SMARTPHONE-BASED ANYWHERE-ANYTIME SIGNALS AND SYSTEMS LABORATORY

Nasser Kehtarnavaz, Fatemeh Saki

Department of Electrical Engineering, University of Texas at Dallas

ABSTRACT

This paper presents a newly developed laboratory paradigm for the laboratory component of signals and systems courses. It involves the implementation of signals and systems algorithms that are written in MATLAB on smartphones using their ARM processors. This smartphone-based approach enables an anywhere-anytime environment for students to conduct signals and systems experiments. The steps involved for running MATLAB codes on smartphones are discussed together with the laboratory experiments that are normally covered in signals and systems courses. It is shown that this paradigm still keeps the programming language for conducting signals and systems experiments to MATLAB while enabling a truly mobile laboratory environment for students to learn the implementation aspects of signals and systems concepts.

Index Terms—Smartphone-based signals and systems laboratory, conducting signals and systems laboratory anywhere anytime, mobile laboratory platform for signals and systems courses

1. INTRODUCTION

A typical undergraduate electrical engineering curriculum incorporates a signals and systems course where students normally first encounter signal processing concepts of convolution, Fourier series, Fourier transform and discrete Fourier transform. For the laboratory component of such courses, the conventional approach consists of a laboratory environment that involves running signals and systems MATLAB codes on computer platforms. There exist a number of textbooks or manuals for the laboratory component of signals and systems courses based on MATLAB, e.g. [1-4].

The use of mobile technology for educational purposes has been steadily growing in the last few years, e.g. [5-7]. As an alternative to the conventional laboratory component of signals and systems courses, this paper discusses the use of the mobile technology of smartphones as an anywhere-anytime laboratory platform for students to carry out signals and systems experiments via their own apps that are written in MATLAB. This approach eases the requirement of using a dedicated laboratory room for signals and systems courses and would allow students to use their own smartphones and laptops as the laboratory platform to conduct signals and systems experiments.

The challenge in developing this alternative approach has been to limit the programming language required from students to MATLAB and not require them to know any other programming language. MATLAB is extensively used in engineering curricula and students are often asked to use it for various courses they take during their undergraduate studies.

The approach discussed in this paper meets the above challenge via the software shells we have developed, thus enabling students to run their own signals and systems MATLAB codes on smartphones as apps. It should be noted that creating an app whose code is written in MATLAB differ from running MATLAB scripts on smartphones via tools such as Octave app [8]. Besides mobility, another advantage of using smartphones is that the software development environments of smartphones are free of charge and students can download and run them on their own laptops. Although the emphasis in this paper is placed on Android smartphones, it should be noted that the same approach is applicable to iPhone smartphones as per the guidelines covered in [9].

The rest of the paper is organized as follows. Section 2 provides an overview of the software tools used to transition from MATLAB to smartphones. Then, a brief description of the labs that are normally encountered in a semester-long signals and systems laboratory course is covered in section 3. These labs include linear time-invariant systems and convolution, Fourier series, continuous-time Fourier transform, and discrete Fourier transform. The paper is concluded in section 4.

2. FROM MATLAB TO SMARTPHONES

This section includes the software tools and the steps that are needed to run signals and systems algorithms written in MATLAB on Android smartphones. The main challenge when using smartphones lies in the fact that their programming environments are different than MATLAB. To meet this challenge, we have developed the software shells that enable running MATLAB codes on smartphones in an easy manner. In other words, our developed software shells

ease the requirement for students to have a prior knowledge of the programming environment of smartphones.

The alternative laboratory approach to the conventional laboratory approach covered in this paper involves using the MATLAB Coder [10] utility of the MATLAB programming environment to generate equivalent C codes from MATLAB codes. This step is then followed by using our developed software shells to run the generated C codes in a seamless manner on smartphones. This way, the MATLAB programming knowledge expected from students is kept the same as the conventional approach while enabling experimentations to be conducted at any place or at any time via smartphones. As stated earlier, the emphasis in this paper is placed on Android smartphones noting that similar shells are developed for iPhone smartphones in [9].

