AC 2008-1613: ENHANCING DIGITAL SIGNAL PROCESSING EDUCATION WITH AUDIO SIGNAL PROCESSING AND MUSIC SYNTHESIS

Ed Doering, Rose-Hulman Institute of Technology

Edward Doering received his Ph.D. in electrical engineering from Iowa State University in 1992, and has been a member the ECE faculty at Rose-Hulman Institute of Technology since 1994. He teaches courses in digital systems, circuits, image processing, and electronic music synthesis, and his research interests include technology-enabled education, image processing, and FPGA-based signal processing.

Sam Shearman, National Instruments

Sam Shearman is a Senior Product Manager for Signal Processing and Communications at National Instruments (Austin, TX). Working for the firm since 2000, he has served in roles involving product management and R&D related to signal processing, communications, and measurement. Prior to working with NI, he worked as a technical trade press editor and as a research engineer. As a trade press editor for "Personal Engineering & Instrumentation News," he covered PC-based test and analysis markets. His research engineering work involved embedding microstructures in high-volume plastic coatings for non-imaging optics applications. He received a BS (1993) in electrical engineering from the Georgia Institute of Technology (Atlanta, GA).

Erik Luther, National Instruments

Erik Luther, Textbook Program Manager, works closely with professors, lead users, and authors to improve the quality of Engineering education utilizing National Instruments technology.

During his last 5 years at National Instruments, Luther has held positions as an academic resource engineer, academic field engineer, an applications engineer, and applications engineering intern. Throughout his career, Luther, has focused on improving education at all levels including volunteering weekly to teach 4th graders to enjoy science, math, and engineering by building Lego Mindstorm robots. In his current position he manages the NI Textbook program and has had the opportunity to collaborate on numerous textbooks including "Learning with LabVIEW" by Robert H. Bishop and "Embedded Signal Processing with the Micro Signal Architecture" by W.S. Gan, and S.M. Kuo.

Luther graduated from the University of Missouri - Columbia in 2003 with a bachelor's degree in Electrical Engineering. He, his wife and son live in Austin, Texas.

Enhancing Digital Signal Processing Education with Audio Signal Processing and Music Synthesis

Abstract

Audio signal processing and music synthesis are familiar, accessible and engaging motivators for students to study digital signal processing (DSP). Hands-on project activities encourage deeper understanding of DSP concepts, and are used regularly in ECE481, a course that covers music synthesis for engineering majors at Rose-Hulman Institute of Technology. Students implement and experiment with music synthesis algorithms on a computer to gain a better appreciation for relationships between theory, sound, and visual representation of signals (time series, spectrum, and spectrogram). The LabVIEW graphical programming platform provides extensive support for DSP programming and soundcard operations, enabling students to quickly implement algorithms using graphical dataflow programming. The interactive user interface elements (controls and indicators) appear automatically while creating the graphical program, so the result is inherently interactive. ECE481 was recently revised to use LabVIEW, and students reported a high degree of satisfaction with the new approach.

Introduction

Music synthesis and audio signal processing offer students exciting applications of DSP concepts. As students implement synthesis and filtering algorithms, they develop a deeper understanding of the inter-relationships between the physical sound, its visual representations such as time-domain waveform and spectrogram, and its mathematical model. ECE481, Electronic Music Synthesis, an elective course in music synthesis for electrical and computer engineering majors initially offered in 1998, helps students tie together numerous conceptual threads from the required curriculum and strengthens their DSP programming skills with a series of mini-projects[1].

The choice of computational platform for mini-projects and in-class demonstrations is a key design decision for the course. Richard Hamming once said “the purpose of computing is insight, not numbers,” and students gain manifold insights when they interact with a signal processing system of their own creation. The choice of a development environment for optimum student learning revolves around two critical issues: (1) the total time required to transform a concept into a working system, and (2) the degree to which the system is interactive. Frustration results when the development process takes too long or is too complicated. In addition, the finished system should be as real-time and interactive as possible in order for the benefits of “computing for insight” to be fully realized. Text-based programming using m-scripts and the MATLAB program have traditionally enabled ECE481 students to implement a computer music algorithm as an m-script to produce an audible waveform. However, the additional development effort required to transform that script into a user-friendly, GUI-based, and interactive “musical instrument” is prohibitive for most student projects.

LabVIEW by National Instruments offers a unique graphical dataflow programming environment in which an interactive user interface automatically emerges during the programming process. A finished "VMI" (virtual musical instrument) built with LabVIEW is immediately and inherently interactive by virtue of LabVIEW's front panel controls-and-indicators paradigm. This aspect and extensive built-in functionality for signal processing, analysis, math, and sound card control come together in LabVIEW to give students an unparalleled means to create an interactive virtual musical instrument with a reasonable development effort.

Three years ago LabVIEW was adopted as the programming environment for the DSP prerequisite course,[2,3] and LabVIEW was likewise adopted as the programming tool for ECE481 to take advantage of student programming experience as well as the technical advantages offered by LabVIEW.

This paper will describe the music synthesis course topics and mini-projects. Several LabVIEW examples will be presented, and end-of-term assessment results of the new course format will be discussed.

ECE481: Electronic Music Synthesis

ECE481, Electronic Music Synthesis, is 10-week course that explores the technical side of electronic music synthesis, including topics such as analog synthesis, MIDI (Musical Instrument Digital Interface) standard, modulation techniques, additive and subtractive synthesis, physical modeling, and reverberation. Analog synthesis has largely been replaced by digital methods, but the mathematical models of analog synthesis still form foundation for many synthesis algorithms, including signal sources (periodic as well as noise), envelopes, and filters. MIDI is an interesting subject in its own right, and also serves as the basis for translating standard MIDI music files into control information to "play" a synthesis algorithm as a musical instrument. Hearing an algorithm in a musical context provides many new insights not otherwise possible by simply listening to individual tones. Modulation techniques (AM and FM) familiar from the communications systems course take on new meaning when the modulation frequency is in the audio range. Additive synthesis generalizes the Fourier series to permit independent time-varying frequency and amplitude trajectories of the harmonics (known as "partials"). Subtractive synthesis invokes much of the DSP material learned in the prerequisite course, and introduces time-varying filters. Reverberation also broadens student DSP background by covering comb filters and all-pass filters.

