The Use of MATLAB Live as a Technology-enabled Learning Environment for Computational Modeling Activities within a Capstone Engineering Course

Mr. Joseph A. Lyon, Purdue University, West Lafayette

Joseph A. Lyon is a Ph.D. student in the School of Engineering Education and a M.S. student in the School of Industrial Engineering at Purdue University. He earned a B.S. in Agricultural and Biological Engineering from Purdue University. His research interests include models and modeling, computational thinking, and computation in engineering education.

Ms. Aparajita Jaiswal, Purdue University, West Lafayette

Aparajita Jaiswal is a Ph.D. student at Purdue Polytechnic at Purdue University, West Lafayette. Her research interests are in educational technology, embodied learning, student engagement and motivation in active learning environments.

Dr. Alejandra J. Magana, Purdue University, West Lafayette

Alejandra Magana is a Professor in the Department of Computer and Information Technology and an affiliated faculty at the School of Engineering Education at Purdue University. She holds a B.E. in Information Systems, a M.S. in Technology, both from Tec de Monterrey; and a M.S. in Educational Technology and a Ph.D. in Engineering Education from Purdue University. Her research is focused on identifying how model-based cognition in STEM can be better supported by means of expert technological and computing tools such as cyber-physical systems, visualizations, and modeling and simulation tools.

The use of MATLAB Live as a technology-enabled learning environment for computational modeling activities within a capstone engineering course

Abstract

This full paper presents an implementation of a technology-enabled learning environment such as MATLAB Live, used to enhance student experience when engaging with computational modeling activities within a capstone engineering course. Computational modeling and simulation are key aspects of engineering education. As such, continued progress towards understanding and improving student experiences within computational modeling activities is paramount for engineering educators. MATLAB Live, allows the users to mix together data visualizations, working code, explanations, and other forms of media. This form of discipline-based technology-enhanced learning environment allows for maximizing scaffolding and potentially lessening the cognitive load experienced during the learning activity.

The paper describes how a computational modeling intervention based on educational theories and frameworks such as the model-eliciting activity and productive failure, was scaffolded and delivered through MATLAB Live. The paper quantitatively and qualitatively identified different ways in which students engaged with MATLAB Live and how those differed between student programmers' comfort levels. Additionally, quantitative analysis was used to understand the effects the intervention had on student self-efficacy. The guiding research questions were: (1) How did such technology-supported scaffolded (MATLAB Live) modeling activity experiences impact student self-efficacy regarding programming and computational modeling? (2) Based on student comfort level with programming (self-efficacy), how did students vary in their reported experiences of MATLAB Live? The results of this analysis show that the MATLAB Live scaffolding proved beneficial to both novice and experienced programmers, yet student-reported benefits differed in key areas. The paper concludes with recommendations for using MATLAB Live and similar programs derived from the results of the study.

Introduction

Modeling and simulation are becoming a more fundamental skillset across engineering disciplines and other disciplines more broadly. There have been multiple national calls for increased computation and computational thinking within the classroom [1], [2]. However, within engineering contexts, it has been shown that computation and computational thinking are best taught when embedded within disciplinary contexts [3], [4]. With the national calls for discipline-based education research, the time has never been better to understand how computation can be taught in new and innovative ways within the engineering classroom. However, teaching computation and programming within the engineering classroom does not come without challenges. Often times, engineering instructors are dealing with overpacked curriculum, students who lack confidence or ability within their programming, or lack familiarity with the programming environments available and useful to students today [5]. Furthermore, gaps continue to exist within racial, gender, and socioeconomic differences [6]. However, recent work has shown that integrating programming into the curriculum through the lens of popular applications of its use can help broaden participation within computing disciplines [7].

Innovations in computing environments have given rise to technology-enhanced learning within programming environments such as MATLAB Live. These scaffolding techniques allow students to directly input equations, text, graphs, and results directly next to the code being written. Additionally, instructors can give instructions or break the programming up into sections directly within the programming environment. Similar programs have been used within research to scaffold computer programming learning [8].

