PRACTICAL MATLAB EXPERIENCE IN LECTURE-BASED SIGNALS AND SYSTEMS COURSES

Peter Milder and Mónica F. Bugallo

Department of Electrical and Computer Engineering
Stony Brook University, Stony Brook, NY 11794-2350
{peter.milder,monica.bugallo}@stonybrook.edu

ABSTRACT

In this paper we report our efforts to streamline the curriculum of a lecture-based course on signals and systems with exercises using the Matlab computing environment. We use a computer framework to generate individualized variations of problems, which are assigned to teams of students as well as to individual students. Feedback from students revealed that the new components were helpful for better understanding of the materials and hold strong promise in our new approach to interactive and hands-on learning. Furthermore, we discuss an auto-grading system that will provide students with instantaneous feedback and ease the task of evaluating projects in the next offering of the course.

Index Terms— Hands-on exercises, signals and systems, Matlab, computer-generated problems, auto-grading

1. INTRODUCTION

At Stony Brook University, Deterministic Signals and Systems is a traditional introduction to signal processing, covering continuous- and discrete-time signals and systems in the time and frequency domains. This is a required course for undergraduate students majoring in Electrical Engineering (EE) or Computer Engineering (CE). In Fall 2015, 108 students enrolled in the course, as described in Table 1.

Traditionally, this course has been taught at Stony Brook as a lecture-only course, without a lab or programming component. This format has offered students the possibility of gaining a thorough theoretical understanding of the material, but various elements of the curriculum proved difficult to understand without practical exercises. Moreover, with the current lecture-only offering, hands-on practical skills such as Matlab were pushed to later courses such as communication systems or DSP, which burden those courses as students entered without sufficient experience using computer tools and technologies for signal processing and related areas. Finally, many similar courses at other institutions offer computer lab components associated with the lectures [1, 2, 3, 4].

This work has been supported by the National Science Foundation under Award CCF-1617986 (M. F. Bugallo).

Year		Major	
Sophomore	34	EE	59
Junior	41	CE	29
Senior	30	Biomed. Eng.	7
Other	3	Other	13

Table 1. Student breakdown for Fall 2015.

Motivated by these reasons, we sought to include experience with Matlab in the form of basic exercises as a first step toward including a permanent lab component. The goal was to provide students with practical hands-on experiences that could be done in a largely self directed manner, to keep within practical constraints of the course. We accomplished this through a combination of careful design of assignments and the use of technology.

In this paper, we describe our contributions to this effort. First, we created a new set of original Matlab exercises associated with the theoretical course topics. Second, we built a computer framework to automatically generate unique variations of lab problems for students. Third, we are prototyping an automatic grading system that will provide students with instantaneous feedback on their work and will ease the evaluation process.

The rest of this paper is organized as follows. Section 2 describes the new lab components, as well as our mechanism for providing computer-generated problems and an automated grading system that we are developing. In Section 3, we discuss student performance and feedback, and we explain how we are correspondingly adapting these activities in future offerings. Lastly, Section 4 offers concluding thoughts.

2. NEW COURSE DESIGN

This section describes how we structured and supported assignments, as well as the technological aspects of our approach: automatically generating unique problem instances and an ongoing effort to build an automated system to evaluate student work and provide feedback.

2.1. Hands-on Labs and Assignments

The Fall 2015 semester marked our first trial of the new course material. We created four assignments, which students were encouraged to complete in groups of two.

Labs 1 and 2: Tutorial. First, we began with two tutorial assignments designed to introduce students to Matlab within the first two weeks of the course. These were the only assignments structured as a traditional lab, when students were required to attend a specific session. For each week, we created explanatory documents covering relevant topics and presenting simple exercises. In week 1, we covered: how to interact with Matlab, simple mathematics, arrays and matrices, plotting signals, and the use of functions and scripts. Week 2 covered: the representation of signals as arrays in Matlab, writing scripts to generate signals, and working with complex numbers. In each of the first two weeks, students signed up to attend a 90 minute session where they would complete the assignment under the supervision of course staff.

Lab 3: Signal Transformation. Next, students were given an assignment to reinforce concepts related to the mathematical manipulation of signals and system properties. First, students read a short tutorial providing examples for writing code for time-based transformation of signals; this discussion connected the topic closely to concepts from class. Then, we provided each team of students with a different signal $x[n]$ and a mathematical description of a system. (The specification of the unique signal and transformations were automatically generated using the framework described below in Section 2.2.) Each team was tasked with writing a Matlab script to generate $x[n]$, manipulate it based on the system described, and plot the results. Lastly, each team demonstrated that their system was not time invariant with a counter-example.