Most students choosing this elective course play a musical instrument, sing, or are avid music listeners. Examples of historically significant musicians, instruments, and compositions are frequently presented in class. As an end-of-term project, students create a two-minute composition using synthesis techniques learned throughout the quarter, and then present their work as a concert for interested students, faculty, and staff. Hatfield Hall auditorium is a relatively new world-class facility on the Rose-Hulman campus, and students enjoyed hearing their work in this setting.

All ECE481 course material including lecture notes and mini-projects has recently been converted to on-line format. Hosted on Connexions[4], *Musical Signal Processing with LabVIEW* (http://cnx.org/content/m15510) is a multimedia educational resource for students and faculty that augments traditional DSP courses and supports dedicated courses in music synthesis and audio signal processing. Each of the learning modules blends video, text, sound clips, and LabVIEW *virtual instruments* (VIs) into explanation of theory and concepts, demonstration of LabVIEW implementation techniques to transform theory into working systems, and hands-on guided project activities. Screencasts – videos captured directly from the computer screen with audio narration and a hallmark of this resource – use a mixture of hand-drawn text, animations, and video of the LabVIEW tool in operation to provide a visually rich learning environment. A companion paper[5] discusses delivery and production aspects of these learning modules.

Mini-Projects

Students engage the course material by completing a series of mini-projects. The course does not have a dedicated lab time, but selected class days are devoted to the projects which then become assigned homework to be finished outside of class. The mini-projects are implemented in LabVIEW and include the following activities:

Musical Intervals and the Equal-Tempered Scale – www.cnx.org/content/m15440 – Discover why musical intervals are specified as ratios, and derive the equation to convert a MIDI note number to frequency.

Analog Synthesis Composition – www.cnx.org/content/m15443 – Emulate a modular analog synthesizer using signal sources (sinewave, squarewave, triangle wave, and noise) and amplitude envelopes to create a simple piece of music.

MIDI File Writer – www.cnx.org/content/m15054 – Implement a set of low-level routines to create MIDI messages, timing information, and tracks. Combine the routines to assemble a binary standard MIDI file.

DSB-SC AM (Ring Modulation) and SSB-AM (Pitch Shifting) – www.cnx.org/content/m15468 – Use double-sideband suppressed carrier AM to modulate tones and speech. Use single-sideband AM as a pitch shifter for speech signals.

FM Synthesis and Chowning Instruments – www.cnx.org/content/m15495 – Experiment with the FM equation when the carrier and modulation frequencies are in the audio frequency range. Use FM to emulate a variety of physical instruments.

Additive Synthesis and Spectrogram Art – www.cnx.org/content/m15446 – Create a general-purpose sinusoidal oscillator whose frequency and amplitude tracks user-defined trajectories in time. Sum the outputs of multiple oscillators to model physical instruments and to create sounds by drawing arbitrary spectrograms.

Linear Predictive Coding (LPC) and Cross Synthesis – www.cnx.org/content/m15479 – Create a time-varying filter by extracting the time-varying formants (resonant peaks) of a speech signal, and then excite the filter using a music signal to create the effect of a "talking instrument."

LabVIEW Implementation Examples

Originally developed in 1986, LabVIEW is a cross-platform software development environment used by scientists and engineers for instrument control, data acquisition, control design / simulation, automation, and a variety of other technical computing applications. As a full programming language with extensive built-in functionality for signal processing, analysis, math, sound card I/O, LabVIEW is well suited to the needs of signal processing education.

Programming with LabVIEW involves working with Virtual Instruments (VIs), which are LabVIEW programs that consist of a block diagram and a front panel. The block diagram can contain graphical and textual components that represent the algorithm. The front panel component of a VI can include numerical inputs, dials, sliders, graphs and other user interface elements.

The following examples illustrate several LabVIEW implementation styles suitable for in-class demonstrations and student mini-projects.

<u>Chowning FM Instrument</u>
Pioneered by John Chowning in the 1970s, FM synthesis can emulate a wide range of natural instrument sounds using only two sinusoidal oscillators and suitable time-varying envelopes[6]. The FM equation is

$$y(t) = a(t)\sin(2\pi f_C t + i(t)\sin(2\pi f_M t)), \tag{1}$$

where f_C is the carrier frequency in Hz, f_M is the modulation frequency in Hz, $a(t)$ is the amplitude envelope, and $i(t)$ is the time-varying modulation index. The carrier frequency specifies the center of the FM spectrum, the modulation frequency determines the spectral density, and the modulation index controls the bandwidth. Figure 1 illustrates several choices for numerical parameters and envelope shapes $a(t)$ and $i(t)$ to emulate the sounds of a bell, wood-drum, and brass instrument.

Figure 2 shows the LabVIEW implementation of Equation 1. The block diagram is grouped into front-panel controls (user inputs), envelope generators, time basis generator, FM equation calculation, and front-panel indicators (waveform plots). The FM equation implementation uses "G code" graphical programming style, i.e., the algorithm is coded by wiring together native LabVIEW graphical icons. Thicker orange wires distinguish arrays from scalar values.

Pressing the "run" button ⇨ on either the block diagram window or front panel window (or pressing Ctrl+R) runs the program to create and play the audio waveform one time. Variations on a theme can be explored by changing the front-panel numerical values and re-running the program. This example typifies the "run-once" programming style; changing a front-panel control does not cause a response until the VI is run again.

Continuous interaction with the equation encourages deeper insight into the relationship between equation parameters and the corresponding spectrum and sound. The block diagram of Figure 3 embeds the FM equation, front-panel controls and indicators, and soundcard output icon in a while-loop structure to create continuous audio output with interactive controls. The spectrum display is also continuously updated. Sliders and a knob on the front panel (Figure 3b) allow point-and-click control of the equation parameters; numerical values may also be entered directly. Clicking on the various tabs of the "TripleDisplay" indicator (www.cnx.org/content/m15430) visualizes the output signal in both the time and frequency domains.

