

# ANIMATIONS AND GUIs FOR INTRODUCTORY ENGINEERING COURSES

Jordan Rosenthal<sup>1</sup>, James H. McClellan<sup>2</sup>

**Abstract** *Since 1994 we have developed and used a variety of multimedia adjuncts in an introductory Electrical Engineering course at Georgia Tech. Among the resources now available are animations and demonstrations that address nearly every major conceptual issue in the course. We have found it most convenient to present these tools to beginning students in the form of graphical user interfaces (GUIs) and animations programmed in MATLAB<sup>®</sup>. In this paper, we describe the features of the resources and the scenarios in which they have successfully been deployed.*

**Index Terms** *graphical user interface, animation, education, signal processing*

## INTRODUCTION

Since 1994 we have developed and used a variety of multimedia animations and demonstrations as an integral part of the entry-level signal processing course at Georgia Tech. We have found it most convenient to present these tools to beginning students in the form of graphical user interfaces (GUIs) and animations programmed in MATLAB.

The GUIs can serve a variety of purposes. First of all, they address nearly every major conceptual issue in the course, so we hope that students will use them to gain insights that are not possible from a printed page. In addition, we have found that instructors benefit as much as the students, because they can employ the GUIs as lecture aids that visually strengthen a student's connection with the material. The GUI tools are convenient because they allow the instructor to quickly pose alternate cases or respond to student questions. Animations and movies serve a similar purpose and even though they are less versatile than a GUI, they can be created more quickly.

Ideally, the GUIs would engage students in learning through interaction, but this is difficult to accomplish in all cases. We have had some success with GUIs that generate drill problems. However, one interesting finding is that students are reluctant to use these new tools when studying. Therefore, we have started to create homework and lab assignments that require active use of the GUIs to compare computer-generated results to analytical derivations.

In this paper, we will describe the features of a few GUIs and animations and the scenarios in which they have successfully been deployed.

## DEVELOPMENT USING MATLAB

GUIs that illustrate engineering concepts typically require an advanced numerical engine, a sophisticated plotting package, and professional user-interface capabilities. When choosing a computer language for developing GUIs, these requirements must be balanced with other issues such as the complexity of the programming language, the portability of the code produced, and widespread availability to other users. We have chosen MATLAB [2] as the environment for developing our GUIs and animations because we feel it combines many features attractive to both the developer and the user.

While developers want to produce professional products, they do not want to spend an inordinate amount of time writing code. These conflicting desires can only be satisfied if there exist large libraries of pre-written code from which the developer can draw. While code repositories exist for almost all languages, most engineering GUIs require the use of multiple libraries. For example, a single GUI can easily involve three libraries: one to do the back-end numerical calculations, a second to create professional quality plots, and a third to handle the user-interface aspects of the GUI. Tracking down the necessary libraries and integrating them into a cohesive application can require a substantial amount of time.

This time can be reduced if one develops the engineering GUI in a language such as MATLAB because the extensive library of engineering functions greatly simplifies the back-end programming. As a simple but important example, consider an application that needs to handle complex data, such as an application involving the Fourier transform. Because a complex number is a built-in datatype in MATLAB, there is nothing special a developer must consider. Coping with complex numbers in other languages, however, requires a significant amount of consideration.

Aside from the numerical libraries, MATLAB also has extensive graphical and user interface tools that rival the capabilities of most alternatives. Plots, animations, and GUIs can be quickly developed and they will run on any platform that can run MATLAB. While these capabilities exist in other languages, few integrate the numerical, graphical, and user-interface aspects as easily as MATLAB. With MATLAB, a developer can spend less time worrying about how a GUI is implemented, and more time thinking about what will make it a useful tool.

<sup>1</sup> Jordan Rosenthal, Georgia Institute of Technology, Atlanta, GA 30038 jr@ece.gatech.edu

<sup>2</sup> James H. McClellan, Georgia Institute of Technology, Atlanta, GA 30038 james.mcclellan@ece.gatech.edu

Another reason we chose to build our GUIs using MATLAB is that students in our curriculum are already using MATLAB for many courses. Thus, the students can use a single tool for all their work, and we can safely assume that our students either own a copy, or have easy access to, an installation of MATLAB.

One drawback to using MATLAB is that its GUIs are not as portable as GUIs written in a language like JAVA. For example, you cannot run a MATLAB GUI inside a web page. The code can still be distributed via the web, and because MATLAB files are just small text files, the code downloads quickly. When choosing a language for developing our GUIs, we felt the small reduction in portability was overshadowed by the superior way in which MATLAB integrates the different capabilities necessary to quickly develop an engineering GUI.

