A JAVA SIGNAL ANALYSIS TOOL FOR SIGNAL PROCESSING EXPERIMENTS

Axel Clausen, Andreas Spanias, Anand Xavier, and Maya Tampi

Department of Electrical Engineering
Distance Learning Initiative
Arizona State University
Tempe, AZ 85287-7206, U.S.A

ABSTRACT

In this paper, a program to simulate discrete time linear systems is presented. The software is written as a Java[1] applet and can be accessed on the internet. Object oriented programming allows the user to construct and simulate a variety of systems. The program uses a graphical user interface which is easy to learn and it provides a visualization of the system and signal flow. The software is currently being used at Arizona State University to support an online software laboratory for a senior level digital signal processing (DSP) course. This paper presents our experiences gained by using the program in a class setting and gives examples of possible laboratory problems.

1. INTRODUCTION

1.1. Motivation

It is becoming increasingly apparent that university education in the next century will not be constrained to the traditional "lecture-only" classroom environment, but will rely on a technology-based distributed environment where instructors and students interact with electronic courseware. This trend is also supported by science education research [1, 2] which has shown overwhelmingly that classical non-interactive lecturing is very ineffective. In addition, education is becoming geographically dispersed with academic programs reaching students outside the localities and state borders. Nowadays, almost every student enrolled in a college program has access to the internet. Many college classes use the internet to provide information about homework, upcoming exams, etc. In this paper, we introduce a Java program to simulate simple discrete time systems. At Arizona State University, this program is used in a software lab for an introductory signal processing class. The advantage of a Java program is that it can be integrated in a page on the worldwide web (WWW). Therefore, students can perform the lab exercises wherever internet access is available.

In the case of Arizona State University, many of our students work in the industry and view lectures over a closed circuit television network. Usually, these students do not come to campus and hence it is convenient for them to complete the software laboratory from work or home.

1.2. Software

Programs which allow the user to design and simulate discrete time systems are widely used. Excellent examples are SPW[2] or System View[3]. However, these programs have a limited exportability. The use of the internet as a medium for active simulations and virtual experiments opens up new possibilities.

In recent years, it has become popular to develop internet based education modules and course material for asynchronous learning and active simulations (for example, JavaBayes[4] by Fabio Cozman). In the field of signal processing projects have been introduced as for example [4, 5]. However, these two approaches lack the capabilities and flexibility offered by our program.

The software we are presenting is written as a Java applet [5] [3]. As mentioned before, the main advantage of the programming language Java is that it produces code which is system independent. This means that the program can run on almost any computer as long as the web browser has a Java interpreter installed.

This work was supported in part by a grant from the Arizona State University Distance Learning Initiative. For information about the program and this software, please contact Andreas Spanias at spanias@asu.edu

[1]Java is a registered trademark of Sun, Inc.

[2]SPW is a registered trademark of Comdisco Systems, Inc.

[3]System View is a registered trademark of Elanix

[4]http://www.cs.cmu.edu/~javabayes/Home/

[5]Information about Java can be found at http://java.sun.com

1.3. Paper Organization

The rest of this paper is organized as follows: section 2 introduces the implementation of the software lab followed by a section about using the Java Signal Processing Editor in an online software lab for an undergraduate signal processing course. Section 4 gives a summary of student's feedback. Concluding remarks and a discussion of our plans for future work are presented in the final section.

2. IMPLEMENTATION

The Java Signal Processing Editor[6] is written as a standalone Java applet. It can be included in a webpage or it can be installed on a local system. It uses a graphical user interface (GUI) which can be seen in figure 1. The program requires a new version of a world wide web browser, like Netscape Navigator[7] 4.0 or Microsoft Internet Explorer[8] 4.0.

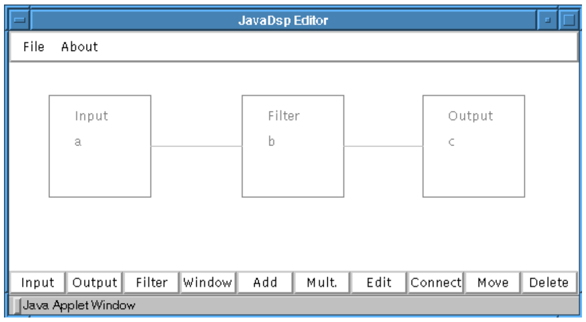
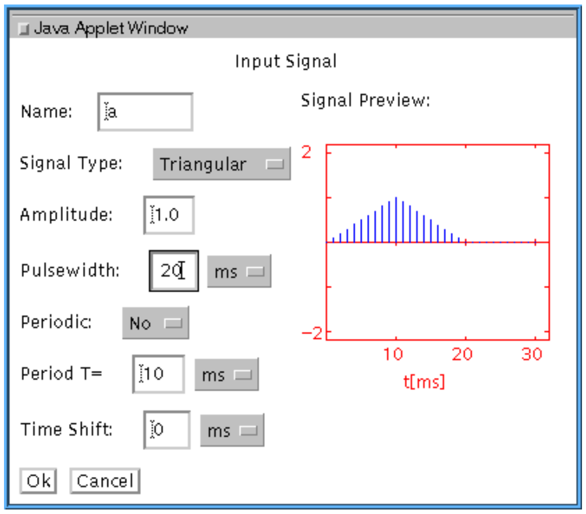


Figure 1: The graphical user interface of the Java Signal Processing Editor. Blocks are placed in the window and connected by using the mouse.