2.1. C code generation via MATLAB Coder

This subsection states how an equivalent C source code is generated from a MATLAB code. The MATLAB Coder (or simply Coder) can be found in the MATLAB toolbar under Apps. As illustrated in Figure 1, the process begins by first writing a MATLAB function for a signals and systems problem. Then, the response of the MATLAB function on the smartphone platform is verified via a test bench MATLAB script.

Next, an equivalent C code is generated by running the Coder on the MATLAB function and changing the Numeric Conversion option to single precision floating-point arithmetic. After the function is selected, the input types need to be specified. This can be done either manually or by using the test bench script to automatically determine the data types. After the data types are set, the Coder then checks to see whether a C code can be generated from the MATLAB function. Once the MATLAB function passes the Coder checks, an equivalent C source code is generated by pressing the Generate button. At the conclusion of this step, a folder named *codegen* gets created in the directory of the MATLAB files, which includes the equivalent C code and the accompanying header files.

2.2. Android smartphone environment

To allow C codes to be compiled and run on the ARM processor of Android smartphones as apps, the following cost-free publicly available development tools are utilized: Android Studio Integrated Development Environment (IDE) along with Android Software Development Kit (SDK) [11], and Android Native Development Kit (NDK) [12]. Android Studio provides a comprehensive development environment,

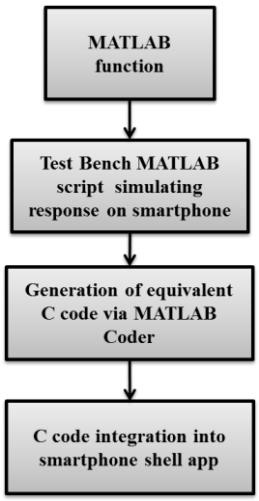

Figure 1. Steps to run MATLAB codes on smartphones as apps

SDK plug-ins, and an Android smartphone emulator. The NDK tool provides the support for incorporating C/C++ codes within the Android environment.

The Android shells developed in [9, 13, 14] for running C codes are written in Java and consist of the following three parts:

User Interface – This part comprises the main activity of the shell which controls its operation and displays outputs on Android smartphones.

I/O Handler – This part consists of the audio i/o which is split into three modules depending on their functionality. Microphone recording is handled by the module WaveRecorder, audio file reading is handled by the module WaveReader, and speaker and debug outputs are handled by the module WaveSaver.

Processing – The processing part is performed by the generated C code which is placed within the developed shell. The shell includes code segments that are written to interface a C code with the Java environment using the Java Native Interface (JNI) tool.

Note that students are not required to know any Java programming and they can simply follow the instructions in [9] regarding how to use the Android programming environment and the above tools to run C codes on Android smartphones as apps.

2.3. C code integration into smartphone shell

To run a generated C code on an Android smartphone, the following steps need to be taken. From the *codegen* folder, all the files with *.h* and *.c* extensions need to be copied to the *jni* folder. The generated C code needs to be checked for having the correct input and output variable data types.

