The Signals and Systems Toolbox: Comparing Theory, Simulation and Implementation using MATLAB and Programmable Instruments

John M. Spinelli
Union College

Abstract

A software system to facilitate rapid comparison among theoretical models, simulations, and implementations of signals and systems can help engineering students develop physical intuition and an understanding of the capabilities and limitations of each. Using programmable instruments in laboratory experiments can improve the efficiency and accuracy of such comparisons. MATLAB and SIMULINK already provide students with easy methods to model and simulate systems and to specify arbitrary input functions. This paper describes a toolbox of MATLAB functions, called the "Signals and Systems Toolbox" that can automatically apply a specified input to a physical system using an arbitrary waveform generator and then use a programmable oscilloscope to measure the resulting output. A comparison of the simulated versus actual response of the system can then be performed. Other more specialized functions allow comparison between the frequency response of a model and that of an actual system by stepping through a desired range of frequencies and measuring the response. Basic functions allow easy generation and manipulation of complex signals. The toolbox is well suited to introductory laboratory courses since all students need to know is how to connect the input and output correctly. More advanced students can use a variety of software options to improve the accuracy or usefulness of the input/output comparison.

The software system is designed to work with commonly available HP (Agilent) laboratory equipment, but it can be easily modified to use programmable instruments from other manufacturers. It is available for downloaded via the Web.

1. Introduction

In introductory courses in Electrical Engineering instructors must decide on the appropriate balance between theory, simulation, and experimentation. In recent years, the availability of sophisticated and easy to use software such as MATLAB and Electronics Workbench has led to an increased reliance on simulation at many schools. At Union College, we have sought ways to emphasize meaningful physical experiments in our introductory courses in Electrical and Computer Engineering because we believe that they are essential for the development of physical intuition. The availability of inexpensive programmable instruments from manufactures such as Agilent and Tektronix together with support for instrument communication by software packages such as MATLAB provides an opportunity to bridge the gaps between theory, simulation, and experiment.[2]

Exploiting the capabilities modern test equipment and software in an introductory or lower level engineering class requires a software interface with the following special properties.

Simple: The interface should be very simple to use and should provide common defaults wherever possible so that students who have yet to master signal or system concepts are not discouraged or confused.

Flexible: The interface should be flexible and configurable enough to be of continued use after students have mastered basic concepts.

Familiar: The interface should use notation that is as similar as possible to what students will see in standard introductory textbooks.

Because MATLAB in commonly used in introductory courses for simulation and analysis, it is the natural choice for adding an interface to programmable instruments. Other software packages, such as LabView, have more sophisticated methods of communicating with instruments, but this level of sophistication is not needed in introductory classes. Minimizing the number of different software packages to which students are exposed in introductory classes and laboratories avoids “software overload” and allows them to develop some level of confidence with and mastery of a small number of software packages.

2. Creating and Downloading Input Functions

In many introductory linear systems textbooks [1, page 69], students are frequently exposed to simple piecewise functions such as the one shown in Figure 1. These are often used as inputs to systems models since the output they produce can often be calculated manually.

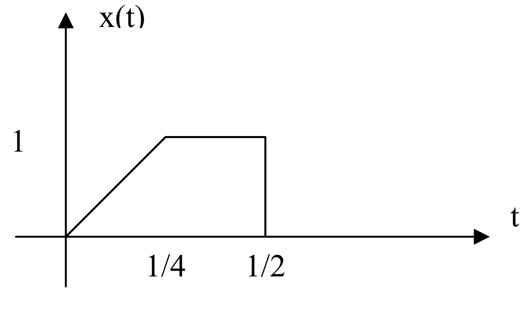

Figure 1: A simple piecewise function.

