INTEGRATING ANALYSIS, SIMULATION, AND IMPLEMENTATION TOOLS IN ELECTRONIC COURSEWARE FOR TEACHING SIGNAL PROCESSING

Roberto H. Bamberger[†] *Brian L. Evans*[‡] *Edward A. Lee*[‡] *James H. McClellan*[§] *Mark A. Yoder*[*]

[†]Dept. of EECS, Washington State University, Pullman, WA 99164-2752
[‡]Dept. of EECS, University of California, Berkeley, CA 94720-1770
[§]School of ECE, Georgia Institute of Technology, Atlanta, GA 30332-0250
[*]Dept. of ECE, Rose-Hulman Institute of Technology, Terre Haute, IN 47803-3999

ABSTRACT

A typical path in learning digital signal processing begins at the theoretical end and progresses toward the practical constraints imposed by implementation in hardware or software. On this path, the student would learn how to convert mathematical theory into algorithms and then algorithms into efficient implementations. In this paper, we first summarize the electronic courseware we have already developed in Mathematica, MATLAB, and Ptolemy to teach DSP theory, algorithms, and implementation, respectively. Then, we discuss ways to integrate our efforts to help students discover the connections between these topics.

1. INTRODUCTION

A typical path in learning digital signal processing begins at the theoretical end and progresses toward the practical constraints imposed by implementation in hardware or software. On this path, the student covers mathematical transforms and algebraic operations on signals and systems, simulation and analysis of algorithms that realize DSP theory, and hardware architectures and programming strategies that implement DSP algorithms efficiently. Different design tools can support learning along the way.

Among the tools that could help students learn DSP theory, we have primarily used the symbolic mathematics environment Mathematica [1]. Mathematica has limited signal processing abilities, so Evans and McClellan developed the Signal Processing Packages [2] to perform algebraic operations (e.g. convolution and transforms) on signals expressed as mathematical formulas Based on these packages, we have written interactive tutorials on convolution, filter design, and transforms [3], interactive tutorials on complex numbers, sampling, and modulation [4], and interactive solution sets [5] for a signals and systems textbook [6].

Using MATLAB [7], McClellan and others [8] have developed laboratory exercises to help students make the transition from mathematical formulas to programs that process data. The exercises help students convert theory into algorithms by having them write programs in MATLAB, at a higher level than Fortran or C.

Lee and others [9] have researched the use of block diagram languages to represent the dataflow of algorithms, organize algorithms hierarchically into systems, and generate efficient hardware and software implementations. They have encoded their research ideas in the visual block diagram environment Ptolemy [10]. Based on Ptolemy, Lee has developed exercises for undergraduate [11] and graduate [12] DSP classes. Lee has also created a multidisciplinary course to bring together a variety of topics in hardware/software implementation of DSP systems [13].

In this paper, we first discuss electronic courseware we have developed to help students learn DSP theory, algorithms, and implementation. Then, we discuss ways to integrate these separate efforts in order to help students convert mathematical theory into algorithms and algorithms into efficient implementations. At the end, we give information on how to obtain the electronic courseware we have developed.

R. H. Bamberger (bamberg@eecs.wsu.edu) was supported in part by the National Science Foundation under contract MIP-9116683. B. L. Evans (ble@eecs.berkeley.edu) and E. A. Lee (eal@eecs.berkeley.edu) were supported by the Ptolemy Project. J. H. McClellan (mcclella@eedsp.gatech.edu) was supported in part by the Joint Services Electronics Program DAAH-04-93-G-0027. M. A. Yoder (Mark.A.Yoder@rose-hulman.edu) is currently on Sabbatical at the Georgia Institute of Technology.

The Ptolemy project is supported by the Advanced Research Projects Agency and the U.S. Air Force (under the RASSP program, contract F33615-93-C-1317), Semiconductor Research Corporation (project 95-DC-324), National Science Foundation (MIP-9201605), Office of Naval Research (via Naval Research Laboratories), the State of California MICRO program, and the following companies: Bell Northern Research, Dolby, Hitachi, Mentor Graphics, Mitsubishi, NEC, Pacific Bell, Philips, Rockwell, Sony, and Synopsys.

