7-2015

Enhancing Freshman Engineering Instruction with In-Class Interaction Systems and e-Books

Branimir Pejcinovic
Portland State University, pejcinb@pdx.edu

Phillip K. Wong
Portland State University

Enhancing Freshman Engineering Instruction with In-Class Interaction Systems and e-Books

Branimir Pejčinović[1], and Phillip K. Wong[2]

[1]*Portland State University, Portland, Oregon, USA, pejcinb@pdx.edu*
[2]*Portland State University, Portland, Oregon, USA, pkwong@pdx.edu*

Abstract

Electrical engineering students in our department take a year-long series of courses which introduces electrical engineering as a discipline and provides good grounding in engineering problem solving and programing. We have recently attempted to make the second course in the sequence more engaging by applying active learning techniques, including assigned reading and exercises prior to lectures, in-class exercises using a classroom interaction system, and programming exercises during lectures. Our results are mixed: while we think that students have learned more than if we had not used these techniques, we have not completely won over our students. While using an e-book was valuable, we believe that exercises within the e-book were not sufficient and their combination with in-class exercises did not provide sufficient training for students to feel comfortable with the programming. In terms of student problem solving skills, we continue to be puzzled by their difficulties despite their already quite extensive math background. There are also uncertainties with respect to our students' preparedness to take larger responsibility for their education as evidenced by their comments and the fact that reminders were required to keep students doing their assigned preparation work.

Keywords: *freshman engineering, flipped classroom, programming.*

1. Introduction

Our university's first year electrical engineering sequence includes two courses that involve programming and hardware interfacing. Both are taught within the electrical and computer engineering (ECE) department. The first requires the student to solve engineering problems using MATLAB. The follow-on course introduces the C language. One interesting feature of the sequence is the use of MATLAB as a programming environment and introduction to C, in addition to its usual role as a problem-solving tool. To make it less abstract and to establish a real-life connection, we also use MATLAB for interfacing with a data acquisition device called LabJack. MATLAB scripts to control the LabJack can be written in the standard MATLAB environment.

This environment has enabled students to participate in some interesting hands-on projects that combine problem-solving, programming, and interfacing. Historically, student participation in the course consisted of attending lectures, three laboratory exercises related to LabJack and MATLAB interfacing, and participation in team-based projects. We recently decided to increase student involvement, especially during lecture time, because current educational research supports the idea that active learning and active student participation lead to better learning [1,2]. We have modified the course to increase student participation by requiring that: a) MATLAB reading and exercises be done in advance of the lecture time, b) students utilize an in-class interaction system from Learning Catalytics (LC), and c) students use MATLAB on their laptops for in-class exercises. We have also hoped that these changes would result in better attainment of course outcomes, such as improved problem solving or programming skills. In the following sections we will discuss the details of implementation and initial assessment of the effect of these changes.

2. Existing Course Overview

Prior to 2010, our electrical engineering (EE) program's first year experience was provided by a pair of general engineering courses that gave a conventional introduction to engineering analysis and computer programming. Based on feedback from industry partners, alumni, and students, we replaced the original courses with EE-specific versions to emphasize electrical engineering and computing topics and to increase student motivation and engagement. More detailed reasons and background for these curriculum changes were reported in [3,4]. The subject matter from the original two courses was expanded into three new courses: ECE 101 Exploring Electrical Engineering, ECE 102 Engineering Computation, and ECE 103 Engineering Programming. ECE 101 introduces incoming students to the electrical engineering field, its many applications in society, and possible career opportunities. The analysis material was transferred to ECE 102, with most non-EE topics removed to

make time for more EE focused material. ECE 103 took on the role of teaching intermediate-level programming in C. Surveys from industry and former students made it clear that the single programming course required of EE students was not meeting the expectations of prospective employers. So, it was decided that ECE 102 would expand the MATLAB portion of the course to include general programming in addition to covering its calculation and graphing tools. Effectively, in our courses MATLAB has become a primer for C due to similarities in syntax. While teaching MATLAB as an introduction to programming is not new [5], direct interfacing between MATLAB and hardware still remains novel and non-trivial [6]. After going through systematic examination of our goals for the course, we came up with the revised course outcomes for ECE 102 as shown in Table 1.

