6-2017

Evolution of an Introductory Electrical Engineering and Programming Course

Branimir Pejcinovic
Portland State University, pejcinb@pdx.edu

Phillip Wong
Portland State University

Evolution of an Introductory Electrical Engineering and Programming Course

1. Introduction

Portland State 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, ECE 102, requires the student to solve engineering problems using MATLAB. The follow-on course introduces the C language. To make programming less abstract and to establish a real-life connection, we use MATLAB for interfacing with a data acquisition device called LabJack. Students use MATLAB's integrated development environment to write scripts that control the LabJack.

This environment has enabled students to participate in some interesting hands-on projects that combine problem-solving, programming, and interfacing. Early on, student participation in the ECE 102 course consisted of attending lectures, three laboratory exercises related to LabJack and MATLAB interfacing, and participation in team-based projects. Given that research in student learning consistently shows that active learning and higher student participation leads to better learning outcomes [1][2], we have recently modified the course to increase student participation by requiring that students: a) do MATLAB reading and exercises in advance of the lecture time, b) utilize an in-class interaction system, c) use MATLAB on their laptops for in-class exercises, and d) attend programming labs. Given that ECE 102 does not deal with programming alone, we have faced a problem of students passing the class without learning basic programming skills - a common problem in any course in which students can collect partial credit. We are attempting to address this through pass/no-pass competency tests that are required for passing the course. This ensures some minimal competency before students get into more advanced programming in later classes. In the following sections we will discuss the details of implementation and initial assessment of the effect of these changes.

2. Course Evolution and Overview

We will first provide an overview of what compelled us to revise our curriculum. This is followed by a discussion of student learning outcomes and our overall goals. Goals, however, are only as good as their implementation, so we review our schedule and discuss the components of the class: active learning using classroom interaction devices, e-book readings, projects, and labs.

2.1 Motivation for Change

Prior to 2010, our electrical engineering (EE) program's first year experience was provided by a pair of general engineering courses that covered engineering analysis and computer programming. Details of the curriculum changes were reported in [3][4], but the primary desire was to increase student engagement with realistic electrical engineering problems that emphasized programming. The topics from the original two courses were 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 and its many applications in society. The analysis material was transferred to ECE 102, which now focused on circuit applications. ECE 103 took on the role of teaching intermediate-level programming in C. Feedback from industry and alumni indicated that the

single programming course required of EE students was insufficient. To address this problem, we expanded the MATLAB portion of ECE 102 to include general programming in addition to covering its calculation and graphing tools. While teaching MATLAB as an introduction to programming is not new [5], we also chose to integrate MATLAB with hardware interfacing to provide a more concrete application of design, circuits, programming, and teamwork [4].

2.2 Student Learning Outcomes

Our initial student learning outcomes (SLOs) for ECE 102 are given in Table 1 as "Old." After going through a systematic examination and performing backward design starting with our goals for the course, in 2014-15 we revised the course outcomes as shown in Table 1 under the "New" column. While the new set of SLOs are not radically different from the old one, it is more focused and we found this kind of periodic examination of higher-level goals very helpful in making decisions about details of instruction.

Table 1: Changes in ECE 102 student learning outcomes, starting in 2014-15 academic year

Old	New
1. Analyze and find solutions to engineering problems by applying the engineering method. 2. Analyze DC circuits using Ohm's law, Kirchhoff's laws, and current-mesh methods. 3. Use software tools to process data, perform calculations, and create graphs. 4. Develop algorithm design skills for writing MATLAB scripts to solve simple engineering problems. 5. Investigate data acquisition and control techniques via MATLAB programming. 6. Document a design project in a technical report.	1. Solve engineering problems by applying the engineering method 2. Analyze DC circuits using Ohm's law, Kirchhoff's laws, and 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

In light of the new SLOs, the course had to evolve from traditional classroom teaching methods to incorporate more active learning and online components. A timeline of these changes is shown in Table 2. Our rationale and discussion of these changes are provided in section 3.

