Signals and Systems Using MATLAB: An Integrated Suite of Applications for Exploring and Teaching Media Signal Processing

Bob L. Sturm and Jerry D. Gibson
Department of Electrical and Computer Engineering and the Media Arts & Technology (MAT) Graduate Program
3431 South Hall, University of California Santa Barbara, Santa Barbara, CA 93106

Abstract— **Effectively teaching introductory media signal processing (MSP) to students requires a means of communicating complex concepts without advanced mathematics. At the Media Arts and Technology program at UCSB we are teaching graduate media arts students concepts of MSP. These students, while interested in learning how their digital tools work, have an incredibly tough time working through the material the way traditional engineering students do. To address this we have used MATLAB to build a comprehensive suite of exploratory demonstrations and applications tailored to illustrate sophisticated concepts, and to inspire students that may not possess a strong background in mathematics. Our application, "Signals and Systems Using MATLAB " (SSUM), is presented here, and its use in a course designed to teach MSP to media arts students is discussed. Its usefulness extends beyond media arts to engineering and computer science curriculum. SSUM can be obtained for free from `http://www.mat.ucsb.edu/~b.sturm`.**

Index Terms— **Computer based education, MATLAB demonstrations, interactivity.**

I. Introduction

There is no doubt that learning media signal processing (MSP) should be a required portion of any media arts technology program; students should at least understand the algorithms behind the software they use, specifications of their hardware, and be able to communicate with engineers. To begin to work with these concepts however, a student needs an ability and confidence in mathematics beyond what most media arts students have. This creates the difficult problem of how to successfully teach MSP to students who do not satisfy the prerequisites that even most freshmen engineering students do. The question "what should be taught" becomes "what can be taught?"

Without employing mathematics more complex than algebra, a class of MSP can become dull and pedantic. At a level just above comfort the students can only be taught to add sine waves in the complex domain, convolve two short signals on paper, and ultimately derive the magnitude response of a low-order system. But how beneficial is it just to teach a student in the media arts these mechanical and undemonstrative skills? As expected, most of the students need constant motivation, and learn the minimum motions necessary to slide by. Instead of finding creative applications for the concepts they are learning, students spend most of their time working on unrewarding elementary problems. By the end, a student may be able to perform convolution on paper, but has little knowledge of how it can, or even why it should, be applied.

There are several published texts that attempt to make concepts of MSP accessible [1]–[5]. These texts, however, are either too specific or too general to be of interest to a media arts student. Among these [1] is perhaps the most popular text, and attempts to make MSP more accessible by including a CD-ROM that has tutorials, movies, and MATLAB[1] demonstrations. The laboratories and movies included on the CD-ROM are good, but are geared more for the engineering student rather than the media arts student.

Using the computer to teach signal processing is not a new idea. Clausen and Spanias describe the creation and use of an on-line digital signal processing (DSP) laboratory programmed in Java [6]. The application is used to present visualizations and interactive demonstrations to students. Radke and Kulkarni have designed a similar application for their DSP lab, but programmed in MATLAB [7]. Rahkila and Karjalainen describe the benefit of computer-based education (CBE) for teaching DSP by virtue of it being multimedia [8]. Illustrating complex functions like filtering by applying it to a sound and hearing its effects can enhance comprehension and leave a longer-lasting impression than just deriving frequency responses on the chalk board.

While these computer-based signal processing demonstrations are good, none are appropriate for the type of students we are teaching. We have thus created a large suite of exploratory demonstrations and applications in MATLAB designed to motivate and inspire the introductory student. "Signals and Systems Using MATLAB" (SSUM) is designed first for practical effective demonstrations, second to provide an interactive experience, and third to serve as a repository of algorithms and code. Using SSUM, a lecturer can quickly illustrate concepts, and a student can gain a deeper understanding of the material. The code behind the application is open and free to be used in student projects. Within this paper we present this suite of

J. Gibson is Professor of Media Arts and Technology, and Electrical and Computer Engineering.
Please direct correspondence to `b.sturm@mat.ucsb.edu`.