Table 1. Course Outcomes – Students have the ability to …

ECE 102 Engineering Computation
1. Solve engineering problems by applying the engineering method
2. Analyze DC circuits using Ohm's law, Kirchhoff's laws, current-mesh methods
3. Process data using software
4. Develop algorithms in MATLAB to solve simple engineering problems
5. Use MATLAB programming for data acquisition and control
6. Communicate technical information in written and graphical format

To reinforce understanding of fundamental programming topics, a final practical project was added to both ECE 102 and 103. It requires the student to write programs that interface to and control sensors, LEDs, motors, and other electronic hardware. The projects are tailored to the students' skill level and are designed to be fun and interesting. While using microcontrollers or single board computers in a freshman engineering sequence is not a revolutionary concept anymore, the reasons for our choosing a particular type of interface device are somewhat unique. In many education scenarios in which an Arduino or Raspberry Pi controller is used, the interfacing task is the ultimate objective. In contrast, our primary goal is the teaching of the computing language, with hardware interfacing in a support role to increase student engagement.

After looking at various alternatives, we decided on an uncommon choice, the LabJack U3-LV from the LabJack Corporation. The LabJack is a measurement and automation unit that provides multiple analog/digital input and output lines, timers/counters, and SPI/I2C support. The LabJack connects to a host computer via a USB cable. For our purposes, the LabJack's multi-language support was a key feature that distinguished it from other, similar devices. It has a common API for both MATLAB and C/C++ (e.g., GCC and Microsoft Visual Studio). Students can program in these industrial-strength languages using each language's native development environment on the host computer, which is what they already use for homework assignments.

1.1. Course Organization

Because our university is based on a quarter system, it was a difficult task to fit all of the envisioned components into a ten week session. A further complication is that close to two-thirds of the students are transfer students, with most coming from local community colleges. Therefore, our student population has very diverse backgrounds, from experienced programmers to complete novices. Students also have varying degrees of prior college work, anywhere from true freshman to post-bac. For logistical reasons, we do not require ECE 102 as a prerequisite for ECE 103. It is, therefore, challenging to design a course sequence to accommodate all students so that some are not bored while others are overwhelmed with new material. As an illustration, our latest iteration of the ECE 102 schedule is given in Table 2.

Table 2. ECE 102 Course Schedule

Week	Topics	Week	Topics
1	Introduction / Units Problem solving	6	MATLAB branching LabJack overview / Interfacing 1 Final project introduced
2	MATLAB ops, variables, scripts Error analysis (significant figures)	7	MATLAB loops LabJack API / Interfacing 2
3	MATLAB vectors, matrices Tables / Graphs	8	Interfacing 3 MATLAB modular programming
4	MATLAB graphs Circuits 1	9	MATLAB strings / file I/O MATLAB data structures
5	MATLAB user-defined functions Circuits 2	10	MATLAB advanced topics Applications
		Finals	Project demonstration and report

The LabJack is first presented about mid-way through the quarter, once branching and loop commands have been covered. Hands-on lab sessions are held to introduce the LabJack hardware and its programming functions. Students learn the procedures for sensing voltages, controlling LEDs, and reading switches. The lab exercises are synchronized with the lectures to provide a "real-world" application of each new programming topic. There is a multi-week final project in which two-person teams program a game that requires a hardware interface. Students are loaned a LabJack and an additional parts kit for constructing circuits.

In past years, the final project has been based on either the television game show "Wheel of Fortune" or the 80's electronic toy "Simon". For Wheel of Fortune, students build a circuit using a decoder chip and eight LEDs to simulate the wheel. Their MATLAB script monitors a switch that activates the wheel, controls which LED is lit as the wheel "spins", and provides the game logic. In the Simon project, students implement the game with four LEDs and four switches. Their script generates the ever-increasing, randomized LED sequences, responds to switch presses, and performs the comparisons to see if the input matches the sequence. For extra credit, sound effects, music, or more elaborate displays can be added. A full design report with commented source code is required, and each team is expected to demonstrate and discuss their work with the instructor.

