SPE Workshop
October 15–18, 2000

AN INTEGRATED MATLAB SUITE FOR INTRODUCTORY DSP EDUCATION

Richard Radke and Sanjeev Kulkarni

Department of Electrical Engineering
Princeton University
Princeton, NJ 08540
{rjradke,kulkarni}@ee.princeton.edu

ABSTRACT

This paper describes an integrated suite of Matlab tools designed for laboratory exercises in the introductory electrical engineering course at Princeton University. Our goal was to design intuitive and flexible tools that the students could use to experiment freely with signals and algorithms, without getting overly involved in programming. We use similar design elements in the graphical user interfaces which appear in the labs from week to week to guide the students through the basic concepts of signal representation, the frequency domain, sampling and interpolation, and time- and frequency-domain filtering. The students' response to the labs was very positive and we hope to refine the suite of tools for use in the coming years.

1. INTRODUCTION

Today's undergraduates are becoming more computer-literate and accustomed to seeing applications of signal processing in their daily lives (e.g. MP3 music files). At the same time, it has become much easier to design a user-friendly front end to professional numerical analysis software such as the MathWorks' Matlab. The result is that it has recently become possible to introduce high-level digital signal processing concepts at an earlier stage in the electrical engineering curriculum.

In this paper, we describe a series of laboratory exercises designed for a sophomore-level class at Princeton which all electrical engineers are required to take.[1] The class is meant to familiarize students with the basic mathematical and computational tools they will use in solving signal processing problems in the coming years. While the scope of the course is reasonably broad, the relevant concepts are covered in substantial depth. Our goal was to create a series of 3-hour laboratory exercises which would complement the lectures and reinforce the concepts with interactive, hands-on experimentation using real-world signals. The exercises encourage the students to experiment with different signals, algorithms, and parameters in order to obtain an intuition for the concepts involved in processing digital signals.

[1] ELE 201, Introduction to Electrical Signals and Systems.

2. LAB DESIGN PHILOSOPHY

Current introductory electrical engineering texts seem to fall into two general categories: broad but less technically oriented texts (e.g. [1, 2]), and higher-level DSP textbooks that focus on the DFT, filter design, the z-transform, etc. (e.g. [3, 4]. The exercises in neither type of book seemed wholly satisfactory to convey both the breadth and depth of coverage that we wanted. Computer laboratory exercises available for introductory signals and systems courses also seemed to either allow little student exploration (e.g. many of the demonstrations and labs in [5]) or require copious amounts of programming (e.g. [6]). Therefore, we set out to design our own laboratory experiments.

For this introductory course, we felt it was important to develop lab exercises which allowed the students to do meaningful experiments without getting bogged down in the minutae of coding. We felt there was little to be gained by using precious lab time to write conceptually straightforward Matlab code. In fact, some of the implementations of our user-friendly interfaces involve non-trivial Mat-

lab coding which would be well beyond the scope of the class. The idea was to expose the students to the applications and effects of digital signal processing using a series of graphical user interfaces, with the understanding that in future courses they would learn the programming skills and algorithms behind the interfaces.

We also felt it was important to create a feeling of continuity between the labs, so that students did not need to spend time at the beginning of each exercise learning the interface to an entirely new tool. Having similar design elements which cut across several weeks of the lab also makes it easier for students to make connections between concepts learned at different times.

A good signal processing tool allows the user to pose and answer reasonably deep questions without sensing the layer of implementation between the interface and the Matlab computational engine. In fact, as we developed the tools, we found ourselves changing the tasks in the lab in response to interesting phenomena the tools allowed us to observe.

The laboratories were implemented in Matlab 5.0, using some elements from the Signal Processing Toolbox 4.0 [7]. The graphical user interfaces were initially designed and laid out using Matlab's `guide` (Graphical User Interface Developer) tool and the "internal wiring" was coded afterwards. The tools were designed in the spirit of the Signal Processing Toolbox's `sptool` suite, which includes tools for viewing signals, designing filters, and creating spectra using an integrated set of graphical user interfaces. Our tools operate as independent windows which coexist with the Matlab command prompt. All of the important data to generate the plots in each figure window is hidden inside a data structure built into the figure, and does not clutter the Matlab workspace.