All this has led to the current study, where a discipline-based computing intervention was implemented into a capstone level engineering course, in the form of modeling and simulation. The study was conducted after seeking and obtaining permission from the university institutional review board (IRB). The researchers then took pre-survey and post-survey measurements to understand how self-efficacy changed in terms of students modeling and simulation skills. Likewise, post-survey data was collected to understand how students experienced the MATLAB Live environment. This has led the research to two research questions: (1) How did such technology-supported scaffolded (MATLAB Live) modeling activity experiences impact student self-efficacy regarding programming and computational modeling? (2) Based on student comfort level with programming (self-efficacy), how did students vary in their reported experiences of MATLAB Live?

Background

The use of modeling is not new to engineering education, having been studied extensively with all levels and disciplines of engineering [3], [9], [10]. For this study, two bodies of literature were primarily used to design the learning intervention for which MATLAB Live was to be implemented within model-eliciting activities (MEAs) and modeling-based learning. The structure of this learning intervention has been previously reported in the literature [11]. This learning intervention was then investigated for impact on student self-efficacy.

Modeling-based learning

Modeling-based learning, or the use of models for learning in educational environments, has been studied broadly in education and specifically STEM education for a long time [12]–[14]. Traditionally, many researchers have tried characterizing the modeling cycle into a series of steps or phases. For example, Shiflet and Shiflet [15] described the modeling process into six phases: (1) analyze the problem, (2) formulate a model, (3) solve the model, (4) verify and interpret the model's solution, (5) report on the model, and (6) maintain the model. Magana [16] broke the process into (1) construct models, (2) use models, (3) evaluate models, and (4) revise models. Similar modeling processes were described by Louca and Zacharia [17], which directly informed the design of this study. Their modeling process was: (1) collection of observations and experiences, (2) construction of the model, (3) evaluation of the model and (4) revision of the model. This four-phased process is referred to as modeling-based learning (MbL). During *Collection of Observations and Experiences*, students work to build off of their own experiences and collect any necessary data needed to build the model. Often during this phase students begin to look into the details of the physical phenomenon that they intend to model. Then, in the second phase of *Construction of the model*, students use that information to formally build the model symbolically by putting to work the information and experiences that were gathered during the first phase of the process. During the *Evaluation of the Model* step, students take their

built model and evaluate through verification and validation. And then finally, in *Revision of the Model*, students modify the model based on the evaluation.

Self-efficacy within Engineering

It has long been known that self-efficacy can have an impact on student learning [18], [19]. However, more recent studies have looked into the effects of self-efficacy within the engineering classroom [20], [21]. Within engineering classrooms, gender gaps in self-efficacy have been observed [21]. When looking at first-year engineering students, Hutchison et al. [21] found that there were nine primary factors that influenced student self-efficacy within the engineering classroom: (1) understanding or learning, (2) drive and motivation, (3) teaming, (4) computing abilities, (5) availability of help, (6) problem-solving abilities, (7) enjoyment, (8) interest, and (9) satisfaction with the course material. In this study, we dive deeper into the computing abilities factor to understand how engaging with a modeling-based learning experience impacts student self-efficacy and how initial self-efficacy can impact student experiences with scaffolding frameworks such as MATLAB Live within computing environments.

Methods

This study utilizes a mixed-methods approach to analyze the impacts of a model-based learning experience to self-efficacy, as well as the effects of self-efficacy on experiences with a technology-enabled learning environment such as MATLAB Live.

Participants

The institution from which data was collected was a large midwestern land-grant university, with a large engineering program. Participants were primarily senior-level undergraduate students enrolled in a capstone course overviewing food and pharmaceutical process engineering. The majority of students had some experience with computer programming, as well as many of the prerequisite engineering courses needed to complete a senior design course for their major (thermodynamics, fluid mechanics, heat and mass transfer, etc.). The class had slightly more females than males and the sample for this study is approximately representative of the entire class (n=25, 14 females, 11 males)

For the course, students were expected to develop computational models for various physical phenomena (food canning line, freeze-drying process, etc.). Additionally, external to these modeling projects, students had a senior design project they worked on, as well as periodic homework assignments and quizzes throughout the semester. The class had both a lecture and lab portion, with the lecture portion being traditional lecturing with assigned homework and quizzes, with the lab section of the course being the time for modeling projects and the senior design project.

