Interactive Signal Processing Education Applications for the Android Platform

Suhas Ranganath, Jayaraman J. Thiagarajan, Deepta Rajan, [1]Mahesh K. Banavar, Andreas Spanias, Jie Fan, Kristen Jaskie and Cihan Tepedelenlioglu
SenSIP Center, School of ECEE, Arizona State University, [1]Dept. of ECE, Clarkson University

Abstract—**In this paper, we present a unique Android-DSP (AJDSP) application which was built from the ground up to provide mobile laboratory and computational experiences for educational use. AJDSP provides a mobile intuitive environment for developing and running signal processing simulations in a user-friendly manner. It is based on a block diagram system approach to support signal generation, analysis, and processing. AJDSP is designed for use by undergraduate and graduate students and DSP instructors. Its extensive functions enable instructors and students to simulate DSP concepts including convolution, Fourier transforms, z-transform and filter design. We describe the AJDSP functions and the process of using them in computer exercises, classroom demonstrations, and undergraduate remote laboratories. We include quantitative and qualitative assessments of AJDSP at the end of this paper. We also include descriptions of outreach efforts where we used AJDSP in middle schools and high schools to demonstrate how DSP algorithms enable several applications.**

Index Terms—**AJDSP, Signal Processing, Android.**

I. Introduction

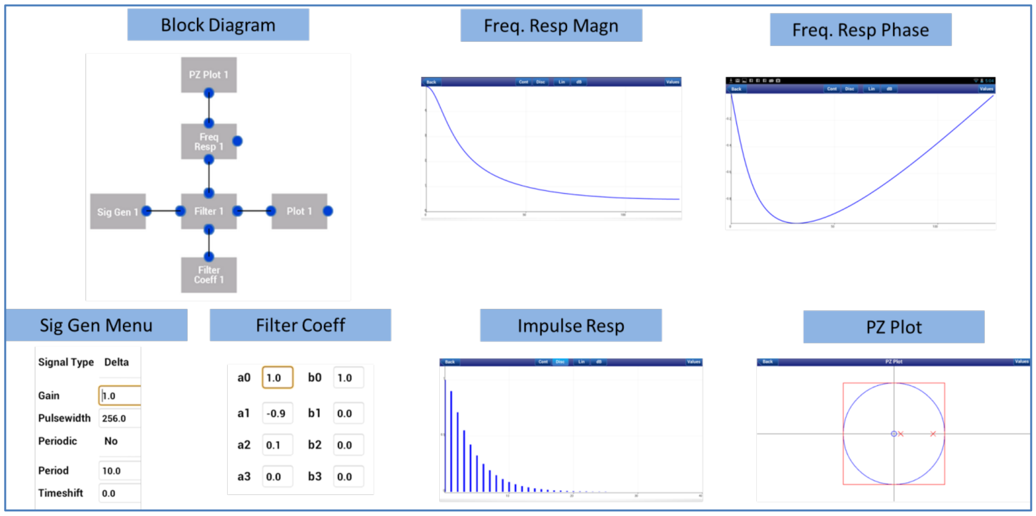
Modern educational software tools have been shown to assist student understanding of complex systems through simulation and visual representation of algorithms [1, 2]. Several software tools have been developed to take advantage of the processing and graphical capabilities of modern computers [3, 4].

It has been shown that interactive animations, real-time demonstrations, interactive dashboards, and audio-visual features enhance student understanding [5]. Incorporating visual representations of algorithms in classroom teaching can foster a better understanding of complex technical concepts and cultivate student interest [6]. Furthermore, increasing the level of interactivity in classrooms is shown to improve learning potential [7]. These factors motivate the usage of computer-based interactive tools in engineering education. During the last decade, we have witnessed the exponential growth of mobile computing in the form of smartphones and tablets. The fast processing speed and intuitive user interface makes these devices a suitable platform to develop tools for engineering education.

In this paper, we present a novel graphical programming application (app) for undergraduate and graduate signal processing, signals, and systems classes. Considerable attention has been given to this subject in recent literature [8-11]. Our proposed standalone Android app, AJDSP (Android-DSP) provides a mobile DSP lab environment for students both on and off campus and was inspired by the Java-DSP (J-DSP) desktop platform [2, 12, 13]. J-DSP uses signal processing algorithms to assist in problems such as speech and image processing [14], health monitoring [15], echolocation [16, 17], online learning [18, 19], power amplifier linearization [20] and machine learning [21]. Although some of the signal analysis functions were ported from Java to Android, several functions, and specifically the AJDSP GUI (graphical user interface), had to be designed and developed from the ground up. Challenges in the GUI design include designing and building the application architecture specifically for compact Android platforms and making the application compatible with diverse Android devices (in size and capability). Adapting the block functions and the graphical orientation of the objects to the Android platform was also a challenge as operations had to be secured on both small cell phone devices and larger tablets. Initial versions with limited functions of our Android app were presented in [22]. In this paper, we present the latest comprehensive version that conforms with the most recent Google Play technical requirements [23]. The latest version has been used in our signal processing, and signals and systems classes as well as in training sessions of our NSF REU program [24-26].

A. Graphical programming tools in STEM Education

Graphical programming is intuitive to learn and use. Previous assessments of J-DSP have shown that students and practitioners learn to set up block diagrams and run DSP simulations quickly and easily. Interactivity provided by menu/button driven simulations proved to be a valuable tool for building educational applications. It provides an extra layer of abstraction between general programming syntax and the end user, allowing program flow visualization in a manner consistent with block diagram representation of algorithms. This is performed by connecting graphical modules in a workspace, which is then interpreted into machine-readable code. Diagrammatic modeling of processes and tasks provides a more intuitive programming experience to the end user, allowing the user to focus on underlying concepts rather than the minutia of algorithm



Signal Type	Delta
Gain	1.0
Pulsewidth	256.0
Periodic	No
Period	10.0
Timeshift	0.0

a0	1.0	b0	1.0
a1	-0.9	b1	0.0
a2	0.1	b2	0.0
a3	0.0	b3	0.0

Fig. 1. Filter design simulation in AJDSP shows the impulse and frequency response of a filter. The “Block diagram” section shows the blocks in the given simulation: Signal Generator, Filter, Plot, Filter Coefficient, Frequency Response, and PZ Plot. The “Sig Gen Menu” section shows the user interface of the Signal Generator block. The “Filter Coeff” section shows the user interface of the Filter Coefficient block. The parameters are the coefficients of the transfer function of the filter. The “Freq Resp Magn” displays the magnitude of the frequency response of the input filter through the Frequency Response block. The Freq Resp Phase” section displays the phase of the frequency response of the input filter through the Frequency Response block. The “Impulse Resp” section shows the impulse response of the input filter through the Plot block. The “PZ Plot” section illustrates the position of the poles and zeros of the transfer function of the filter through the PZ Plot block.

implementation.