3. THE `splay` FAMILY OF TOOLS

The first lab in the sequence introduces a signal viewer called `splay` (Figure 1) that can used to play audio signals at different sampling rates.

The exercises for the first lab involve the creation and manipulation of signals in Matlab, and interpretation of some signals as sounds. The students create sums of sinusoids that are discovered to be the dial tone, busy signal, and ringing sounds heard on U.S. telephone receivers.

The second lab focuses on the frequency domain. The student is presented with the familiar signal player interface, with an additional panel which shows the frequency domain representation of the selected signal (Figure 2).

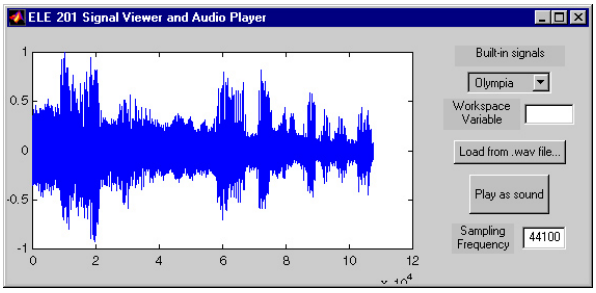

Figure 1: The `splay` signal viewer.

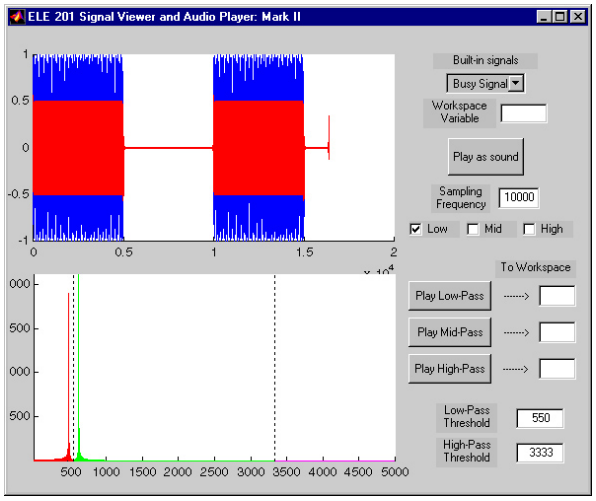

Figure 2: The `splay` signal viewer with frequency domain.

By moving vertical dividing bars in the frequency domain, the students can partition the signal into low, middle, and high frequency components, and play these as audio signals. In this way, the phone signals from the first lab can be separated into their pure tone components. The interface also features buttons for importing variables from and exporting signals to the Matlab workspace, where more complicated manipulation can be performed. The final task in the lab involves using the signal viewer to isolate a hidden signal from the middle band of a carrier signal, demodulate it to the proper frequency, and report the content of the message.

The graphical interface allows the students to

do signal processing with a minimal amount of effort. The Matlab code required to duplicate the results of the final task at the command line or in an m-file would have taken many of the students a sizable amount of time to write. We felt this time was better spent allowing the student to experiment graphically with the signals, in order to gain intuition about the frequency domain.

The third lab deals with the aliasing that is introduced by different methods of interpolating subsampled signals. Again, the students work with the familiar time and frequency domain signal viewer from the previous lab, which has been altered to let the user specify the subsampling rate and the method of interpolation (Figure 3) .

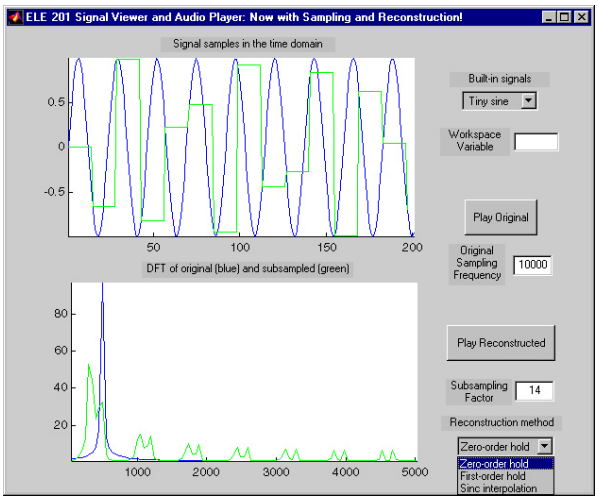

Figure 3: The `splay` signal viewer for sampling and interpolation.