[1]Created by The Mathworks, Inc., MATLAB is a popular, high-level, scientific programming environment that works on several operating systems.

0-7803-9077-6/05/$20.00 © 2005 IEEE

applications and review its use in a class teaching MSP.

II. Choice of MATLAB

There are several criteria in the development of our suite of demonstrations and applications of MSP. First, the concepts must be presented clearly with little interfering information. Second the applications should be direct, flexible, and fast. Third, sound and visuals must be used to demonstrate concepts. Fourth, the demonstrations should allow the user to explore the topic by changing parameters. Fifth, a student should be able to look into the code to understand how it works. Sixth, the demonstrations should be compatible with as many computer platforms possible. And finally, the cost to students to be able to run the applications on their own computer should be minimal.

There are a several low-level programming languages that can be used to demonstrate the application of DSP, such as C++, and Java; but these require a high proficiency in programming if students want to learn the implementation, not to mention the tangle of cross-platform issues. Sound processing languages SuperCollider [9] or the graphical programming applications Max/MSP,[2] or pd [10], can also be used to create interesting demonstrations, but only for sound. Though these have excellent real-time capability, they have marginal abilities for visual data display.

There are several high-level software packages that can be used to teach signal processing, such as Mathematica,[3] Octave,[4] and MATLAB. A good overview of these and other packages in terms of engineering education can be found in [11]. Mathematica is meant more for symbolic mathematics than creating applications, and it cannot easily produce sound. Octave, a free open-source mathematics software application, is quite compatible with MATLAB code, but it lacks much of the rich library of functions available in MATLAB. In addition there is no easy way to create graphical user interfaces (GUIs).

MATLAB provides a flexible integrated programming environment that is easy to use and understand, and inexpensive for students.[5]. MATLAB is platform independent, has superior graphics handling and visualization capabilities, and has a great GUI development environment for creating applications. In addition it offers unparalleled functionality with many different data formats of sound and images. It has an extensive library of routines, and "toolboxes" can be purchased to add specialized functions, such as advanced signal and image processing routines. Applications written in MATLAB are open; any user can look at the code. Furthermore, countless institutions, both academic and corporate, as well as many independent users worldwide, use MATLAB for algorithm development, prototyping, and complex problem solving. A drawback to using MATLAB, however, is its lack of real-time functions, like tracking a sound as it plays, or visualizing a spectrogram directly from the sound input. Though there should be some familiarity with vectors and matrices, the MATLAB programming language is easy to learn and intuitive. For these reasons it is clear that MATLAB is a good choice for developing applications that satisfy our criteria.

III. SSUM: Signals and Systems Using MATLAB

SSUM is a continually expanding suite of exploratory demonstrations and applications. It demonstrates essential principles and concepts of MSP without requiring advanced mathematics. To use SSUM, MATLAB must be installed,[6] as well as the signal processing toolbox.

SSUM currently has 37 programs illustrating concepts of waveforms, modulation, sampling and interpolation, aliasing, the frequency domain, convolution and filtering, pole-zero diagrams, analysis and synthesis, signal features, and many others. Many of these are applicable to sounds and images. There are also demonstrations of curious topics such as sound cross-synthesis, reverberation using all-pass filters, additive synthesis of birdsong, sine wave speech synthesis, and modeling of musical instruments.

SSUM is perfect for use in lectures, labs, and assignments. All SSUM programs are wrapped in GUIs, so there is no need for typing unwieldy commands at the prompt. Many of the applications are integrated as well. For instance, if one is creating a waveform in an application, it can be sent to another application for filtering, and to another to see its frequency content.

SSUM uses and extends the programming style used in the excellent "MATLAB Auditory Demonstrations" application [12]. Modularizing the code and keeping the GUI separate from the functionality makes SSUM much more manageable. When a new application is desired, it is quite easy to copy and reuse the functionality.