Several popular graphical programming languages have been developed in the last few decades. LabVIEW [3], developed by National Instruments, provides toolboxes for programming and is used at many universities. Scratch [1], a web-based programming language developed by the Lifelong Kindergarten Group at the MIT Media Lab, enables users to create interactive stories, games, and art. J-DSP [2], developed at the SenSIP Center of Arizona State University, allows online digital signal processing analysis.

B. *Interactive Software Tools in DSP Education*

DSP simulations help students understand the concepts, mathematics, and implementation details associated with signal processing applications. For this reason, several educational platforms for DSP have been developed in the last decade. Morrow, et al. integrates a basic waveform generator, audio receiver, and graphic equalizer into an application to teach real-time DSP [27]. MentorDSP [28] is an interactive learning resource for signal processing techniques and statistical analysis. MATLAB^{TM} [29] by Mathworks, Inc. is a commercial simulation environment that enables numerical computations and visualization of signals in a scripting environment. Other mobile signal processing environments were developed in [16, 17, 26, 30, 31]. J-DSP was developed as a Java applet allowing users to run programs directly on an internet browser [2] and is available online in [13]. Several recent developments of the desktop J-DSP were reported in [18, 21].

C. *Emergence of Mobile Devices*

The emergence of mobile smartphones and tablets with substantial computational abilities, along with advanced cloud capabilities is a game changer in education. Since the first iPhone was introduced in 2007, the popularity of smartphone devices and tablet computers has increased exponentially. Extensive research has been focused into understanding the benefit that smartphones and tablets bring to classroom instruction. Handheld devices have been demonstrated to outperform personal computers in K-12 education [32] as their easy accessibility boosts student interest. The application of mobile technologies in higher education in [33] indicates that smartphones assist students in grasping complex principles and lead to an increase in class participation. Use of smartphones in engineering and mathematics courses has illustrated how mobile devices can broaden the scope and effectiveness of technical education in classrooms [30, 34-36]. Some mobile applications related to signal processing education are available on the Android platform including a basic circuit toolkit in Electrical Engineering [37] by GK Soft, an audio analysis application form AndroidSP and Speedy Spectrum Analyzer [38], an FFT-based spectral analysis visualizer.

A well-known tool for signal processing simulations is MATLAB Mobile [39]. It provides a lightweight version of MATLAB that runs on a remote computer. However, this application relies on a command line interface and is dependent on a stable internet connection. To overcome these limitations, iJDSP, a standalone iOS version of the J-DSP software was proposed in [22]. As mentioned before initial versions of AJDSP was proposed in [40, 41] as an application that exploits the resources and features of the Android platform to create a compelling anywhere/anytime signal processing learning environment. To the best of our knowledge, this is the first platform specific graphical programming environment for engineering signal processing education applications for Android devices and was developed further recently by adding several new DSP functions, improving precision, and establishing the app operations to conform with the recent technical requirements of Google Play. The most recent version of AJDSP (Oct. 2018) can be downloaded freely from Google Play [23].

The rest of the paper is organized as follows. Section 2 presents an overview of the AJDSP application and describes its various features. Section 3 describes the architecture of the application. A comprehensive review of signal processing functions implemented in the application is presented in Section 4. A description of various laboratory exercises is provided in Section 5. An assessment of AJDSP from student workshops and evaluations is given in Section 6. Possible extensions of the software in interdisciplinary areas, health monitoring, and social network applications along with concluding remarks are given in Section 7.

II. AJDSP Overview

AJDSP is a standalone, graphical programming application that exploits the resources of the Android platform to create a compelling mobile learning environment. It uses smartphone technology to develop a novel learning environment for the signal processing community. The core of this application is to use smartphones to solve signal analysis exercises in signals and systems classes, as instruments to perform labs in DSP courses, and as dashboards to display/plot data. Developing the application for Android devices enables us to reach more educators and students. Using AJDSP, the students learn using a block diagram system-level approach using inputs and outputs, establish and run DSP algorithms with various configurations on their Android devices, access the application anywhere/anytime without the need for internet connectivity, use the multitouch interface to view interactive demos on convolution and *z*-transforms, and more. Interoperability of the application is preserved by adapting the user interface to suit a diverse set of Android devices.

Signal processing functions including signal generation, signal processing filters, and signal display units are incorporated into the application. AJDSP has capabilities to generate deterministic and random signals. In addition to conventional signals such as rectangular, triangular, and exponential, MIDI and DTMF waveform generators are also integrated into the software.

AJDSP has a rich suite of time and frequency domain signal processing functions. Modern mathematical techniques are used to implement signal processing algorithms like the fast Fourier transform, filter design, and *z*-domain transformations on the Android platform. Signal block parameters during simulation. We employ a modified version of the MVC paradigm to design the architecture of AJDSP as shown in Fig 4.

A. Model

The graphical modules along with the block parameters, visualization tools for time, frequency, and *z*-domain signals are included in the application. Figure 2 shows the function list along with the search functionality provided to better locate the modeling blocks. Two demonstrations, one describing the process of convolution and the other illustrating the relationship between different signal representation domains have been included. A comprehensive description of the various signal processing functions in AJDSP is given in Section 4.

