Work in Progress - Mathematical software and programming preparation of undergraduate engineering students in mathematics courses

Mrs. Johannah L. Crandall, Washington State University

Johannah Crandall is a PhD student in Mathematics and Science Education, with research interests including undergraduate engineering education, especially transferable mathematical knowledge for use in modeling software and discrete solvers.

Dr. Kristin Lesseig, Washington State University Vancouver

Kristin Lesseig is an Associate Professor of Mathematics Education in the College of Education at Washington State University Vancouver. She earned her PhD at Oregon State University and currently teaches elementary and secondary mathematics content and methods courses as well as doctoral level courses focused on research in mathematics and science learning at WSUV. Kristin's research focuses on mathematical knowledge for teaching proof and the design of professional learning experiences that can support teachers' ability to promote mathematical reasoning in middle and high school classrooms. Kristin served as principal investigator on a 3-year Math Science Partnership grant designed to support middle grades teachers in implementing STEM Design Challenges aligned with CCSSM and NGSS content and practice standards and continues to collaborate on STEM-related projects.

Work in Progress: Mathematical software and programming preparation of undergraduate engineering students in mathematics courses

Introduction

This paper is reporting on work in progress investigating the perceived and actual contributions mathematics and engineering departments make to the software and programming preparation of undergraduate engineering students. Engineering students often must depend on multiple departments within a university for the various components of their degree program, including not only the department housing their core engineering courses, but also the mathematics department, among others. Recognizing that these departments can function differently, and that courses within them can focus on entirely disparate tool sets, this study draws on principles of situated cognition to frame questions about the development of computing proficiencies across disciplinary and departmental boundaries [1]. We investigate how mathematics courses which support the engineering curriculum may or may not contribute to important repeated and early exposure to software and programming tools in contextualized ways that help engineering students develop the ability to skillfully leverage domain-specific software, practice algorithmic thinking, and become familiar with the behavior and limitations of computational tools [2].

Even when engineering-inspired examples are used to motivate or practice a solution method in differential equations courses, for example, it may be that math instructors' treatment of those examples does not ultimately advance engineering students' understanding of how to engage with the problem when it arises in an engineering scenario [3]. For example, the goal in a typical mathematical treatment of a problem is to find the form of the 'solution,' often neglecting exploration of how the given problem behaves as a *system* undergoing variable inputs. Practical exploration of many system behaviors, however, requires computing tools, notably absent in the classrooms of this study's participants. Furthermore, if students encounter a tool too infrequently, they may also experience the burden of "relearning" the tool, and lack of facility with a programming platform can make its use in a mathematics setting a greater challenge than the actual mathematics being tackled [4].

Objectives

The objectives of the study are to characterize the present condition of computing within the mathematics curriculum at multiple institutions and its relationship to software and programming preparedness of undergraduate engineering students. The following research questions have guided data collection and analysis:

To what extent do engineering students encounter computing tools within their mathematics coursework that they perceive as industry-critical?

To what extent is computing within mathematics education perceived as relevant to engineering students' engineering coursework and future careers?

To what extent does computation within mathematics education contribute to engineering students' proficiency with domain-specific computing paradigms in the engineering curriculum?

Methods

Participants in the study included mathematics faculty and students in upper-division mathematics courses at two western United States public universities housing ABET-accredited engineering programs, referred to when necessary as universities I and II. The data reported were gathered during the fall of 2019.

Faculty participants. Mathematics faculty members who were currently teaching or until recently had taught upper-division mathematics courses aligned with typical engineering tracks completed semi-structured in-person interviews. Seven participants from university I and four from university II were able to meet for in-person interviews. One additional faculty member from II was not able to meet for an interview, but did provide a written summary of their thoughts on the interview topics, and their views have been included in the overall analysis. Interviews lasted an average of 15 minutes and most were conducted in the office of the interviewee. Audio recordings were made with permission of the participants and were later transcribed for thematic coding and analysis. The interview protocol for faculty participants is included in Appendix A. The interview was designed to gather information about the programming tools and languages faculty used in their courses and in their own work as well as elicit faculty perceptions of the tools they felt were most valuable in industry.