2. Course changes

In preparing for 2014-15 school year we decided to take advantage of some new opportunities to redesign the course. First of all, two instructors were assigned to teach separate sections of the course. This opened up opportunities for collaboration and testing of various hypotheses. Secondly, class size was set to a maximum of 50, which meant that experimentation with new pedagogy and tools would be easier and lower-risk than in 90+ sized classes of the past. After some initial discussions and given the newly agreed upon course outcomes, we decided to implement several changes:

1. Bring more in-class interaction by using an in-class response system
2. Partially "flip" the classroom by requiring students to read a MATLAB e-book and doing exercises before coming to class
3. Assign another mini group project to give students experience with group work and writing before they start working on the final LabJack-based projects

For the in-class interaction system, we settled on Learning Catalytics (LC) from Pearson [7]. It is a fairly flexible system that is more applicable in physical sciences and engineering due to its ability to handle LaTeX formatting and specialized questions, such as graphical input from students. Example of a working screen with all of the "lectures" that we set up for the course is shown in Figure 1. The system makes it easy to observe student performance in class so that the pace and questions can be adjusted to student progress. Within the system it is also possible to track individual students so that their participation and progress can be evaluated, as shown in Figure 2. This evaluation was done at the end of the class mainly due to lack of time, but to take full advantage of it, the evaluation would best be done periodically during the course.

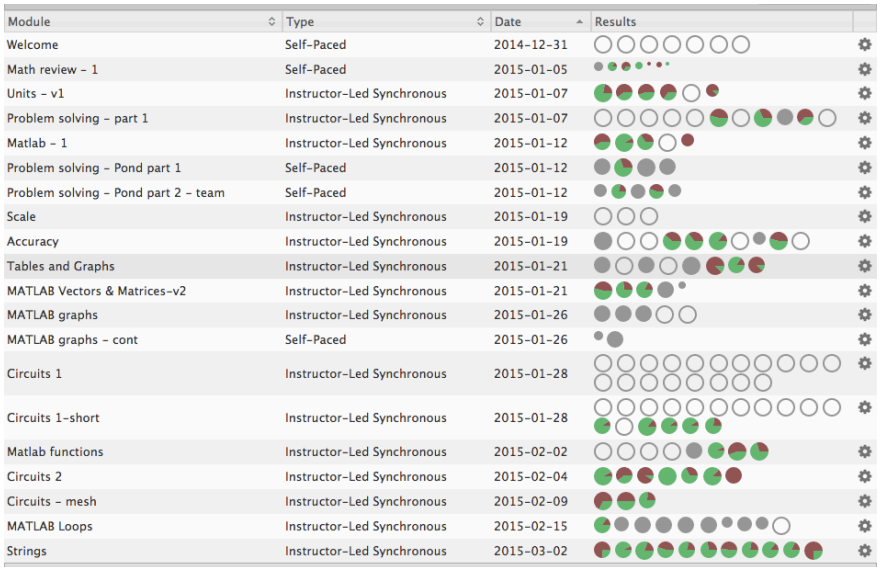
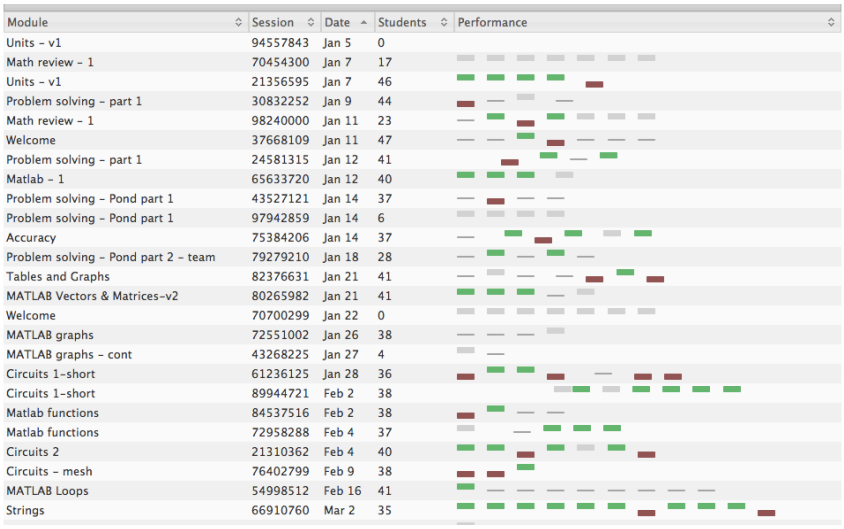


Figure 1. Illustration of sessions used in Learning Catalytics for in-class exercises. Pie-charts indicate completion and correctness of answers.