A. Performing Simulations

To illustrate a simulation in AJDSP, we will look at one now that uses a Fast Fourier Transform (FFT). The functions blocks are accessible by pressing the '+' button in the main workspace. The Functions List view displays a comprehensive list of available functions.

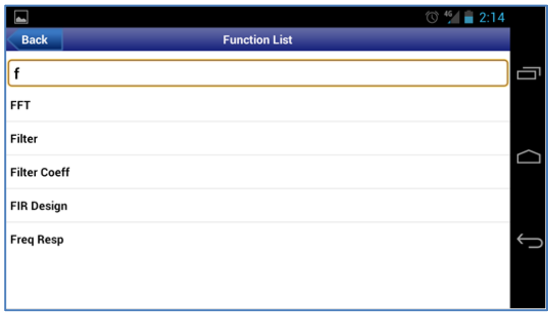


Fig. 2. Function Search Feature in AJDSP. Modules can be accessed by typing an appropriate query on the Function List view.

To perform an FFT simulation, the Signal Generator, FFT, and the Plot blocks are selected and added to the main workspace as shown in Fig 3(a). Double tapping on a block allows access to the function parameters. The fields of each block would now be set to the required values. Next, a connection is established between two blocks by tapping on the output pin of the first block and then tapping on the input pin of the second block. This is shown in Fig 3(b). The blocks are connected to complete the simulation as shown in Fig 3(c). The signal display can be viewed by double tapping on the Plot block and selecting the appropriate option in the menu. The Fast Fourier Transform of a rectangular signal is shown in Fig 3(d).

III. System Architecture

AJDSP is based on the Model-View-Controller (MVC) Paradigm illustrated in Fig 3(a). This separates the representation of information from the user's interaction with it. The model stores application data blocks and updates the algorithm implementations, and the input interface constitute the Model portion of our MVC paradigm. It contains the configuration information of the blocks as well as the algorithm parameters. The algorithm parameters are set by the user through the user interface. Each module, in turn, has a Part Calculator that implements the algorithms related to the function. More details about these DSP modules are presented in Section 4 below.

B. View

The View section renders blocks into a form suitable for interaction - providing functionalities for selection, movement, and deletion. It constitutes the main workspace and acts as the user interface for the connection of blocks, designing block diagrams, and performing graphical simulations.

C. Controller

The main control functions of the program are handled by the Controller. The Controller has two main components: the creation of blocks and updating the block parameters during simulations. Creating new blocks adds new graphical modules in the workspace. Updating blocks is mainly performed during block connection and to assist in editing block parameters.

IV. Signal Processing Modules

This section describes the signal processing algorithms in AJDSP. DSP algorithms for filter design, FFT, convolution, and z-domain transforms are implemented as graphical modules which can be connected together to perform simulations. The concepts for developing the functions were derived from [12]. We divide functionalities the functionality of AJDSP into signal generation, signal processing, and signal plotting blocks and describe each in detail here.

A. Signal and Speech Generation Functions

Signal generation functions in AJDSP provide a range of input signals for simulations. AJDSP provides a signal generator block along with MIDI, DTMF signal generation, and a long signal generator for speech and music signals. The desktop version of AJDSP, J-DSP, has a series of speech processing functions focusing on audio signal processing [42], perceptual coding [43], linear prediction [44], and speech coding [45] developed in Java and are in the process of being ported to AJDSP.

The Signal Generator is the basic input module of AJDSP. It supports a variety of discrete time-domain signals such as rectangular, triangular, delta, exponential, sinusoid, and random signals. Uniform, normal and Rayleigh random distributions are supported. Users can customize the signal by defining pulse width, amplitude, time shift and the periodicity of the signal.

B. Signal Processing Functions

A comprehensive set of DSP algorithms for convolution, sampling, filter design, FFT, and z-transform are implemented in AJDSP. In the time domain, the graphical convolution function allows the user to convolve and visualize two input signals in discrete time. In the frequency domain, we have a module for the Fast Fourier Transform (FFT), using the Cooley-Tukey [46] implementation. These modules allow users to calculate and visualize the FFT of a signal and to observe its changes based on the modification of the input signal.

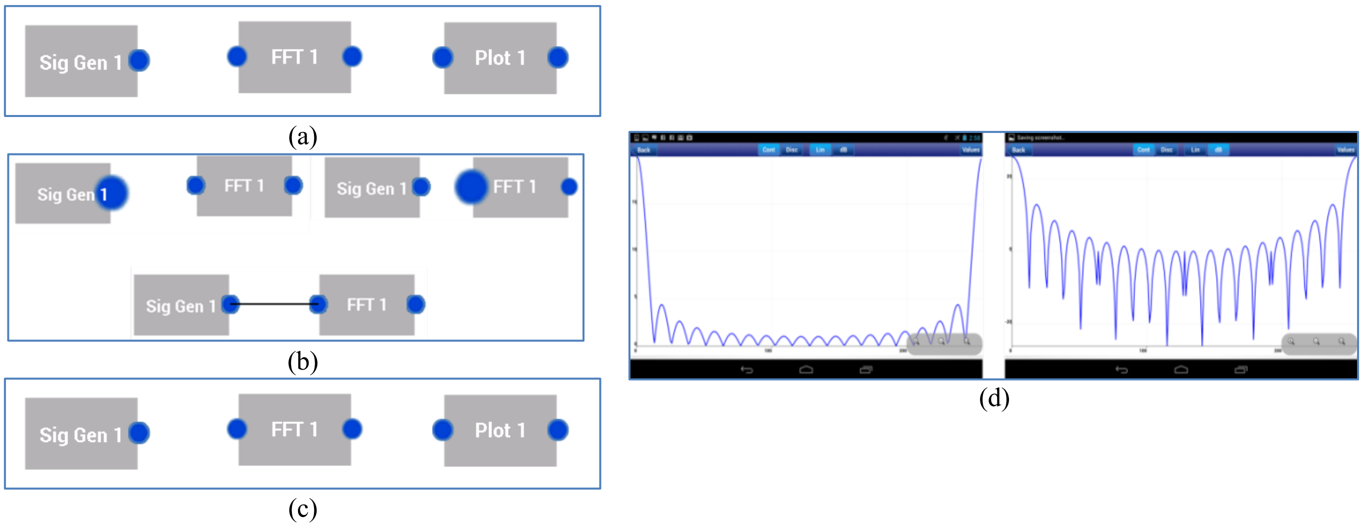


Fig. 3. A guideline to perform simulations with AJDSP. This figure shows an FFT simulation: (a) Placement of the blocks, (b) Procedure for block connection, (c) Block Diagram for FFT, and (d) FFT of a rectangular signal using Plot module.