0-7803-2431-5/95 $4.00

2. DSP THEORY IN MATHEMATICA

Mathematica is a symbolic mathematics environment. Its algebraic abilities are sufficient to teach introductory circuit analysis [14] but are insufficient for teaching signal processing. The Signal Processing Packages [2] extend Mathematica by defining common signals and systems. The packages also perform algebraic operations such as continuous and discrete piecewise convolution, Laplace and Fourier transforms, and z, discrete Fourier, and discrete-time Fourier transforms. The balance between continuous and discrete operations enabled us to write an interactive companion to [6].

Students can use the packages to compute transforms and convolutions symbolically. They can investigate properties of transforms; e.g., the z-transform of $a^n x[n]$ is returned as $X(\frac{z}{a})$. Students can also see how to perform computations by hand. For example, the packages can convolve piecewise descriptions of signals to produce a piecewise result and animate the flip-and-slide process. This ability has enabled us to take a visual approach in introducing convolution.

In teaching convolution, an instructor would traditionally show students the convolution integral and then spend a class period or two explaining how to evaluate the integral. Once mastered, the student would convolve many signals to get a feel for *what* convolution is. The instructor could turn the students loose on the computer to compute discrete approximations of continuous convolution. Although these programs improve insight by showing the flip-and-slide process, they do not produce piecewise answers in the form the students get when convoluting by hand.

Long before the students see the convolution integral, we show them how to use the Signal Processing Packages to convolve two continuous signals. Then, we ask them to convolve two pairs of signals, sketch the results next to the input signals, and predict the result of the convolution of a third pair. We start by asking them convolve a pulse function and then a triangular function with a Dirac delta function and then ask them to guess at the convolution of a finite-extent downward parabola and a Dirac delta function. They then convolve with a shifted Dirac delta function, a pair of shifted Dirac delta functions, etc. After convolving a dozen pairs of signals, most students are able to predict the convolution of a pulse train and a decaying exponential. Once they have a good understanding of *what* convolution does, then we show them *how* to compute it using the convolution integral. Although some students still cannot perform piecewise convolution by hand without making a simple error, most students now have an idea if their answer looks correct or not.

3. DSP ALGORITHMS IN MATLAB

When a student needs to make the step from mathematical formulas to actual processing programs that can act on real data, MATLAB becomes very appropriate. In MATLAB, the student writes programs, but at a higher level than FORTRAN or C. MATLAB combines just enough programming at just a low enough level to get across the computability issues.

Perhaps using the Discrete Fourier Transform (DFT) as an example would make this point clear. This transform can be derived mathematically by examining a sampled version of the Fourier transform for discrete-time signals. It has numerous properties which are presented as theorems in a typical DSP course, including the circular convolution property. However, the most important property of the DFT is that it is computable, but not only that, it is computable by an extremely efficient algorithm— the Fast Fourier Transform (FFT). For this reason, any presentation of the DFT without an accompanying lesson in using it for numerical processing misses a golden opportunity to explore the true meaning of issues such as circular convolution via a program for overlap-save which amounts to a few lines of MATLAB code. Another time-tested example is filtering in the DFT domain:

```
%-- # freq. samples in passband
Npass = 13;
HH = [ ones(1,Npass),
       zeros(1,512-2*Npass+1),
       ones(1,Npass-1) ];
%-- filter in the freq. domain
YY = HH .* fft( xin, 512 );
yout = ifft(YY);
```

Analysis of this code fragment leads to a discussion of the frequency sampling method of filter design and how `HH` controls the frequency response of the lowpass filter.

Writing programs in MATLAB can introduce students to two fundamental implementation issues:

1. *Functional programming*: many MATLAB functions are exactly equivalent to a module in a block diagram of a signal processing algorithm. For example,

```
plot(log10(abs(fft(sin(0.123*[0:99]), 512))))
```

is a single line of MATLAB code that amounts to a cascade of several operations: a sine-wave generator, a 512-point FFT, a log magnitude operation, and a plot.

2. *Vector programming*: operations underlying many DSP algorithms may be expressed directly as matrix-vector operations, thereby avoiding loops and improving performance. However, it may still be necessary

to manage blocks of data using buffering strategies in algorithms such as overlap-save convolution.