Module	Session	Date	Students	Performance
Units – v1	94557843	Jan 5	0	
Math review – 1	70454300	Jan 7	17	
Units – v1	21356595	Jan 7	46	
Problem solving – part 1	30832252	Jan 9	44	
Math review – 1	98240000	Jan 11	23	
Welcome	37668109	Jan 11	47	
Problem solving – part 1	24581315	Jan 12	41	
Matlab – 1	65633720	Jan 12	40	
Problem solving – Pond part 1	43527121	Jan 14	37	
Problem solving – Pond part 1	97942859	Jan 14	6	
Accuracy	75384206	Jan 14	37	
Problem solving – Pond part 2 – team	79279210	Jan 18	28	
Tables and Graphs	82376631	Jan 21	41	
MATLAB Vectors & Matrices-v2	80265982	Jan 21	41	
Welcome	70700299	Jan 22	0	
MATLAB graphs	72551002	Jan 26	38	
MATLAB graphs – cont	43268225	Jan 27	4	
Circuits 1-short	61236125	Jan 28	36	
Circuits 1-short	89944721	Feb 2	38	
Matlab functions	84537516	Feb 2	38	
Matlab functions	72958288	Feb 4	37	
Circuits 2	21310362	Feb 4	40	
Circuits – mesh	76402799	Feb 9	38	
MATLAB Loops	54998512	Feb 16	41	
Strings	66910760	Mar 2	35	

Figure 2. Illustration of one student's performance and participation using Learning Catalytics.

So far, student participation in LC exercises is good at 70% to 80% of the whole population. Percentages are higher if we count only students attending the class. It usually takes at least one to two weeks for students who have never used the system to become comfortable and proficient in using it.

For assigned MATLAB readings, we settled on the e-book "Programming in MATLAB" by zyBooks [8]. This e-book system has been in use for several years and their MATLAB book was claimed to have improved student learning in a typical class environment [9]. The most attractive feature of the book is that it comes with interactive features and quiz-like questions that students can answer to test their understanding of the material they have just read. The system keeps track of how many exercises a student performs and generates reports that we can utilize to track student usage, as explained in the next section.

Finally, introducing a mini-project was reasonably successful. Most student groups were able to produce good quality reports, but we will not discuss this change in detail.

3. Course Assessment and Discussion

Student performance in ECE 102 is assessed in multiple ways: in-class exams (quizzes), homework sets, projects, and surveys. Final project success among the student teams is typically high, with completion rates in the 95% range, as shown in Table 3. We consider a project successfully completed if students can demonstrate its use as a "real" game machine, and if hardware and software satisfied all the requirements listed in our project guide. For example, all lighting and audio signals behave as specified. Students are given extra credit if they improve the game in some fashion but only if the core program performs as expected.

Table 3. ECE 102 Final Project Success Rates

Quarter	Year	# of Teams	% Successful
Winter	2011	31	94
Summer	2011	9	100
Winter	2012	35	94
Winter	2013	37	97
Winter	2014	36	89
Winter	2015	40	95

In-class exams have proven to be more challenging for students with a much wider variation in scores and are usually the most important contributor to variations in student final grades. We have been trying to identify variables that would explain success and failure in ECE 102 and other freshman classes, and math preparation is the usual suspect. However, as our previous work [10] has shown, the correlation between previous success in math and grades in ECE 102 is weak. We are currently examining some alternative hypotheses that we hope will be more predictive of student success.

3.1. e-Book Readings

We monitored student reading of the e-book through weekly reports generated by the system. Figure 3 shows weekly completion rates for activities associated with MATLAB readings assigned for that week. Note that we eliminated from the pool the students who dropped the class. There were 43 students in Section 1 and 31 in Section 2. There is a significant difference in time evolution of completion rates among the two sections: starting from roughly the same point one stabilized at around 75% and the other at 40%. The main difference turns out to be that the instructor in Section 1 constantly reminded students of their assignments and posted weekly reading completion reports while the other one did not. Reading completion, however, was part of the overall grade in both sections. These two numbers seem to provide upper and lower bounds on what can be expected in terms of reading and activity completions. One may be tempted to explain these results by the fact that these are freshmen but our student population includes many "seasoned" students and only 38% are true freshman. Therefore, if it is deemed important to have students finish their readings it will require constant reminders of some type, independent of their status.