Figure 1 shows Sampling Explorer, demonstrating how continuous signals are sampled, quantized, and reconstructed. Using Sampling Explorer one can investigate the cause and effect of aliasing, the effects of quantization, and the process of making digital signals continuous using ideal lowpass filtering. The top plot represents the continuous input signal and the position of samples. The bottom plot shows in bold the result of interpolating the samples back to a continuous signal. With the sliders and text boxes the user can change input frequency, amplitude, phase, and offset, as well as sampling rate and sample word-length. The input can also be changed to square, triangle, sawtooth, and random waveforms. The plots can be altered as well by changing the number of periods to plot, hiding the grid, lollipops marking the samples, and the interpolation.

Filtering images can be explored using Image Filter Explorer (Figure 2). Once an image has been loaded, its two-dimensional Fourier transform is displayed. Several filters are available including the moving average, Gaussian, and median filter. The spatial frequency response of each linear filter can be plotted. The filters can be applied to only the horizontal or vertical

[2]Distributed by Cycling74: `http://www.cycling74.com`

[3]Created by Wolfram Research Inc.

[4]Available at `http://www.octave.org`

[5]Currently, the student version of MATLAB costs US $99.

[6]Preferably version 6.5 or greater.

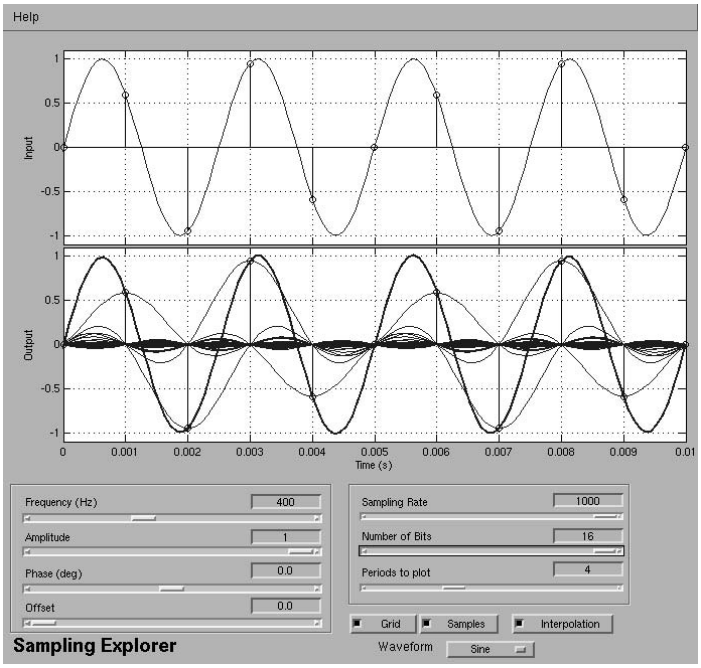

Fig. 1. Sampling Explorer

directions, or over blocks. Noise can be added to an image and the effects of filtering seen. Our students find the ability of the median filter to remove speckle noise startling.

Figure 3 shows an application demonstrating how any periodic waveform can be created by adding together sinusoids. The user is able to adjust frequency, amplitude, and phase for fifteen sinusoids, as well as select predefined waveforms like square and sawtooth. There are buttons to play and write the sound to a file. Additionally, the user can send this waveform to other SSUM applications like Sonogram Explorer or Fourier Spectrum Explorer. This way the student can begin to understand superposition and spectra. This integration of tools within each application is important for giving the student several views of the same thing—i.e. time and frequency domain.