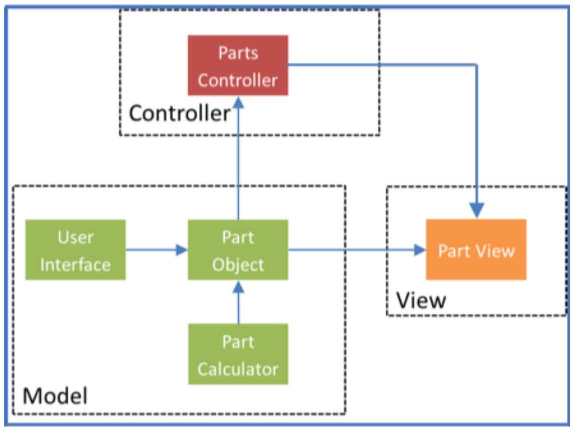


Fig. 4. The architecture of AJDSP based on the Model-View-Controller paradigm.

A wide range of capabilities for finite (FIR) and infinite impulse response (IIR) filters are provided. Specialized FIR filter design methods such as Kaiser and Parks-McClellan are available as well. The Kaiser Filter block designs FIR filters based on the windowing method. The design process involves calculating the Fourier series of the ideal filter and then multiplying it by a Kaiser window that best fits the filter specifications. The Parks-McClellan filter block uses the min-max implementation of the Parks-McClellan algorithm. This algorithm designs FIR filters by minimizing the maximum difference between the impulse response of the designed filter and that of the ideal filter for a given filter order.

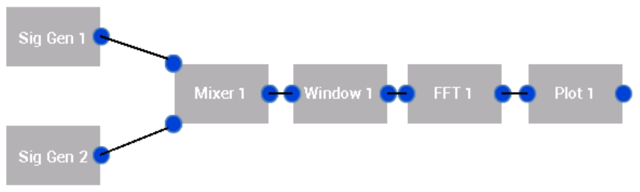


Fig. 5. Block diagram to demonstrate signal resolution properties windows

In addition to these standard methods, AJDSP also features a design toolkit which allows students to design filters by specifications. Students can design FIR filters including low-pass, high pass, band pass, and band stop filters using different windows such as Bartlett, Rectangular, Hamming, and Hanning windows. They can also design common IIR filters based on bilinear transformation such as Butterworth, Chebyshev, and Elliptic filters. AJDSP also supports multi-rate signal processing using dedicated blocks for upsampling and downsampling algorithms.

C. *Signal Display Functions*

Rich signal plotting capabilities have been incorporated into AJDSP to provide students with a visual representation of signal processing concepts. Three types of signal display modules have been implemented. The Plot function renders the input signal to a two-dimensional graph. The module can display both real and complex signals in both linear and dB scales. Zooming features have been provided to help students examine signals at different resolutions. The Freq-Resp block is the primary frequency domain display function of AJDSP. Freq Resp takes filter coefficients as its input and displays the frequency spectrum of the signal. The z-domain representation of a signal can be displayed by the PZ Plot module which calculates the position of the poles and zeros by applying the *z*-transform to the input filter coefficients.

D. *Interactive Demonstrations*

Two interactive demonstrations are included in AJDSP to assist students in visualizing DSP related concepts. One of these involves a convolution and the other illustrates the relationship between the z-domain and the frequency domain representations.

The first interactive demonstration involves the convolution of two signals and provides a graphical illustration of both discrete and continuous convolution processes. Two input signals are selected from a drop-down menu and added into the workspace. Each step of the graphical convolution procedure including flipping, shifting and progressive multiplication of the signal samples is illustrated with animations. The second demonstration in AJDSP is designed to assist students in understanding the relationship between the *z*-domain and frequency domain representations of a signal. Users can design filters by moving the poles and zeros around and examining the corresponding changes in the frequency response.

V. Exercises

In this section we describe some of the exercises that have been developed to be used with AJDSP. We also compare user experiences between the desktop (JDSP) and mobile versions (AJDSP). Command line vs graphical interfaces for signal processing applications are also compared with regards to ease of use and ability to foster better learning environments. In this section, brief descriptions of the exercises and the objectives of the exercises are given. The next section will describe the evaluation results and insights gained in this process. We will also discuss student feedback and planned improvements for the software.

A. *Discrete and Continuous Convolution*

This first exercise introduces the concepts of continuous and discrete time convolution. The Convolution Demo block helps the students to visualize the process of convolution through an animated demonstration. The effect of causality in continuous and discrete convolution is also illustrated. For a given set of input signals, the Convolution Demo block animates the convolution procedure and plots the resulting impulse response. Two sets of signals: (a) causal triangular and non-causal triangular signals, and (b) causal delta and causal exponential signals are examined for discrete convolution while another two signal sets: (a) causal rectangular and non-causal rectangular signals, and (b) non-causal exponential and non-causal triangular signals are used as input for continuous signals. The students are asked to make observations and comparisons of both the continuous and discrete convolution processes. This exercise evaluates the effectiveness of the convolution demo block in helping students understand the process of convolution.

B. *Fast Fourier Transform*

The second exercise focuses on frequency and z-domain representations. This is aimed at evaluating the utility of the FFT and PZ (Plot Z-domain) placement modules of AJDSP. First, we look at the effectiveness of different window types in resolving the frequency spectrum. The input signal is the sum of two sinusoids with closely spaced frequencies. Different types of windows, such as Rectangular and Bartlett, are applied to the input signal. The resulting signal is then transformed to the frequency domain using the FFT block. The resolution capabilities of the two windows are then compared. Students are asked to relate the window properties to the respective resolution capabilities.