Table 2: Chronology of changes in the ECE 102 course

Year	Changes
2010-2014	● Removed non-EE topics and expanded MATLAB coverage ● Interleaved MATLAB and engineering analysis topics ● Added LabJack interfacing project
2014-15	● Added Learning Catalytics ● Added interactive MATLAB e-book ● Added separate mini-design project at start of term ● Began experimenting with competency testing
2015-16	● Changed schedule to cover all MATLAB topics during first five weeks ● Made MATLAB labs mandatory ● Used online homework assignments (e-book with MathWorks Cody) ● Replaced mini-design project with two-part HW/SW project ● Added kanban for project management ● Added mandatory MATLAB Competency Tests (CT)
Winter 2017	● Added "analytical thinking" pre- and post-test

3. Course Organization

Our university is a relatively large, urban, public university based on a quarter system, i.e., each term is 10 weeks long with the 11th week reserved for final examinations. There are three community colleges within city limits and all of them have articulation agreements with our university and ECE programs. The university and community colleges have dual-enrollment agreements, further blurring the line between so-called native and transfer students. These and other factors make our student population very diverse in terms of the amount of experience in problem solving, programming, writing, and other areas relevant to engineering. In the past, we avoided enforcing the pre-requisite relationship between freshman courses, i.e., ECE 101 was not listed as a prerequisite for ECE 102, and 102 is not a prerequisite for 103. For reasons to be discussed below, we have started enforcing the first prerequisite. This background present serious challenges for designing this sequence overall and also for designing individual courses.

As an illustration, our latest iteration of the ECE 102 schedule is given in Table 3. Note that we have concentrated MATLAB instruction up front whereas in the past it was interspersed with problem solving topics, which are now covered in the second half. Many students have minimal or no programming experience, which is exacerbated by their lack of problem solving skills. We believe that these students will benefit from the reduced cognitive load of the new schedule. This also enables more focused lab sessions for programming exercises. The potential drawback is that students will not see the application of the programming concepts until the fifth week or so. We have attempted to address this issue by making lab exercises more applied, for example by analyzing sample electrocardiogram data with MATLAB. More concentrated MATLAB instruction may also help reading assignments, as discussed in section 4.

Table 3: Current ECE 102 course schedule (2017)

ML =MATLAB topic, E-# =eBook reading; T-# =Textbook reading, HW-m =MATLAB homework, HW-s =regular homework, CT =competency test, EX =in-class exam

Wk	Topic	Readings	Lab	HW	Exam	Project
1	Intro / ML Ops, Variables, Scripts	E-1	Lab-1	HW-m1		
	ML Vectors, Console I/O, Plots 1	E-2				
2	ML Functions, Matrices	E-3	Lab-2	HW-m2		
	ML File I/O, Logical expressions	E-4				
3	ML Selection	E-5	Lab-3	HW-m3	CT-1	
	ML Loops	E-6				
4	ML More selection & loops, Strings	E-7	Lab-4	HW-m4	CT-1b	
	ML Programs, Subfunctions	E-8, T-1				
5	ML Miscellaneous topics	T-2	Lab-5	HW-s1	CT-2	Trello eval.
	Basic circuits					
6	Circuit analysis	T-3	LJ-1	HW-s2	CT-2b	Part 1 Report
	More circuits / LabJack intro					
7	Problem solving / Units	T-4	LJ-2		EX-1	
	Exam 1					
8	Error analysis	E-9	LJ-3 Quiz	HW-s3	Practical Lab Quiz	Trello eval.
	Tables / Graphs / ML Plots 2					
9	Problem solving practice		Team meetings		EX-2	
	Exam 2					
10	TBA		Team meetings			Demo +Code
	Project Demonstration Day					
11	Project Final Report Day					Part 2 Report

3.1 In-class Activities

To monitor student understanding of lecture material, we use Learning Catalytics (LC), which is a web-based method for assessing student responses to questions in real-time [6]. It is a fairly flexible system that can handle LaTeX formatting and specialized questions, such as graphical input from students. 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. So far, student participation in LC exercises is good at 70% to 80% of the whole population but nearly everyone who is present participates. It usually takes one to two weeks for students who have never used the system to become comfortable and proficient in using it. One issue that we encountered is that submission of programming exercises, i.e., student code, can be done only as text. This is still helpful because it requires students to commit to an answer but also means that the instructor

must evaluate these submissions on-the-fly. Typically, we resolve this by picking one of the wrong submissions and discuss it in detail. In classrooms that are equipped with additional projectors, we may also ask one of the students to present their solution and then dissect it with the rest of the class. Overall, in-class activities are very important for immediate formative feedback and to expose weaknesses early on.

3.2 E-book

For assigned MATLAB readings, we selected the e-book "Programming in MATLAB" by zyBooks [7], which is accessible through a web browser. This e-book system has been in use for several years, and the publisher claims their MATLAB book improves student learning in a typical class environment [8].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. Interactive activities in the e-book are divided into two categories: 1) *Participation*, which are simple questions related to a reading section, and 2) *Challenge*, which are more substantial questions which require writing and testing small snippets of code that are automatically evaluated through an integrated version of MathWorks Cody Coursework. Cody allows a student to execute and validate their code with sample test cases as many times as they wish before submitting their solution. Since the e-book records a student's work, we use participation activities to gauge the student's weekly progress in reading the material and assign 5-10% of the total grade to these activities. A study reported in [8] found that this was necessary to ensure good student participation. We also follow the recommended practice of assigning readings ahead of lectures and homework afterward [8].

