INTERACTIVE DSP LABORATORIES ON MOBILE PHONES AND TABLETS

Jinru Liu, Shuang Hu, Jayaraman J. Thiagarajan, Xue Zhang, Suhas Ranganath, Mahesh K. Banavar and Andreas Spanias

SenSIP Center, School of ECEE, Arizona State University, Tempe, AZ, USA.

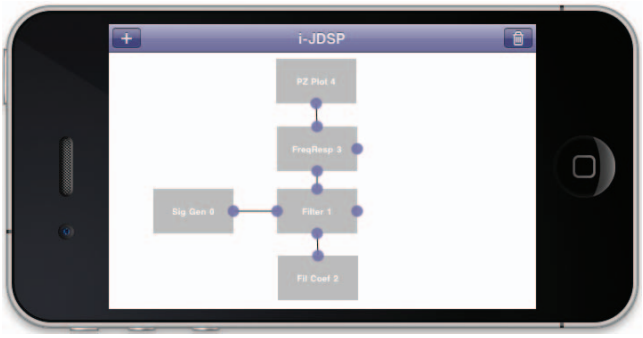
ABSTRACT

The use of mobile devices and tablets in engineering education has been gaining lot of interest, due to its interactive capabilities and its ability to stimulate student interest. On the other hand, this technology can also enable instructors to broaden the scope of their curriculum and increase student participation. In this paper, we describe an interactive application to perform signal processing simulations on iOS devices such as the iPhone and the iPad. Furthermore, we describe two laboratory exercises to introduce continuous/discrete convolution and filter design. The exercises and the proposed application will be evaluated by students of an undergraduate DSP course at Arizona State University during Fall 2011. Finally, we describe the planned assessment methodology which will enable us to provide prescriptive recommendations for using i-JDSP in DSP courses.

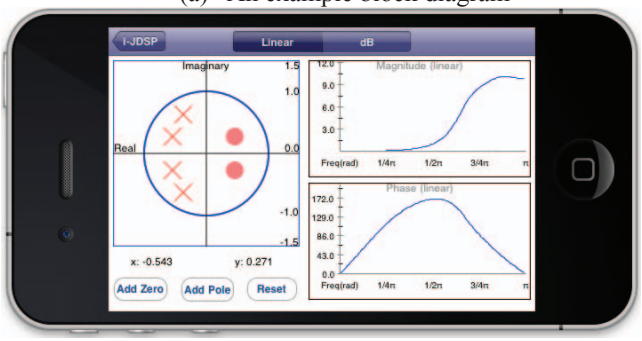
Index Terms— iOS devices, mobile applications, DSP education, laboratory exercises.

1. INTRODUCTION

The pervasive nature and interaction capability of mobile devices present a great potential for instructors to stimulate student interest. The benefit of using mobile devices in classroom has been the subject of considerable research. The advantages of handheld devices over personal computers in K-12 education have been investigated in [1]. The study has found that out that the easy accessibility and maneuverability of handheld devices led to an increase in student interest. By incorporating mobile technologies into mathematics and applied mathematics courses, it has been shown that smartphones can broaden the scope and effectiveness of technical education in classrooms [15]. The crucial role of interactivity in knowledge acquisition, and the development of cognitive skills are discussed in detail in [14]. Furthermore, in [2] it is emphasized that education tools typically require a higher degree of interactivity in order to enhance learning potential. Several software applications that incorporate visual components in learning to simplify the understanding of complex theoretical concepts have been developed [3-6]. These applications are characterized, in general, by rich user interaction and ease of accessibility. In order to enable students to perform DSP labs over the Internet, the authors in [7] developed J-DSP, a visual programming environment. J-DSP was designed as a zero footprint, standalone Java applet that can run directly on a browser [8]. Several interactive laboratories have been developed and successfully used in undergraduate courses [9], [10]. Though J-DSP is a light-weight application, running J-DSP over the web on mobile devices can be data-intensive. Hence, executing simulations directly on the mobile devices is a suitable alternative. The current mobile devices possess abundant memory and powerful processors, in addition to providing highly interactive interfaces and hence can support the implementation of signal processing algorithms. These factors have motivated the design of an interactive application to perform digital signal processing



(a) An example block diagram



(b) Pole-zero plot and magnitude/phase responses of the filter.

Fig. 1. Illustration of frequency domain analysis of a filter using i-JDSP on an iOS device.