Fourier Spectrum Explorer, shown in Figure 4, enables one to look at the spectrum of a sound. As the user drags a window

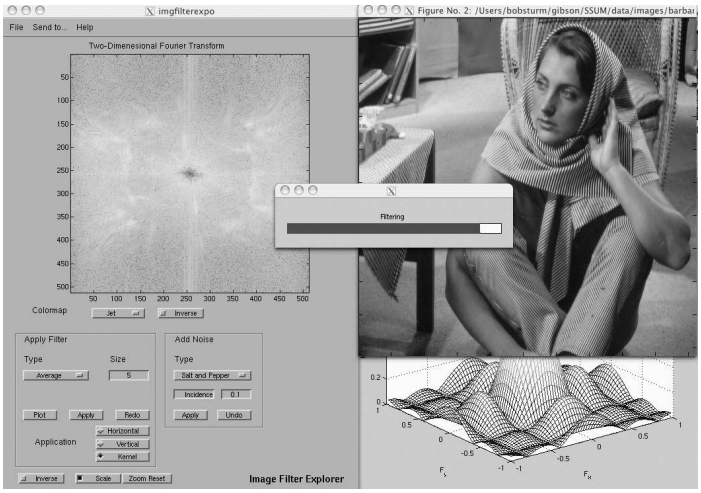

Fig. 2. Image Filter Explorer

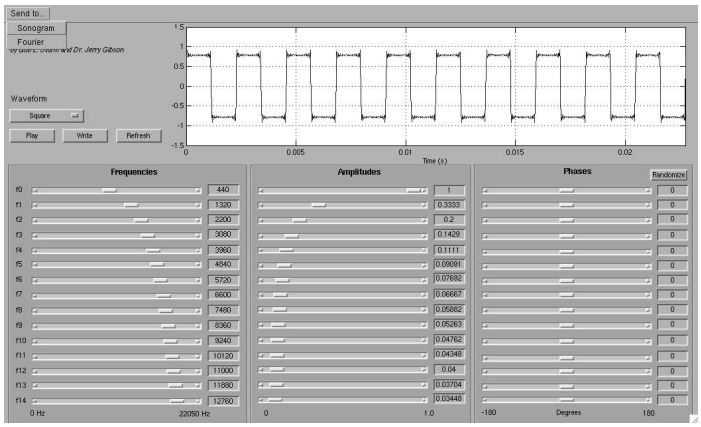

Fig. 3. Waveform Explorer

across the time-domain representation of a signal, demarcated by two red vertical lines, the spectrum changes. The window size and shape can be changed. It would be ideal to have the window sweep as the sound plays, but currently MATLAB cannot handle such tasks. A similar application is Sonogram Explorer, which presents the user with the short-time Fourier transform (STFT) of a sound.

Figure 5 shows Pole Zero Explorer. In this application one can add any number of poles and zeros (conjugate-symmetric pairs) onto the z-plane, drag them around or outside the unit circle, and watch how the frequency response and impulse response of the system changes. The constructed transfer function can be sent to other applications, such as Pole-Zero Filter Explorer, and Finite Difference Equation Explorer.

Figure 6 shows Cross-Synthesis Explorer. This application allows three different methods for cross-synthesizing sounds: convolution of the sounds, amplitude enveloping of one signal by the other, and linear prediction—using one sound as a model and the other as a source. Students really enjoy this demonstration and begin to realize the effects of convolution; suddenly the mystery of digital reverberation disappears.

SSUM nicely satisfies our design goals, and provides a fruitful multimedia experience for teaching and learning MSP. All of the applications are quick to compute and display results, so there is little worry for the learning process to come to a halt. In addition to its effectiveness for illustrating concepts