The second part of this exercise has two main goals: (a) to evaluate the PZ Placement module and (b) to compare the desktop and mobile interfaces regarding ease of use and support of conducive learning. An illustration of the mobile environment GUI is illustrated in Fig. 6. Students are asked to design a low pass filter using three sets of zeros and two sets of poles. This exercise was performed both on the desktop using JDSP and mobile tablets using AJDSP. Students were asked to compare their experience. The results of this evaluation are presented in the next section.

C. *Filter Design*

The third exercise focuses on filter design. Here, we want to (a) evaluate the filter design modules, (b) measure the capability of the software to support learning filter design, and (c) compare the command line and graphical programming interfaces for signal processing applications on mobile devices.

The first part of this exercise focuses on IIR filter design. Students are asked to design low pass filters using the IIR design block. Filter specifications including the passband and stopband cutoff frequencies and tolerances are provided. The filter is then designed using Butterworth, Chebyshev I, Chebyshev II, and Elliptic filters. Students examine and compare the different characteristics of the filters including pass and stop band ripples and phase characteristics.

The second part of this exercise concentrates on FIR filter design and compares graphical programming to command line interfaces in mobile devices. Students are asked to design a 14th order band pass filter with a rectangular window. A code snippet of MATLAB is given, and students were asked to execute it on the Android devices. The two modes of programming are then compared for ease of use, consistency with previously known methods, intuitiveness of the user interface, and supportiveness of the learning environments. The results of these evaluations are presented in the next section.

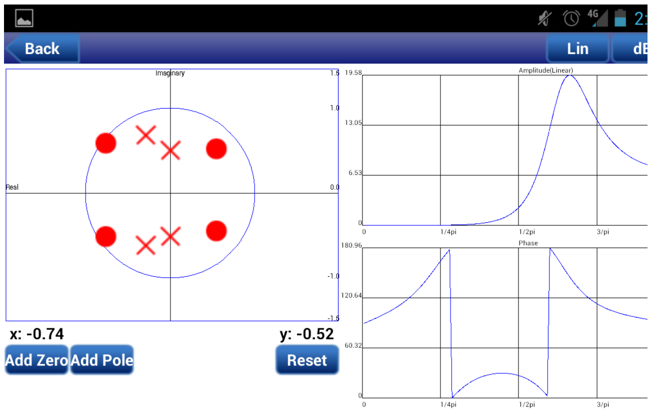


Fig. 6. User Interface of the PZ Placement Demo. The z-domain representation, with the poles and zeros of the filter, is shown in the left-hand side. The magnitude and frequency response of the filter is shown on the right-hand side. The poles and zeros can be moved around the z-plane resulting in changes to the frequency response.

VI. Assessments

A detailed evaluation methodology was designed to assess the AJDSP application. The evaluations are based on the exercises presented in Section 5. These exercises assess various aspects of the software including educational value addition, robustness, and usability.

Two workshops were conducted, one for graduate and one for undergraduate students, along with a K-12 outreach program described below. The goal of the graduate student workshop was to assess the robustness and the accuracy of the software. The undergraduate workshop assessed the teaching ability of the application to help students understand and learn various signal processing concepts. In addition, the evaluation also determined the interactive capabilities of the AJDSP framework and its ability to sustain student interest. General assessments on the aesthetics and usability of AJDSP are combined to obtain an overall subjective opinion rating for the application. The pedagogy adopted for these tests include: (a) a lecture on the pertinent signal processing concepts, (b) having the students take a pre-quiz on the concepts involved in the laboratory exercise, (c) having the students perform the described simulation exercises and laboratories using AJDSP, (d) and finally having the students then take a post-quiz to test conceptual understanding.

In addition to working with graduate and undergraduate college students, a K-12 outreach program has been used for K-12 outreach activities in collaboration with a local high school Mathematics teacher. We have worked with Corona Del Sol (CDS) High School and held outreach sessions in Spring 2015, Fall 2016, and Spring 2017. Images of the workshop with students learning signal processing tools through AJDSP are shown in Fig 9.

(a)

(b)

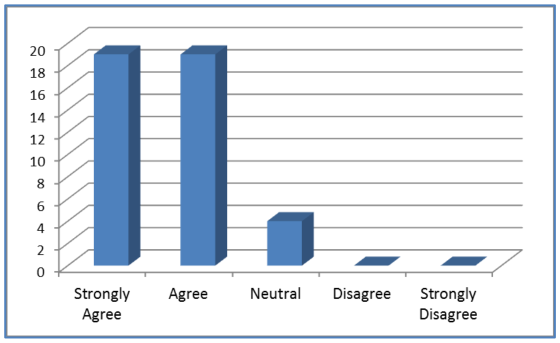
Fig. 7. Images from the K-12 outreach to enable students to understand signal processing concepts through mobile phones: (a) CDS high school, (b) Hermanas conference at Phoenix College

A. Workshop for Graduate Students

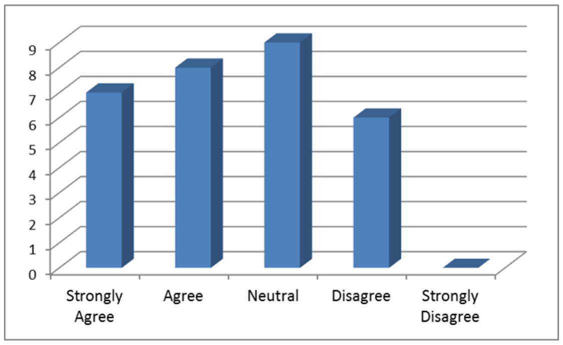
The workshop for graduate students was attended by 20 students. These students were mainly graduate students working in signal processing and related fields such as communication, machine learning, computer vision, and audio processing. These students had the experience to gauge the effectiveness of the application in teaching DSP concepts and to give valuable feedback on the robustness and usability of the application.