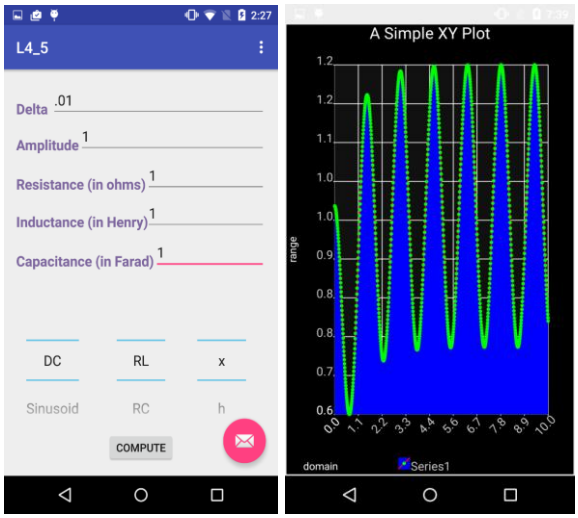

Figure 2. App screen of linear RLC circuit analysis

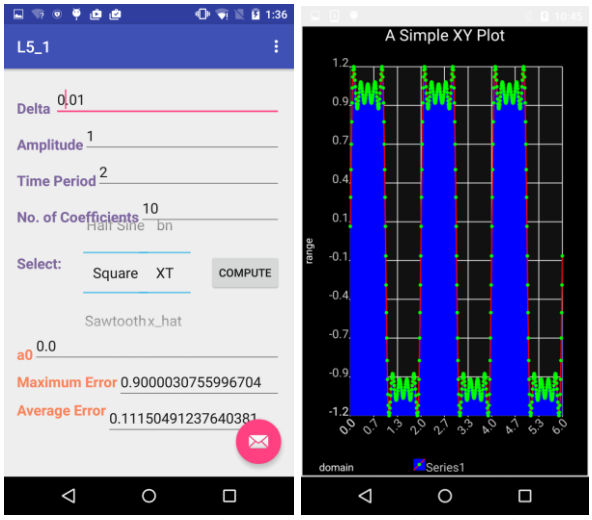

Figure 3. App screen of the Fourier series signal decomposition and reconstruction experiment

A modification of the generated C code may be needed if inconsistencies are observed. More specifically, the following two situations need to be checked. In case of array inputs, the generated code incorporates static array sizes and thus they need to be modified to be accessed by array pointers. Also, a persistent variable storage needs to be established by declaring a persistent variable and performing a one-time initialization. It is worth noting that it is not necessary for students to know C to make these modifications. More details regarding these simple modifications of the generated C code are provided in [15], which is a newly written textbook by the authors for signals and systems laboratory courses.

3. SIGNALS AND SYSTEMS LABORATORY EXPERIMENTS

The approach presented in this paper and described in detail in [15] constitutes the first time smartphones are used as a mobile laboratory platform to run MATLAB codes as apps for signals and systems courses. A representative set of the laboratory experiments that are included in a typical signals and systems course are described below. The labs are designed such that the input signal can originate either from a file or from the smartphone microphone. To allow for real-time operation on Android devices, the processing is carried on a frame by frame basis.

It is worth mentioning that these mobile laboratory experiments were utilized in the course EE3102 Signals and Systems Laboratory as part of the Electrical Engineering Curriculum at the University of Texas at Dallas for the first time in Fall 2016 with the enrollment of 120. A feedback or assessment provided by the students consistently was that the use of smartphones kept them interested and engaged in the lab experiments.

3.1. Linear time-invariant systems and convolution

This lab involves solving linear time-invariant (LTI) systems by using the convolution integral involving continuous-time signals. Due to the discrete-time nature of programming, an approximation of the convolution integral is made via summation. The first experiment involves examining the numeric approximation of the convolution integral for an input signal *x(t)* and a unit impulse response *h(t)*, followed by additional experiments involving different signal shapes.

Another experiment examines commutative, associative and distributive properties of convolution. As an application of convolution, RLC circuits which are linear time-invariant systems are solved via convolution by finding the voltage or current as output in response to an input voltage or current source. Figure 2 shows the app screen on an Android smartphone for this experiment. From the app, one can change the circuit type (e.g., RL, RC) and the input voltage type (DC or AC). The circuit response can be studied by changing the following settings: time interval Delta for approximating the convolution integral, circuit element values, and Amplitude of the input signal. Additional experiments including echo cancellation, noise reduction using mean filtering, and impulse noise reduction using median filtering are stated in [15].