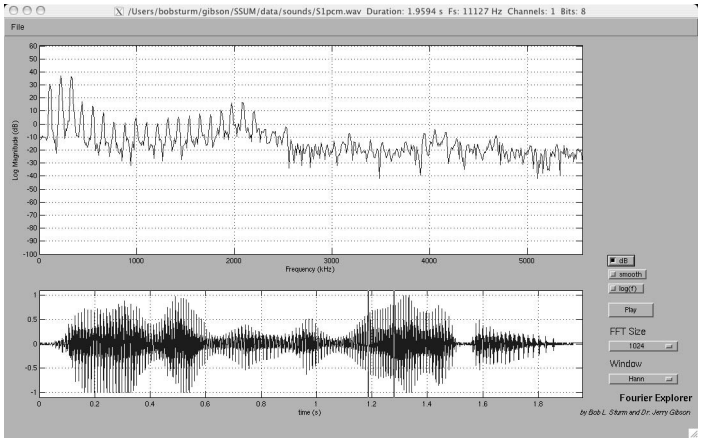

Fig. 4. Fourier Spectrum Explorer

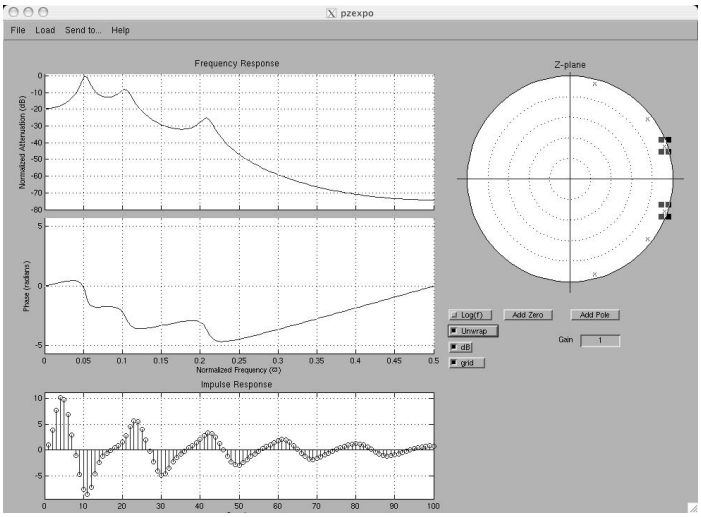

Fig. 5. Pole Zero Explorer

of MSP, SSUM also shows how to program MATLAB. Using these applications as models, students can easily and quickly construct their own interesting programs.

IV. Using SSUM in a Course

One of the core courses in the graduate MAT curriculum presents MSP to students with no background in engineering. The syllabus is not intended to be exhaustive, but the students should finish with at least an understanding of digital signals (e.g. samples, and sampling), the frequency domain (e.g. spectra), conversion between analog and digital signals (e.g. interpolation), filtering (FIR, IIR), and time-frequency transformation and analysis (e.g. DFT). This course, offered once a year during ten weeks, has been taught twice by the authors together.

In the first year we used the text DSP First [1], assigned written homework, and administered a midterm and exam. Seven assignments used problems from the text, but only two problems had a MATLAB-specific, but elementary, component. MATLAB was used only a few times to illustrate a concept, such as frequency content and aliasing. For this year the first hour of class was reserved for lecturing, and the second hour for addressing problems with the material, especially the homework.

After having an incredibly difficult quarter getting the students comfortable with the mathematics, and keeping their interest in the material, we realized the need for a more effective way to teach these students. SSUM was created to satisfy that goal, and was integrated into the course the second year it was offered. This time however the only required text was the student version of MATLAB. During the quarter, notes of the material were handed out, as well as copies of articles of interest, like [13]–[15]. Five long assignments were created that required MATLAB programming and work with SSUM. Instead of a midterm or final, the students were given the freedom to pursue projects exploring MSP during the last four weeks of the course. In the final class they presented these projects.