Learning Design

The final learning design was developed based on modeling-based learning. The development of a four-phase process from these frameworks has previously been reported on [citation blinded for peer review]. The four phases of the modeling process that students used during their modeling activities were: (1) planning the model, (2) building the model, (3) evaluating the model, and (4) reflecting on the model. Table 1 below overviews the tasks that students did during each phase of the modeling process.

Table 1. Overview of learning design for the modeling projects during the course.

Phase	Student Activity	Deliverable
Planning the model	Students meet in teams to determine the equations, properties, and solution elements that are needed to solve the problem.	Each student individually turns in a plan of their computational model.
Building the model	Students individually (over 2-3 weeks) build their model using MATLAB Live templates.	Each student individually turns in a MATLAB Live file of their computational model.
Evaluating the model	Students meet with their original team as well as rotate to meet with other teams to understand the differences between their own and other models.	Each student individually turns in a write-up that indicates identified differences between the different models seen.
Reflecting on the model	Students individually reflect on how they would change their model in the future and what they would do differently during the modeling process.	Each student individually turns in a reflection report.

Students were asked to list their assumptions and answer disciplinary questions from within the MATLAB Live file. Additionally, they were asked to include results within the same file. Whereas previously students may have had to write a report and turn in their code separately, MATLAB Live allows for students to be able to answer questions and explain their solution right next to the code itself.

Data Collection

A pre-survey looking at the self-efficacy of students was administered prior to the initial modeling activity and a post-survey looking at both the self-efficacy as well as the self-reported experiences of the students regarding MATLAB Live. Each survey question used a five-point Likert scale. Table 2 lists the questions regarding student self-efficacy around computational modeling.

Table 2. Pre-survey and post-survey questions evaluating student self-efficacy.

		Very confident	Fairly confident	Neither	Not very confident	Not at all confident
Q1	I have the ability to model systems with MATLAB					
Q2	I have the ability to design an algorithm					
Q3	I have the ability to write a computer program					
Q4	I have the ability to visualize data using a computer					
Q5	I have the ability to implement a graphical user interface					

Additionally, student self-responses regarding experiences with MATLAB Live software were measured using additional survey questions after the final modeling assignment of the semester. The survey had five multiple-choice questions to evaluate student experiences as well as multiple open-ended questions for students to explain their ratings. Table 3 outlines both the multiple-choice and open-ended survey questions.

Table 3. Post-survey questions looking at MATLAB Live experience.

		Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
Q1	I believe MATLAB Live was more beneficial to my learning than traditional programming environment.					
Q2	I believe MATLAB Live made the coding process easier for me than a traditional programming environment					
Q3	I believe MATLAB Live enhanced my ability to visualize my computational model.					
Q4	I would prefer to use MATLAB Live as opposed to traditional programming environments on future projects.					
Q5	I would recommend the use of MATLAB Live to other instructors for their classrooms.					
Q6	What do you feel were the biggest benefits to using MATLAB Live as opposed to traditional programming environments during this course?					
Q7	What do you feel were the biggest challenges to using MATLAB Live as opposed to traditional programming environments during this course?					
Q8	How could your instructors make these challenges easier to overcome in the future?					

The results of these surveys were then analyzed to understand the effects that the learning design had on student self-efficacy as well as what students' experience with MATLAB Live had been as a scaffolding framework.

Data Analysis

The study used descriptive and inferential statistics to answer the research questions. Paired *t*-tests were used to answer the first research question and intended to explore the difference in students' prior experience with computing and self-efficacy in programming ability. Students' responses on the pre-test and post-test regarding their self-efficacy were compared at $\alpha = 0.95$. Any students who did not complete all survey data were not included in the data sample. Table 6 and Figure 1 demonstrate the results obtained from the *t*-tests. Cohen *d* effect size was used to compare the effect of the intervention on student self-efficacy. We considered a strong effect size when $d \geq 0.8$; a medium effect size when $d = 0.5$ and weak effect size when $d = 0.2$ [22].