This example illustrates the "run continuously" programming style using polled front-panel controls. The two code sections that produce inputs to the while-loop structure – soundcard setup and time basis setup – execute one time when the VI starts. Execution then continues indefinitely inside the while-loop structure until the front-panel "stop" button is pressed, causing execution to transfer to soundcard cleanup and error handling. Audio is produced one block at a time, with a block length of 0.25 seconds. The sound card "write" icon in the lower right of the while-loop structure sets the pace for the while-loop, causing the front-panel controls contained within the loop to be polled once every quarter of a second.

Algorithm in Musical Context

Playing a synthesis algorithm as if it was a musical instrument provides more insight than is possible by listening to individual tones. As an example, the subtractive synthesis technique known as the Karplus-Strong plucked string algorithm[7] produces a remarkably life-like sound using only a delay line, a low-pass filter, and a noise source. The signal flow diagram of the algorithm (Figure 4) reveals an IIR filter structure suitable for student analysis as a project or in-class activity[1]. A real-time interactive LabVIEW implementation is used to listen to individual notes, which provides an initially satisfactory result.

The "MIDI JamSession" LabVIEW application VI (www.cnx.org/content/m15053) provides a means to listen to the plucked string algorithm played as a "virtual musical instrument" (VMI) using a MIDI music file to serve as the musical score. The VMI is a LabVIEW *sub-VI* (analogous to a subroutine or function call) that produces an array of audio samples when given information about the desired duration, frequency, and amplitude of the note. Figure 5 shows the plucked string algorithm signal flow diagram implemented as a VMI. The LabVIEW MathScript node illustrates how standard m-script can be incorporated directly into the block diagram.

Students enter the filename of a MIDI file in the upper left text entry box of MIDI JamSession (Figure 6) to load the note and timing information, and then specify the name of the VMI file in the associated track number the text entry boxes on right panel. Pressing "Render Audio" causes MIDI JamSession to repeatedly invoke the student's VMI to render one note at a time and insert each note into the finished audio file at the proper time. MIDI JamSession includes additional features such as rendering a subrange of the total duration, left/right pan controls to place an instrument in the stereo sound field, individual channel mutes, and the option to save the audio to a .wav sound file.

Listening to the MIDI JamSession rendering of a MIDI file using the basic Karplus-Strong implementation of Figure 4 instantly reveals that the pitch accuracy is poor, especially for higher frequencies. Chords sound especially out of tune. The students are now able to better appreciate the role of the delay line in setting the pitch of the synthesized notes. Since the delay line length

N is an integer, and the pitch is f_S/N (f_S is the sampling frequency in Hz), the pitch cannot be set with sufficient precision necessary to render the MIDI music file. An all-pass filter is then introduced as a means to insert a fractional delay into the signal flow diagram. After some additional analysis, the MathScript node text of Figure 5 can be modified slightly to introduce the all-pass filter, whose filter coefficients are selected to fine-tune the pitch frequency for each note. Rendering the same MIDI file with the updated algorithm yields impressive results.

Assessment Results

An end-of-term student survey was developed to assess the effectiveness of the course elements. Twenty questions were developed to which the student would respond "strongly agree," "agree," "weakly agree," "weakly disagree," "disagree", and "strongly disagree:"

1. The course helped me to better understand the relationship between frequency-domain concepts and time-domain concepts;
2. I am more confident in my ability to work with binary computer files (e.g., bit manipulations, file read/write);
3. * I am now more skilled at implementing my ideas on a computer;
4. * I have a better understanding of how DSP can be applied in real systems;
5. I have developed an increased interest in music;
6. * I spend more time thinking about what I hear;
7. * I had fun in this course;
8. * It was helpful to see the details of LabVIEW coding techniques discussed in class;
9. Implementing the concepts on the computer helped me to fully understand the concepts;
10. This was a useful course;
11. * I have a better understanding of digital filtering concepts;
12. * It was important to be able to hear examples of the various techniques while discussing them in class;
13. * The musical examples (CDs at the beginning of selected classes) stimulated my interest in the material;
14. * I would recommend this course to my friends;
15. The miniprojects were effective at helping me to practice the concepts;
16. LabVIEW is an effective way for me to implement the concepts learned in class;
17. * My LabVIEW skills have improved as a result of this course;
18. The miniprojects were effective at helping me to understand the concepts;
19. * The interactivity offered by the LabVIEW frontpanel user interface was important for lab projects in this course; and
20. The interactivity offered by the LabVIEW front-panel user interface would be important for lab projects in any DSP-related course.

Of 19 enrolled students in the course, 13 participated in the survey, or 68% of the class. Admittedly the total number of students surveyed is low, making statistical analysis inappropriate. Class sizes are limited to 30-40 students at Rose-Hulman, and the elective course ECE481 has a typical enrollment of 20 to 25 students. Additional assessment data will continue to be collected for future offerings of the course.

Figure 7 shows the student response histograms for each question. The student response to all survey questions was positive, with only a few "disagree"-type responses. Statements generating the strongest responses, defined as a majority of the students responding "strongly agree," are flagged with an asterisk in the list above.

According to the survey results, the large majority of students agreed that they had developed improved understanding of DSP concepts (Questions 4, 11) as well as the ability to implement those concepts (Questions 3, 17). The students also indicated that they were motivated by the course topics and structure (Questions 6, 7, 13, 14). The students also agreed that the interactivity offered by LabVIEW was important (Question 19).

