signal.

Highpass Filter Design

Recordings of wildlife sounds often include unwanted sounds from the environment. The file set includes a recording (recorded by the author) of birds chirping while a jet flies overhead. The spectrogram in Figure 8 can be used to determine that the jet noise is a constant low frequency signal below about 1507 Hz, and that the bird chirps appear at frequencies above about 1895 Hz.

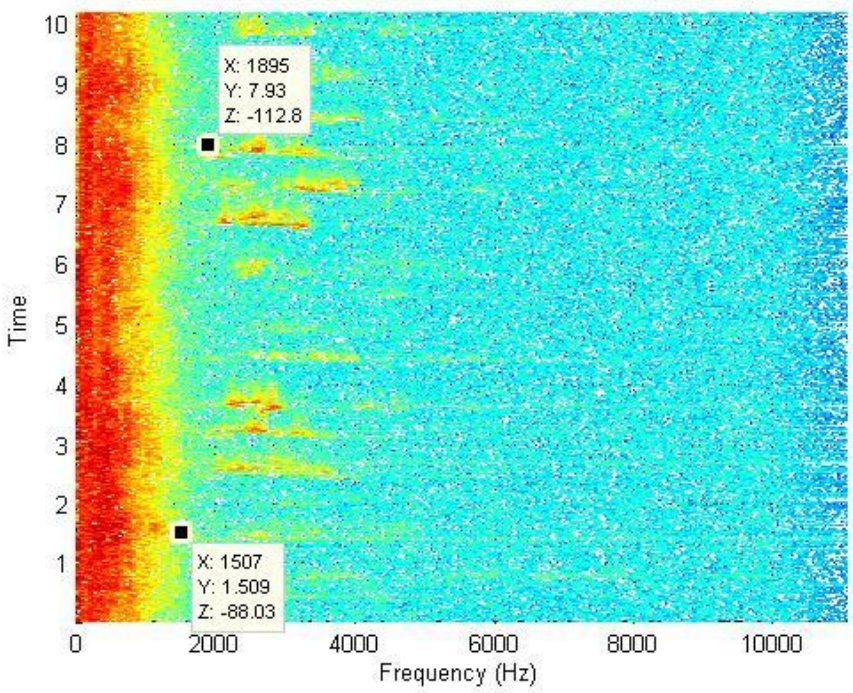

Figure 8: Spectrogram of Birds and Jet Noise

The students are asked to design an FIR filter to eliminate the jet noise and pass the bird chirps. The MATLAB commands to design and test one possible FIR filter with a cutoff frequency of 1700 Hz are shown below.

```
[x,fs] = wavread('birds_jet_noise.wav');    % load audio file
sound(x,fs)                                 % listen to the input file
b = fir1(200,1701/(fs/2),'high');           % design the filter
y = filter(b,1,x);                          % filter the signal
sound(y,fs)                                 % listen to the output of the filter
```

Assessment

The author uses the projects presented in this paper in an undergraduate DSP lecture course, and the impact of the projects was measured using anonymous course evaluations. After the latest offering of the course (Fall 2011), the students were asked to "Describe the strengths of this instructor and course" on the course evaluations. The comments related to the projects are listed below from the ten students who submitted the evaluation.

- "The course itself gives us a great understanding of how signals (most often audio signals) are stored and processed in the real world."

- "the projects were a very great learning experience"
- "The project was relevant and interesting, because it was simple examples of how DSP can be used in the real world."

When asked "What changes, if any, would you suggest for this instructor and course?", the only comment related to the projects was as follows:

- "I think we should do more programming assignments."

There was also a question "The projects helped me learn" which was scored on a 1 to 5 point numerical scale (where 5 is Strongly Agree and 1 is Strongly Disagree). The average score of the ten responses was 4.3 on a 5.0 point scale, and the breakdown of responses is shown below:

- 50% (5 students) Strongly Agree (5 points)
- 40% (4 students) Agree (4 points)
- 10% (1 student) Disagree (2 points)

Although the sample size for these results is small, they suggest that most of the students thought that the projects were interesting and useful for learning DSP.

Conclusions

This paper presented several simple projects that can be used to spark interest in the FFT and filter design. The files needed for these projects are available for free by emailing the author.

References

1. Li, Tan; Jean, Jiang; "Improving digital signal processing course with real-time processing experiences for electrical and computer engineering technology students", 2010 ASEE Annual Conference and Exposition; 2010.
2. Barkana, Buket; "A graduate level course: Audio Processing Laboratory", 2010 ASEE Annual Conference and Exposition; 2010.
3. Adams, J.; Mossayebi, F.; "Hands on experiments to instill a desire to learn and appreciate digital signal processing", 2004 Annual Conference and Exposition, 2004.
4. Ossman, Kathleen; "MATLAB/Simulink lab exercises designed for teaching digital signal processing applications", 2008 ASEE Annual Conference and Exposition; 2008.
5. Ossman, Kathleen; "MATLAB exercises to explain discrete fourier transforms", 2003 ASEE Annual Conference and Exposition; 2003.

6. Pierre, John W.; Kubichek, Robert F.; Hamann, Jerry C.; “Enhancing the comprehension of signal processing principles using audio exercises with MATLAB”, 1999 ASEE Annual Conference and Exposition; 1999.
7. Atti, Venkatraman; Spanias, Andreas; Panayiotou, Constantinos; Song, Yu; “Teaching digital filter design techniques used in high-fidelity audio applications”, 2004 ASEE Annual Conference and Exposition; 2004.
8. Piano Key Frequencies, Wikipedia, http://en.wikipedia.org/wiki/Piano_key_frequencies
9. Musical Instrument Samples, Electronic Music Studios, University of Iowa, http://theremin.music.uiowa.edu/MIS.html
10. www.owling.com