3.3 Homework Sets

For the MATLAB section of the course, we assign the challenge activities from the MATLAB e-book as homework, which are automatically graded and recorded. To supplement the online exercises, students are also asked to solve "extra" MATLAB problems that are sourced from the instructors and various textbooks. These require students to write more substantial scripts and functions, which are manually graded by a teaching assistant for correctness and adherence to good program practices. We are exploring ways to extend Cody Coursework to automate some of these tasks. When the course moves on to engineering analysis and problem solving, students work on the typical types of problems offered by a first year engineering curriculum.

3.4 Projects

To reinforce understanding of fundamental programming topics, a final multi-week, hands-on project has been an integral part of ECE 102 since the beginning. 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. Three-person teams are formed using the CATME system [9][10], which is also used for peer assessment of contributions to the team. Recent projects included modeling a road intersection with traffic light control, and modeling of a home security system. The projects are split into two parts with the goals of:

1. Researching a topic; collaborating with teammates; using project management software for planning and building a rough model out of appropriate materials (particleboard, hardboard, plastic, etc.); and writing a short report on accomplishments and plans.
2. Building all the necessary electronic hardware; developing a MATLAB program to drive the hardware; testing the whole system; and writing a full report on the project.

While minimum requirements and specifications are given, students are encouraged to add features for extra credit, such as sound effects, music, or more elaborate displays. A full design report with commented source code is required, and each team is expected to demonstrate and discuss their work with the instructor. A friendly design competition is held on the demonstration day and teams vote for the best overall design.

In earlier iterations of ECE 102, the project was a single part that was assigned more than half-way through the quarter. This was necessary since students still needed training to use the interfacing device, which did not happen until the midpoint of the course. With our latest course design, splitting the project into two parts has been beneficial, since the first research-only part allows teams to meet each other and start working together earlier than was possible before. We are also able to provide more timely feedback regarding writing, project management, and teamwork.

We also had to specify a type of interface device that would be used in projects and labs. Our primary goal is the teaching of the computing language, with hardware interfacing in a support role to increase student engagement. The device also had to be relatively easily controllable from MATLAB. Given these constraints, we settled on the LabJack U3-LV from the LabJack Corporation. The LabJack is a measurement and automation unit that provides multiple analog and digital I/O lines. The LabJack connects to a host computer by a USB cable and supports multiple languages, including MATLAB and C/C++. This allows students to program using each language's native development environment, which is what they already use for homework assignments. LabJack units have proven to be reliable and have withstood regular use by students over several years, which makes loaning them to students a practical option.

3.5 Teamwork and Project Management

Initially, project teams consisted of pairs of students. This made for relatively simple administration and less potential for team conflicts. It also seemed to be simple in terms of determining any social loafing, i.e., students not contributing to the team. On the other hand, it limited the range of difficulty or productivity that could be expected from a team, and experience with a more realistic team environment would be beneficial in follow-on courses. In addition, providing students with a more collaborative environment is generally expected to improve their learning [11]. This led us to increase the size of teams to three, which we feel is still practical. One concern is the increased potential for social loafing, but we have successfully used CATME peer evaluation to identify non-functioning teams early on during the first part of the project. Similarly, CATME peer evaluations are used to adjust individual project scores. Typically, we also have our own observations of team functioning to triangulate peer evaluations. Overall, adding CATME has been very valuable, but students need training in its proper use.

At this stage we do not expect, nor do we teach, full project management techniques. Our goals are more limited and include: a) making sure that student plan their activities, b) training students to keep track of their own and their colleagues' work, and c) making teamwork and individual contributions as transparent as possible. Kanban boards offer a simple and effective solution for these goals [12]. Once students go through training on how to use Kanban boards during all three courses in the freshman year, we hope that they will be ready for more formal Scrum-like project management in their sophomore and later courses [13]. Our tool of choice is Trello.com, which offers free accounts and easy setup of project boards, lists, and cards [14]. We have developed a simple rubric that we use to evaluate the quality of boards and the level of activity. Most teams take this approach seriously, but without constant vigilance and reminders on the instructor's part, there is great temptation to neglect it. Our initial results are encouraging, and we will continue with applying this style of project management to student projects. The biggest payoff, however, will be in the follow-on classes.