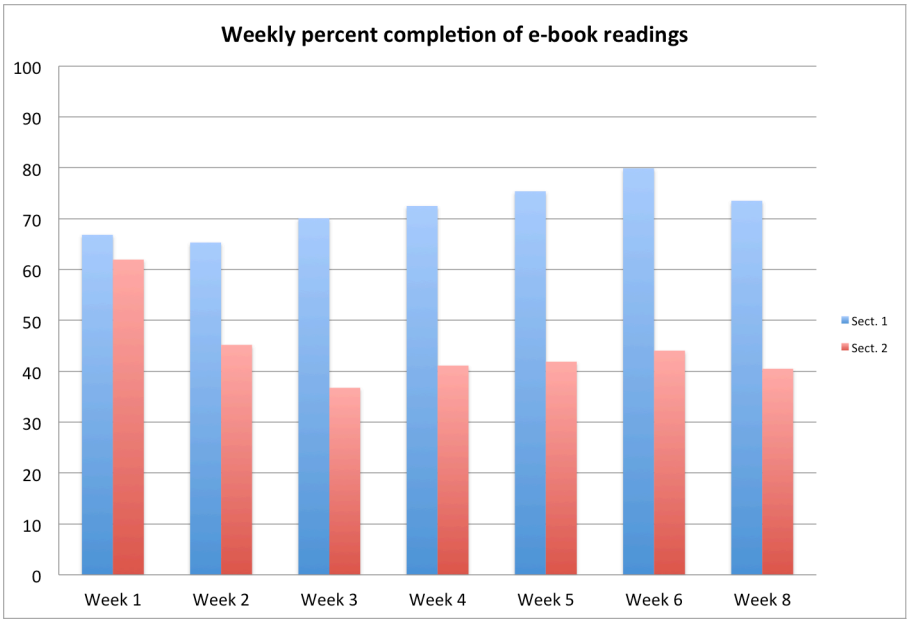


Figure 3. Weekly evolution of completion of activities from MATLAB e-book.

3.2. Survey

In order to assess effectiveness of various additions and changes to the course, we designed a new survey to probe student opinions about various components of the course. The full list of questions is given below and it is broken into two components: a) assessing student self-efficacy, i.e. their perception of their own ability to perform certain tasks, and b) perceived effectiveness of instructional techniques used in the class. Survey questions include:

A) *Self-efficacy ("I am confident that …")*
Scale: Strongly disagree (1), Disagree (2), Neutral (3), Agree (4), Strongly Agree (5)
1. I can program and use MATLAB to solve problems
2. I can use MATLAB to control LabJack
3. I can solve DC electric circuits problems
4. I can solve general engineering problems
5. I can write good quality reports

B) *Effectiveness of instructional techniques*
Scale: Complete waste of time (1), Not helpful (2), Neutral (3), Somewhat helpful (4), Very helpful (5)
6. Labs for building and testing circuits using LabJack
7. Doing in-class exercises and problems (incl. Learning Catalytics)
8. Solving homework problems
9. Listening to instructor lecture
10. Class projects

Figures 4 and 5 show the results collected from two sections, the first one with 31 out of 43 and the other with 32 out of 36 respondents.

Data from our survey regarding student self-efficacy is presented in Figure 4 where average scores for each question are presented. In general, there are no dramatic differences between the two sections with the possible exception of student confidence in solving DC electric circuits. More interesting is the fact that students seem to be fairly confident in their ability to solve problems. This is not entirely borne out by other assessment results. As mentioned above, project completion rates are very high. However, the quality of the final programs varies considerably, though it is difficult to judge consistently. We believe that this variation in quality reflects variation in student understanding of programming and problem solving and would indicate that students may be overly optimistic about their abilities. Quiz results provide another possible point of comparison. We have collected quiz results by question but have not yet completed a numerical analysis that would provide a comparison with student self-assessment. Our first impression is that students are overestimating their abilities, but we cannot quantify this observation at this time.