3.2. Fourier series

In this lab, Fourier series or the representation of periodic signals via sinusoidal signals is examined. More specifically, two experiments are conducted: Fourier series signal decomposition and linear circuit analysis via Fourier series. The first experiment allows students to gain an understanding of Fourier series decomposition and reconstruction for periodic signals. For programming purposes, analog signals are simulated or approximated by considering a very small time interval.

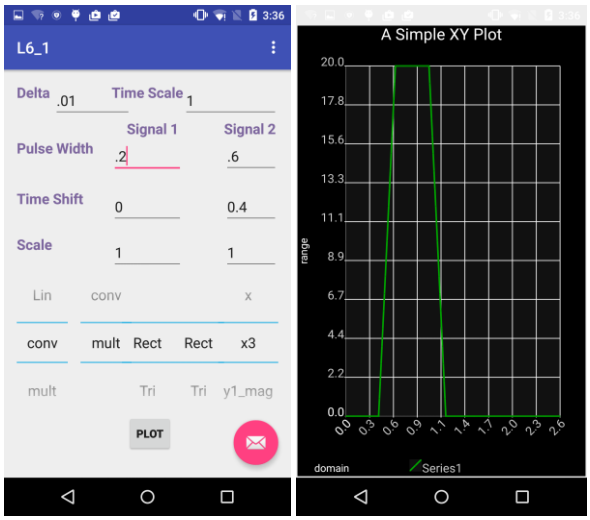

Figure 4. App screen of the CTFT properties experiment

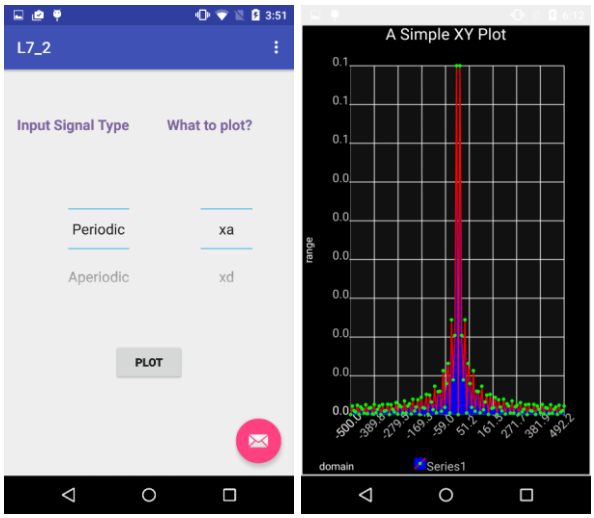

Figure 5. App screen of the experiment computing DFT and Fourier series for periodic signals and DTFT and CTFT for aperiodic signals

The second experiment involves linear circuit analysis via trigonometric Fourier series. This experiment demonstrates that the representation of any periodic signal into a number of sinusoids enables linear circuit analysis. It is shown that by knowing the response of a linear circuit to a sinusoidal input signal, the response to any periodic signal can be obtained by decomposing the signal into sinusoidal signals and performing a linear superposition of the sinusoidal signals. Figure 3 shows the app screen for one of the Fourier series experiments on an Android smartphone together with a number of control parameters including the effect of changing the number of Fourier series coefficients towards more accurate signal reconstruction. Additional experiments including circuit analysis, Doppler effect and synthesis of electrical music are stated in [15].

3.3. Fourier Transform

The continuous-time Fourier transform (CTFT), often referred to as simply Fourier transform, is computed numerically in this lab. In addition, its properties of time shifting, time scaling, linearity, time convolution and frequency convolution are examined. As applications of Fourier transform, a noise cancellation and an amplitude modulation experiment are conducted. Figure 4 shows the app screen for one of the experiments that examines the equivalency of multiplication of Fourier transforms in the frequency domain with convolution in the time domain. This figure shows the inverse of the product of two Fourier transforms is equivalent to the convolution of their corresponding signals in the time domain. Additional experiments including circuit analysis, Morse coding, Doppler effect and light diffraction are stated in [15].