3.6 Labs

Until the 2015-16 academic year, the course did not incorporate MATLAB programming labs. However, we realized early on that the students who most needed help were not showing up to talk with the instructor or teaching assistant. Instead, they silently struggled to comprehend the material on their own or just gave up altogether. Without feedback, we often assumed the students were understanding the programming concepts, but results from in-class questioning and exams clearly showed they did not. This led us to implement labs, which were optional at first but then made mandatory. A programming assignment is handed out at the start of the lab session, which makes the students write programs that are related to the lecture topics of that week. The instructor, teaching assistant, and an undergraduate student helper all attend the lab to provide guidance. In the less formal lab environment, students are more willing to ask questions, and the instructor has the chance to reply with more detail than is possible during a time-limited lecture. With the ability to "look over the shoulder" of students as they attempt to write a program, we have gained valuable insight into areas of confusion and misunderstanding.

The LabJack interface device used in the project is first presented about mid-way through the quarter, once branching and loop commands have been covered. So in addition to the standard MATLAB programming labs, two hands-on lab sessions are dedicated to introducing the LabJack hardware and its programming functions. Students learn the procedures for sensing voltages, controlling LEDs, and reading switches, which reinforces DC circuit concepts that were discussed in lectures and readings. Skills learned in these labs are directly applicable to team projects. Overall, labs have been instrumental in providing students additional programming practice and feedback. It is difficult to state how much of an improvement can be attributed to their introduction because historically we did not do a separate assessment of programming skills. However, we believe that their combination with competency testing, discussed below, has great potential to improve student learning.

4. Course Assessment

Student performance in ECE 102 is formally assessed in multiple ways: readings, surveys, in-class exams (quizzes), homework sets, projects, and competency tests. The first two are primarily formative and the rest mostly summative in nature. Projects contribute the most to the final grade. Final project success among the student teams is typically high, with completion

rates in the 95% range. We consider a project successfully completed if a team's hardware and software satisfied all the requirements listed in our project guide. Students are given extra credit if they improve the project in some fashion, but only if the core program performs as expected. Separate assessment of project demo, report, and MATLAB code is performed. Individual contributions to the team effort are assessed through CATME and final scores are adjusted.

In-class exams, which focus primarily on problem solving, 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 the problem solving area and in class overall. Our initial guess was that math preparation may be the culprit. However, as previous work [15] has shown, the correlation between previous success in math and grades in ECE 102 is not very strong. We are still looking for a reliable and valid instrument that would help us guide students better and maybe even offer individualized instruction. So far, some locally developed math and programming tests seem to hold the most promise, but the data is preliminary. Some of our other assessments are discussed in the following sections.

4.1 e-Book Readings

We monitored student reading of the e-book through weekly reports that the e-book system generates. Figure 1 shows weekly completion rates for activities associated with MATLAB readings associated with that week. The figure represents three sections taught by the same instructor in consecutive course offerings. We conjectured that the low completion rates for the Spring 2015 and Winter 2016 sections may have been partly due to our poor organization of the e-book content. One feature of the MATLAB e-book is its ability to move chapters around and to delete unwanted sections. With this in mind, we reorganized the e-book to group related topics together in a way that exactly matched our lecture schedule. This was done to see if it could reduce the confusion students experienced. The Spring 2016 results indicate that this did make a difference. Additional factors that contributed to improvement were the increase in assigned number of points to 8% of the total and more concentrated MATLAB instruction. Note that if it is deemed important to have students finish their readings, it will require constant reminders for the first few weeks.

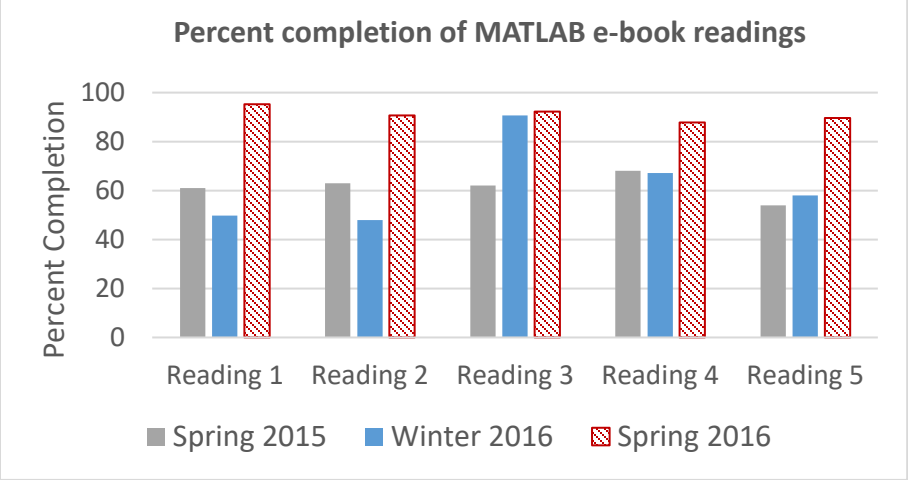

Figure 1. Weekly evolution of completion of activities from MATLAB e-book.

