AC 2011-456: IPHONE/IPAD BASED INTERACTIVE LABORATORY FOR SIGNAL PROCESSING IN MOBILE DEVICES

Jinru Liu, School of ECEE, SenSIP Center, Arizona State University
Jayaraman J Thiagarajan, School of ECEE, SenSIP Center, Arizona State University
Prof. Andreas S Spanias, Arizona State University, ECEE, SenSIP Center

Andreas Spanias is Professor in the School of Electrical, Computer, and Energy Engineering at Arizona State University (ASU). He is also the founder and director of the SenSIP center and industry consortium (NSF I/UCRC). His research interests are in the areas of adaptive signal processing, speech processing, and audio sensing. He and his student team developed the computer simulation software Java-DSP (J-DSP - ISBN 0-9724984-0-0). He is author of two text books: Audio Processing and Coding by Wiley and DSP; An Interactive Approach. He served as Associate Editor of the IEEE Transactions on Signal Processing and as General Co-chair of IEEE ICASSP-99. He also served as the IEEE Signal Processing Vice-President for Conferences. Andreas Spanias is co-recipient of the 2002 IEEE Donald G. Fink paper prize award and was elected Fellow of the IEEE in 2003. He served as Distinguished lecturer for the IEEE Signal processing society in 2004.

Karthikeyan Natesan Ramamurthy, Arizona State University
Shuang Hu, ASU
Mahesh K. Banavar, SenSIP Center, School of ECEE, Arizona State University

iPhone/iPad Based Interactive Laboratory for Signal Processing in Mobile Devices

1. Introduction

The demand for using advanced mobile devices in education, business and research has resulted in several powerful processors with an array of capabilities and large multi-touch screens[1]. Advanced mobile devices are capable of handling tasks that are relatively complex such as word processing, complex Internet transactions, and even human motion analysis[2]. Furthermore, this compelling technology has become a part of everyday student life. Hence, the design of exciting mobile applications and software represents a great opportunity to build student interest and enthusiasm in science and engineering.

Apple's iOS devices, including the iPhone, the iPod touch and the latest in the family – the iPad, are among the most popular today[3]. In particular, the iOS platform is emerging into an important tool for engineering and STEM online education and web-based simulations[4]. There are several educational applications related to science and engineering some of which are available commercially[5].

The field of Digital Signal Processing (DSP) has occupied researchers and students because of its enabling nature, modern applications and the need for sophisticated digital implementations[6]. DSP algorithms have been used in the processing of a wide range of signals such as speech, audio, image, video, radar, sonar, biomedical and seismic data. In undergraduate engineering education signal processing courses involve complex mathematical concepts and in some cases students have difficulty making the connections to real-world problems and applications. Although several software components and web courses have been designed for education, they are expensive and often lack interactive capabilities. To address these problems, a web-based, platform-independent simulation environment, Java-DSP (J-DSP), has been developed at Arizona State University. This visual programming software enables users to perform online signal processing calculations and simulations[7]. J-DSP was built from the ground up in Java to provide free and universal access

to an array of functions that can be used for research and education[8]. Several interactive laboratories have been developed and successfully used in undergraduate courses[9, 10]. In addition, several toolboxes have been developed for image processing[11], control systems[12], genomics[13], time-frequency analysis[14], analog/digital communications[15], earth system signal processing[16] and RF amplifier linearization[17]. Furthermore, hardware interfaces to TI DSK[18] and Crossbow sensor motes[19] have also been developed. The interfaces to MATLAB[20] and LabVIEW[21] allow synergies with other simulation environments.

The recent efforts to facilitate simulation of signal processing algorithms in iOS devices include the MATLAB Mobile interface[22]. This provides a lightweight mobile desktop that can connect via the Internet to the MATLAB software running on a remote computer. It features a traditional command line interface and allows users to type commands, execute scripts and view the corresponding results. 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. Furthermore, rendering of the MATLAB figures relies heavily on the internet connection, since figures are not generated on the mobile device but simply transferred from the MATLAB instance running on the remote machine.