Most students enrolled in the course had exposure to both m-script programming as well as LabVIEW coding prior to taking the course. Two additional survey questions were included to address the conversion of ECE481 to the LabVIEW programming platform:

21. Answer these questions only if you would have preferred to use m-script programming at the beginning of the term:
    - Now that I have completed this course, I believe that [m-scripts would have been | LabVIEW is] a better way to implement the mini-projects
    - Now that I have completed this course, I believe that [m-scripts would have been | LabVIEW is] a better way to implement the in-class demonstrations

22. Answer these questions only if you had preferred using LabVIEW at the beginning of the term:
    - Now that I have completed this course, I believe that [m-scripts would have been | LabVIEW is] a better way to implement the mini-projects
    - Now that I have completed this course, I believe that [m-scripts would have been | LabVIEW is] a better way to implement the in-class demonstrations

Six students (46% of survey participants) answered Question 21, indicating their original preference for m-script programming at the beginning of the course. By the end of the course, all six selected LabVIEW as a better way to implement the in-class demonstrations, and five of the six selected LabVIEW as a better way to implement the mini-projects.

Six students answered Question 22, indicating their original preference for LabVIEW programming. All six maintained LabVIEW as their preference for both the mini-projects and the in-class demonstrations.

Conclusions

Audio signal processing and music synthesis have been successfully applied to motivate students to study digital signal processing. ECE481, an elective music synthesis course for electrical and computer engineering majors, has been enhanced by adopting LabVIEW as the programming platform for student mini-projects and in-class demonstrations. Implementing computer music algorithms as a LabVIEW block diagram and interactive front-panel offers students an ideal way to create "virtual musical instruments" with reasonable design effort. Assessment results for the first offering of ECE481 using LabVIEW as the programming platform show that a large majority of the students report that the course improved their understanding of DSP concepts as well as their ability to implement those concepts, and that the course topics and structure were motivating. The students also concluded that LabVIEW was an effective programming platform for the course.

Bibliography

1. Doering, E. R., "Making Music with MATLAB: An Electronic Music Synthesis Course for Engineering Students," pp. 3581-3584 in *Proceedings of the 1999 International Conference on Acoustics, Speech, and Signal Processing*, Volume 6, 1999.
2. Yoder, M.A., and B.A. Black, "A Study of Graphical vs. Textual Programming for Teaching DSP," *2006 ASEE National Annual Conference Proceedings*, Chicago, IL.
3. Yoder, M.A., and B.A. Black, "Teaching DSP First with LabVIEW," *Digital Signal Processing Workshop, 12th - Signal Processing Education Workshop, 4th* , pp.278-280, Sept. 2006.
4. Baraniuk, R.G., C.S. Burrus, D.H. Johnson, and D.L. Jones, "Sharing Knowledge and Building Communities in Signal Processing," *IEEE Signal Processing Magazine*, pp. 10-16, Sep 2004.
5. Shearman, S., E. Luther and E.R. Doering, "Applying an Interactive, Modular Approach to Effectively Teach Signal Processing Concepts," *2008 ASEE National Annual Conference Proceedings*, Pittsburgh, PA.
6. Dodge, C., and T.A. Jerse, *Computer Music: Synthesis, Composition, and Performance*, 2nd ed., Schirmer Books, 1997.
7. Moore, F.R., *Elements of Computer Music*, Prentice-Hall, 1990.