One of the unique features of zyBook readings is that it is possible to determine the "earnestness" with which students approach various activities within the reading [16]. For certain types of problems, it is possible for students to see the correct answer before entering theirs, which, of course, is not the intended use. Any such uses are flagged and compared to the number of times that student tries to answer a question first and then looks up an answer (if it was incorrect). With help from zyBook staff [17], we collected the completion and earnestness data based on the first four assigned readings, as shown in Table 4.

Table 4: MATLAB zyBook reading earnestness (-1 and -2 are parallel course sections)

	Wi15-1	**Wi15-2**	**Sp15**	**Wi16-1**	**Wi16-2**	**Sp16**
Earnestness (%)	66	79	68	76	77	75
Completion (%)	85	70	72	81	85	91

There has been a definite improvement in earnestness scores, but they are still low compared to pure programming courses at other institutions using similar zyBooks. Such programming courses regularly have earnestness scores around 85% [16]. The reason for this discrepancy is unclear, but we will strive to improve this score because students are not getting the full benefit of these readings if they are circumventing the system. Overall, we have found that assigned e-book readings help student learning, but we need to monitor student performance carefully, especially early on.

4.2 Survey

In order to assess the effectiveness of various additions and changes to the course, we designed a survey to probe student opinions about 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 2 and 3 show the results collected from two consecutive quarters. Note that two sections are offered in Winter (Wi) term and one in Spring (Sp) term.

Data from our survey regarding student self-efficacy is presented in Figure 2, where average scores for each question are presented. In general, there are no dramatic differences between the parallel sections so their data has been combined. 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 for questions related to problem solving 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. There appears to be slight improvement over time and across all questions, except for report writing. Based on our evaluation of the project reports, we believe that students need to improve their writing. However, these reports are team-based, and usually the best writer on the team is assigned this task, which masks individual student's true writing ability. Individual writing skills are emphasized more in the preceding class (ECE 101).

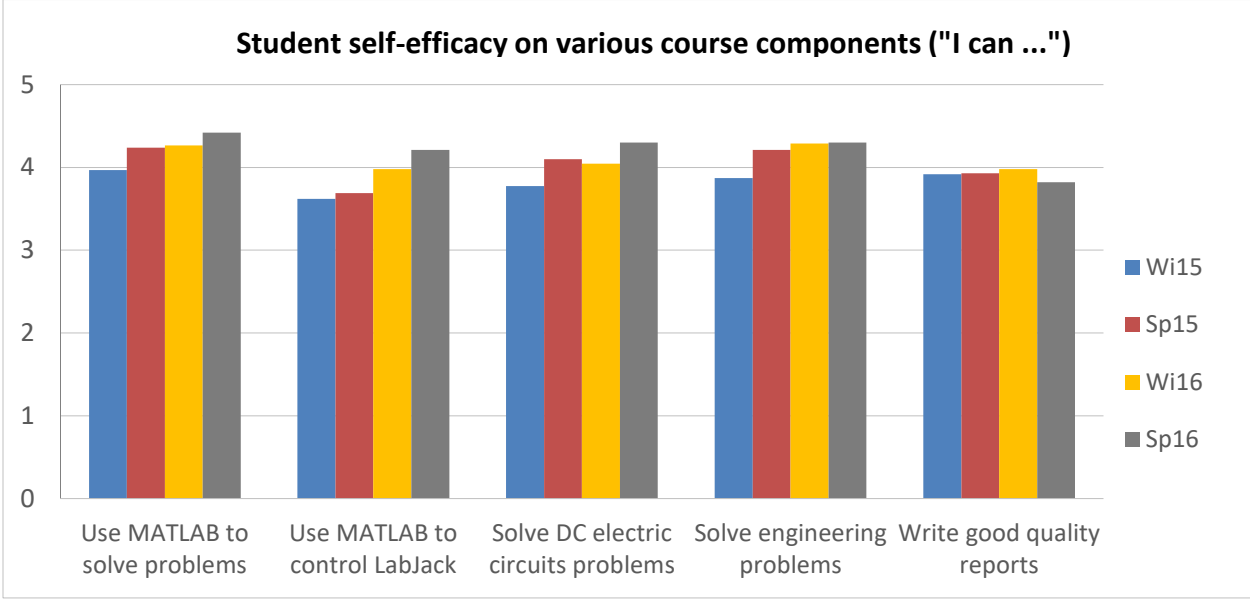

Figure 2. Student self-efficacy as determined by the end-of-term survey from Winter & Spring 2015, Winter & Spring 2016.