In this paper, we present the design and implementation of an interactive signal processing simulation environment that offers an intuitive, all-graphical programming experience on the iOS platform. The proposed application executes functions on the iOS device directly and hence does not require Internet connectivity. Furthermore, the extensive support for user interactivity provides scope for improved learning. This iOS-based object-oriented application is called i-JDSP and is based on the (J-DSP) software. The current version of i-JDSP offers functions for FFT, filtering and spectral analysis through an easy-to-use graphical user interface (GUI) and provides a very compelling multi-touch programming experience. Built-in modules also demonstrate concepts such as MIDI, DTMF and sound capture and playback. All simulations can be visually established by forming interactive

Figure 1. The main simulation view of i-JDSP.

block diagrams through multi-touch and drag-and-drop procedures. Computations are performed on the mobile device when required, making the block diagram execution fast. In the rest of this paper, we describe the architecture of the proposed i-JDSP environment and present some of the functions available in the current version of the software.

2. The i-JDSP Application

i-JDSP has been developed using Apple's Xcode IDE with the iOS SDK. It has been designed as a native Cocoa Touch application[23] that can be run on any iOS device. For performance considerations[24], we have implemented the software using C and Objective-C. The simple and intuitive GUI of i-JDSP is easy to understand with minimal assistance. All functions in i-JDSP are represented 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 (Figure 1) has been designed to provide maximum drawing space with minimal navigation buttons. A detailed workflow that demonstrates the creation of a block diagram in i-JDSP is shown in Figure 2. Since the user needs to navigate through the application to select different DSP function blocks and configure their corresponding user-dialogs, i-DSP framework contains multiple views. During any part of the application execution, users can navigate through the view hierarchy. For example, in Figure 1, the user can push the "+" button on the left corner to open the view 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, when the user is

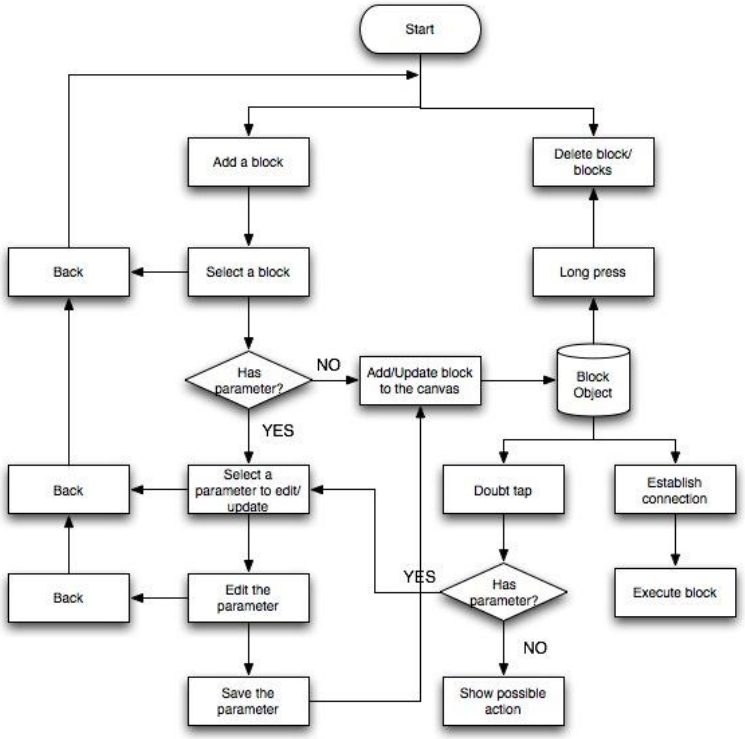

Figure 2. Workflow for creating a block diagram in i-JDSP.

done editing, pressing the "Add" button places the selected function in the main simulation view.

3. DSP Functions in i-JDSP

i-JDSP consists of a set of basic DSP functions that can be used to perform lab exercises that are relevant to a DSP class. All functions are represented as graphical blocks with appropriate connection terminals, as shown in Figure 3.