An exploratory data analysis was conducted to answer the second research question. The intent of the exploratory data analysis was to understand the relationship between the student comfort level with programming, in general, and the reported experience the students had with the MATLAB Lab software. In order to understand the student programming comfort level, the pre-test self-efficacy data was classified into two categories: *comfortable* and *uncomfortable*, based on the median of student mean self-efficacy score. Table 4 shows the breakdown between comfortable and uncomfortable programming categories.

Table 4. Categorization of Student Comfort Level

Comfort Level	**Mean Score**
Comfortable	> 3.4
Uncomfortable	≤ 3.4

Although nearly all students found MATLAB Live to be beneficial in their learning, to further explore the impact of the MATLAB Live software on the students, we classified the MATLAB Live experience survey responses into three categories: *very beneficial, beneficial, and somewhat beneficial* based on 1/3 and 2/3 quantiles. Table 5 demonstrates the categories of the student experience by student comfort level with programming.

Table 5. Categorization of Student Experience with Matlab Live

Student Experience	**Mean Score**
Somewhat Beneficial	< 3.8
Beneficial	$3.8 < \leq 4.2$
Very Beneficial	> 4.2

Results

Effects of intervention on self-efficacy

The results of the analysis showed that students had statistically significant gains reported on both Q1 regarding students' ability to model systems with MATLAB, and Q2 regarding students' ability to design and algorithm ($p<.05$) after participating in our designed intervention scaffolded with MATLAB Live. Additionally, students saw increases in Q3 regarding their ability to write a computer program, Q4 regarding their ability to visualize data using a computer, and Q5 regarding their ability to implement a graphical user interface, although these were not statistically significant increases. Taking all of the scores as a composite self-efficacy score and comparing between pre and post surveys also yielded a significant difference ($p=0.02$*). Figure 1 shows the pre and post-survey mean scores and marks significant differences between pre and post surveys.

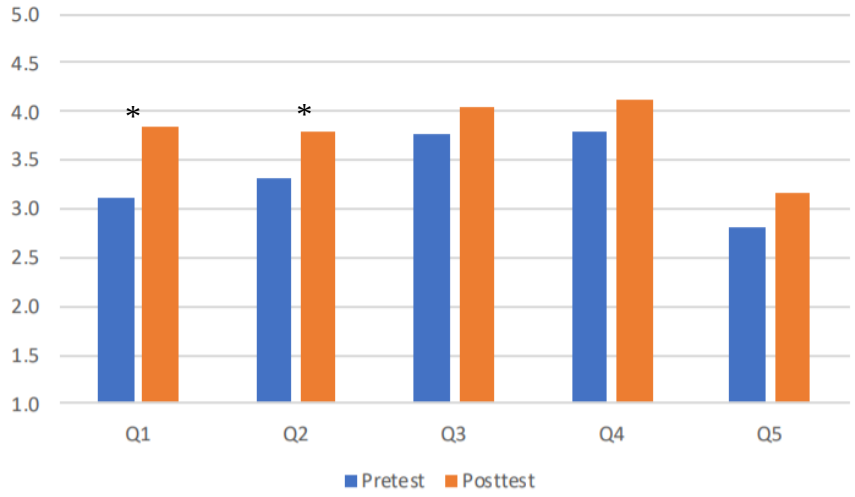

Figure 1. Self-efficacy gains (* denotes statistical significance)

From Figure 1, it is apparent that there were gains across the board for the students as a whole. The researchers looked to see if there were any statistically significant differences between student self-efficacy based on reported gender, however, no significant differences were seen in either the pre-survey or post-survey data.

Table 6. Results of the paired t-test of student self-efficacy before and after the intervention.

Pretest		**Posttest**		**Paired *t*-test**			
Mean	**S.D.**	**Mean**	**S.D.**	**t**	**df**	***p*-value**	**Effect size**
3.12	0.70	3.84	0.85	-2.33	24	0.02*	0.55

Student experience with MATLAB Live software

Additionally, student experiences with the MATLAB Live software were investigated to understand how the scaffolding framework can best be used in the classroom, as well as the reported benefits, challenges, and best instructional methods when using the MATLAB Live software to scaffold computational modeling and simulation problems within an engineering classroom.