By using the unit step function, u(t), x(t) can be expressed algebraically as:

$$x(t) = 4t\left[u(t) - u(t - \tfrac{1}{4})\right] + \left[u(t - \tfrac{1}{4}) - u(t - \tfrac{1}{2})\right]$$

The Signals and Systems toolbox provides a simple MATLAB version of the unit step function, called `u()`, to allow a similar notation to be used. Since we wish to have actual data for this function in MATLAB, a time axis, t, must first be defined with the desired range and a sufficiently large number of points to faithfully represent the signal. The following commands may be used to define a MATLAB version of x(t) ranging from 0 to 1 second and evaluated at 1000 points.

```
t = linspace(0,1,1000);
x = 4*t .* [u(t) - u(t - 1/4)] + [u(t - 1/4) - u(t - 1/2)];
```

The similarity between the algebraic and MATLAB forms helps student to rapidly define and manipulate functions.

Although the above signal is simple, it is not one of the standard waveforms typically available in the laboratory. The Signals and Systems toolbox function `darb()` can be used to download signal x(t) to the hp33120A function generator:

```
darb(t, x);
```

For convenience, a periodic version of x(t) is produced, but a single shot may be selected as well. This gives students the immediate ability to easily produce in the laboratory nearly any type of signal that they see represented in textbook or classroom examples. Limitations of the hp33120A require that the signal have between 8 and 16,384 points, amplitude between -10 V to +10V, and frequency less than 15 MHz. While signals such as x(t) have no important practical application beyond instruction, it is helpful for students to be able to verify experimentally examples that they see worked out in class and in their text. The output of the function generator when producing x(t) is described in Section 3.

2.2 Generating "Real Signals"

In addition to allowing easy generation of "textbook" signals, the Signals and Systems toolbox can also be used to produce laboratory versions of complex waveforms such as audio. While this can also be achieved using soundboards, programmable instruments allow easier data gathering and analysis.

The following MATLAB commands take one second of audio from the MATLAB sample audio file "gong" and download it to the function generator. Triggering on this waveform is accomplished by using the "sync" output of the generator which produces a pulse each time the waveform repeats (once per second in this example). This sound file uses a sample rate, Fs, of 8192 samples per second, so up to two seconds could be held in the arbitrary waveform generator's memory. The Signals and Systems Toolbox provides a function `timeaxis()` that takes a time duration and a sample rate and returns an appropriate time axis.

```
% load the sample "gong" sound,
% this defines sample rate Fs and sound samples y
% "gong" is part of the standard MATLAB distribution
load gong

% define a time axis for 1 s of audio at a sample rate Fs
t = timeaxis(1, Fs);

% select only the first one second of audio samples
y = y(1:length(t));

% download the waveform to the function generator.
darb(t, y);
```

Once the sound file has been downloaded to the generator it can be listened to directly by the students (using headphones) and can also be used as the input to systems under study in the laboratory. Students may also record their own audio using the PC sound board or take a short clip from their favorite CD. The output produced when passing such a signal through a system can be gathered and analyzed using the functions described in the next section.

Using standard MATLAB operations such as random number generation and trigonometric functions, it is easy to generate audio signals that are corrupted by a specified amount of noise or to modulate, mix, or combine signals in arbitrary ways. This provides enormous flexibility for laboratory demonstrations and experiments.

3. Gathering Output Waveforms

Once a signal, has been displayed on an oscilloscope, the Signals and Systems toolbox provides a simple interface to transfer waveform data into MATLAB. The function `swave()` takes a specified oscilloscope channel and returns the displayed waveform. The function is designed to work with the Agilent (formally HP) 54xxx family of oscilloscopes, but can be easily modified to work with the scopes of other manufacturers such as Tektronix.

Assume that the function x(t) that was downloaded to the arbitrary waveform generator in Section 2 is displayed on channel 1 of a supported oscilloscope. The waveform, called sx with a time axis st, can be transferred to MATLAB using the `swave()` function of the Signals and Systems toolbox. The argument to `swave` specifies the selected channel.

```
[st, sx] = swave(1);
```

Figure 2 shows a comparison of this signal to x(t) generated in section 2. In a similar manner, a portion of the signal audio signal y(t) generated in section 2 may be transferred to MATLAB. A comparison of this signal with the corresponding portion of the "gong" audio file is shown in Figure 3. It can be seen that the arbitrary function generator does a reasonably good job of reproducing both simple and complex waveforms.