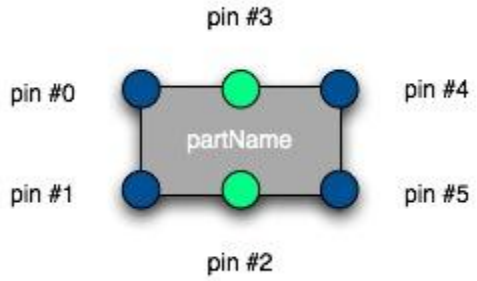

Figure 3. Pin layout of a function in i-JDSP.

A function can allow upto two input terminals and two output terminals, which can each

support a time or frequency domain signal. In addition, pin #2 and pin #3 can be used to input and output filter coefficients respectively. The DSP functions implemented in the current version of i-JDSP are listed below:

- Signal generator
- Fast Fourier Transform (FFT)
- Filter
- Filter coefficient
- Frequency response
- MIDI
- Pole Zero placement
- Plot
- Sound recorder and sound playback

As described earlier, adding a function to the simulation view can be performed by a simple drag-and-drop procedure. By establishing connections between different blocks, a variety of DSP systems can be simulated. The parameters corresponding to each function can be edited by double-tapping on a block and navigating to the edit view. Changing the parameter settings of the functions will automatically update the data in all later blocks that use these data and parameters. Individual blocks can be deleted along with their connections to other blocks, by a long-hold press. i-JDSP can support an unlimited number of functions and multiple instances of the same function in a block diagram. However, the size of the block diagram is limited by the screen space of the mobile device.

3.1. Signal Generator

The signal generator block supports a suite of basic deterministic signals such as rectangular, triangular, delta, sinusoid, and sinc with configurable parameters and can generate random signals from Gaussian and uniform distributions. The maximum length of the time domain signals generated by this function is 256. The other parameters of the signals that can be modified include the period and time-shift. This function can be extended to include real-life signals such as speech, music and other time-domain signals. Furthermore,

users can be provided with options to record real-time sounds using the microphone.

3.2. Fast Fourier Transform (FFT)

The FFT block computes the discrete Fourier transform (DFT) of the input signal using the Cooley-Tukey decimation-in-time FFT algorithm. The FFT block takes in a signal vector and computes both magnitude and phase of the FFT. Possible FFT vector sizes that can be provided by the user should be radix-2 (8 to 256). Figure 4 demonstrates the computation of FFT in i-JDSP.

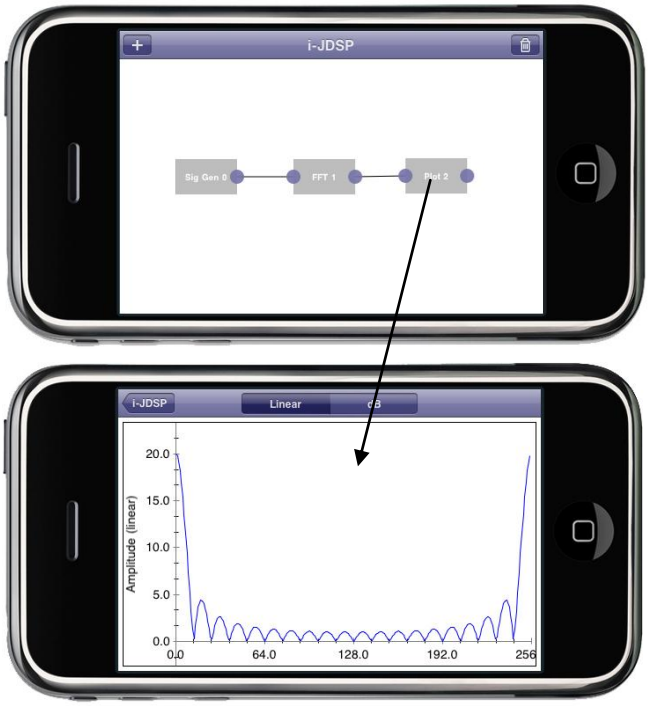

Figure 4. Computing FFT of an input signal in iJDSP.