Figure 2 details the responses of the students for each question on the MATLAB Live experience survey.

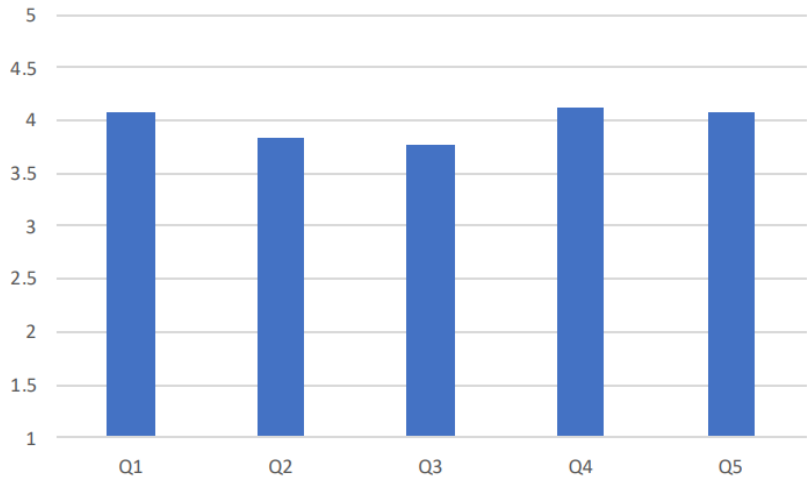

Figure 2. Mean responses to the MATLAB Live Experience survey.

From Figure 2 it could be inferred that students generally responded very favorably to the MATLAB Live framework, with the mean score for each question falling between 3.5 and 4.5. However, there were students that found there were some drawbacks to MATLAB Live and fell into the somewhat beneficial category for experiences with the program. Table 7 below shows the breakdown of students in each category and the reported gender breakdown in each of the categories.

Table 7. Student experience by student comfort level with programming and gender.

	Very Beneficial	**Beneficial**	**Somewhat Beneficial**
Comfortable	3 (3f)	4 (2m,2f)	5 (3f,2m)
Uncomfortable	5 (4m,1f)	3 (2f, 1m)	5 (3f, 2m)

*f is female, m is male

As can be seen in Table 7, there were multiple students that fell into both the comfortable and uncomfortable categories given their prior programming self-efficacy. There are no immediate trends in the data, as students of both comfort levels had reported MATLAB Live to be very beneficial while others reported MATLAB Live to be only somewhat beneficial. Upon initial inspection, it appears that there could be some diminishing benefits for programmers who reported they were comfortable with programming and computational modeling, as comfortable students are skewed towards somewhat beneficial experiences with MATLAB Live.

In order to better understand why students had either high or low reported benefit to the use of MATLAB Live, qualitative questions were analyzed for common reasons both among students of low and high comfort with programming to bring out the biggest themes as to benefits students saw to the software package and the biggest challenges they saw.

Benefits

The most prevalent benefit that students reported on was the (1) organization it provides, (2) the ability to put text and graphs directly next to the code, and (3) the ability to easily identify and debug errors in the code. Regarding the organization it was able to provide students wrote that:

> *Organization is more visually striking so its easier to follow. – Student K*
> *It organizes the code way better and allows for write-up sections. – Student X*

Therefore, students felt that the MATLAB Live framework better organized their solutions as opposed to traditional learning environments. Along the same lines, students wrote that the solutions were more easily debugged, likely stemming from this increased organization. Regarding debugging their solutions, students wrote that:

> *[MATLAB Live] allowed users to see their work as they went so problems could be found earlier and quicker. -Student R*
> *You can see results immediately and can see what series of code has errors…-Student AO*

Additionally, students reported that the ability to easily switch between code and text was useful. This again may stem from the organization, in that having these two entities next to each other allows for less cognitive processing. Students wrote that:

> *I liked being able to easily switch between text and code. – Student Q*
> *I could code and type when I needed to, which just made things easier. – Student F*

Previous research has reported that proximity and limiting the amount of representational holding that students must do in their minds [23]. MATLAB Live puts the graphs, text, and code all into one place, thus students no longer have to move between documents, remembering what they have seen on previous screens but rather get all information in one place.