This work was sponsored by the NSF TUES Phase 3 grant 0817596.

(DSP) laboratories using iOS devices. In this paper we present i-JDSP, a graphical application to simulate DSP concepts. The proposed simulation framework is compatible with all iOS devices (iPhone, iPad). Fig. 1 illustrates the frequency domain analysis of a filter using i-JDSP.

The other recent efforts to facilitate simulation of signal processing algorithms on iOS devices include the MATLAB Mobile interface [11]. This provides a lightweight mobile desktop that can connect via the Internet to the MATLAB software running on a remote computer. However, this application requires a running instance of MATLAB on a remote machine and does not support sophisticated User Interfaces (UI). More importantly, this implies that the application cannot be used without Internet connectivity

In this paper, we will describe the architecture of the i-JDSP application and briefly discuss the set of DSP functions developed. Furthermore, we will present two laboratory exercises in i-JDSP that can be used to introduce continuous/discrete convolution and basic filter design in undergraduate DSP courses. Finally, we will discuss the set of planned assessments in order to understand the impact of using mobile devices in DSP education. The set of designed exercises will be evaluated by students of an undergraduate DSP course at Arizona State University and we will present the results of the assessment at the conference

2. THE I-JDSP ARCHITECTURE

The i-JDSP environment has been developed using Apple's Xcode IDE with the iOS SDK. It has been designed as a native Cocoa Touch application [12] that is compatible with all iOS devices. For performance considerations [13], we have implemented the software using both C and Objective-C. The simple and intuitive UI of i-JDSP is easy to understand with minimal assistance. All functions in i-JDSP are organized as graphical blocks that can be visually added to the main simulation view of the application. The main simulation view of the i-JDSP environment (Fig. 1(a)) has been designed to provide maximum drawing space with minimal navigation buttons. Since the user needs to navigate through the application to select different DSP function blocks and configure their parameters in the corresponding user-dialogs, i-JDSP framework contains multiple views. During any part of the application execution, users can easily navigate through the view hierarchy. For example, in Fig. 1(a), the user can push the "+" button on the top left corner to open the view (Fig. 2.) that lists the set of available functions. After selecting a function from the list, the user is prompted with an edit view, in which the parameters of the function can be modified to the desired values. Finally, pressing the "**Add**" button confirms the changes and places the selected function in the main simulation view.

Fig. 2. i-JDSP interface listing the set of available functions.

By establishing connections between different blocks, a variety of DSP systems can be simulated. The parameters corresponding to each block can be edited by double-tapping on a block and navigating to the edit view. Changing the parameters of a block will automatically re-execute all blocks that are dependent on the settings of the current block. Individual blocks can be deleted along with their connections to other blocks, by a long-hold press.

The current version of i-JDSP supports the following functions that can be used to simulate a variety of DSP systems: *Signal Generator, Plot, Fast Fourier Transform, Filter Design, MIDI, and DTMF*. In addition to the functions described, i-JDSP supports blocks to perform basic arithmetic operations, the *Junction* block create multiple copies of an input signals and stand-alone blocks for demonstration purposes. The exercises described in Section 3 will be based on this set of functions.

3. LABORATORY EXERCISES

In this section, we describe a set of exercises that introduce continuous/discrete convolution and basic filter design using i-JDSP. The laboratory exercises are typically performed after the students are introduced to the basic concepts of convolution and filtering in their lecture sessions.

3.1. Continuous/Discrete Convolution

This exercise introduces the concepts of continuous and discrete time convolution and enables students to visualize the same using demonstrations in i-JDSP. To begin with students are asked to analytically compute impulse responses obtained by convolving two signals. Following this, the *Conv. Demo* block in i-JDSP is used to demonstrate the effect of causality in continuous convolution. For a given set of input signals, the *Conv. Demo* block animates the convolution procedure and plots the resulting impulse response on the same plot screen. Fig. 3(a) illustrates an example case of convolving two rectangular (Rect.) signals. The students are asked to make observations on the convolution results in the following cases:

(a) Continuous convolution (b) Discrete convolution

Fig. 4. Illustrations of continuous and discrete convolution using the Conv. Demo block in i-JDSP. Note that *Conv. Demo* is a stand-alone demonstration block that animates the convolution procedure and plots the resulting impulse response.

- Signal 1: Causal (Rect.); Signal 2: Causal (Rect.).
- Signal 1: Causal (Rect.); Signal 2: Non-causal (Rect.).
- Signal 1: Non-causal (Rect.); Signal 2: Non-causal (Sinc).