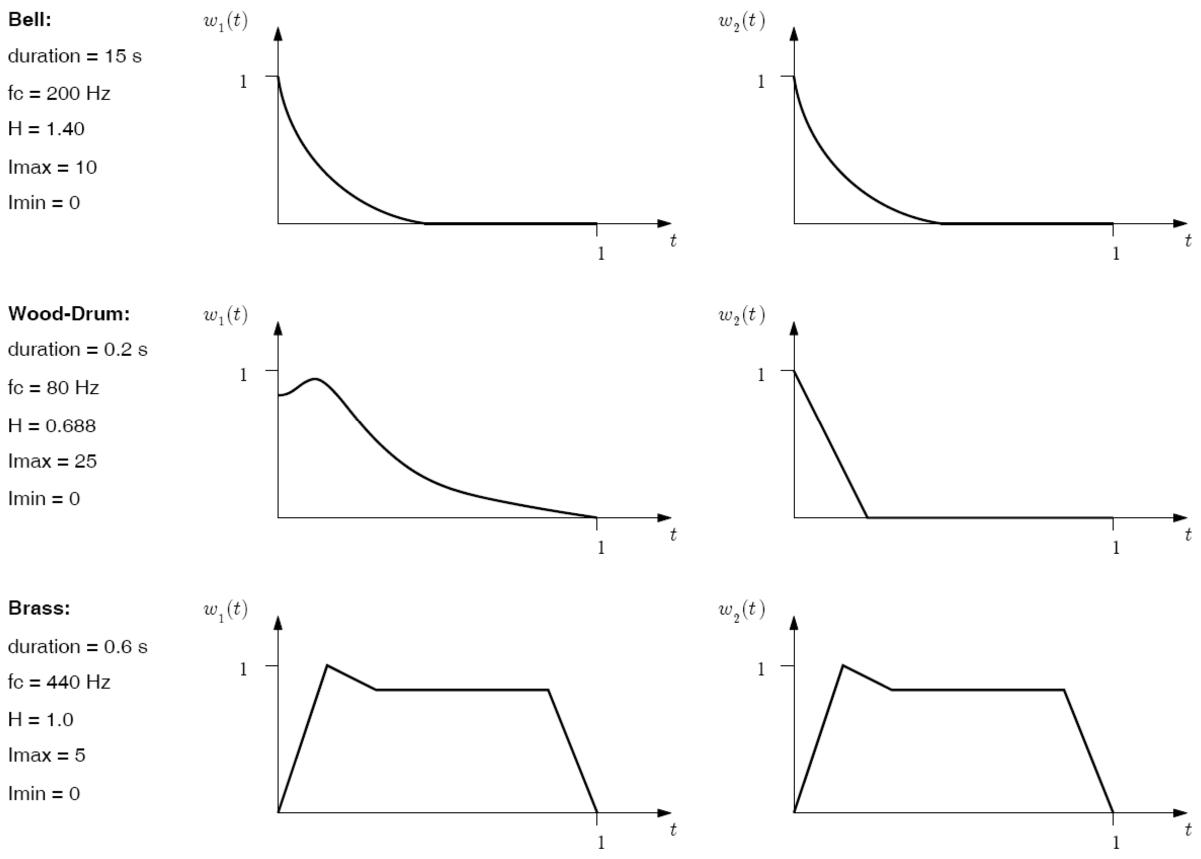
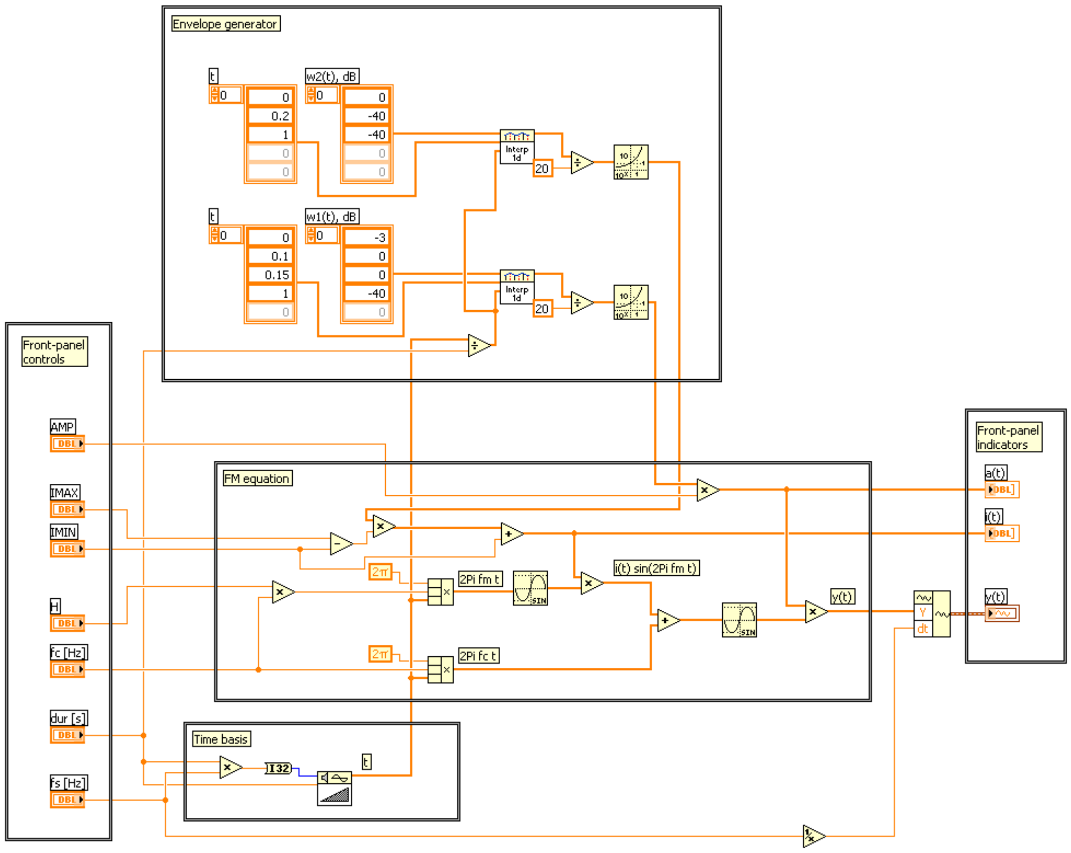
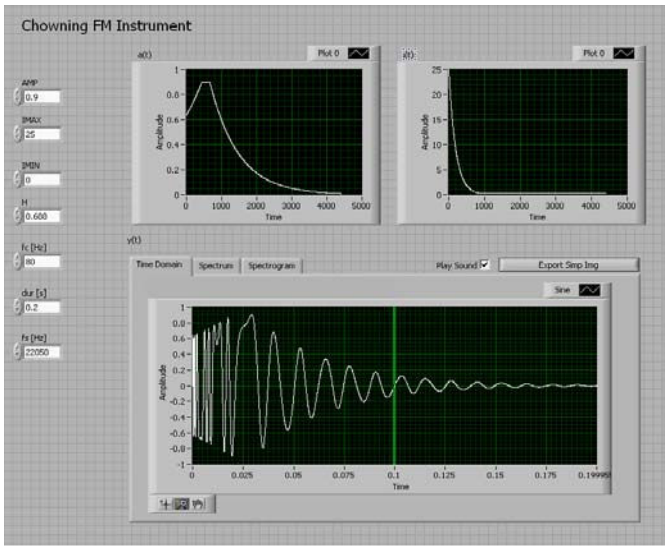


Figure 1: Chowning FM instrument parameter choices for three types of emulated instrument sounds. *H* denotes the harmonicity ratio fm/fc, and Imax and Imin indicate the modulation index maximum and minimum values, respectively. w1(t) and w2(t) depict the envelope shape for the amplitude envelope a(t) and modulation index i(t), respectively.



(a)