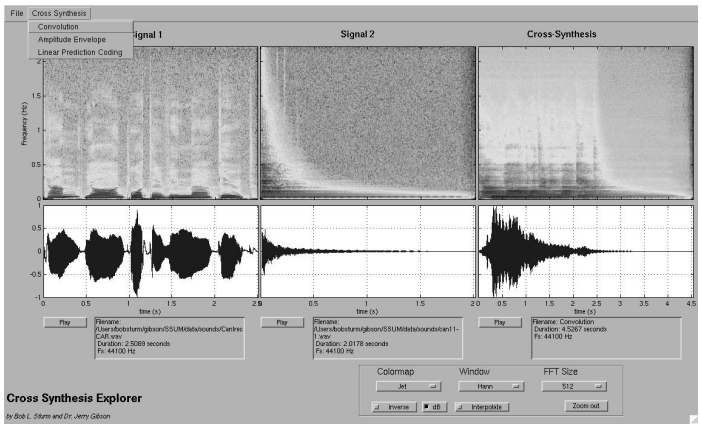

Fig. 6. Cross-Synthesis Explorer

During the second year every lecture was illustrated by several demonstrations from SSUM. The assignments placed more emphasis on programming with MATLAB, than on solving problems by hand. We relied on the illustrative ability of SSUM and MATLAB to increase comprehension and inspire the students to create their own applications using SSUM as a model. The class was therefore transformed from one requiring only mathematical practice, to one requiring programming in MATLAB. In this way the class provided a theoretical and practical experience, rather than just a pedantic one.

Response to SSUM was very positive, and the students were noticeably more motivated than those from the previous year. SSUM was essential for quick demonstrations of complex concepts. The integration of applications in SSUM provided a natural progression of topics. It was quite easy to move between applications to show, for instance, a finite difference equation, its poles and zeros, the frequency and impulse responses of the system, and its effects on arbitrary sounds.

Finding a balance between class topics and increasing the student's skill with MATLAB was difficult. As the students did not have enough knowledge to begin working with signals, some other topic needed to be used as a conduit for learning MATLAB. The best topic was in learning how SSUM works, from making the GUIs to writing the functions. Many complained during the beginning that too much time and importance was being spent on making interfaces. But as soon as the students had enough background, the homework focus shifted from learning MATLAB to creating and working with signals. By the fourth assignment they were given the task of building their own complete application using frequency modulation to synthesize musical instrument timbres.

Most of the final projects displayed a high level of sophistication. Topics included pitch detection, image filtering, music genre recognition, sound morphing, and transforming images into sound. All projects demonstrated hard work, and the students showed genuine interest in the topics they chose. The applications they wrote in MATLAB made good use of the functionality and GUI programming taught in the homework. A few students mentioned that had we not taught GUI programming, their applications would have been less interesting and manageable. Several students admitted the process was

tough, but were pleased and proud of their results. All students demonstrated a clear progression in their comprehension of MSP.

V. Conclusion

The use of SSUM in our classroom has proved to be indispensable for quickly and effectively illustrating concepts. SSUM is designed first for practical demonstrations, second to provide an interactive experience to enhance comprehension of MSP, and third to serve as a repository of algorithms and code. SSUM nicely satisfies these three goals, and creates a fruitful multimedia experience for teaching and learning MSP. All the applications are quick to compute and display results. As SSUM is used more in the classroom, both at MAT and other programs around the world, its collection of demonstrations and applications continues to grow.

By using SSUM as tool for a course, one is able to introduce applications first, and thus motivate the students to experiment and learn how they work, as well as create applications of their own. [16] describes the use of MATLAB to "help reconcile the declarative (what is) and imperative (how to) points of view on signals and systems." Students working with SSUM should be encouraged to program their own applications using it as a model. Having working examples at their disposal demonstrates that interesting and complex applications are possible.

It might be stated that by focusing on MATLAB in a syllabus one is replacing the difficulty of learning mathematics with programming, making the class more an exercise of programming MATLAB than learning MSP. However, due to the multimedia nature of MSP, it is more attractive to an introductory student to learn the concepts by building applications, than working problems by hand.

In our case at MAT, by catering to the creative motivations of the media arts student the difficult concepts of MSP can be approached with enthusiasm rather than dread. SSUM provides a practical experience with great examples, and artistically inspiring demonstrations. "If the teachers can create an enduring fascination for the subject-matter, the job's almost over: the more the students love the subject, the less help they need in their studies" [17]. This reflects our experience with our students.

Much more information on SSUM can be found in [18]. SSUM can be downloaded for free from `http://www.mat.ucsb.edu/~b.sturm`.