Lab 4: LTI Systems. The final assignment related to discrete-time linear time invariant (LTI) systems. This reinforced knowledge of the DTFT, convolution, and the concept of the frequency response of a system. In this assignment, students were given formulas specifying a discrete-time signal $x[n]$ and the impulse response $h[n]$ of a discrete-time LTI system. Students were tasked with examining the input and output of this system in both the time and frequency domains. First, they wrote code to generate $x[n]$ and $h[n]$ signals based on mathematical specifications. Then, they used convolution to compute the system's output in the time domain. Next, they used the DTFT to examine the frequency representation of the signals, and to verify that taking the DTFT of the output signal was equal to taking the product of the DTFT of the input and the system's frequency response. As in Lab 3, each team of students was provided with a unique instance of the problem: in this case, a unique $x[n]$ and $h[n]$.

2.2. Computer-Generated Problems

To facilitate self-directed Matlab assignments, we developed a framework to enable automated generation of unique instances of Matlab problems, along with their solutions. This allows each team of students to solve a different variation of the problem than other classmates solved, and it can give students the ability to continue to practice by generating new problems. Because we additionally generate the solution (e.g., the final graph of the result the student's function should create), instructors can choose to distribute solutions (allowing students to work toward the correct output) or keep them for grading. Alternatively, an instructor could take advantage of the automated nature of the system and create several problems per student, distributing some with solutions (for practice) and some without (for evaluation).

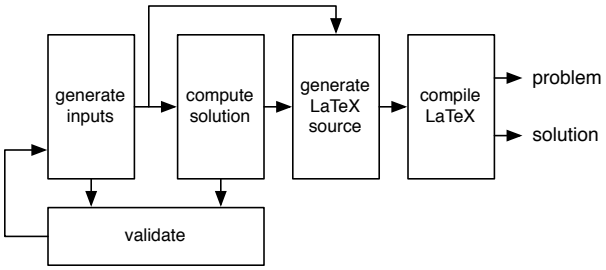

Fig. 1. Template-based framework for automatic generation of Matlab problems.

Figure 1 shows a high-level view of our problem generator framework. The system is based on templates—to create a new assignment, the instructor must provide a function for each step of the framework. The modular nature of this system means it is easy to develop or share a library of different assignments. First, the system generates a set of inputs that will be used for a particular instance of this problem. The instructor provides a list of variables to generate as well as constraints on the randomization process (e.g. "a should be a random integer between -10 and 10, and b should be no more than 4 above or below a"). Based on the specification, our system uses Matlab's built-in random number generation functions to set variables' values. A set of random variables represents a specification of one customized instance of the given assignment.

Next, the system computes the problem's solution; this represents the work that the student will when completing the assignment. To accomplish this, the instructor provides the framework with a function that takes as input the variables described above and solves the desired problem. Additionally, the instructor is able to specify optional validation functions that will check the inputs or solution. (For example, one can specify checks to ensure that if the solution becomes trivial or uninteresting, a new set of input parameters will be created.) This is implemented simply as a series of conditional statements the instructor can provide.

Lastly, the values of parameters (and any desired Matlab graphs) are plugged in to an instructor-provided LaTeX template, which is compiled to a PDF. Our framework en-

parameter(s)	potential values
a, b, c	$10, 11, \ldots, 20$
ω_1	$\pi/10, \pi/5$, or $3\pi/10$
ω_2	$\omega_1 + (\pi/10, \pi/5$, or $3\pi/10)$
ω_3	$\omega_2 + (\pi/10, \pi/5$, or $3\pi/10)$
g, h	0 or 1

Table 2. Parameters generated for example assignment.

ables this by automatically generating LaTeX macros for the random variable values; the instructor then writes a template that uses those macro names. The system can generate separate PDFs for the problem and its solution, or it can combine them into a single document. Additionally, functionality is implemented to allow the instructor to provide a list of names or identification numbers; for each item in the list, a unique problem and solution will be produced, with the identifier optionally included at the top of the document and in the generated file name.

Example. To illustrate, we describe how this system functions for the Lab 4 assignment discussed in Section 2.1. In this assignment, each team is given specifications of an input signal $x[n]$ and the impulse response of an LTI system $h[n]$. Students are asked to find and plot time domain signals $x[n], h[n]$, and $y[n]$ (the output of the system), as well as frequency domain representations $X(e^{j\omega}), H(e^{j\omega})$, and $Y(e^{j\omega})$, using the discrete-time Fourier transform. To create an instance of the problem, we first generated constrained random numbers according to the constraints in Table 2. These specifications are added into the system in the form of a Matlab function. Then, the signal $x[n]$ is generated according to

$$x[n] = a\cos(\omega_1 n) + b\sin(\omega_2 n) + c\cos(\omega_3 n).$$

This results in a signal with three frequency components ω_1, ω_2, and ω_3. Then $h[n]$ is chosen as the impulse response of a low-pass or high-pass filter (depending on the value of g) with cutoff-frequency halfway between ω_1 and ω_2 or halfway between ω_2 and ω_3 (depending on the value of h).