## GUI DESCRIPTIONS

In this section, we will discuss the details of five GUIs that we have developed recently [5].

### Convolution GUIs

Two GUIs have been developed to illustrate the principle of convolution. The first, CCONVDemo, targets the concept of continuous-time convolution while the second, DCONVDemo, targets the concept of discrete-time convolution. The two GUIs are operated in an identical manner — the only difference between the two being the domains from which the signals are drawn. Features of these GUIs include:

- Users can choose from a variety of different signals.
- Signals can be dragged around with the mouse with results displayed in real time.
- Tutorial mode lets students hide convolution results until requested.
- Various plot options enable the tool to be effectively used as a lecture aid in a classroom environment.

When either GUI loads, the user is prompted to choose the two signals that will be convolved together. The process of choosing a signal involves opening a dialog box in which the signal class can be chosen from a drop-down box, and then adjusting the signal's parameters using a set of on-screen controls. The GUI not only plots a graph of the signal, but it also provides the written formula that describes the signal. We believe that this juxtaposition of the visual and the written representations of the signal is essential in allowing a student to discover the connection between the analytical paper-and-pencil world and the hands-on world of numerical computing.

After initializing the GUI by choosing the two signals to be convolved, the user is presented with plots that describe the various components of convolution. Figures 1 and 2 illustrates the two GUIs in this state. Note the formula

legend in the lower right provides yet another link to the analytical world.

One of the trickiest parts of learning convolution is understanding the “flip and slide” viewpoint. The GUI supports this visualization by letting the user drag the signal around with the mouse. Thus, the user can change the sample for which the convolution output is being calculated. Another feature allows the user to quickly change which signal is being flipped, illustrating that convolution is commutative.

As mentioned before, these GUIs can be quite useful to a lecturer. The sliding nature of convolution is much easier to illustrate with the GUIs than with a static blackboard approach, and a large variety of examples can quickly be explored. In addition to explaining the convolution process, one can also illustrate ideas such as the transient response of a filter, or how filtering with a long pulse is related to integration.

To further aid the lecturer, specific features have been added that enhance the GUI for use in large classrooms. One menu allows the instructor to quickly increase the widths of all the plot lines to make the graphics more visible when displayed through an overhead projector. Another menu enlarges the three main plots to cover the entire figure while hiding the other elements from view.

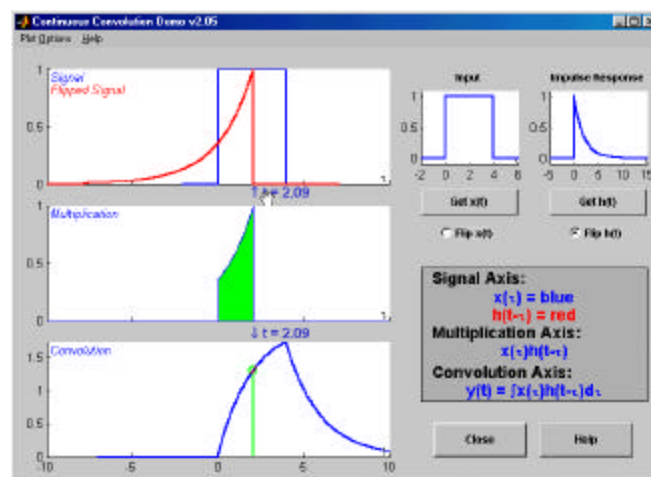


FIGURE 1  
CONTINUOUS CONVOLUTION DEMO

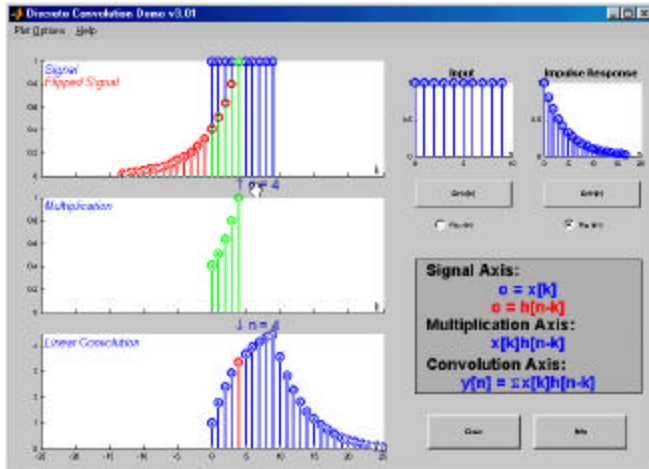


FIGURE 2  
DISCRETE CONVOLUTION DEMO

**LTIDemo**

LTIDemo is a GUI that illustrates the idea that when the input to an LTI system is a sinusoid of a particular frequency, then its output will also be a sinusoid of the same frequency, but with a possible change in its amplitude and phase. Figure 3 contains a screen shot of this GUI. The input sinusoid is shown on the left, the frequency response of the filter in the middle, and the output sinusoid on the right.