3.3. Filtering

i-JDSP supports both Infinite Impulse Response (IIR) and Finite Impulse Response (FIR) filter design by the combination of the filter block and the filter coefficient block. The filter block simulates a digital filter that performs mathematical operations (multiply / accumulate and convolution) on the incoming signal to reduce or enhance certain frequency-

domain components in that signal. More specifically, it filters the input signal based on the filter parameters (numerator and denominator polynomials) specified in the filter coefficient block. The filter coefficient block simply accepts the coefficients and thereby determines the characteristics of the filter. In the current setup, a maximum of 11 coefficients can be used. Users can change and edit the coefficients at any time by double tapping the block. When updated, the block outputs the filter coefficients through pin #3 to the filter.

The frequency response block measures the frequency spectrum of the system output in response to a unit impulse. The frequency response is displayed in both linear and decibel (dB) scales, and the phase, measured in radians. Figure 5 illustrates the process of filtering and plotting the frequency response in i-JDSP. Students can use this simulation to visualize impulse and frequency responses of digital filters.

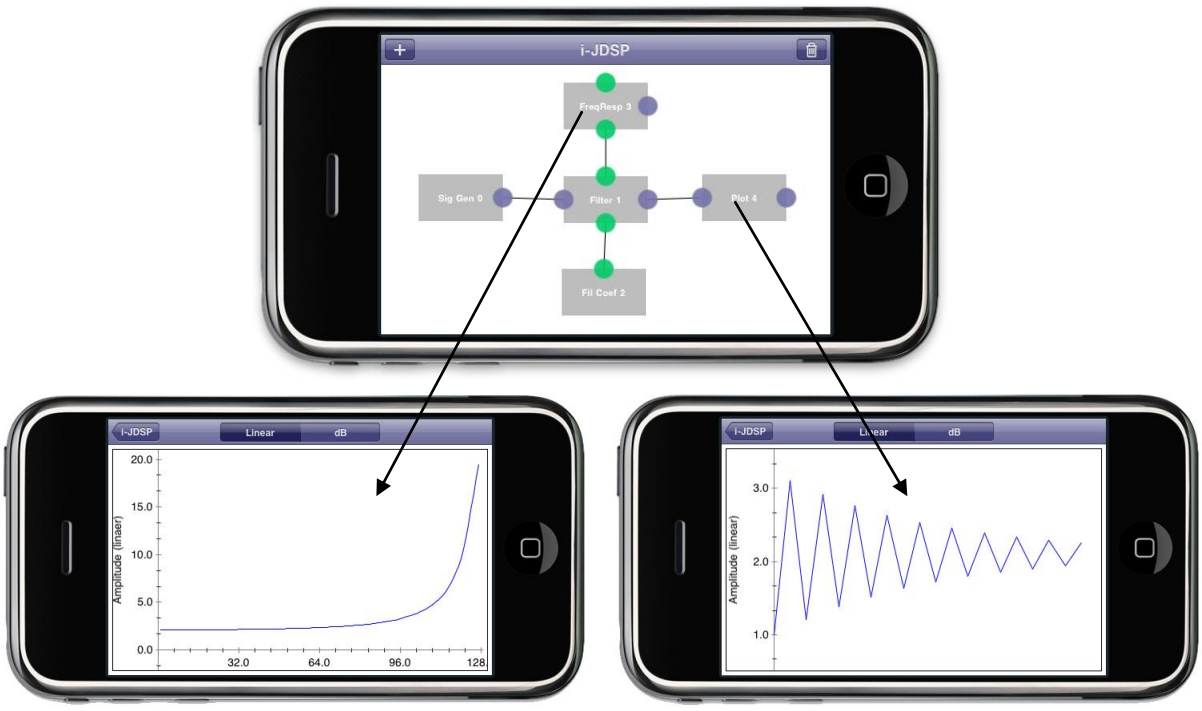

Figure 5. Filtering of a rectangular signal in i-JDSP with filter coefficients filter coefficients: a = [1, 0.9], b = [1, 3].

Figure 6. MIDI synthesis in i-JDSP.