For each problem instance, a set of values is generated; this set is passed to code that performs the tasks students are asked to accomplish. In this example, that means generating Matlab representations of the signals $x[n]$ and $h[n]$ based on the specification, using Matlab's convolution function to compute the system's time-domain output $y[n]$, using a DTFT to compute the frequency representations of each signal, and creating plots for these six signals following a prescribed format. We provide a Matlab function that the "compute solution" block uses to perform these steps and generate the results. All parameters and results are then sent to a block that produces the LaTeX source by connecting these values to a template that we created. This step additionally produces image files of the solution graphs which are embedded in the

$$x[n] = 13\cos(0.7\pi n) + 15\sin(-0.6\pi n) + 18\cos(0.3\pi n)$$

$$h[n] = \begin{cases} \frac{\sin(0.65\pi n)}{\pi n}, & \text{if } n \neq 0 \\ 0.65, & \text{if } n = 0 \end{cases}$$

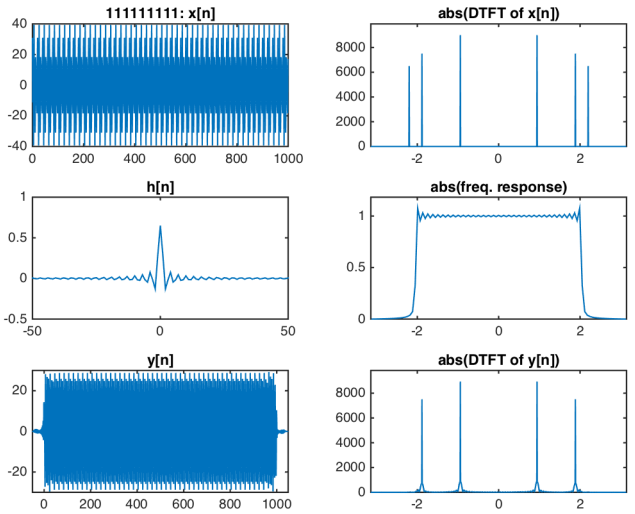

Fig. 2. Example of a generated problem and solution.

LaTeX. Lastly, the LaTeX sources are compiled to produce PDFs of the problem instance (for the student) and the solution (for the answer key). An example from the answer key is provided in Figure 2. For this problem (and others we tried), the entire process of generating parameters, computing the results, exporting the graph and LaTeX, and creating the final PDF takes under two seconds; problems and handout material for an entire class can be produced in just minutes.

2.3. Auto-Grading System

A natural next step is to extend the problem generation framework previously described into a system that automatically grades assignments, often called an "Auto-grader." Auto-graders aim to provide students and instructors with automated evaluation and feedback of student work [5, 6, 7, 8]. By allowing students to immediately receive feedback, auto-graders can encourage students to iterate over multiple trials, improving the quality of their work [6]. Also related is [9], which describes recent offerings of undergraduate and graduate signal processing courses in the form of massively open online courses.

We are currently prototyping a web-based auto-grading system for Matlab code. Our prototype auto-grader begins with a web-based front end, built using Ruby on Rails [10]. Each student has an account where they can track their progress and download assignment files. When a student starts an assignment, the system first generates a problem instance (Section 2.2), and it also provides the student with a customized "checker" as a protected Matlab `.p` file. The checker contains a function that inspects the outputs of a

Lab 1	Lab 2	Lab 3	Lab 4
95.2%	97.1%	84.8%	93.3%

Table 3. Percentage of students who successfully completed each lab assignment.

student's work and gives feedback. On completion of the assignment, the checker can send a message to our server to indicate success. We note that it may be possible for students to reverse-engineer the checker code to "short-circuit" the checks; for a safer approach, we plan to enable an option where checking is performed directly on the server.

We are currently constructing this prototype and plan to test it with the class during the next year. After development and testing, we intend to make this system available to faculty at other institutions.

3. RESULTS AND FEEDBACK

In this section, we first describe the students' performance on the assignments, and then we discuss student feedback given in anonymous end-of-semester surveys and how this feedback has influenced our future plans.

3.1. Student Performance

We assess student performance through the successful completion of the lab assignments, and through performance on small questions included on written quizzes. Table 3 shows the percentage of students who correctly completed each lab assignment. First, we note that Lab 1 and Lab 2 were completed in the presence of course staff, who were available to help and check each teams' results, so the few instances where these assignments were not successfully completed were caused by carelessness or incomplete work.