The questionnaires used in this evaluation were designed to obtain feedbacks on three main categories: the ability of the application to assist in DSP education, the usability of the application, and the efficiency and robustness of the program. The feedback from this workshop is divided into these categories and is summarized in Fig. 8. The questions followed the Likert scale formatting style and were given as a statement having five possible responses: strongly agree, agree, neutral, disagree, and strongly disagree.

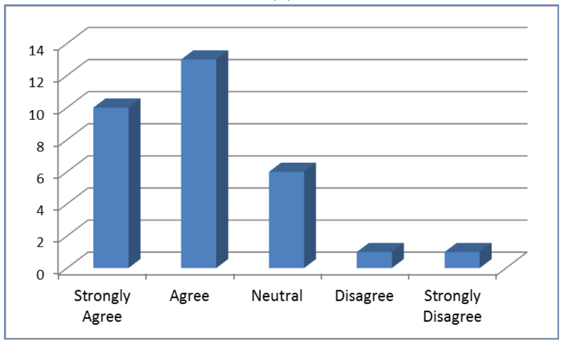
Questions meant to test the educational value of the software focused on the effectiveness of the application in teaching the concepts specified in the exercises described in section 5. Results on this topic are shown in Fig. 8(a) and indicate that the application is expected to be highly useful in teaching basic DSP concepts like convolution, the Fourier transform and filtering.



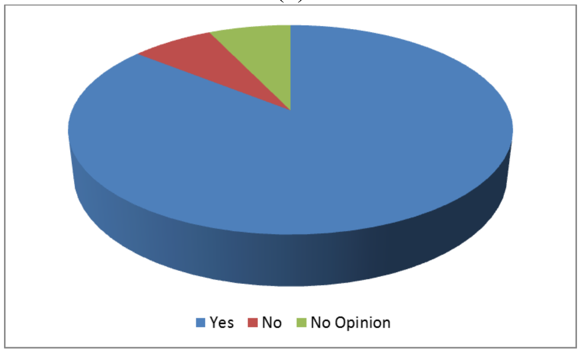
(a)



(b)



(c)



(d)

Fig. 8. Evaluations with graduate students on (a) Educational value (b) User Interface (c) Robustness (d) Satisfaction of Speed

Questions related to the usability of the application focused on the ease of use, screen size and the visual appeal of the user interface. The feedback is given in Fig. 8(b) and shows that the user interface is mostly satisfactory. The small amount of negative feedback was related to the block proportions being off due to screen size variation. These issues have since been resolved.

B. *Workshop for Undergraduate Students*

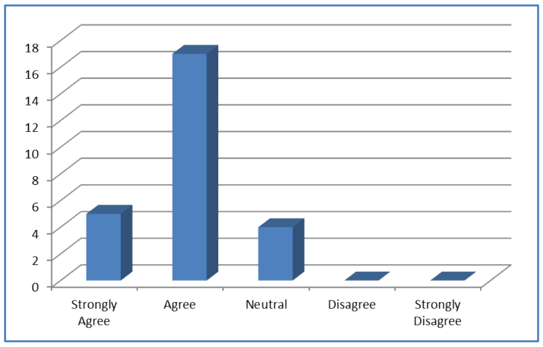
Fifteen students attended the workshop for undergraduates. The attendees were mostly senior undergraduates concurrently enrolled in an introductory DSP course at Arizona State University (EEE 407). The students had been previously introduced to the basic concepts of DSP such as convolution, filter design, and frequency transformation. The objectives of this workshop were as follows: 1) evaluate AJDSPs effectiveness in teaching signal processing concepts to students, 2) compare the suitability of command line and graphical programming interfaces in signal processing education, and 3) test the robustness of the application. Questions were in a similar format as the graduate student workshop, with two additions: 1) the questions of the workshop were more detailed and were designed to put forth the concepts behind the simulations and 2) an additional exercise was designed asking the students to design band pass filters in MATLAB and AJDSP. This was introduced to compare the suitability of command line vs graphical user interfaces for signal processing educational laboratories on mobile devices.

To measure the effectiveness of the application in conveying various DSP concepts, the pre-quiz and the post-quiz results are compared together in Fig 12(d). Students achieved an overall improvement of 11.36% after using AJDSP. Application effectiveness was also measured across different concepts including FFT, convolution, filtering, and the z-transform. Improvements in student performance were observed in all topics, with FFT and convolution registering the highest with improvement of approximately 30% and 25% respectively.

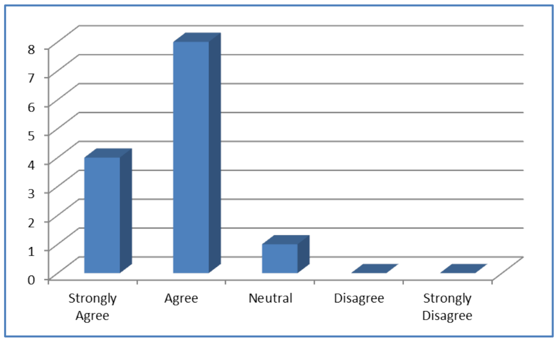
VII. Conclusions

Technological education can benefit dramatically from the emergence of educational apps on mobile phones and other mobile devices. These devices provide mobility, good processing capabilities, and an intimate learning environment. Employment of graphical programming enables the student to concentrate on the underlying concepts rather than programming syntax and other unrelated details, such as compilation environment customizing and program debugging.