References

[1] J. H. McClellan, R. Schafer, and M. A. Yoder, *DSP First: A Multimedia Approach*. New Jersey: Prentice Hall, 1998.

[2] ——, *Signal Processing First*. New Jersey: Prentice Hall, 2003.

[3] K. Steiglitz, *A DSP Primer: with Applications to Digital Audio and Computer Music*. Menlo Park: Addison Wesley, 1996.

[4] P. R. Cook, *Real Sound Synthesis for Interactive Applications*. Massachusetts: A. K. Peters, 2002.

[5] U. Zölzer, Ed., *DAFx: Digital Audio Effects*. New York: Wiley, 2002.

[6] A. Clausen and A. Spanias, "An internet-based computer laboratory for DSP courses," in *Proc. of 28th ASEE/IEEE Frontiers in Education*, 1998. [Online]. Available: http://fie.engrng.pitt.edu/fie98/

[7] R. J. Radke and S. Kulkarni, "An integrated MATLAB suite for introductory DSP education," in *Proc. of the First Signal Processing Education Workshop*, 2000. [Online]. Available: http://www.ee.princeton.edu/~rjradke/papers/radkedsp00.pdf

[8] M. Rahkila and M. Karjalainen, "Considerations of computer based education in acoustics and signal processing," in *Proc. of 28th ASEE/IEEE Frontiers in Education*, 1998. [Online]. Available: http://fie.engrng.pitt.edu/fie98/

[9] J. McCartney, "Supercollider: A new real-time sound synthesis language," in *Proc. of the Int. Computer Music Conference*, 1996. [Online]. Available: http://www.audiosynth.com/

[10] M. Puckette, "Pure data," in *Proc. of the Int. Computer Music Conference*, 1996. [Online]. Available: http://www.crca.ucsd.edu/~msp/Publications/icmc96.ps

[11] M. Nagrial, "Education and training in engineering software and applications," in *Int. Conference on Engineering Education*, 2002. [Online]. Available: citeseer.nj.nec.com/560624.html

[12] M. Cooke, H. Parker, G. J. Brown, and S. N. Wrigley, "The interactive auditory demonstrations project," in *Eurospeech Conference*, 1999. [Online]. Available: http://www.dcs.shef.ac.uk/~martin/MAD/docs/articles.htm

[13] F. R. Moore, "An introduction to the mathematics of digital signal processing: Part I: Algebra, trigonometry, and the most beautiful formula in mathematics," *Computer Music Journal*, vol. 2, no. 1, 1978.

[14] J. Harvey, "Mortuos Plango, Vivos Voco: A realization at IRCAM," *Computer Music Journal*, vol. 5, no. 2, 1981.

[15] F. J. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," in *Proc. of the IEEE*, vol. 66, no. 1, 1978.

[16] E. A. Lee, "Designing a relevant lab for introductory signals and systems," in *Proc. of the First Signal Processing Education Workshop*, 2000. [Online]. Available: http://ptolemy.eecs.berkeley.edu/publications/papers/00/spe2/

[17] J. Koumi, "Designing for learning—effectiveness with efficiency," in *Effective Screenwriting for Educational Television*, R. Hoey, Ed. U.K.: Kogan Page Ltd., 1994, pp. 230–239.

[18] B. L. Sturm, *SSUM: Signal and Systems Using MATLAB; Creating an Effective Application for Teaching Media Signal Processing to Artists and Engineers*, 2004, (M.S. Project) University of California, Santa Barbara, Graduate Program in Media Arts and Technology, USA. [Online]. Available: http://www.mat.ucsb.edu/~b.sturm

Acknowledgments

The MathWorks, Inc., the makers of MATLAB, has supported this research by providing the authors with full multiplatform licenses to MATLAB. Support also provided in part by the National Science Foundation under grant numbers #DGE-0221713 (IGERT in Interactive Digital Multimedia), #CCF-0429884.

VI. Appendix

This is a list of exploratory demonstrations and applications currently implemented in SSUM.