The success rate was lower for Lab 3. By examining these submissions, we see that the most common problem was where teams would produce an incorrect implementation of the system (e.g., shift incorrectly), but fail to notice this when checking results by hand on paper. That is, because of students' incorrect understanding of the underlying mathematics, they were unable to identify mistakes in their Matlab functions. We view this result as strong motivation for producing the prototype auto-grading system described in Section 2.3; in these instances, student learning would be improved with immediate feedback. By programming the auto-grader to recognize these types of common mistakes, we will carefully tailor the system's feedback to help identify and correct the problem. For Lab 4, we observed similar patterns, however to a smaller extent.

We also included two short written questions related to this material on quizzes, in order to evaluate how well students retained the knowledge they gained from completing these assignments. While completing the lab assignments, students were aware that questions of this type would be included on the exam. This also served to discourage students from over-relying on their partners in completing the assignment. Both questions presented students with a short piece of Matlab code related to a lab assignment, and asked them to explain the code's function. For the first question, 30.5% of students answered the question completely correctly, with an additional 56.2% of the students answering partially correctly. The second question was solved completely correctly by 40.1% of students and partially correctly by an additional 38.1%.

3.2. Feedback and Future Plans

At the end of the term, Stony Brook solicits course feedback from students in the form of course evaluations. Student comments related to this material were almost exclusively favorable, falling into two classes of comments: (1) positive comments about the Matlab assignments, and (2) suggestions that more Matlab work would be valuable. Additionally, one student commented that the knowledge gained in the Matlab assignments was particularly helpful in finding an internship. Only one student had a critical comment about the assignments, fairly pointing out that the timing of the assignments could be adjusted to integrate more closely with the theoretical material.

Our observations of student performance and feedback have led us toward several adjustments in our (current ongoing) second offering of this material. First, the types of problems students encountered (particularly in Lab 3 as described above) strongly suggest that students would benefit from the automatic feedback and guidance that will be provided by the auto-grading system we are developing. Second, we are increasing the number of Matlab exercises, and rather than concentrating the material into four larger assignments, we will assign more smaller problems. Third, in order to more tightly integrate Matlab into the course, we are including these assignments as portions of weekly homework assignments.

4. CONCLUSIONS

In this paper we discussed our efforts to include practical Matlab experience in a lecture-based Signals and Systems course. The new educational components consist of programming activities for teams of students, including unique individual problems that we created using a Matlab-based generator we designed. Feedback from students in the form of surveys and their answers to quiz questions indicated that the added components help in better understanding of the materials, and their comments provide us suggestions for improving the new assignments. Finally, we described our efforts to create an automated grading system that we will use to provide students with instantaneous feedback.

5. REFERENCES

[1] Edward W. Kamen and Bonnie S. Heck, *Fundamentals of signals and systems: using the Web and MATLAB*, Prentice Hall, 2000.

[2] Virginia Stonick and Kevin Bradley, *Labs for Signals and Systems using MATLAB*, Brooks/Cole Pub. Co., 2000.

[3] Won Young Yang, *Signals and Systems with MATLAB*, Springer-Verlag, 2009.

[4] Luis F. Chaparro, *Signals and Systems using MATLAB*, Academic Press, 2010.

[5] Garvit Juniwal, Alexandre Donzé, Jeff C. Jensen, and Sanjit A. Seshia, "CPSGrader: Synthesizing temporal logic testers for auto-grading an embedded systems laboratory," in *Proceedings of the 14th International Conference on Embedded Software - EMSOFT '14*. 2014, ACM Press.

[6] Mark Sherman, Sarita Bassil, Derrell Lipman, Nat Tuck, and Fred Martin, "Impact of auto-grading on an introductory computing course," *Journal of Computing Sciences in Colleges*, vol. 28, no. 6, pp. 69–75, 2013.

[7] Ge Yu, Libin Hong, and Lei Sheng, "A web-based examination and evaluation system for computer programming," in *Proceedings of the Sixth IEEE International Conference on Advanced Learning Technologies*. aug 2006, IEEE.

[8] Stephen H. Edwards and Manuel A. Perez-Quinones, "Web-CAT: Automatically grading programming assignments," in *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education*, New York, NY, USA, 2008, ITiCSE '08, ACM.

[9] Thomas A. Baran, Richard G. Baraniuk, Alan V. Oppenheim, Paolo Prandoni, and Martin Vetterli, "MOOC adventures in signal processing: Bringing DSP to the era of massive open online courses," *IEEE Signal Processing Magazine*, vol. 33, no. 4, pp. 62–83, 2016.

[10] "Ruby on Rails," `http://rubyonrails.org`.