Using the controls in the lower left of the GUI, the user can control the parameters of the input sinusoid. These changes take effect immediately. As with the convolution GUIs, the formulas for the sinusoids are also given in their textual form to solidify the connection between the written and visual representations of the sinusoid.

Using the controls in the lower right of the GUI, the user can control the type of filter and its corresponding parameters. A number of preset filters are available from which the user can choose. These include both ideal and actual filters, thus allowing the instructor to move quickly between an abstract and realistic description of the filtering operation.

Other interesting topics that can be illustrated with this GUI, include:

- the frequency of the input sinusoid can be increased beyond the Nyquist frequency to illustrate aliasing
- the length of an averaging filter can be increased and its narrowing effect on the main lobe of the frequency response observed
- the effect of linear phase filters can easily be observed and related to the time delay of the output sinusoid

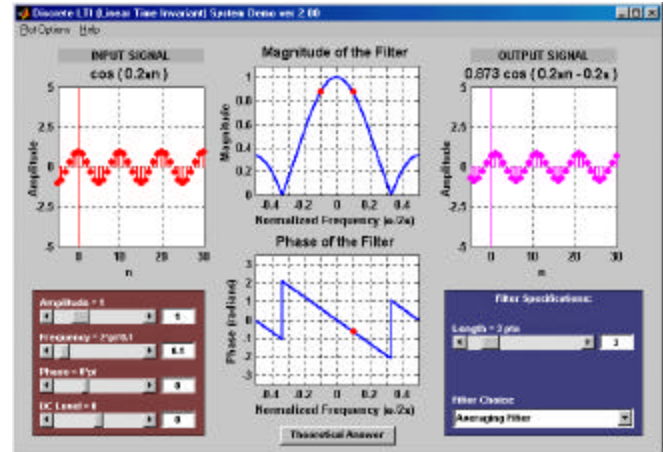


FIGURE 3  
DISCRETE LTI (LINEAR TIME INVARIANT) SYSTEM DEMO

**SinDrill and ZDrill**

SinDrill and ZDrill are GUIs that, as their names suggest, present students with drill problems. SinDrill tests the users ability to determine basic parameters of a sinusoid, while ZDrill tests the users ability to calculate the result of simple operations on complex numbers. Both GUIs can be set to either a beginner or an advanced level. The advantage for using a GUI for this type of problem is that while the number of drill problems found in the back of a textbook is limited, the GUIs can generate an endless supply of new problems.

SinDrill, shown in Figure 4, asks the user to determine the amplitude, frequency, and phase of a test sinusoid from a plot. Aside from the test sinusoid, the user's guess can also be displayed. This visual feedback is important because a student can quickly see how close their guess is to the correct answer.

Visual feedback is even more important in ZDrill (see Figure 5). In fact, one of the main purposes of ZDrill is to emphasize the vector view of a complex number. The student first chooses an operation that will be tested. Supported operations include: addition, subtraction, multiplication, division, inversion, and conjugation. After choosing an operation the user is presented with one or two complex numbers, depending on the operation involved. The numerical values of the complex numbers are displayed in text boxes, and are graphically drawn as vectors in the plane. The user must then provide the result of the operation. For example, in Figure 5, the user is being asked to find the sum of the two complex numbers that are drawn as vectors in the lower left axis. When the student enters a guess, the vector representing that complex number is drawn in blue in the lower right axis. Because the correct answer can also be shown, the student can visually determine how close the guess is to the correct answer. For example, the GUI shown in Figure 5 has its options set to show how the

two input vectors can be placed head to tail to find the overall sum and then displays this sum in black.