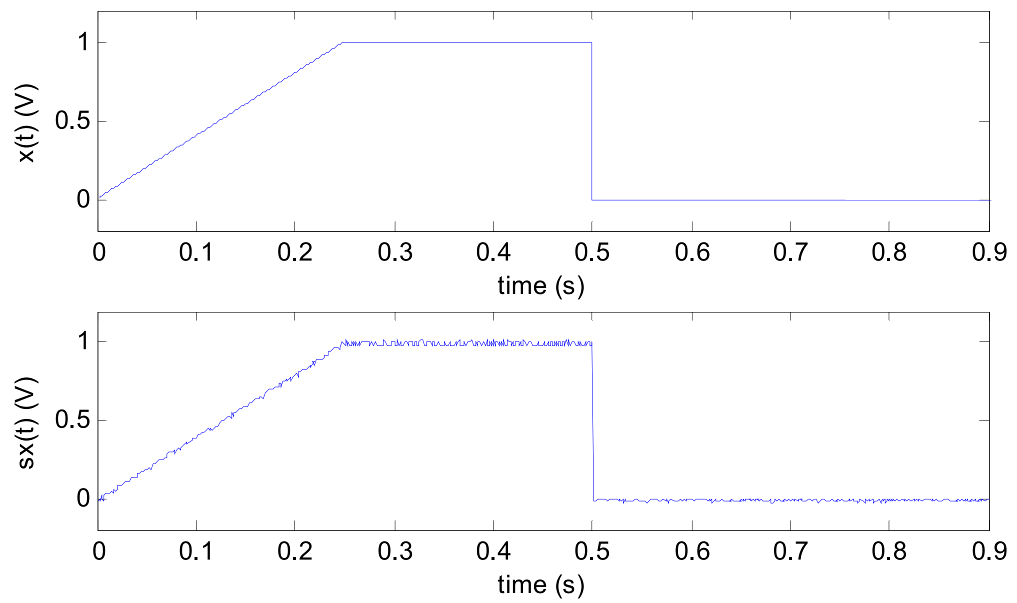

Figure 2: Top: MATLAB representation of x(t).
Bottom: trace of x(t) read from the oscilloscope.

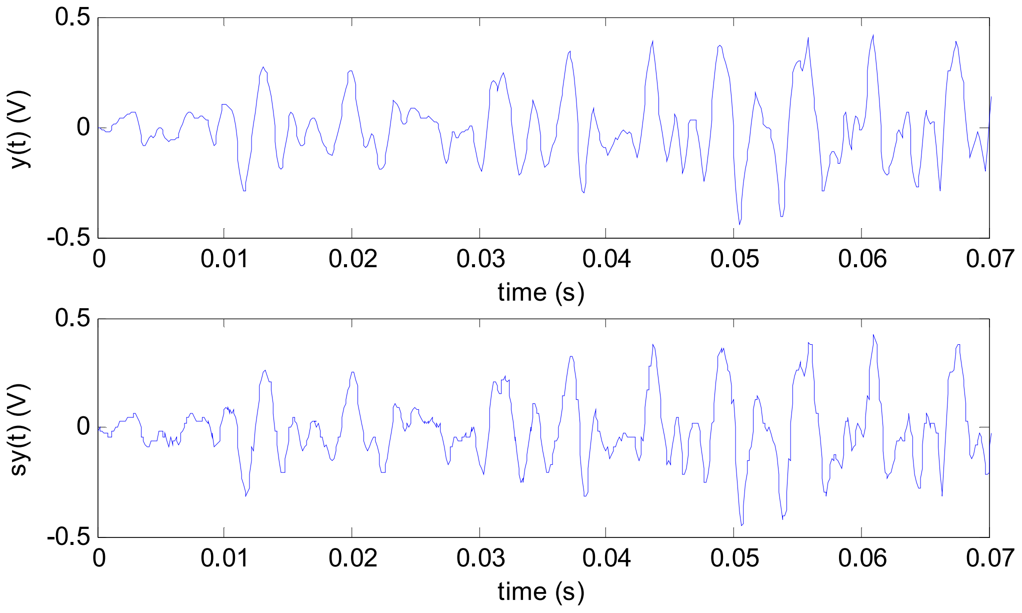

Figure 3: Top: MATLAB representation of y(t).
Bottom: trace of y(t) read from the oscilloscope.