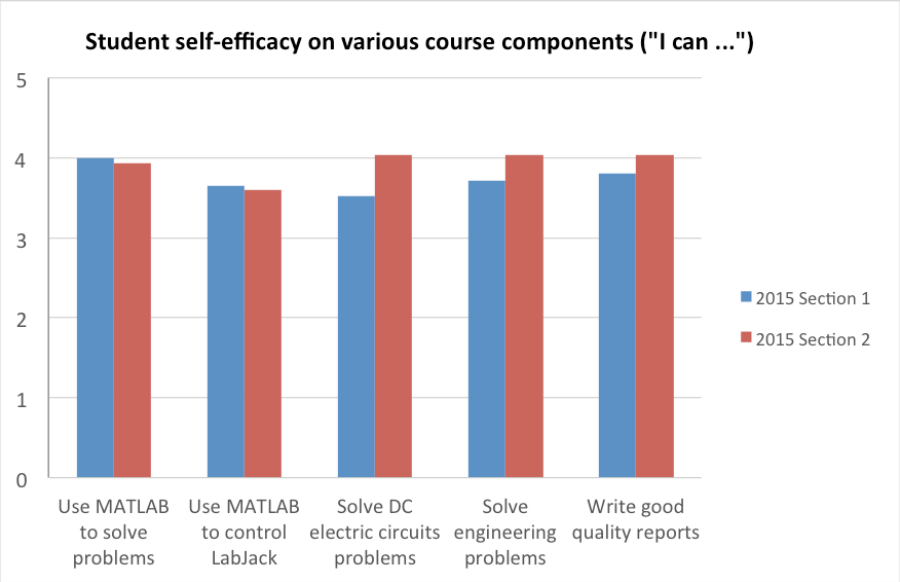


Figure 4. Student self-efficacy as determined by the end-of-term survey in Winter 2015.

Another set of questions on our survey deals with effectiveness of various pedagogical tools and the results are presented in Figure 5. In this case there are some differences between two sections. In particular, lectures in Section 1 were deemed significantly less effective than in Section 2. This may be attributable to the fact that the instructor in Section 1 taught the course for the first time, while the second instructor has many years of experience in teaching this or similar courses. Given that the students are “neutral” (i.e. numerical score around 3) with respect to helpfulness of LabJack in understanding MATLAB, we should strengthen the connection between MATLAB and LabJack possibly by adding another lab or more discussions in class. Interestingly, students still value homework exercises more than doing similar problems in class. This is puzzling because one of the persistent student complaints is regarding lack of feedback on homework problems, but when we provide immediate feedback during class that does not seem to be perceived as an improvement over homework. Alternatively, it is possible that our implementation of in-class exercises is not yet fluent and polished, and we will have to examine trends over time. It is also quite possible, as evident in some student comments, that students are not used to this mode of instruction. Finally, it is interesting to note that students like the MATLAB e-book better than the paper textbook used in 2011. Again, we will monitor how this item develops over time.

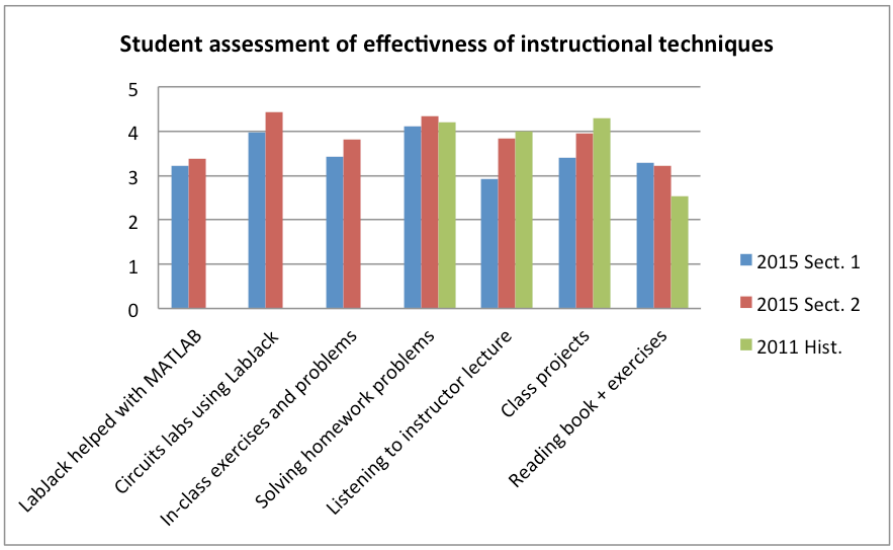


Figure 5. Effectiveness of instructional techniques as determined by the end-of-term survey in Winter 2015.