The *Conv. Demo* block in i-JDSP can also perform discrete time convolution. In addition to providing a set of pre-defined discrete signals, the i-JDSP function allows users to create their own signal by holding and dragging each sample to the desired amplitude. Fig. 3(b) demonstrates a sample case of discrete convolution. In the final part of the exercise, the students compare the results obtained by sampling two continuous signals and performing discrete convolution, with the result obtained by sampling the convolution result obtained by convolving the two continuous signals directly.

3.2. Filter Design

This exercise was designed to demonstrate the different methods used for designing discrete time filters. Using the filter design functions in i-JDSP, the students were asked to examine the following methods:

- *Manual entry of filter coefficients*: This was designed to provide students an intuitive notion of how the filter response varies with change of coefficients.
- *Windowing*: The characteristics of non-parametric windows such as Bartlett and parametric windows such as Kaiser were studied.
- *Parks-McClellan method.*

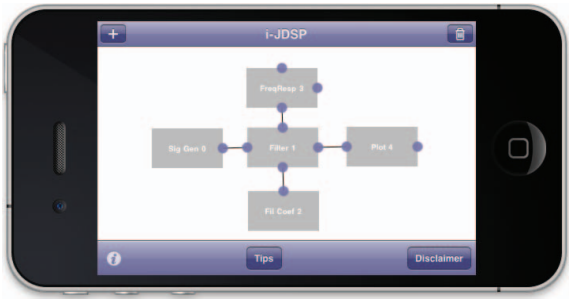
The students perform simulations with each of the filter design methods for different choices of filter parameters and input signals, and comment on the results obtained. The impulse responses, filtered output plots, magnitude/phase responses and pole-zero plots are generated and analyzed. Fig. 4 illustrates the filter design simulation that requires users to manually provide filter coefficients.

4. ASSESSMENTS

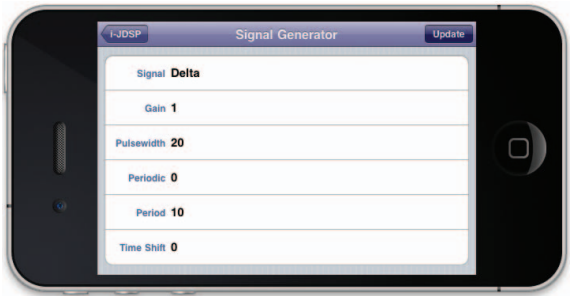
The set of laboratory exercises described in Section 3 will be performed by students of an undergraduate signal processing class at Arizona State University during Fall 2011. Furthermore, we will generate assessment results based on the student evaluation of the application. The goal of this evaluation is to identify the impact of employing mobile devices to perform DSP simulations. The assessments will determine if the i-JDSP framework was interactive and interesting for students to perform different simulations. Furthermore, general assessments about the aesthetics and the usability can be used to obtain an overall subjective opinion about the application. The pedagogy adopted for the use of i-JDSP will include the following: (a) lecture on the pertinent signal processing concepts, (b) a pre-quiz on the concepts involved in the laboratory exercise, (c) a simulation exercise using i-JDSP, (d) post-quiz to test student understanding of the concepts. We list a part of our assessment questionnaire below.

- Did the contents of the i-JDSP exercises improve your understanding of the concepts of convolution and filter design?
- What do you think about the speed of the convolution animation in i-JDSP?
- Did the exercises help you understand the effects of causality?
- If a student colleague is having difficulty understanding convolution, would you recommend the i-JDSP convolution demonstration?
- Is it practical to perform DSP exercises and simulations on an iPhone/iPad?
- By using the iPhone/iPad, does that give you a more compelling reason to finish your lab exercises than using a computer based tool?
- How easy is it to navigate through the application to create a simulation?
- How long did it take for you to learn i-JDSP?

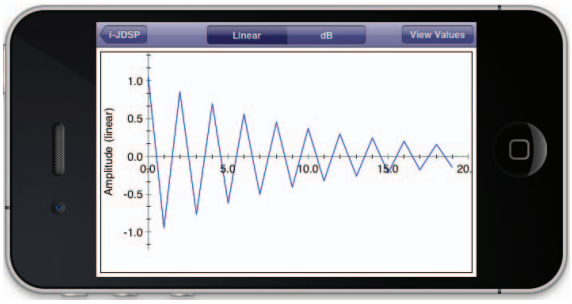
Addressing the issues identified based on the results of the student evaluation will enable us to provide prescriptive recommendations concerning strengths and the sustainability of this mobile education paradigm, and the ability to reuse this set of exercises in other courses.