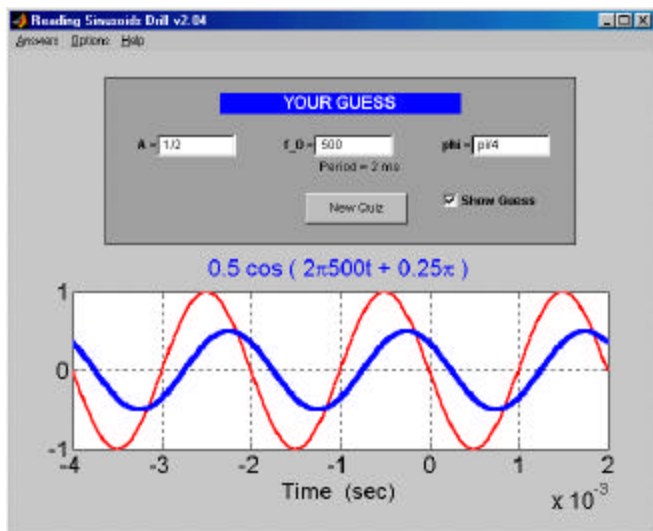


FIGURE 4  
READING SINUSOIDS DRILL

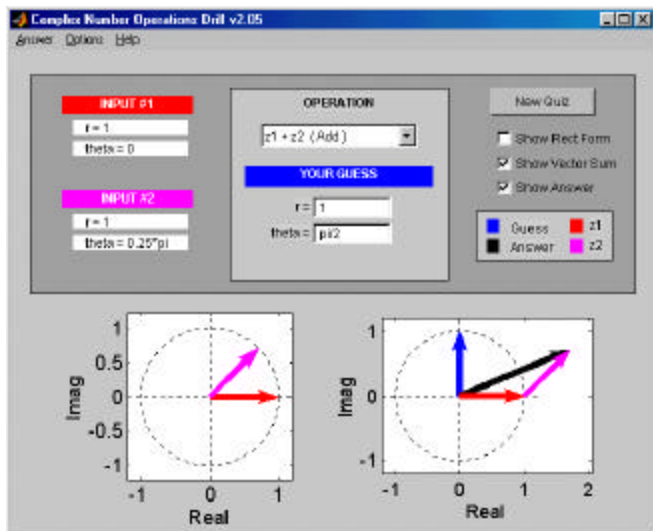


FIGURE 5  
COMPLEX NUMBER OPERATIONS DRILL

### ANIMATION DESCRIPTIONS

While GUIs can enhance the educational experience, the development time required is not always warranted. Often, complicated ideas can just as easily be explored using computer animations, which take substantially less time to produce. Figure 6 contains screenshots of a few selected animations that are included on the DSP First CD-ROM [3]

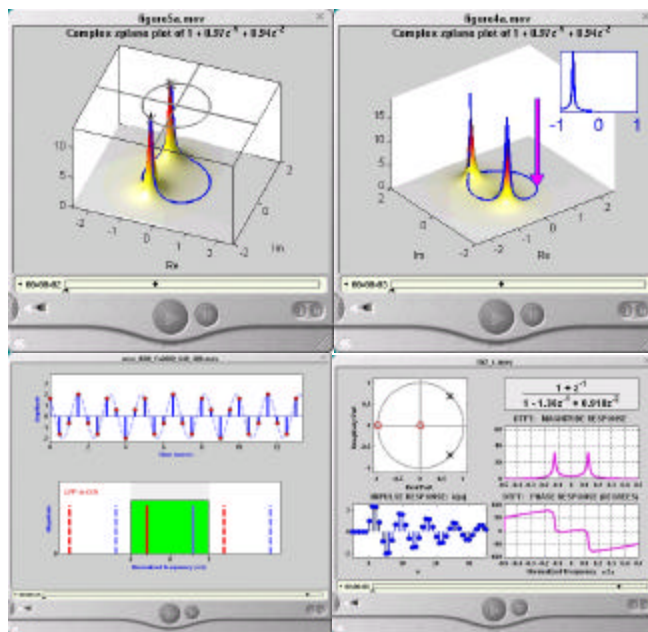


FIGURE 6  
FOUR DIFFERENT ANIMATION SCREENSHOTS

The animation in the top left of Figure 6 contains a three-dimensional view of a z-transform with the associated zero-pole plot superimposed at the top of the plot box. As the animation progresses, the view is rotated and the outline of the frequency response (in blue) peels off the unit circle. The animation in the top right of Figure 6 illustrates the same concept, but in a different way. An arrow traces out the unit circle and the frequency response gets drawn in the inset axes. The animation in the bottom left of Figure 6 illustrates the reconstruction process of the sampling theorem; while the one at the bottom right is a movie that shows the connection between the time domain, frequency domain and transform domain for digital filters. This movie is particularly useful for visualizing these connections, and is used by us in three or four different lectures.

These animations are typical of the kinds of animations one needs to create for an engineering class. Note that these would have been nearly impossible to create without a numerical engine such as MATLAB to do the complex calculations and generate the plots.

The animations shown in Figure 6 were created using Matlab. Scripts generated the movie frames and wrote them out in the QuickTime format [1]. At the time these animations were created, Matlab did not natively support the creation of movie files. It was, therefore, necessary to use a public domain function to write the QuickTime movies [4]. The latest version of Matlab, however, supports the direct creation of AVI movies, obviating the need for this function.