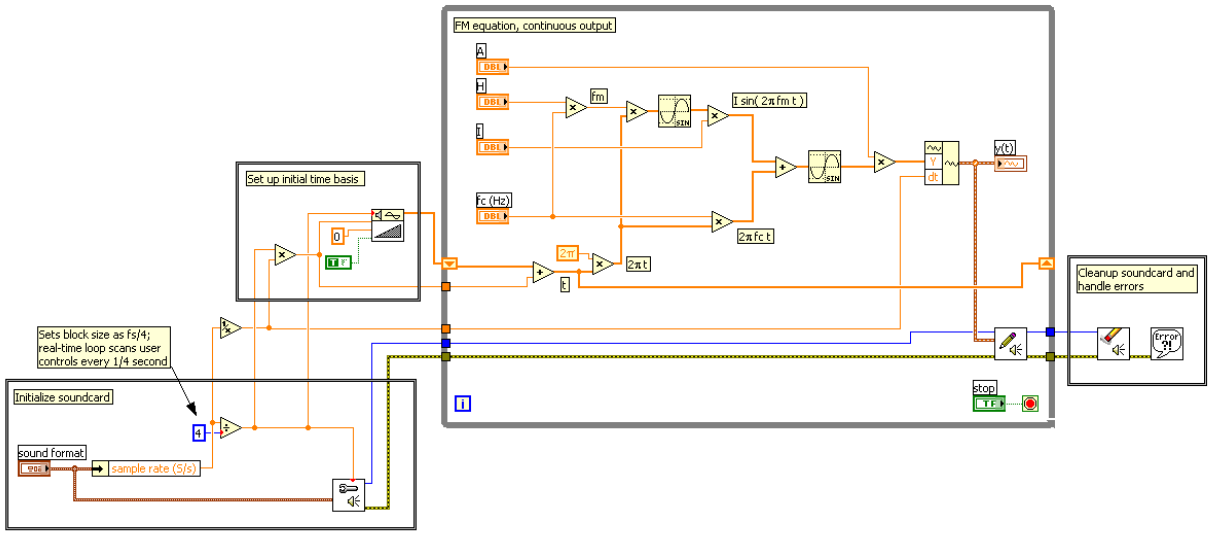


Chowning FM Instrument

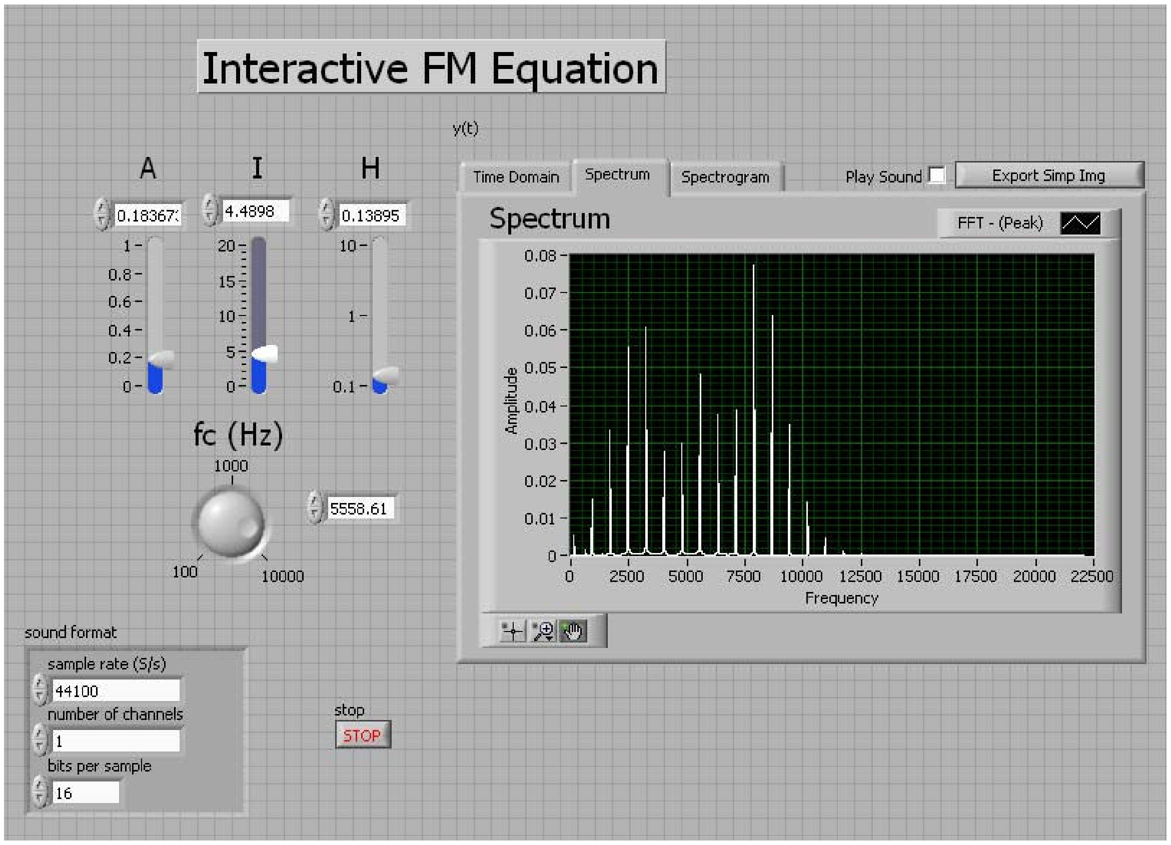
AMP 0.9
IMAX 25
IMIN 0
H 0.600
fc [Hz] 80
dur [s] 0.2
fs [Hz] 22050

(b)

Figure 2: LabVIEW implementation of Chowning FM instrument showing (a) block diagram and (b) front-panel controls configured for “wood-drum” instrument.


FM equation, continuous output
A
H
I
fc (Hz)
fm
I sin(2πfm t)
2πfc t
2π
2πt
t
y(t)
Set up initial time basis
Sets block size as fs/4; real-time loop scans user controls every 1/4 second
Initialize soundcard
sound format
sample rate (S/s)
Cleanup soundcard and handle errors
stop

(a)



(b)

Figure 3: LabVIEW implementation of interactive FM equation showing (a) block diagram and (b) front-panel controls. Sliders and a knob facilitate easy manipulation of the equation’s parameters.