3.4. MIDI Synthesis

The MIDI block provides a simple piano keyboard interface and generates Musical Instrument Digital Interface (MIDI) sounds at frequencies described by the MIDI standard. It can generate MIDI tones of length 256 samples with sampling frequency at 8KHz. This block can be used along with the FFT block in i-JDSP to analyze the spectrum of the MIDI tones. Figure 6 illustrates the interface of the MIDI block. Students can use this to relate musical and MIDI tones to frequencies in Hz.

3.5. Pole Zero Placement

This block allows the user to place poles and zeros graphically in the Z-plane and analyze the corresponding frequency response. As it can be observed from Figure 7, moving the poles and zeros dynamically updates the magnitude and phase responses of the corresponding filter. It is important to note that the poles and zeros are added to the Z-plane as conjugate pairs. In the current version of the software, a maximum of 10 poles and zeros are supported. Students can use this to relate poles and zeros on the z plane to the frequency response.

3.6. Plot

The plot block, as the name suggests, simply generates a 2D plot of the data. When

the user double taps on the block, it stores the data received from its parent block. Finally, the

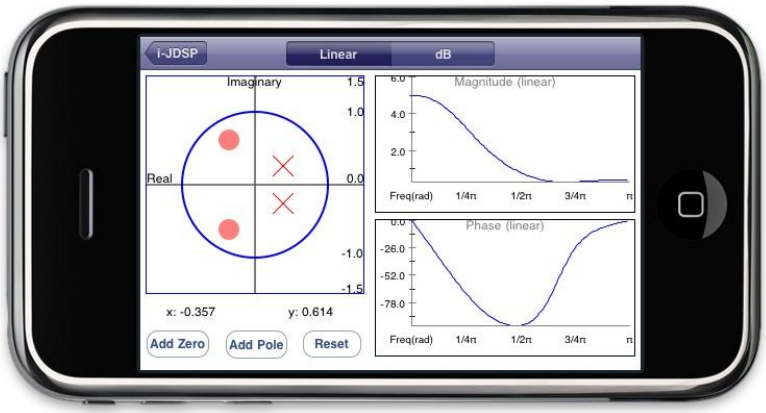

Figure 7. Filter design using pole zero placement in i-JDSP.

plot block utilizes CorePlot[25], an open source 2D visualization framework for iOS, in order to plot and manipulate the figure.

4. Assessments

The i-JDSP software is currently in the final phase of development and alpha testing. The release of the application is planned in early Fall of 2011. The students of the undergraduate DSP course at Arizona State University (EEE 407) will use i-JDSP to perform laboratory exercises and evaluate the software during Fall 2011. Furthermore, we are currently involved in the design of suitable tools to build assessments of the i-JDSP software, based on student feedback. Since the learning to use i-JDSP is self-directed, similar to the online J-DSP software, we present some of the prior assessment results reported in[7].
One of the fundamental problems in DSP classes is the overall perception and attitudes that the students have toward the class material. A summary of the impact of the J-DSP software in the DSP course (EEE 407) is presented below.

- The assessment results described in[7] revealed that the J-DSP laboratories have indeed enhanced and reinforced student learning, particularly in topics such as filter design, FFT-based spectral estimation, and multirate signal processing. In particular, the general and concept-specific assessment results supported the objectives associated with the J-DSP labware and the pedagogy.
- The J-DSP simulations involving seamless animations and interactive GUI have

shown prominent differences in the prelab/postlab assessment.

- Moreover, the Effective Size measures in[7] showed that the effect of J-DSP involvement in student learning is mostly "medium" or "large".
- The J-DSP visualizations involving PZ placement were shown to have prominent impact in student understanding. This has indeed motivated integration of several other seamless animations and interactive demonstrations.

To summarize, it was observed that the J-DSP exercises and software changed the attitudes of the students toward the class, resulting in increased enrollments. Furthermore, the use of the software provided the students hands-on experiences with basic DSP concepts.