The user starts by specifying a sampling rate. The sampling rate is necessary for the frequency domain representation of signals. For students who have not been introduced to the concept of a sampling rate, it is possible to keep the rate at a default value.

The user can build an arbitrary system by creating blocks and connecting them to form an active simulation, for example, input - filter - output as shown in figure 1. The following blocks are available: Input, Output, Filter, Window, Add, and Multiply.

For each block it is possible to specify several parameters. Clicking with the mouse button inside a block causes a dialog window to pop up. Figure 2 shows the dialog box to choose an input signal. The user can



Name: a
Signal Type: Triangular
Amplitude: 1.0
Pulsewidth: 20 ms
Periodic: No
Period T= 10 ms
Time Shift: 0 ms

Figure 2: Dialog box to select an input signal.

specify the type of the signal, the amplitude, the duration, time shift, and periodicity.

The filter box allows the user to specify the coefficients of a rational transfer function of the form

$$H(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_{10} z^{-10}}{1 + a_1 z^{-1} + \ldots + a_{10} z^{-10}} \qquad (1)$$

where the coefficients are real numbers. This notation follows the notation used in [6]. If a higher order filter is desired, the transfer function must be factored into lower order subfilters.

The output box provides the option to view the signal in the time domain, in the frequency domain, or as tabulated values. The frequency domain values are computed by a fast Fourier transform (FFT) where the size of the FFT can be specified. Figure 3 shows the output signal corresponding to the triangular input signal from figure 2 and a filter with $b_0 = 1, a_0 = 1, a_1 = -0.9$. The blocks are connected as shown in figure 1.

The window block allows the user to select different types of windows. The dialog box shown in figure 4 provides all the windows which are presented in [6], including the Kaiser window. The user can specify the length of the window and for the Kaiser window it is possible to adjust the parameter β.

The add and multiply blocks take two incoming signals and add or multiply them.

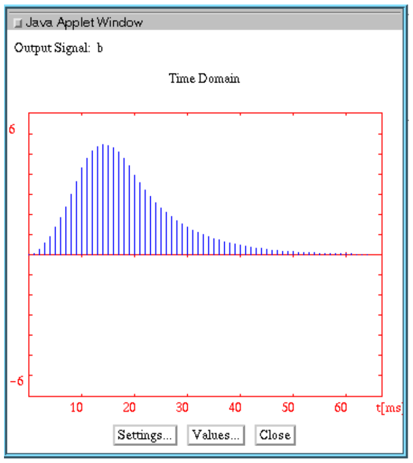
The graphical user interface was designed with the intention to provide a user-friendly environment. Creating, moving, and deleting boxes is accomplished by using the mouse. The keyboard is only required for entering some of the parameters of the boxes.

One drawback of Java is that for security reasons it is not possible to install a client-end filesystem. As far

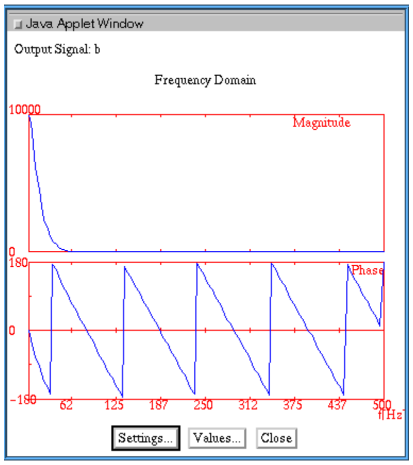
[6]http://www.eas.asu.edu/~eee407

[7]Netscape Navigator is a registered trademark of Netscape, Inc.

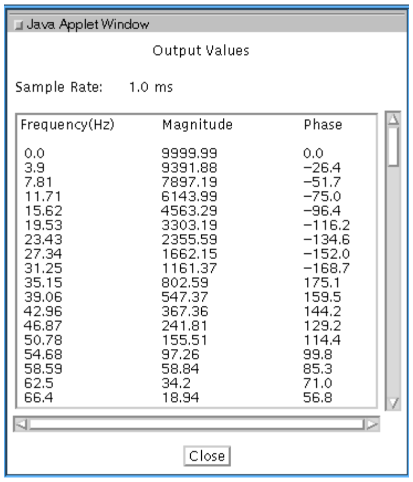
[8]Microsoft Internet Explorer is a registered trademark of Microsoft, Inc.