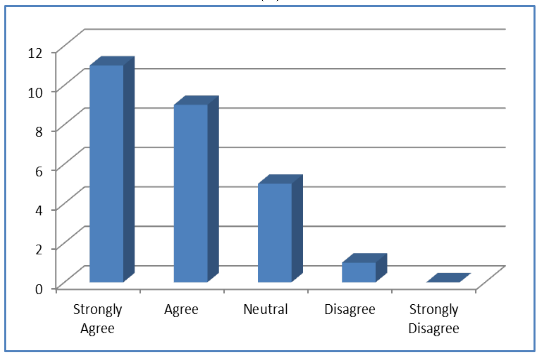
Few educational applications in DSP with any depth have been designed to work on mobile devices. We surveyed those we could find and discovered a need for a high-quality educational signal processing application for mobile devices. In this paper, we introduced and described AJDSP (Android J-DSP) as a graphical programming application for an Android operating system for DSP educators and students. The key features of AJDSP included signal processing modules, interactive animations, and an intuitive graphical



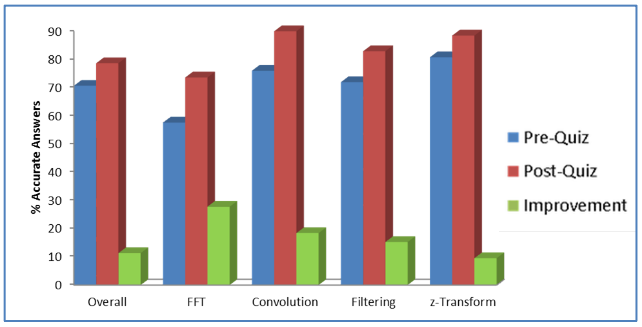
(a)



(b)



(c)



(d)

Fig 12: Suitability of the application in teaching DSP concepts: (a) Convolution, (b) Filtering, (c) FFT, and (d) Undergrad research

user interface. We also described the architecture of AJDSP, which is a derivative of the Model-View-Controller paradigm optimized for the Android platform.

Functional blocks for AJDSP are divided into signal generation, signal connection, time and frequency signal processing, signal display, and demonstration modules. These were briefly described. We discussed exercises developed using the modules, and the evaluation methodology that we used in our assessment of AJDSP. We held graduate and undergraduate workshops and describe how they demonstrated the usefulness, usability, and robustness of our app. Feedback from the workshops helped determine improvements and possible future directions for application development.

So far, AJDSP remains under development and maintenance to provide more education applications using recent Android APIs. Possible future extensions of the AJDSP software include health monitoring and multimedia signal processing among others. Simple filtering and the Fourier transform can be used to demonstrate the basic characteristics of audio signals. Various concepts such as sound compression, spectral distribution, and psychoacoustic modeling can be used by graduate students specializing in speech processing. Extensions to other domains such as machine learning, image processing, computer vision, and social networks can serve to reach a wider audience of educators and students.

Acknowledgments

The work at Arizona State University is supported in part by the NSF DUE award 1525716 and the SenSIP Center. The work at Clarkson University is supported in part by the NSF DUE award 1525224.

References

[1] M. Resnick, J. Maloney, A. Monroy-Hernandez, N. Rusk, E. Eastmond, K. Brennan, A Millner, E. Rosenbaum, J. Silver., B. Silverman, Y. Kafai "Scratch: Programming for All," *Communications of the ACM*, vol. 51, no. 11, pp. 60-67, November 2009.

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

[3] National Instruments, "LabVIEW," http://www.ni.com/labview/.

[4] MathWorks.Inc."Simulink". http://www.mathworks.com/products/simulink/.

[5] W. G. Holliday, Harvey, D. A. "Adjunct labeled drawings in teaching physics to junior high school students*." Journal of Research in Science Teaching*, vol. 13, no. 1, pp. 37–43, January 1976.

[6] P. Thornton, C. Houser, "Using mobile phones in education." *IEEE International Workshop Wireless and Mobile Technologies in Education,* pp. 3 – 10, .JungLi, August 2004.

[7] C. Chou, "Interactivity and interactive functions in web-based learning systems: a technical framework for designers." *British Journal of Educational Technology*, vol.34, no. 3, pp. 265–279, May 2003.

[8] S. Kanna, W. von Rosenberg, V. Goverdovsky, A. G. Constantinides, D. P. Mandic, "Bringing Wearable Sensors into the Classroom: A Participatory Approach." *IEEE Signal Processing Magazine*, vol. 35, no. 3, pp. 110-130, May 2018.

[9] Y. Xi, X. Tang, Z. Li and X. Zeng . "Application of digital signal processing tools for the detection of voltage sag/swell." *International Journal of Electrical Engineering Education*, vol. 55, no. 2, pp. 186-209, Februray 2018.

[10] N. Zhao, M. Wu, J. Chen, "Android-based mobile educational platform for speech signal processing," *International Journal of Electrical Engineering Education*, vol. 54, no. 1, pp. 3-16, April 2016.

[11] B. D. McPheron, C. V. Thangaraj, C. R. Thomas, "A Mixed Learning Approach to Integrating Digital Signal Processing Laboratory Exercises into a Non-Lab Junior Year DSP Course," *Advances in Engineering Education*, vol. 6, no. 1, pp. n1, 2017.

[12] A. Spanias, Digital Signal Processing: An Interactive (Second Edition) Approach: Lulu, 2014.

[13] A Spanias, "Java Digital Signal Processing (JDSP Editor)." http://jdsp.engineering.asu.edu/.

[14] A. Spanias, V. Atti, Y. Ko, T. Thrasyvoulou, M. Yasin, M. Zaman, T. Duman, L. Karam, A. Papandreou and K. Tsakalis, "On-line Laborites for Communication Systems Using J-DSO," *Proc. 10th DSP workshop and 2nd Signal Processing Education Workshop*, Gallaway Gardens, Pine Mountain, Georgia, USA, 2002.

[15] D. Ramirez, D. Rajan, P. Curtis, M. Banavar, H. Braun, C. Pattichis, A. Spanias, "Android Signal Processing for Mobile Health Monitoring," *38th IEEE EMBC 2016*, Orlando, August 2016.

[16] B. Robistow, R. Newman, T. H. DePue, M. K. Banavar, D. Barry, P. Curtis and A. Spanias, "Reflections: An eModule for echolocation education," *Acoustics, Speech and Signal Processing (ICASSP)*, 2017 IEEE International Conference on. IEEE, pp. 1562-1566, March 2017.

[17] M. K. Banavar, H. Gan, B. Robistow and A. Spanias, "Signal processing and machine learning concepts using the reflections echolocation app," *Frontiers in Education Conference (FIE)*. IEEE, pp. 1-5, Indianapolis, Indiana, December 2017.