Another set of questions on our survey deals with the effectiveness of various pedagogical tools, and the results are presented in Figure 3. In this case there were initially some differences between the two sections with respect to instructor lecturing, but those have since disappeared. Hence, we have combined parallel section data. LabJack was initially deemed only "neutral" for its effectiveness in learning MATLAB, but it looks like student assessment of it is becoming more positive. On the other hand, students appreciate "Circuits labs using LabJack" very much.

Interestingly, students used to value homework exercises more than doing similar problems in class. More recently, however, this gap has been closing. It is also quite possible, as evident in some student comments, that students are not used to active-learning mode of instruction. Finally, it is interesting to note that students like the MATLAB e-book better than the paper textbook used in 2011. There has also been a quite a positive change in student attitude about these readings, which we attribute to better organization. We will continue to monitor how all of these items develop over time.

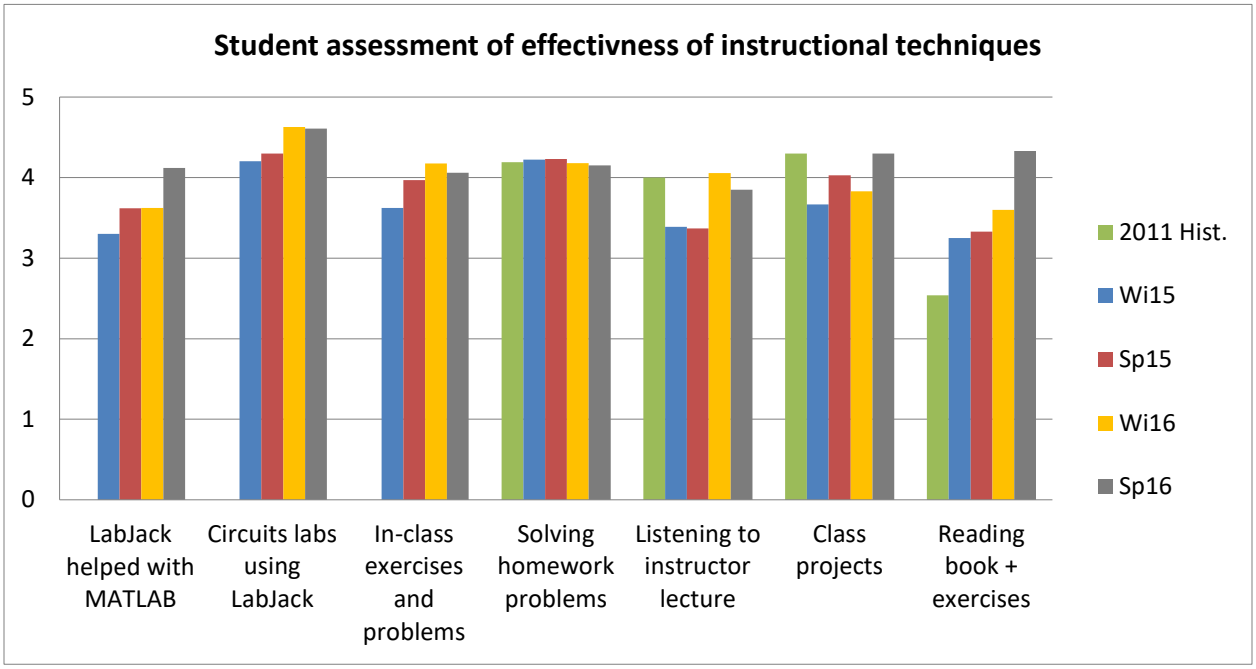

Figure 3. Effectiveness of instructional techniques as determined by the end-of-term survey from 2011 (historical), Winter & Spring 2015, Winter & Spring 2016.