Finally, we also looked at some student comments on the survey. From a previous survey in 2011 we know, for example, that many students stated that they enjoyed the practical aspects of the project, which maintained their interest. On the other hand, a common complaint at that time was the significant amount of time it takes to develop and debug code.

From the 2015 survey it appears that not all students bought into the active learning and flipped approach, so one of them complained "The teacher needs to teach, not just show and have us read an online book" but others felt that "In-class and group activities are extremely helpful." For using the e-book, there are similarly mixed opinions, from: "The zyBook was not very helpful for me. The book was a good idea but I think something else could help more", to: "The zyBook was alright, but grading the reading on that is a little lame as it doesn't really indicate if the material was learned." Many students complained about lack of practicing MATLAB and we have starting implementing a set of labs that will help train students better. In parallel with that development we will experiment with competency testing.

4. Conclusions

The ECE 102 Engineering Computation course has evolved over time in both content and teaching philosophy. It was always meant to provide students with practical problem solving skills while at the same time teaching them the basics of programming and electrical engineering. In 2015 we incorporated more active learning tools, such as interactive online textbooks and real-time assessment of student responses to questions using the web-based Learning Catalytics. We have presented our implementation and discussed some problems that have arisen. From our initial assessment it seems that some students have not yet fully accepted this instructional approach. Furthermore, just assuming that students will take over the responsibility for their learning is likely to result in disappointment. In our case this was reflected in the fact that without constant reminders in the form of weekly grades, students were much less likely to complete their reading assignments. Overall, this was a successful first test of the effectiveness of more active in-class engagement and the utilization of an e-Book as primary reading material for MATLAB programming. Despite the initial difficulties, we feel that the time and effort invested in this change was well worth it. Our first corrective action will be introduction of programming labs and competency testing, both of which are aimed at strengthening student programming skills.

References

[1] M. Prince, "Does active learning work? A review of the research," Journal of engineering education, vol. 93, no. 3, pp. 223–231, 2004.

[2] S.A. Ambrose, et al., *How Learning Works: Seven Research-Based Principles for Smart Teaching*. John Wiley & Sons, 2010.

[3] P. Wong, et al. “Redesign of Freshman Electrical Engineering Courses for Improved Motivation and Early Introduction of Design,” in ASEE Annual Conference, and Exposition, Vancouver, Canada, 2011.

[4] P. Wong and B. Pejcinovic, “Teaching MATLAB and C Programming in First Year Electrical Engineering Courses Using a Data Acquisition Device,” accepted for publication, ASEE Annual Conference and Exposition, Seattle, Washington, USA, 2015.

[5] S. Attaway, “Using MATLAB to Teach Programming to First-Year Engineering Students at Boston University.” Mathworks, available at http://www.mathworks.com/tagteam/65082_91847v00_NN10_FA_BU.pdf .

[6] M. A. Rubio, R. Romero-Zaliz, C. Mañoso, and P. Angel, “Enhancing an introductory programming course with physical computing modules,” in Frontiers in Education, Madrid, 2014.

[7] Details available at learningcatalytics.com

[8] Details on MATLAB e-book are available at https://zybooks.zyante.com

[9] A. Edgcomb et al. “Student Performance Improvement using Interactive Textbooks: A Three-University Cross-Semester Analysis,” Technical report number: UCR-CSE-2014-10030 http://static.cs.ucr.edu/store/techreports/UCR-CSE-2014-10030.pdf, October 2014.

[10] B. Pejcinovic et al. “Assessment of Student Preparedness for Freshman Engineering Courses Through Assessment of Math Background,” Frontiers in Education (FIE), Madrid, Spain, 2014.

Authors

Branimir Pejčinović: Branimir Pejčinović received his PhD and MS degrees from University of Massachusetts, Amherst and BS from University of Zagreb. He is a Professor and former Associate Chair for Undergraduate Education at Portland State University, Electrical and Computer Engineering department. He has led department-wide changes in curriculum with emphasis on project- and lab-based instruction and learning. His research interests are in the areas of engineering education, semiconductor device characterization, design and simulation, signal integrity and THz sensors. He is a member of IEEE and ASEE.

Phillip K. Wong: Phillip Wong received an M.S. degree in electrical engineering from Carnegie Mellon University in 1990. Since then, he has been with Portland State University, Oregon, USA, where he is currently the ECE Lab Coordinator and an instructor.