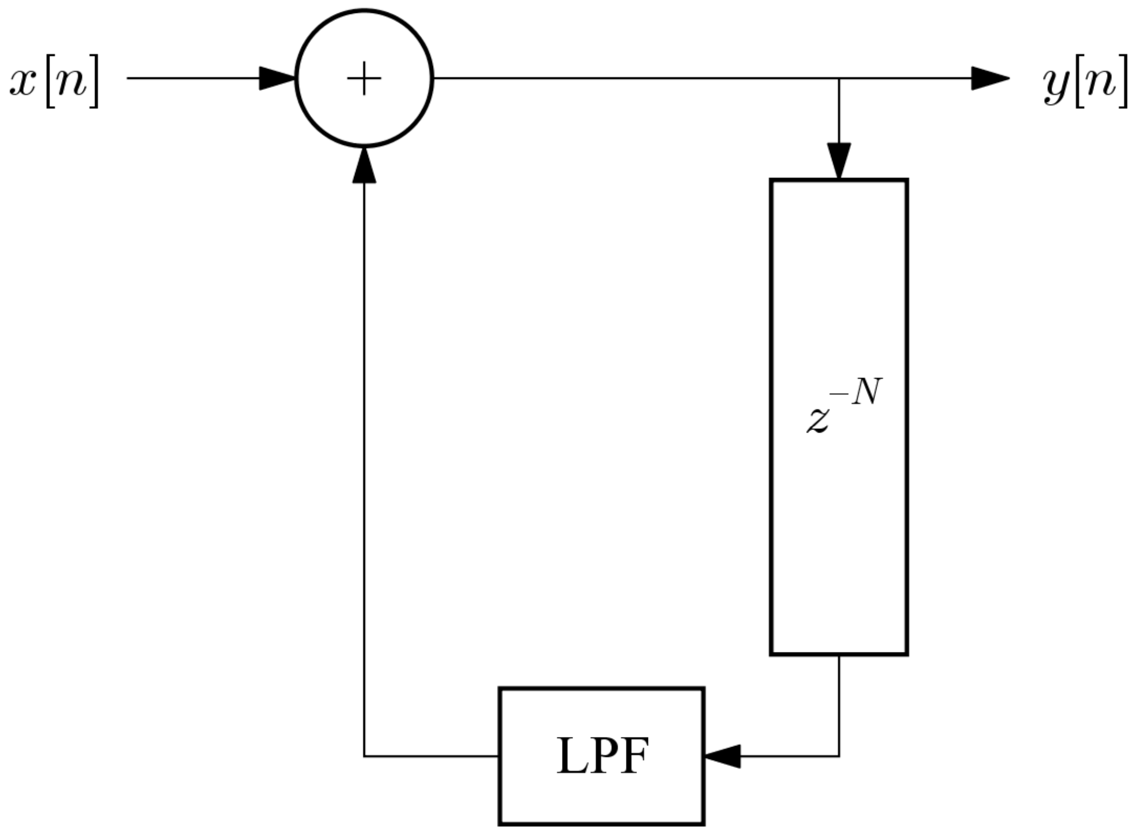
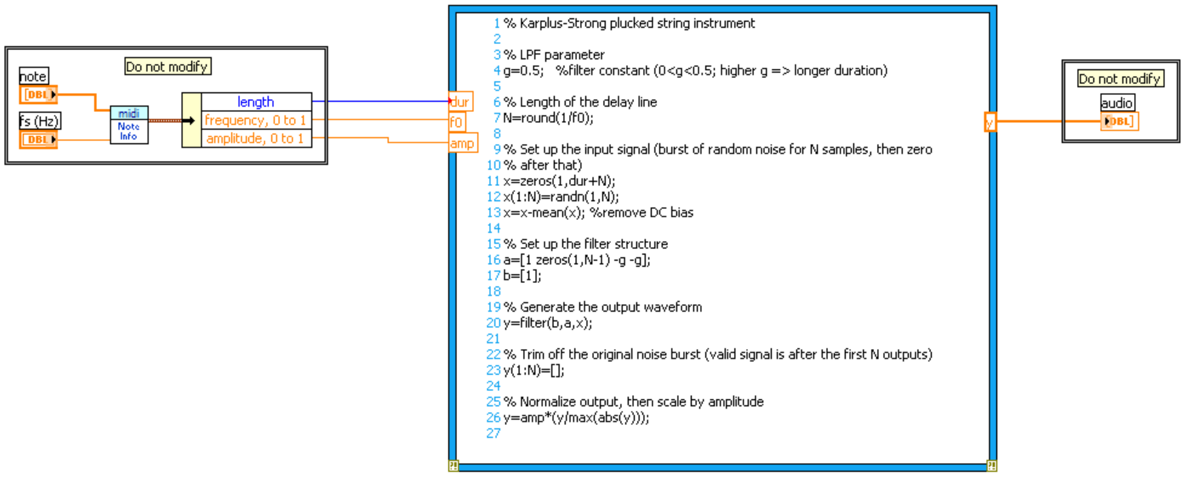
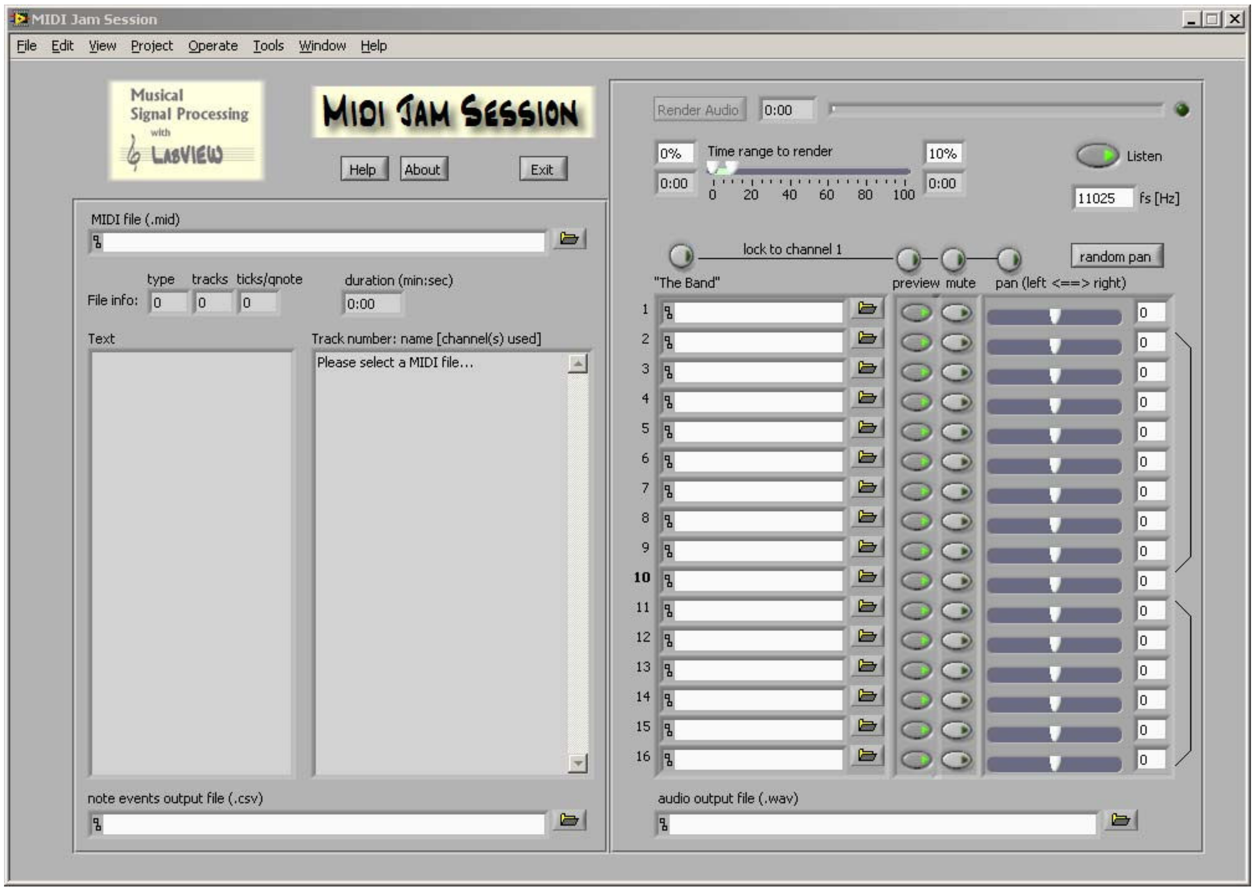