3.1 Quantization Issues and Averaging

Examination of Figures 2 and 3 shows a fair amount of quantization noise in the signals transferred from the oscilloscope. This is due to the crude 8 bit vertical resolution of the scope, and can be somewhat reduced by gathering several copies of a given waveform and averaging them within MATLAB. The Signals and Systems Toolbox provides a function `savgwave()` which gathers a specified number of copies of a waveform and averages them. By default, 10 copies are averaged. The use of this function, as an alternative to swave, greatly improves the comparison between simulated and experimental waveforms.

4. Frequency Domain Analysis

MATLAB provides a wealth of functions to perform frequency domain analysis of signals, but analyzing a physical system can be somewhat more involved. To aid in understanding systems in the frequency domain, the Signals and Systems Toolbox provides a function `sfreq()` that generates the magnitude and phase frequency response of a system under test by programming the arbitrary waveform generator to send a sequence of sinusoids at various frequencies into a system, and then uses the oscilloscope to measure the amplitude and phase of the resulting outputs. The data gathered is then presented as magnitude and phase response graphs. The frequency range and increment are user controlled. While this function is no replacement for a high quality spectrum analyzer, it allows a simple and rapid exploration of the frequency response of a system in an introductory laboratory. The results obtained may be compared to system bode plot models to determine how well a systems conforms to its model or design specifications. This function has been used extensively in the laboratory experiments described in [3].

5. Time Domain Analysis

The Signals and Systems toolbox provides a number of other functions to aid in the time domain analysis of systems. All operate of pairs of vectors such as (t, x) which define an approximation of a continuous signal x over a time vector t. While the details are beyond the scope of this paper, listed below are the names and purposes of some of these functions.

`ct_conv()`:	Provides "continuous time" convolution of two waveforms defined on the same time axis.
`ct_diff()`:	"Continuous time" derivative.
`ct_int()`:	"Continuous time" integral
`smooth()`:	Zero-phase distortion low pass filtering used to remove high frequency noise

6. Conclusions

Versions of the Signals and Systems toolbox have been used for several years in both the classroom and laboratory portions of an introductory systems course at Union College. Student assessment has been quite favorable, and a number of students have used parts of the toolbox independently in later laboratory and project classes. The toolbox may be freely downloaded at http://grinch.union.edu/spinelli/SST. Using it requires a MATLAB license that includes the Signal Processing and Instrument Control toolboxes.

Bibliography

1. B.P. Lathi, "Signal Processing and Linear Systems," Berkeley-Cambridge Press, 1998.
2. David McDonald, et al., "Improving the Laboratory Experience with Modern Computer-Based Instrumentation," Proceedings of the 1997 ASEE Annual Conference and Exposition, Session 1559.
3. John M. Spinelli and Kevin LaFerriere, "A Discovery Based Systems Laboratory using LabView and MATLAB," Proceeding of the 2000 ASEE Annual Conference and Exposition, Session 2532.

JOHN SPINELLI
John M. Spinelli is an Associate Professor in the department of Electrical and Computer Engineering and the department of Computer Science at Union College, Schenectady, New York. He teaches in the areas of linear systems, digital communication and computer networks, and does research on fault-tolerant communication protocols. He received the B.E. degree (summa cum laude) in electrical engineering from The Cooper Union, New York, in 1983, and the S.M. and Ph.D. degrees in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology, Cambridge, Massachusetts, in 1985 and 1989, respectively.