3.4. Digital signals and discrete Fourier transform

This lab involves studying Fourier transforms for digital signals. Digital signals are formed by sampling and quantizing analog signals. The conversion of analog to digital signals is normally performed by using an analog-to-digital (A/D) converter, and the conversion of digital to analog signals is performed by a digital-to-analog (D/A) converter. The first experiment in this lab involves examining the Nyquist sampling rate for achieving a proper analog-to-digital signal conversion. In another experiment, the concept of aliasing caused by not having enough number of samples is examined. The error due to the quantization process is also examined in another experiment. Then, the reconstruction from digital signals to analog signals is conducted by using linear interpolation.

In another experiment as part of this lab, the discrete-time Fourier transform (DTFT) for analog signals and discrete Fourier transform for digital signals are computed and they are compared with the continuous-time Fourier transform (CTFT) and Fourier series for analog signals. Figure 5 shows the app screen of an experiment for computing DFT and Fourier series for periodic signals and DTFT and CTFT for aperiodic signals. Additional experiments including dithering, image filtering and DTMF decoder are stated in [15].

4. CONCLUSION

A newly developed paradigm for the laboratory component of signals and systems courses has been presented in this paper. Considering that MATLAB is the programming language normally used in the laboratory component of such courses, this paradigm is designed to allow codes written in MATLAB to be run on smartphones as apps. In other words, smartphones are used as the implementation platform for signals and systems concepts while retaining the programming language to MATLAB. The introduced smartphone-based paradigm enables an anywhere-anytime environment for students to conduct signals and systems experiments. A textbook on this newly developed paradigm is written by the authors which can be used as the manual for the laboratory component of signals and systems courses.

5. REFERENCES

[1] S. Mitra, *Signals and Systems*, Oxford, 2016.

[2] M. Sadiku and W. Ali, *Signals and Systems: A Primer with MATLAB*, CRC Press/Taylor and Francis, 2015.

[3] A. Palamides and A. Veloni, *Signals and Systems Laboratory with MATLAB,* CRC Press, 2011.

[4] N. Kehtarnavaz, P. Loizou, and M. Rahman, *An Interactive Approach to Signals and Systems Laboratory*, Connexions (now OpenStax), 2009.

[5] J. Potts, N. Moore, and S. Sukittanon, "Developing Mobile Learning Applications for Electrical Engineering Courses," *Proc. of IEEE Southeastcon*, Nashville, 2011.

[6] G. Engel, R. Palloff, K. Pratt, "Using Mobile Technology to Empower Student Learning," *Proc. of 27th Annual Conference on Distance Teaching & Learning*, Madison, 2011.

[7] A. Spanias, *Java Digital Signal Processing (J-DSP) Editor*, http://jdsp.engineering.asu.edu/jdsp.html, 2014.

[8] https://github.com/corbinlc/octave4android/

[9] N. Kehtarnavaz, S. Parris, and A. Sehgal, *Smartphone-Based Real-Time Digital Signal Processing*, Morgan and Claypool Publishers, 2015.

[10] http://www.mathworks.com/products/matlab-coder/

[11] http://developer.android.com/sdk/index.html

[12] http://developer.android.com/tools/sdk/ndk/index.html

[13] N. Kehtarnavaz and S. Parris, "Teaching Digital Signal Processing on Smartphones: A Mobile DSP Laboratory," *IEEE ICASSP Show&Tell*, Italy, 2014.

[14] N. Kehtarnavaz, S. Parris, and A. Sehgal, "Using Smartphones as Mobile Implementation platforms for Applied Digital Signal Processing Courses," *Proc. of IEEE Signal Processing Education Workshop*, Salt Lake City, 2015.

[15] N. Kehtarnavaz and F. Saki, *Anywhere-Anytime Signals and Systems Laboratory: From MATLAB to Smartphones*, Morgan and Claypool Publishers, 2016.