Challenges

The students, however, did list a few challenges associated with their use of MATLAB Live as opposed to traditional programming environments. The most prominent challenges students reported facing were (1) the running time of code seemed longer, (2) basic understanding of the features of MATLAB Live, and (3) support of MATLAB Live files across different versions of MATLAB. The biggest and most prominent challenge that a student faced was processing time of their code within the MATLAB Live files, writing:

> *While the live scripts had a few nice features, they were much much slower than just a standard [MATLAB] file. - Student Z*
> *It took way to long to run. I had to copy and paste graphs because my code never finished running…-Student H*

Students reported long computing times for their MATLAB Live scripts, and thus even in some instances circumnavigated the software in order to avoid these long computation times. This could have been related to the hardware students were running, or potentially trying to run the software through remote software on campus. Students reported that MATLAB Live did not easily integrate with all version of MATLAB code, writing:

My computer did not support [MATLAB Live] so I had to come to campus to code. – Student AO
Sometimes it was hard to save and send files between older version of MATLAB, software remote, and [building] computers. – Student Q.

These version differences are well-known limitations of MATLAB Live, however, the students may not have been as knowledgeable as they could have been of these limitations. Students reported overall struggling to pick up all of the new features of MATLAB Live writing:

[A challenge was] getting new users familiar with the capabilities of MATLAB Live. – Student R
The steps were a bit confusing, as to what would go where and why. -Student V

Overall this may point to additional training students need prior to the intervention in order to both understand the limitations of MATLAB Live but also get acquainted with the features that MATLAB Live has to offer.

Discussion

These results have multiple implications for continued research as well as for teaching and learning of programming and computational modeling. Additionally, feedback from students leads to recommendations for future implementations of using technology-enabled learning environments such as MATLAB Live in the future.

Self-efficacy in engineering and computing

Our implementation within this capstone class along with the use of MATLAB Live increased self-efficacy gains significantly within the class. This aligns with the literature in that mastery experiences, which is in our case completing a programmed computational model, often are effective factors in increasing self-efficacy [18], [21]. Additionally, the students worked in teams and had the opportunity to see the computational models of other students, thus seeing other students succeed in building their own computational models. By doing so this provided vicarious experiences for the students, thus also reinforcing each student's self-efficacy [18].

Our results did not indicate any significant differences between males and females either in pre-survey or post-survey. Numerous studies have indicated a significant difference between females and males in terms of self-efficacy and confidence regarding programming and computational modeling [24], [25]. Our lack of difference between genders could be due to the fact that the students within this study were upper-division undergraduates, where both females and males had previously completed mastery experiences that narrowed any previous gap. Additionally, the small sample size may have made it difficult to detect any small differences that may have remained.

MATLAB Live as a technology-enhanced learning framework

It appears that students overall found that MATLAB Live was a beneficial framework for integrating computational modeling into a capstone engineering course. This speaks to the fact

that we saw significant self-efficacy gains with computational modeling and programming. Therefore, the students did, in fact, feel more comfortable programming computational models at the end of the semester. There are multiple reasons that MATLAB Live may be useful in supporting student learning and self-efficacy in computational modeling and programming, including the potential lessening of cognitive processing and the increased organization.

MATLAB Live may allow students to lower some of the cognitive processing that is needed in order to solve the complex modeling problems encountered by engineering students in a capstone engineering course. Namely, proximity and the elimination of representational holding between screens are ways to lower cognitive processing [23]. In doing so, students are more able to focus their cognitive processing on the essential processing that is needed to complete the task. However, there was a slight decrease in benefits to the students most comfortable with programming. This suggests that we may be observing the expertise reversal effect, or that the MATLAB Live environment added cognitive load to those comfortable with programming [26].

Additionally, the organization allows users to more easily find sections of their code, but also potentially eliminates extra time in the debugging process as well. Any time spent searching for areas of code or searching for errors in the code is time that is potentially pulled away from solving the actual problem. In conjunction, every time a student had trouble finding a section of their code, they disengaged with the actual solution process, which they must have reentered in order to continue solving the problem.