Figure 4: Signal flow diagram for Karplus-Strong plucked string algorithm. The input sequence *x*[*n*] is a burst of white noise (*N* samples) followed by zero to initialize the delay line with noise. The low-pass filter is a two-point running sum.



```
% Karplus-Strong plucked string instrument

% LPF parameter
g=0.5;   %filter constant (0<g<0.5; higher g => longer duration)

% Length of the delay line
N=round(1/f0);

% Set up the input signal (burst of random noise for N samples, then zero
% after that)
x=zeros(1,dur+N);
x(1:N)=randn(1,N);
x=x-mean(x); %remove DC bias

% Set up the filter structure
a=[1 zeros(1,N-1) -g -g];
b=[1];

% Generate the output waveform
y=filter(b,a,x);

% Trim off the original noise burst (valid signal is after the first N outputs)
y(1:N)=[];

% Normalize output, then scale by amplitude
y=amp*(y/max(abs(y)));
```

Figure 5: Karplus-Strong plucked string algorithm implemented by a MathScript node to create a "virtual musical instrument" (VMI) suitable for MIDI JamSession. The sections labeled "do not modify" appear in a standard VMI template and ensure proper connectivity to MIDI JamSession.



Please select a MIDI file...

Figure 6: “MIDI JamSession” front panel. Upon entering the MIDI file name in the upper left text-entry box, the application reads the MIDI file and extracts note and timing information. One or more “virtual musical instruments” (specially-formatted LabVIEW VIs) entered in the text-entry boxes on the right side specify the algorithms to be used to render the note and timing information to an audio waveform.

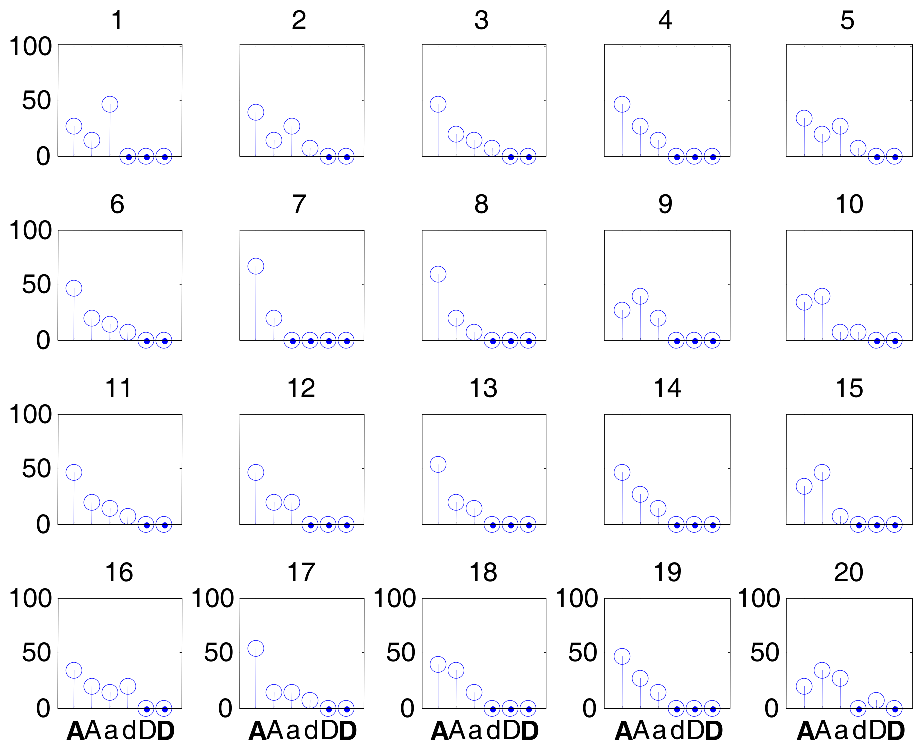


Figure 7. Student survey results for assessment questions. Response categories are **A** = strongly agree, A = agree, a = weakly agree, d = weakly disagree, D = disagree, and **D** = strongly disagree. Vertical scale is percentage response in each category.