4.3 Competency Testing

Two out of six student learning outcomes deal with MATLAB programming. In the rest of the curriculum, students are primarily using their existing programming skills, for example using C to program embedded systems chips. It is, therefore, critical that students get a solid foundation and practice in basic programming skills. Many students find programming very hard and end up with a piecemeal understanding of it [18][19]. Another issue is that due to the possibility of partial credit for other work, they may devote less time and effort to programming. One potential solution is to introduce competency or proficiency testing (CT) [20], which aspires to provide the following benefits:

- Ensures that students develop a solid programming foundation
- Provides explicit and detailed guidance on what is expected
- Provides useful, timely, and frequent feedback to students
- Improves the effectiveness of our teaching

To accomplish these, we had to implement significant changes in our teaching, with the most significant being introduction of programming labs. Other techniques to improve programming

competency include use of the MATLAB e-book, in-class exercises, and homework assignments. Many of these exercises are relatively small, and we need train students to write larger pieces of code. This is accomplished through separate homework assignments, labs, and the final project.

There are two major CT events, and students are given two chances to pass each one (see Table 3). Competencies covered are given below. The first three are on the first test and all six are on the second test.

1. Variable usage
2. Vector manipulation
3. I/O functions
4. Branching
5. Loops
6. Function definition & calling

Testing for the first three competencies consists of simple programming tasks that take several lines of code to accomplish. Students have to produce correct intermediate steps and final results. Problems for the second test involve 20-30 lines of code and require some thinking and planning but can be accomplished in 45 minutes. Student competency is determined by direct observation of their programming performance during an in-lab test. Students who do not pass CT1 or CT2 automatically fail the class. Those who pass CT get the grade based on other segments of the class.

Our results for 2015-16 academic year are shown in Figure 4. The passing rates for the winter and spring quarters were 85% and 92%, respectively. While the numbers are respectable, we still believe that the overall level of student programming skills needs to be improved further.

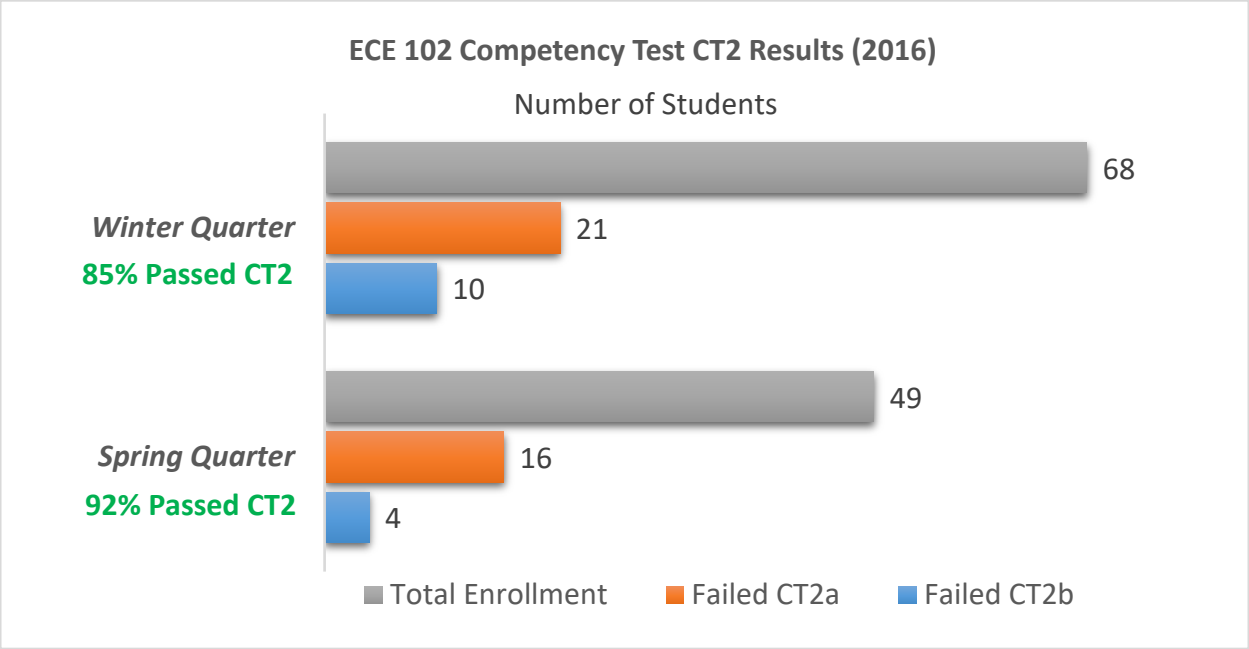

Figure 4: Competency Test results for the 2015-16 academic year

Other features of CT include:

- It makes expectations clear to students but has to be supplemented with other improvements in teaching.
- Students benefit by making sure they actually mastered the basics and can perform programming tasks before moving on to more complex concepts and courses.
- Implementation requires additional resources, such as trained TAs and helpers.