Given some of the challenges that students listed regarding MATLAB Live, it is apparent that not enough pre-training was given to students in order to lessen the cognitive processing required to complete the assignment. Pre-training is often able to lessen the amount of information that a student needs to process at any given time during a learning task [23]. Pre-training may help eliminate some of the frustrations with moving files between versions of MATLAB, as well as the initial issues students experienced with learning the new features of the software.

Implications for teaching and learning

Overall, this study has many implications for instructors who are looking to integrate programming into their engineering courses. First, the instructional design using modeling-based learning along with the technology-enabled learning environment of MATLAB Live seemed to promote student learning of the material and students felt more confident in their programming abilities as this set up provided mastery experiences for the students, embedded in real-world contexts. If instructors are looking to integrate modeling and simulation exercises into upper-division engineering classrooms, this set-up provides both the necessary scaffolding for the programming activities while providing openness with the disciplinary material so that students can fully explore the solution space.

However, when using MATLAB Live instructors must be aware of the learning curve for operating the software, in order to avoid causing too much cognitive load on the student while they are doing the assignment. Students spoke to the need for adequate pre-training of the software to lessen the amount of time and challenge it took to learn the new features of the program. Whereas pre-training is needed, the payoff seems well worth it in that students overall

found the program very beneficial in the organization and condensing of report writing that is provided.

Conclusion

Moving forward, instructors must find ways to incorporate programming into the engineering classroom and computational modeling provides an efficient way to teach programming along with disciplinary content. One limitation of this study is the relatively small sample size. However, the study provides useful insights into the ways that technology-enabled learning evironments such as MATLAB Live are useful tools in helping promote learning of programming in educational settings. However, instructors should be cautious when using this or similar technology-enabled learning environments to make sure students have enough pre-training, or else they may just add more cognitive load to the students learning process. Yet, if implemented properly, modeling interventions along with these technology-enabled learning environments can significantly impact student self-efficacy moving forward on computational modeling and programming tasks.

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. (DGE-1842166) as well as the National Science Foundation under Grant No. (EEC-1449238). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] PITAC, "Computational science: Ensuring America's competitiveness." President's Information Technology Advisory Committee, Arlington, VA, 2005.

[2] [NRC], "Report of a Workshop on the Pedagogical Aspects of Computational Thinking," National Academies Press, Washington, D.C., 2011.

[3] A. J. Magana, M. L. Falk, and M. J. Reese Jr., "Introducing Discipline-Based Computing in Undergraduate Engineering Education," *ACM Trans. Comput. Educ.*, vol. 13, no. 4, pp. 16:1-16:22, 2013.

[4] J. D. Ortega-Alvarez, W. Sanchez, and A. J. Magana, "Exploring Undergraduate Students' Computational Modeling Abilities and Conceptual Understanding of Electric Circuits," *IEEE Trans. Educ.*, vol. 61, no. 3, pp. 204–213, 2018.

[5] A. J. Magana and G. Silva Coutinho, "Modeling and simulation practices for a computational thinking-enabled engineering workforce," *Comput. Appl. Eng. Educ.*, vol. 25, no. 1, pp. 62–78, 2017.

[6] A. Scott, F. McAlear, A. Martin, and S. Koshy, "Broadening participation in computing: Examining experiences of girls of color," *ACM Inroads*, vol. 8, no. 4, pp. 48–52, 2017.

[7] C. Alvarado, Z. Dodds, and R. Libeskind-Hadas, "Women ' s Participation in Computing at Harvey Mudd College," vol. 3, no. 4, pp. 55–64, 2012.

[8] A. Madamanchi, M. Thomas, A. Magana, R. Heiland, and P. Macklin, "Supporting Computational Apprenticeship through educational and software infrastructure . A case study in a mathematical oncology," *bioRxiv*, pp. 1–28, 2019.