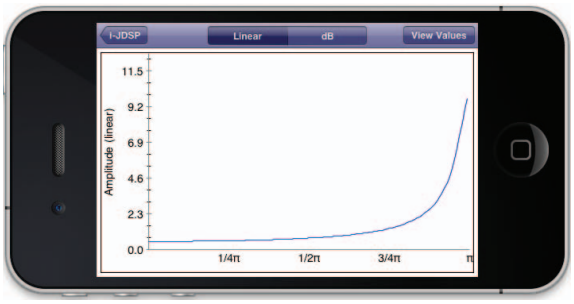
(a) Block diagram.



(b) Editing the parameters of the *Signal Generator*.



(c) Impulse response obtained as the output of the *Filter* block.



(d) Magnitude response of the filter.

Fig. 5. Illustration of designing filters by manually entering the filter coefficients in i-JDSP. The *Filter Coef.* block is used to provide the numerator and denominator coefficients.

5. CONCLUSIONS

In this paper, we described the i-JDSP application, which can be used to perform DSP simulations on mobile phones and tablets. The proposed application has been developed as a native cocoa application and is compatible with all iOS devices. The current set of functions in the application will enable students to perform simulation exercises on convolution, Fourier analysis and filter design. The interface is highly interactive and the block diagrams can be constructed using a simple drag-n-drop procedure. The set of exercises described in this paper will be performed by students of an undergraduate DSP course and the assessment results of the student evaluation will be presented at the conference.

6. REFERENCES

[1] E. Soloway, C. Norris, P. Blumenfeld, B. Fishman, J.Krajcik, & B. Marx, "Handheld devices are ready-at hand", *Communications of the ACM*, Vol. 44, No. 6, pp.15-20, 2001.

[2] C. Chou, "Interactivity and interactive functions in web-based learning systems: A technical framework for designers." *British Journal of Educational Technology*, vol. 34, pp. 265–279, 2003

[3] D. Millard and G. Burnham, "Innovative interactive media for electrical engineering education," in *IEEE FIE Conference*, vol. 3, 2001, pp. S3C –17–22.

[4] "The Infinity Project: Engineering education for today's classroom," Available online at http://www.infinity-project.org/infinity/.

[5] D. J. Brown, M. Covington, and M. L. Swafford, "Mallard: An educational tool for digital signal processing," ser. Proceedings of *Asilomar conference on SSC*, vol. 1, 1996, pp. 231–235.

[6] M. J. Jackson, D. I. Laurenson, and B. Mulgrew, "Developing and evaluating java-based educational tools," in *IEE International Symposium on Engineering Education: Innovations in Teaching, Learning and Assessment*, vol. 2, 2001, pp. 26/1 –26/6.

[7] A. Spanias and V. Atti, "Interactive online undergraduate laboratories using J-DSP," *IEEE Transactions on Education*, vol. 48, no. 4, pp. 735–749, Nov 2005.

[8] A. Clausen et.al., "A Java signal analysis tool for signal processing experiments," in *Proceedings of IEEE ICASSP*, vol. 3, may 1998, pp.1849–1852.

[9] A. Spanias et.al., "Development of a web-based signal and speech processing laboratory for distance learning," *ASEE Computers in Education*, vol. X, no. 2, pp. 21–26, Jun 2000.

[10] V. Atti and A. Spanias, "On-line simulation modules for teaching speech and audio compression techniques," in *IEEE FIE Conference*, vol. 1, Nov 2003, pp. T4E–17–22.

[11] MATLAB Mobile, Available online at: http://www.mathworks.com/mobile/.

[12] Cocoa Frameworks, Available online at: http://developer.apple.com/technologies/mac/cocoa.html.

[13] M. Akten, "NSArray vs. C Array performance," Available online at: http://memo.tv/nsarray_vs_c_array_performance_comparison.

[14] R. Sims, "Interactivity: A forgotten art?" *Computers in Human Behavior*, vol. 13, no. 2, pp. 157–180, 1997

[15] Yerushalmy, M & Oshrat Ben-Zaken, (2004), "Mobile phones in education: A case of mathematics," *Tech report,* Institute for Alternatives in Education, University of Haifa.