(a)



(b)

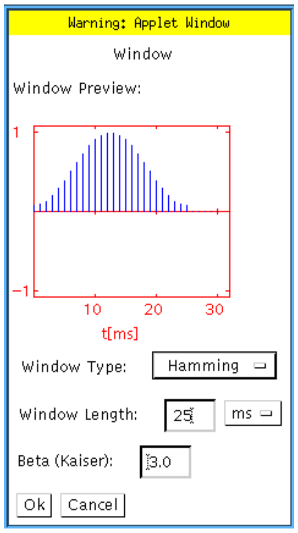


Sample Rate: 1.0 ms

Frequency(Hz)	Magnitude	Phase
0.0	9999.99	0.0
3.9	9391.88	−26.4
7.81	7897.19	−51.7
11.71	6143.99	−75.0
15.62	4563.29	−96.4
19.53	3303.19	−116.2
23.43	2355.59	−134.6
27.34	1662.15	−152.0
31.25	1161.37	−168.7
35.15	802.59	175.1
39.06	547.37	159.5
42.96	367.36	144.2
46.87	241.81	129.2
50.78	155.51	114.4
54.68	97.26	99.8
58.59	58.84	85.3
62.5	34.2	71.0
66.4	18.94	56.8

(c)

Figure 3: Output dialog box: output signal in (a) time domain and (b) frequency domain. In (c) a table of the values of the signal in the frequency domain is shown.



Window Type: Hamming

Window Length: 25 ms

Beta (Kaiser): 3.0

Figure 4: Dialog box for windowing.

as this software is concerned, it is not possible to save or load a configuration. This option would be desirable for complex laboratory tasks which might need several weeks to finish. This is a problem that we are working on.

3. DSP SOFTWARE LAB

In the Fall semester 1997, the program was used as part of a software lab for a senior level digital signal processing (DSP) course. In this section, we present some of the active experiments which were developed for this course and were supported by this software.

3.1. Discrete Time Convolution

The Java Signal Processing Editor was used to convolve an input signal with the impulse response of a filter. The students were asked to compare their analytical solution of the problem to the output of the program. Some problems used an unstable system to expose the students to the issue of stability.

3.2. Z-Transform, Inverse, and Properties

In another problem, the program was used to invert Z-transforms of rational transfer functions of the form (1). The students had to design a simple system consisting of an input box, a filter box, and an output box as shown in figure 1. Using a delta function as the input resulted in the impulse response at the output. Again, the students were asked to compare their analytical solution to the problem to the output of the program. Several exercises were also assigned to demonstrate various properties of the Z-transform.

3.3. Frequency Response

The frequency-domain output of the Java Signal Processing Editor was used to explain the relation between the time domain and the frequency domain. A similar setup as above can be used to obtain the frequency response of a system function. In other problems we were focusing on the relationship between magnitude and phase, allpass systems, or systems which have a generalized linear phase.

3.4. Future Exercises

For future software laboratory exercises we are planning to introduce new blocks which are "black boxes". For the students, the system function of these blocks is not visible and we might ask them to obtain the impulse response. Other questions using these blocks will address issues related to causality, stability or linearity.

4. REACTION

A questionnaire has been added to the webpage of the program. Students who used the program for the software lab were asked to tell us about their experiences with the software and their opinion regarding the use of the internet for a software lab.

The overall responses were positive and encouraging. Learning how to use the program took most students less than half an hour. The majority of students responded that it is more convenient for them to do the lab on the internet. Since all information about the lab is available on the world wide web, students who miss a class, do not have problems with getting lab handouts on time.

Most students thought that the program helped them to get a better understanding of signal processing and that they were able to visualize some of the concepts.

At the beginning of the semester we had some problems with accessibility of the programs. In particular, students working from off campus industrial sites complained that their computers at work could not access Java applets. The reason was found in firewalls which were set up in their local computer system and which did not allow the Java programs to transfer.

5. CONCLUSION AND FUTURE WORK

A Java signal processing editor has been presented. The program is intended to be used in a software laboratory for an introductory signal processing class. Our experiences in using the program in a class as replacement for a conventional software lab are very promising.

A major advantage of our software is it's modular implementation. Object-oriented programming allows the easy extension of the program to cover other aspects of signal processing. For the future it is planned to add more functionality to the program by including new modules for speech input and output or, random signals.

This program can be considered as a first building block to implement a complete internet-based distance learning course. In the future, the concept of this program could be extended to run real-time simulations wherever internet access is available. This approach might be very useful in situations where engineers who are working at different locations have to collaborate on the same project.

6. REFERENCES

[1] Carlson, "Longitudinal Investigation of the Development of the Function Concept," AMS/MAA Conference Proceedings, January 1995.

[2] Hake, "Interactive Engagement vs Traditional Methods: A six-thousand Student Survey of Mechanics Test Data for Introductory Physics Courses," American Journal of Physics (to appear).

[3] Mary Campione and Kathy Walrath, The Java Tutorial. Reading, Massachusetts, Addison-Wesley, 1996.

[4] Chaouki Abdallah et al., "Interactive DSP Course Development/Teaching Environment," Proceedings ICASSP '97, Munich, pp. 2249-2252, May 1997.

[5] Martti Rahkila and Matti Karjalainen, "An Interactive DSP Tutorial on the Web," Proceedings ICASSP '97, Munich, pp. 2253-2256, May 1997.

[6] Oppenheim, A. V., Schafer, R. W. Discrete Time Signal Processing. Prentice Hall Signal Processing Series. Englewood Cliffs, New Jersey: Prentice Hall, 1989.