By quickly toggling back and forth between the different reconstruction methods, the student discovers that sinc interpolation is the best signal reconstruction method and experimentally verifies that perfect reconstruction is possible when the signal is sampled above the Nyquist rate. The reconstruction can be seen to degrade more or less gracefully, depending on the interpolation method. Again, the actual Matlab implementations of the reconstructions are transparent to the student. The interface reinforces the concept of aliasing by allowing the student to both observe and hear its effects on a variety of real and artificial signals.

The final lab which uses the `splay` interface concerns filtering of digital signals. The interface allows FIR filters to be developed in the time or frequency domain. The time-domain design panel, illustrated in the upper-right-hand corner of Figure 4, uses a simple stem plot whose taps can be dragged up and down. The filters can be normalized to have zero or unity sum depending on the application.

The frequency-domain panel, illustrated in the upper-right-hand corner of Figure 5, is modeled after a graphic equalizer on a stereo system, in which the frequency response in different bands is controlled by dragging bars up and down. The student can create a filter with multiple bandpass regions quickly, and apply the filter with a single button click.

Students explore how to apply filters in the time and frequency domains for the purposes of denoising signals and enhancing them (for example, applying a "bass boost").

4. CONCLUSIONS AND FUTURE WORK

In their first semester of use, the introductory lab exercises using the interfaces described here were very well-received. In course evaluation forms, the students gave the laboratories an average rating of 4.6 out of 5. We are currently in our second semester of implementation.

We are considering making a single tool with a menu that changes which panels are displayed, so that there is really only one 1-D signal viewer for the entire sequence of labs. Different panels would be activated depending on the task.

Aside from the audio applications discussed here, the introductory electrical engineering course for which these interfaces were designed also includes laboratories on two-dimensional image processing (sampling, quantization, halftoning, JPEG, MPEG, etc.) as well as signal processing algorithms (convolution, Fourier series, variable-length compression). Next, we hope to unify the two-dimensional signal processing lab exercises using the same type of consistent interface described in this paper.

Eventually, we hope to distribute our lab exercises and Matlab tools as a set of small lab modules which can be combined together for a short or long laboratory session.