Most modern movie formats, including both the AVI and the Quicktime formats, allow the designer to choose from a wide variety of video compression codecs. The different codecs determine the nature of the compression algorithm used, and therefore directly affect the quality of the movie.

There is no one codec that will work well for every animation. For example, a codec optimized for animations in which most of the screen remains the same from frame to frame will not produce quality output for animations in which rapid changes occur. For the Matlab generated animations, we found that Apple's animation compressor or the compact video compressor (cinepak) both create good quality outputs at a reasonable size. The former was useful for animations with hard edges and distinct colors, such as that shown in the lower left of Figure 6. The latter was useful for plots with smoothly varying colors, such as the surface plots shown in the top of Figure 6.

When used during a lecture, the animations do not need to be played continuously from beginning to end. Using modern media players, it is possible to scroll through an animation frame by frame, or even rewind if necessary. This is a much more flexible way for an instructor to use the animations and point out specific behavior.

**ASSESSMENT**

Students have taken surveys in this class for each for the past five years. The results, shown in Figures 7-9, were taken at the end of the Fall semester, 2000. The students' responses to these questions are typical of our experience over a longer period of time. First of all, we see that a majority of the students agree that the computer demos aided understanding of the material. The question in Figure 7 refers to instructor usage of the demos during a lecture, so this indicates passive acceptance by the students. The perceived utility is a little bit less when the question is posed independent of classroom usage, as shown in Figure 8. On the other hand, Figure 9 shows clearly that students are not naturally attracted to the demos. They do not actively use the demos in conjunction with their other learning activities. More than likely this is because most of the learning tasks are traditional, e.g., homework problems, lab reports, etc.

**Question: 15. Class computer demos**

The computer demos in class/recitation helped me understand the concepts being taught.

- 1. Strongly agree
- 2. Agree
- 3. Do not agree or disagree
- 4. Disagree
- 5. Strongly disagree

**Response Summary**

Answer	Value	Frequency Distribution
1	0%	18
2	0%	110
3	0%	62
4	0%	14
5	0%	7

FIGURE 7

STUDENT REACTION TO LECTURE PRESENTATION OF GUIs

**Question: 6. CDROM/Web demos aided understanding**

The demos on the DSP-First CDROM/Web page helped me understand the concepts being demonstrated.

- 1. Strongly agree
- 2. Agree
- 3. Do not agree or disagree
- 4. Disagree
- 5. Strongly disagree
- 6. I did not view them

**Response Summary**

Answer	Value	Frequency Distribution
1	0%	11
2	0%	63
3	0%	54
4	0%	18
5	0%	3
6	0%	53

FIGURE 8

STUDENT LEARNING FROM DEMOS AND GUIs

**Question: 16. Personal viewing of computer demos**

I viewed computer demos used in class later via the web or the DSP-First CDROM.

- 1. Almost always
- 2. Frequently
- 3. Occasionally
- 4. Seldom
- 5. Never

**Response Summary**






Answer	Value	Frequency Distribution
1	0%	4 
2	0%	17 
3	0%	56 
4	0%	65 
5	0%	68 

FIGURE 9  
STUDENT USAGE OF DEMOS AND GUIs

Therefore, we have a challenge to improve this situation and to promote active use of these visualization tools. In order to get the students more involved, we are now designing two labs that will require dual usage: run the GUIs to generate observations and then perform a companion analysis to explain the observations with theoretical results. We hope that this effort will increase every student's active participation in using these visualization tools.

**REFERENCES**

- [1] Apple Computer, Inc., 1 Infinite Loop, Cupertino, CA, *QuickTime*, Available: <http://www.quicktime.com> [2001, May 21].
- [2] The Mathworks, Inc., 24 Prime Park Way, Natick, MA, *MATLAB*, Available: <http://www.mathworks.com> [2001, May 21].
- [3] McClellan, J. H., Schafer, R. W., Yoder, M. A., *DSP First: A Multimedia Approach*, Prentice-Hall, Inc., Upper Saddle River, NJ, 1998.
- [4] Slaney, Malcolm, "MakeQTMovie – Create QuickTime movies in Matlab", Interval Technical Report, 1999-066, 1999. Available: <http://rvl4.ecn.purdue.edu/~malcolm/interval/1999-066/> [2001, May 21].
- [5] Rosenthal, J., *Educational MATLAB GUIs*, available at <http://users.ece.gatech.edu/mcclella/matlabGUIs> [2001, June 1].