[18] A. Dixit, S. Katoch, P. Spanias, M. K. Banavar, H. Song and A. Spanias, "Development of signal processing online labs using HTML5 and mobile platforms," Frontiers in Education Conference (FIE). IEEE, pp. 1-5, Indianapolis, Indiana, December 2017.

[19] A. Spanias, P. Curtis, P. Spanias, M. Banavar, "A new signal processing course for digital culture," *IEEE FIE 2015*, El Paso, Texas, October 2015.

[20] R. Santucci and A. Spanias, "JAVA Functions for Power Amplifier Linearization Techniques," *The ASEE Computers in Education (CoED) Journal*, vol. 5, no. 2, 2014.

[21] A. Dixit, U. S. Shanthamallu, A. Spanias, V. Berisha, and M. Banavar, "Online Machine Learning Experiments in HTML5," *IEEE Frontiers in Education (FIE)*, San Jose, California, October 2018.

[22] J. Liu, S. Hu, J. J. Thiagarajan, X. Zhang. S. Ranganath., M. K. Banavar and A. Spanias, "Interactive DSP Laboratories on Mobile Phones and Tablets," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Kyoto, August 2012.

[23] AJDSP Google Play Link: https://play.google.com/store/apps/details?id=com.prototype.ajdsp1

[24] A. Spanias, J. Blain Christen, T. Thornton, K. Anderson, M. Goryll, H. Arafa, U. Shanthamallu, E. Forzani, H. Ross, W. Barnard and S. Ozev, "The Sensor Signal and Information Processing REU Site, " *Proc. 2018 ASEE Annual Conference,* Salt Lake City, Utah, June 2018.

[25] A. Spanias and J. Blain Christen, "A STEM REU Site on The Integrated Design of Sensor Devices and Signal Processing Algorithms," *Proc. IEEE ICASSP 2018,* Calgary, April 2018.

[26] N. Kehtarnavaz and F. Saki, "Smartphone-based anywhere-anytime signals and systems laboratory," 2017 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6334 - 6338, March 2017.

[27] T. B. Welch, M. G. Morrow, C.H.G Wright, and R. W. Ives, "commDSK: a tool for teaching modem design and analysis," *ASEE Computer Education Journal*, 2003, vol. XIV, pp. 82–89, 2003.

[28] E. Sicard, ni2designs, "MentorDSP", http://www.mentordsp.com/.

[29] MathWorks.Inc."MATLAB". http://www.mathworks.com/products/matlab

[30] J. Potts, N. Moore and S. Sukittanon. "Developing mobile learning applications for electrical engineering courses," *Proceedings of IEEE in Southeastcon*, pp. 293 –296, Nashville, April 2011.

[31] N. Zhao, M. Wu, and J. Chen, "Android-based mobile educational platform for speech signal processing," *International Journal of Electrical Engineering Education,* 54(1), 3-16, April 2016.

[32] E. Soloway. C.Norris. P. Blumenfeld, B. Fishman, J. Krajcik and R. Marx, "Log on education: Handheld devices are ready-at-hand," *Communications of ACM*, vol. 44, no. 6, pp. 15–20, June 2001.

[33] P. Thornton and C. Houser, "Using mobile phones in English education in Japan," *Journal of Computer Assisted Learning*, vol. 21, no. 3, pp. 217–228, May 2005.

[34] J. Yerushalmy and O. Ben-Zaken, "Mobile phones in education: A case of mathematics," *The Institute for Alternatives in Education*, University of Haifa, October 2004.

[35] A. Gero, Y. Stav, and N. Yamin. "Increasing Motivation of Engineering Students: Combining "Real World" Examples in a Basic Electric Circuits Course," *International Journal of Engineering Education* 32, no. 6 (2016): 2460-2469, July 2016.

[36] J. Martín-Gutiérrez, C. E. Mora, B. Añorbe-Díaz, and A. González-Marrero, "Virtual technologies trends in education," *EURASIA Journal of Mathematics Science and Technology Education* 13, no. 2, 469-486, August 2016.

[37] GK Soft, "Electrical Engineering," Google Play Link: https://play.google.com/store/apps/details?id=an.ElectricalEng

[38] Electron Chaos. "Speedy Spectrum Analyzer". Amazon Link: https://www.amazon.com/Electron-Chaos-LLC-Spectrum-Analyzer/dp/B004YFNY78

[39] MathWorks.Inc. "Matlab Mobile." Google Play Link: https://play.google.com/store/apps/details?id=com.mathworks.matlabmobile

[40] S. Ranganath, J. J. Thiagarajan, K. N. Ramamurthy, S. Hu, M. K. Banavar and A. Spanias, "Work in progress: Performing signal analysis laboratories using Android devices," *Frontiers in Education Conference*, pp. 1,2, Seattle, October 2012.

[41] S. Ranganath , J. J. Thiagarajan., K. N. Ramamurthy, S. Hu. M. K. Banavar and A. Spanias, "Undergraduate Signal Processing Laboratories for the Android Operating System," arXiv preprint arXiv:1502.07026, 2015.

[42] A. Spanias, T. Painter, V. Atti, "Audio signal processing and coding," *John Wiley & Sons,* 2006.

[43] T. Painter and A. Spanias, "Perceptual coding of digital audio," *Proceedings of the IEEE*, vol. 88, no. 4, pp. 451-515, April 2000.

[44] A. Spanias and S. Urban, A. Constantinou, M. Tampi, A. Clausen, X. Zhang, J. Foutz and G. Stylianou "Development and evaluation of a web-based signal and speech processing laboratory for distance learning," *Proc. IEEE Acoustics, Speech, and Signal Processing* (ICASSP), vol. 6, pp. 3534-3537, Istanbul, June 2000..

[45] A. S. Spanias A, "Speech coding: A tutorial review," *Proceedings of the IEEE* vol. 82, no. 10, pp. 1541-1582, October 1994

[46] J. Cooley and J. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, April 1965.