The CT process and results have made it clear that we need to focus our attention on fundamental programming concepts and avoid covering too many topics until students are firmly grounded.

5. 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 started incorporating more active learning tools, such as an interactive online textbook from zyBook and real-time assessment of student responses to questions using the web-based Learning Catalytics. This was followed up by the introduction of programming labs, reformulated projects and project management, and competency testing. We have presented our implementation and discussed some problems that have arisen and possible solutions. We hope that some of our experience can be helpful to others teaching similar courses. Major lessons learned include:

1. assessing students' pre-existing knowledge and abilities is very important but still underdeveloped,
2. we need to focus on only the most important programming concepts,
3. careful sequencing of material is critical, especially for a very busy schedule like ours,
4. in-class activities are very important for immediate formative feedback and to expose weaknesses early on,
5. assigned e-book readings are helpful but we need to carefully monitor student performance,
6. competency testing can be successfully organized and is useful in sending a clear message about expectations.

The course is still under development, and we are exploring additional pre- and post-assessments of programming skills which we hope will help us further refine the course and lead to some changes on the curricular level. Despite the initial difficulties of incorporating various instructional tools, we feel that the time and effort invested in these changes was well worth it.

Acknowledgments

We would like to thank Alex Edgcomb of Zyante for providing us with data on earnestness in our courses and for helping us better analyze what is going on.

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, M.W. Bridges, M. DiPietro, M. Lovett, and M.K. Norman, *How Learning Works: Seven Research-Based Principles for Smart Teaching*. John Wiley & Sons, 2010.

[3] P. Wong, M. Holtzman, B. Pejcinovic, M. Chrzanowska-Jeske, "Redesign of Freshman Electrical Engineering Courses for Improved Motivation and Early Introduction of Design," 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," 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] Details available at learningcatalytics.com

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

[8] A. D. Edgcomb, F. Vahid, R. Lysecky, A. Knoesen, R. Amirtharajah, and M. L. Dorf, "Student Performance Improvement using Interactive Textbooks: A Three-University Cross-Semester Analysis," ASEE Annual Conference & Exposition, 2015, p. 26.1423.1-26.1423.17.

[9] M. L. Loughry, M. W. Ohland, and D. D. Moore, "Development of a Theory-Based Assessment of Team Member Effectiveness," Educational and Psychological Measurement, vol. 67, no. 3, pp. 505–524, Jun. 2007.

[10] T. Kijewski-Correa, M. Ohland, L. McWilliams, K. Meyers, and S. Silliman, "Comparison Of Two Peer Evaluation Instruments For Project Teams," ASEE Annual Conference & Exposition, 2008, pp. 13.315.1–13.315.19.

[11] B. A. Oakley, D. M. Hanna, Z. Kuzmyn, and R. M. Felder, "Best Practices Involving Teamwork in the Classroom : Results From a Survey of 6435 Engineering Student Respondents," IEEE Transactions on Education, vol. 50, no. 3, pp. 266–272, 2007.

[12] H. Kniberg and M. Skarin, Kanban and Scrum - making the most of both. lulu.com, 2010.

[13] R. B. Bass, B. Pejcinovic, and J. Grant, "Applying Scrum project management in ECE curriculum," IEEE Frontiers in Education (FIE), Erie, PA, USA, 2016.

[14] Trello on-line project management software www.trello.com, accessed 2/11/2017

[15] B. Pejcinovic, D. Duncan, P.K. Wong, M. Faust, and G. Recktenwald, "Assessment of Student Preparedness for Freshman Engineering Courses Through Assessment of Math Background," Frontiers in Education (FIE), Madrid, Spain, 2014.

[16] J. Yuen, A. Edgcomb, and F. Vahid. "Will Students Earnestly Attempt Learning Questions if Answers are Viewable?," ASEE Annual Conference, New Orleans, LA, USA, 2016. DOI 10.18260/p.27205 https://peer.asee.org/27205

[17] A. Edgcomb, Zyante Inc., private communication.

[18] T. Jenkins, “On the difficulty of learning to program,” in Proc. of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences, 2002, vol. 4, pp. 53–58.

[19] M. J. Scott and G. Ghinea, “Educating programmers: A reflection on barriers to deliberate practice,” in Proc. 2nd HEA Conf. on Learning and Teaching in STEM Disciplines, 2013, p. 028P.

[20] K.S. Fant, B. Pejcinovic, and P. Wong, “Exploring Proficiency Testing of Programming Skills in Lower-division Computer Science and Electrical Engineering Courses,” ASEE Annual Conference & Exposition, New Orleans, LA, USA, 2016.