5. Conclusions

In this paper, we motivated and presented a signal processing simulation environment for Apple iOS devices including the iPhone, iPad, and iPod. The proposed framework has been designed based on the architecture of the online J-DSP software. This all-graphical programming environment has been developed using C and Objective-C, and runs as a native Cocoa application in any iOS device. We also described a set of functions that can be used to perform fundamental laboratory exercises in an undergraduate DSP class. Finally, a summary of assessment results of the online J-DSP software, reported in[7], was provided. Similar assessment results are being compiled for i-JDSP in the DSP course (EEE 407) at Arizona State University. The release of i-JDSP is planned in the Fall 2011. Plans to develop the application on Android are under way.

5. Acknowledgements

This work has been funded in part by NSF CCLI (TUES) phase 3 award number 0817596.

Bibliography

[1] Benko H., Wilson A.D, and Baudisch P., "Precise selection techniques for multi-touch screens", *Proceedings of CHI '06,* pp. 1263–1272, 2006.

[2] Siewiorek D., Smailagic A., and Furukawa J., et al. "Sensay: A context-aware mobile phone", *ISWC '03 IEEE*, 2003.

[3] Chang E., "How popular is the iPhone", Available online at: http://www.billshrink.com/blog/10071/how-popular-is-iphone/. Last accessed 17 Jan. 2011.

[4] The Elements, Available online at: http://periodictable.com/ipad/.

[5] App Store, Available online at: http://www.apple.com/iphone/features/app-store.html.

[6] A. Spanias, Digital Signal Processing: An Interactive Approach. ISBN: 978-1-4243-2524-5, *Lulu Publishers*, 2007.

[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 Educati*ons, 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 *Proceedings of IEEE Frontiers in Education*, vol. 1, nov 2003, pp. T4E–17–22.

[11] M. Yasin et.al., "On-line laboratories for image and two-dimensional signal processing using 2D J-DSP," in *Proceedings of IEEE ICASSP*, vol. 3, apr 2003, pp. 785–788.

[12] T. Thrasyvoulou et.al., "J-DSP-C, a control systems simulation environement: labs and assessment," in *Proceedings of IEEE Frontiers in Education*, vol. 1, nov 2003, pp. T4E–11–T4E–16.

[13] A. Spanias et.al., "Teaching genomics and bioinformatics to undergraduates using J-DSP," in *Proceedings of ASEE Annual Conference and Exposition*, Jun 2004.

[14] M. Zaman, "Advanced concepts in time-frequency signal processing made simple," in *Proceedings of IEEE Frontiers in Education*, vol. 1, pp. T2E10–15, Denver, Nov 2003.

[15] Y. Ko et.al., "On-line laboratory for communication systems using JDSP," in *Proceedings of IEEE Frontiers in Education*, vol. 1, pp. T3E–13–T3E–18, Denver, Nov 2003.

[16] K. Ramamurthy et.al., "On the use of J-DSP in earth systems," in *Proceedings of ASEE Annual Conference and Exposition*, Jun. 2006.

[17] R. Santucci et.al., "Advanced functions of Java-DSP for use in electrical and computer engineering courses," in *Proceedings of ASEE Annual Conference and Exposition*, jun 2010.

[18] A. Spanias et.al., "Using the java-dsp real-time hardware interface in undergraduate classes," in *Proceedings of IEEE Frontiers in Education*, Oct. 2006, pp. 12–17.

[19] H. Kwon et.al., "Experiments with sensor motes and Java-DSP," *IEEE Transactions on Education*, vol. 52, no. 2, pp. 257–262, may 2009.

[20] A. Spanias et.al., “Java-DSP interface with MATLAB and its use in engineering education,” in *Proceedings of ASEE Annual Conference and Exposition*, Jun 2004.

[21] A. Spanias et.al., “Using JDSP and LabVIEW to perform undergraduate labs,” in *Proceedings of ASEE Annual Conference and Exposition*, Jun 2007.

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

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

[24] Akten, M., “NSArray vs. C Array performance”, Available online at: http://memo.tv/nsarray_vs_c_array_performance_comparison. Last accessed 17 Jan. 2011.

[25] core-plot, “Cocoa plotting framework for Mac OS X and iOS”, Available online at: http://code.google.com/p/core-plot/. Last accessed 17 Jan. 2011.