MATLAB is probably the most widespread and successful language for signal processing exploration related to courses. One notable demonstration is the FFT operation counting demo [8] which exploits MATLAB's floating-point operations counter to graph the trend in the number of multiplications vs. FFT length, and shows the great computational savings that power-of-two FFT lengths yield. An entirely new set of instructional demos is based on the programmable graphical interface of MATLAB 4. Within the latest Signal Processing toolbox is a filter design demo that redesigns a filter as the student moves graphical filter specifications with a mouse. Other demos involve sounds and spectrograms, which help students make connections between audio perception and the mathematics of the frequency domain. Georgia Tech students have built a pole-zero demo in which students can move root positions with a mouse and immediately see the updated time and frequency responses of the system. Development of more canned demos will structure the courseware provided in MATLAB.

Despite the central role that MATLAB plays in simulation for DSP, it has definite limits because it is an interpreted language. Its strength as a general purpose programming environment for scientific calculations has led to increased demand for a MATLAB compiler that would produce better execution times for long-running simulations. This general purpose nature is also one of its primary shortcomings when DSP design is considered. Not only would a compiler for certain DSP chips be desirable, but also it would be useful to have simulation capability that reflected actual processor constraints such as finite word length. This is where other tools must come into play. For example, the Ptolemy interface to MATLAB allows more detailed simulations and a migration of MATLAB algorithms to implementation. Interfacing MATLAB to design environments such as Ptolemy is an attractive path both for education and for actual design practice.

4. DSP IMPLEMENTATION USING BLOCK DIAGRAM LANGUAGES

Block diagram representations of signal processing systems are a pedagogically useful complement to mathematical representations. They are commonly used in informal ways in textbooks, but can be formalized and developed into a programming environment. Such a programming environment can be used by undergraduate and graduate students as a design laboratory, e.g Ptolemy at U.C. Berkeley [10, 11, 12].

Interactivity and animation in block diagram environments can greatly aid the development of intuition. In principle, real-time implementations generated by these environments would strengthen this intuition. The hardware required, however, is still prohibitively expensive for some teaching environments.

Unlike mathematical representations, a block diagram environment works with representations of *systems* that are conceptually closer to implementations. For example, Ptolemy would allow the student to decompose a larger system into its signal processing, communications, and control subsystems and then further decompose each subsystem into computational models (processing uniformly sampled data, discrete-event, finite-state machine, etc.). The student might then carry the design to a working prototype by matching the subsystems to hardware/software implementations. The support of hierarchical system design allows the student to tackle fundamental design problems.

Block diagram programming environments are likely to be closer to the design environment that students will use professionally than a wire-wrapped, soldering-iron hardware environment. They are hopefully closer than the assembly language, development board, in-circuit emulator environment with which so many DSP engineers today struggle. Intuition about semantics, therefore, may prove to be at least as important as intuition about circuits. To explore this issue, Lee has developed a multidisciplinary course on programming languages for real-time reactive (DSP) systems [13].

5. INTEGRATING TOOLS FOR ELECTRONIC COURSEWARE

The primary motivation for integrating analysis, simulation, and implementation tools is to provide students with a rich, interactive, user-friendly environment for "inquiry-based" a.k.a. "discovery-based" learning. Traditionally, signals and systems courses in electrical engineering have been taught using a linear learning model which concentrates on facts and formulas, relegating the student to a passive role. If the student can *apply* these facts and formulas successfully, then the student is considered to have mastered the material. Only recently has an increased emphasis been placed on *understanding* the facts and formulas. In the case of signals and systems theory, students would now understand the relationships between the systems being investigated and the physical properties of the systems and resultant signals. What will the resultant signal sound like, what will it look like, and how difficult will it be to implement this system? By properly integrating modern and evolving computer-based tools such as MAT-

LAB, Mathematica, and Ptolemy, it is possible to convert a course taught using a traditional lecture-based format into one where the student becomes an integral and active member of their education.

Discovery-based learning requires an easy-to-use environment for experimentation. The computer-based tools described in this paper facilitate the design of new exercises for students. Instead of solving contrived problems, the student would now investigate the impact of various system parameters and see how they affect the system's performance. In this fashion, the student can ask, and usually answer, questions such as "What if I change this...." or "Why not do it this way instead...". By physically seeing and experiencing the impact of their proposed changes, retention and understanding are drastically improved. At the same time, the role of the instructor would change. Instead of being a person that only disseminates information, the instructor becomes more of a tour guide, by helping the students along the path to understanding what they are seeing, and by providing suggestions as to how to verify what they think they are seeing and why it is happening. In this fashion, the students and instructors become partners in the learning process.