[9] H. A. Diefes-Dux, T. Moore, J. Zawojewski, P. K. Imbrie, and D. Follman, "A framework for posing open-ended engineering problems: Model-eliciting activities," in *Proceedings*

of the 34th ASEE/IEEE Frontiers in Education Conference, 2004.
[10] A. J. Magana, H. W. Fennell, C. Vieira, and M. L. Falk, "Characterizing the interplay of cognitive and metacognitive knowledge in computational modeling and simulation practices," *J. Eng. Educ.*, vol. 108, no. 2, pp. 276–303, 2019.
[11] J. A. Lyon, A. J. Magana, and M. Okos, "WIP: Designing modeling-based learning experiences within a capstone engineering course," *ASEE Annu. Conf. Expo.*, 2019.
[12] R. Lesh, M. Hoover, B. Hole, A. Kelly, and T. Post, "Principles for developing thought-revealing activities for students and teachers," in *The handbook of research design in mathematics and science education*, A. Kelly and R. Lesh, Eds. Mahwah, NJ: Lawrence Erlbaum Associates, 2000, pp. 591–646.
[13] T. J. Moore, R. L. Miller, R. A. Lesh, M. S. Stohlmann, and Y. R. Kim, "Modeling in engineering: The role of representational fluency in students' conceptual understanding," *J. Eng. Educ.*, vol. 102, no. 1, pp. 141–178, 2013.
[14] H. A. Diefes-Dux, M. Hjalmarson, J. S. Zawojewski, and K. Bowman, "Quantifying aluminum crystal size part 1: The model-eliciting activity," *J. STEM Educ. Innov. Res.*, vol. 7, no. 1–2, pp. 51–63, 2006.
[15] A. B. Shiflet and G. W. Shiflet, *Introduction to computational science: Modeling and simulation for the sciences*. Princeton, NJ: Princton University, 2014.
[16] A. J. Magana, "Modeling and simulation in engineering education: A learning progression," *J. Prof. Issues Eng. Educ. Pract.*, vol. 143, no. 4, 2017.
[17] L. T. Louca and Z. C. Zacharia, "Modeling-based learning in science education: Cognitive, metacognitive, social, material and epistemological contributions," *Educ. Rev.*, vol. 64, no. 4, pp. 471–492, 2012.
[18] A. Bandura, "Self-efficacy: Toward a unifying theory of behavioral change," *Psychol. Rev.*, vol. 84, no. 2, pp. 191–215, 1977.
[19] A. Bandura, "Self-efficacy," in *Encyclopedia of human behavior*, vol. 4, V. S. Ramachaudran, Ed. 1994, pp. 71–81.
[20] G. S. Stump, J. C. Hilpert, J. Husman, W.-T. Chung, and W. Kim, "Collaborative Learning in Engineering Students: Gender and Achievement," *J. Eng. Educ.*, vol. 100, no. 3, pp. 475–497, 2011.
[21] G. M. Hutchison, M. A., Follman, Deborah K, Sumpter, Melissa, Bodner, "Factors Influencing the Self-Efficacy Beliefs of First-Year Engineering Students.," *J. Eng. Educ.*, vol. 95, no. 1, pp. 39–47, 2006.
[22] C. Carson, "The effective use of effect size indices in institutional research," in *Proceedings of the 2004 American Society for Engineering Education Annual Conference & Exposition*, 2012, pp. 41–48.
[23] R. E. Mayer and R. Moreno, "Nine Ways to Reduce Cognitive Load in Multimedia Learning Nine Ways to Reduce Cognitive Load in Multimedia Learning," *Educ. Psychol.*, vol. 38, no. 1, pp. 43–52, 2003.
[24] T. Busch, "Gender Differences in Self-Efficacy and Attitudes toward Computers," *J. Educ. Comput. Res.*, vol. 12, no. 2, pp. 147–158, 1995.
[25] M. Burnett *et al.*, "Gender differences and programming environments: Across programming populations," *ESEM 2010 - Proc. 2010 ACM-IEEE Int. Symp. Empir. Softw. Eng. Meas.*, 2010.
[26] S. Kalyuga, P. Ayres, P. Chandler, and J. Sweller, "The expertise reversal effect," *Educ. Psychol.*, vol. 38, no. 1, pp. 23–31, 2003.