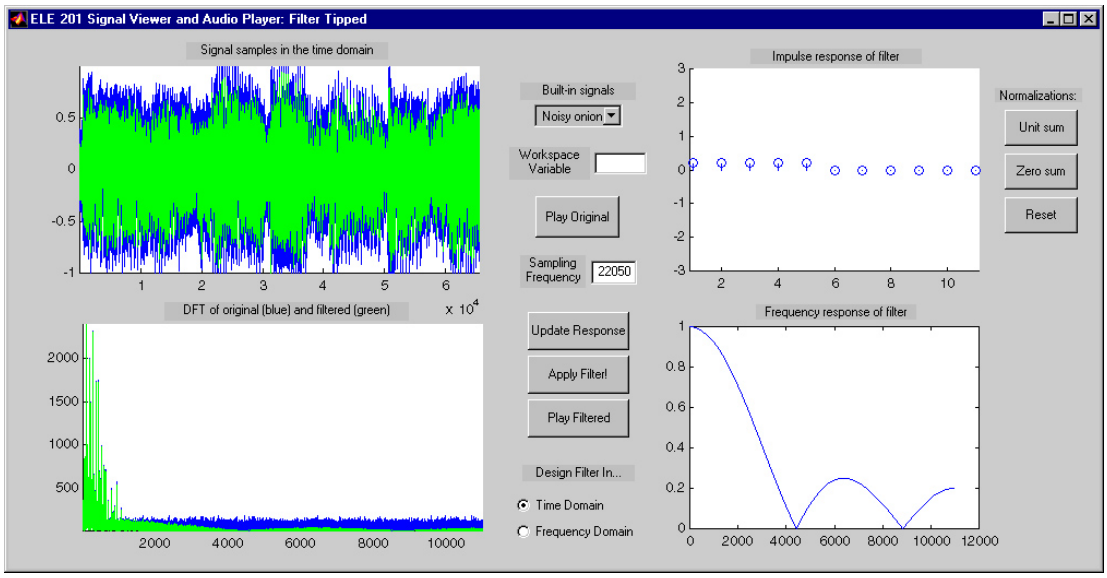

Figure 4: The `splay` signal viewer with the time-domain FIR filter design panel.

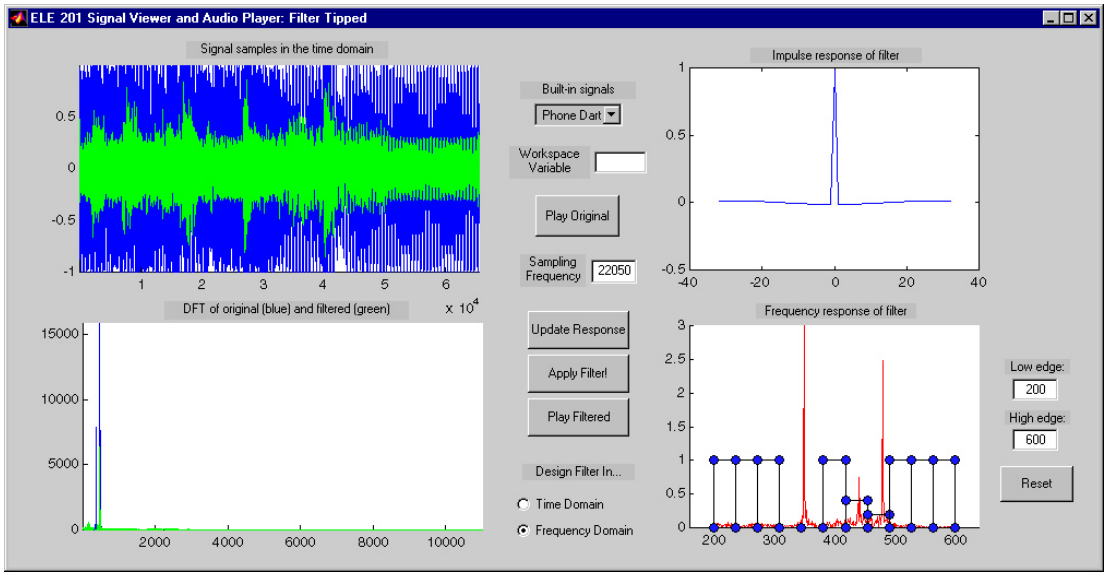

Figure 5: The `splay` signal viewer with the frequency-domain filter design panel.

REFERENCES

[1] R. Kuc. *The Digital Information Age: An Introduction to Electrical Engineering*, PWS Publishing, 1999.

[2] D. White and R. Doering. *Electrical Engineering Uncovered*, Prentice Hall, 1997.

[3] S. Haykin and B. Van Veen. *Signals and Systems*, John Wiley and Sons, 1999.

[4] A.V. Oppenheim, A.S. Willsky, et al. *Signals and Systems*, Prentice Hall, 1996.

[5] J.H. McClellan, R.W. Schafer, and M.A. Yoder. *DSP First: A Multimedia Approach*, Prentice Hall, 1998.

[6] C.S. Burrus, J.H. McClellan, A.V. Oppenheim, T.W. Parks, R.W. Schafer, and H.W. Schuessler. *Computer-Based Exercises for Signal Processing Using Matlab*, Prentice Hall, Englewood Cliffs, NJ, 1994.

[7] The MathWorks, Inc. Matlab 5.0 and the Signal Processing Toolbox 4.0. Natick, MA. `http://www.mathworks.com`.