Of the learning tools discussed in this paper, Washington State University [4] and Rose-Hulman Institute of Technology introduce Mathematica first by means of self-contained tutorial Mathematica notebooks and MATLAB later. Georgia Tech introduces MATLAB first in sound and image processing laboratories and then Mathematica the following year, whereas U. C. Berkeley uses only Ptolemy. The next step is to integrate these learning tools together. Currently, MATLAB and Mathematica can call each other, and we have written a Ptolemy interface to MATLAB. However, one issue for the student is having a standard interface. One choice is to use Ptolemy as the primary interface, with MATLAB commands being run from Ptolemy. We are exploring World Wide Web interfaces as an alternative and for long-distance learning over the Internet.

6. INTERNET ACCESS

A freely distributable version of the Signal Processing Packages and Notebooks for Mathematica, as well as [2], is available by FTP to `gauss.eedsp.gatech.edu`. Commercial versions are available from PWS Publishing Company and Wolfram Research Inc.

Information on Mathematica notebooks on modulation and sampling is available on the World Wide Web (WWW) `http://www.eecs.wsu.edu/~bamberg/`.

Ptolemy Project papers, software, and information is available by FTP at `ptolemy.eecs.berkeley.edu` and WWW at `http://ptolemy.eecs.berkeley.edu` Ptiny Ptolemy, a demo version, is also available.

7. REFERENCES

[1] S. Wolfram, *Mathematica: A System for Doing Mathematics by Computer.* Redwood City, CA: Addison-Wesley, second ed., 1991.

[2] B. L. Evans, *A Knowledge-Based Environment for the Design and Analysis of Multidimensional Multirate Signal Processing Algorithms.* PhD thesis, Georgia Institute of Technology, Atlanta, GA, June 1993.

[3] B. L. Evans, J. H. McClellan, and H. J. Trussell, "Investigating signal processing theory with Mathematica," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, vol. I, (Minneapolis, MN), pp. 12–15, Apr. 1993.

[4] R. H. Bamberger, "Interactive tools for signal processing education: The signal processing instructional facility (SPIF Lab) at Washington State University," *Comp. Appl. in Eng. Education*, vol. 1, Mar. 1994.

[5] B. L. Evans, S. X. Gu, and R. H. Bamberger, "Interactive solution sets as components of fully electronic signals and systems courseware," in *Proc. IEEE Asilomar Conf. on Signals, Systems, and Computers*, (Pacific Grove, CA), Nov. 1994.

[6] R. D. Strum and D. E. Kirk, *Contempory Linear Systems Using MATLAB.* Boston, MA: PWS Publishing, 1994.

[7] C. Moler, J. Little, and S. Bangert, *Matlab User's Guide.* Natick, MA: The MathWorks Inc., 1989.

[8] C. S. Burrus, J. H. McClellan, A. V. Oppenheim, T. W. Parks, R. W. Schafer, and H. Schüssler, *Computer-Aided Exercises for Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1994.

[9] J. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt, "Ptolemy: A platform for heterogeneous simulation and prototyping," in *Proc. of the 1991 European Simulation Conf.*, (Copenhagen, Denmark), July 1991.

[10] The Ptolemy Group, "The Almagest: Ptolemy 0.5 Manual." (four volumes), 1994.

[11] E. A. Lee, "Signal processing experiments using Ptolemy — instructor's manual." (contact the author at `eal@eecs.berkeley.edu`), May 1994.

[12] E. A. Lee, "A design lab for statistical signal processing," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, vol. 4, (San Francisco, CA), pp. 81–84, Mar. 1992.

[13] E. A. Lee, "Computing and signal processing: An experimental multidisciplinary course," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, vol. VI, (Adelaide, Australia), pp. 45–48, Apr. 1994.

[14] M. A. Yoder, "An Electrical Engineer's Introduction to Symbolic Algebra via Mathematica." (contact the author at `Mark.A.Yoder@rose-hulman.edu`), Sept. 1991.