Essentials

- *Complex Number Explorer*
  Visualize complex numbers; add/subtract vectors.
- *Convolution Explorer*
  Visualize linear and circular convolution with different signals.
- *Fourier Series Explorer*
  Inspect the Fourier series of a periodic step function.
- *Image Aliasing Explorer*
  Explore aliasing for images; use anti-aliasing filter for downsampling.
- *Image Sampling Explorer*
  Sample images at different resolutions.

- *Sampling Explorer*
  Demonstrate sampling, quantization, and interpolation.

- *Sinusoidal Explorer*
  Parameters of sine waves; add and multiply two sine waves.
- *Sound Aliasing Explorer*
  Explore aliasing and folding for sound signals; use anti-aliasing filter for downsampling.
- *Waveform Explorer*
  Generate waveforms using fifteen sinusoids.

Systems
- *Finite Difference Equation Explorer*
  Enter finite difference equations and see their frequency and impulse response.
- *FIR Filter Explorer*
  Create finite impulse response filters and apply to a sound.
- *IIR Filter Explorer*
  Create infinite impulse response filters and apply to sound.
- *Image Filter Explorer*
  Apply different filters to images; see linear filter frequency response.
- *LPC Explorer*
  Explore linear prediction for audio. Resynthesize with residual, noise, pulses, or another sound.
- *Model Explorer*
  See the effects of different models for simulating communication.
- *Modulation Explorer*
  Modulate one signal by another and see changes in spectrum and waveform.
- *Pole-Zero Explorer*
  Drag poles and zeros around a unit circle to watch frequency and impulse response change.
- *Pole-Zero Filter Explorer*
  Create filter using pole-zero plot and apply it to sound.

Analysis/Synthesis
- *Formant Explorer*
  Drag window across sound and watch the spectrum and formants change; also displays autocorrelation, cepstrum.
- *Fourier Explorer*
  Drag window across sound and watch the spectrum change.
- *Image Analysis/Reconstruction*
  Spectral analysis of image and reconstruction. Ability to swap magnitudes and phases of other images.
- *Image Spectrum Explorer*
  Explore the spatial frequencies in images.
- *Signal Feature Explorer*
  Explore the statistics of a signal, such as RMS, spectral centroid, and pitch.
- *Sinewave Speech Synthesis Explorer*
  Use linear prediction to reduce sounds to four sine waves.
- *Sonogram Explorer*
  Explore the short-time Fourier transform of a signal; trace partials with mouse clicks.
- *Sound Analysis/Synthesis*
  Spectral analysis of sound and resynthesis. Ability to swap magnitudes and phases of other sounds.
- *Vector Quantization Explorer*
  For arbitrary speech file determine and display codebooks of vectors of cepstral coefficients or LPC parameters.

Applications
- *Additive Synthesis Bird Song*
  Bird song synthesized using additive synthesis.
- *Catastochastic Additive Synthesis Composition Machine*
  Random music generator using additive synthesis with variable partials and envelopes.
- *Concatenative Synthesis Explorer*
  Synthesize sounds from other sounds using feature extraction and matching criteria.
- *Cross-Synthesis Explorer*
  Cross-synthesize two sounds using convolution, amplitude enveloping, or linear prediction.
- *Denoising Explorer*
  Denoise an audio signal using lowpass, median filtering.
- *Dynamic Time Warping Explorer*
  For speech find least-cost path through a reference with a test using cepstral coefficients, or LPC parameters.
- *Karplus-Strong String Explorer*
  Synthesize realistic string tones using lossy integrator. Convolve with recorded impulse response of a real instrument and room.
- *Reverberation Explorer*
  Explore reverberation and echo simulation using up to three cascaded comb and allpass filters.
- *Speech Endpoint Explorer*
  Explore finding the endpoints of recorded speech using loudness and zero-crossing statistics.
- *String Explorer*
  Explore the d'Alembert solution to the wave equation using delay lines.