Themes within faculty responses were identified through a multi-step coding process. First, broad thematic coding was done after multiple readings of all transcripts, and often made use of in vivo codes [5]. For example, the Black Box theme emerged first as an in vivo code, as did the Language Agnostic theme. This first step produced over twenty possible thematic codes. After the initial round of potential codes were generated, transcripts were reread to locate thematic gaps and to identify which ideas could be grouped into larger themes, which ultimately resulted in a condensation of themes into six major categories, each of which emerged repeatedly in interviews with mathematics faculty: Software Development, Programming as Language Agnostic, Mathematical Considerations for Computing, Restrictions on Computing in Classrooms, Institutional Dynamics, and the Black Box theme. The major themes, together with topical subcategories that emerged in interviews, are displayed in Figure 1.

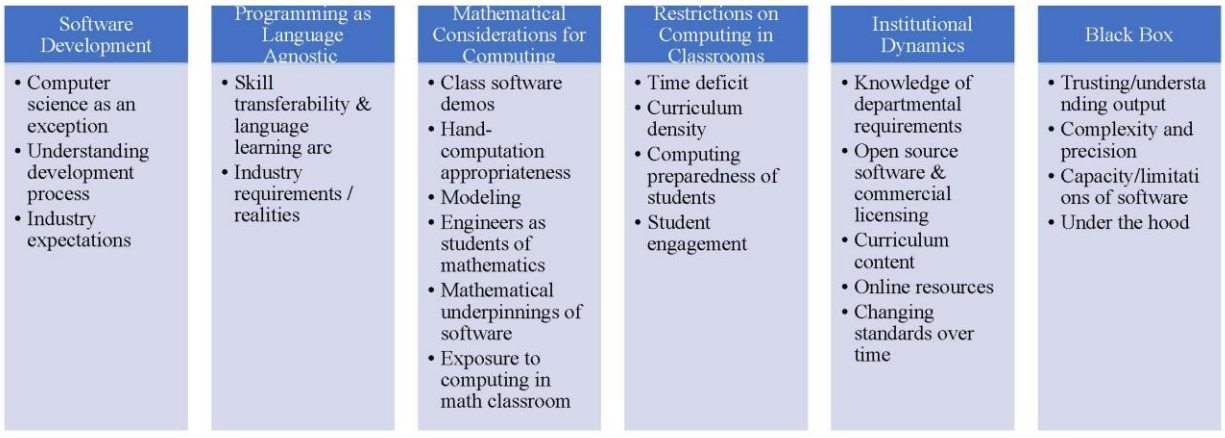

Figure 1 – Mathematics Faculty Interview Themes

Those faculty whose responses included the Software Development theme expressed the opinion that understanding in the development process was valuable to understanding the inner workings

of software tools. They also connected industry-preparedness with the ability to develop software. Programming as Language Agnostic thematic responses centered around the perceived ease of acquiring a second or more programming languages once one has been acquired, as well as around the generalizability of computing concepts beyond any one specific language. Faculty responses in the Mathematical Considerations for Computing theme included concerns that students need to understand the mathematical underpinnings of algorithms implemented in software packages like MATLAB and COMSOL, and that exposure to these things in mathematical contexts was valuable. However, many faculty responses themed as Restrictions on Computing in Classrooms highlighted perceived difficulties in that very mathematical exposure, including students' presumed lack of preparedness and the problem of fitting more into the curriculum. Similar concerns were closely related in the Institutional Dynamics theme, in which faculty cited unfamiliarity with engineering department software needs and the dynamic nature of industry requirements over time. Finally, and acutely, faculty responses in the Black Box theme included concerns that students do not currently consistently develop the means to evaluate output of mathematical software packages, and therefore are not equipped to gauge the trustworthiness or reasonableness of results.

Student participants. A subset of the faculty who agreed to participate were also asked whether the student survey instrument could be distributed to students in their current courses. This resulted in personal visits to six undergraduate classrooms (three differential equations, two mathematical computing, one numerical analysis) where the current study was explained to students and QR codes/URLs to participate in the study were distributed. Also, emails were sent by the instructors of one applied mathematics and one linear algebra course with the same survey information and access instructions to their classes. 78 student responses were received, of which 71 were retained after removing incomplete responses and responses from graduate students. Of those retained, 58 were from university I and 13 were from university II, 47 were engineering degree-seeking students, 8 were pursuing applied math, and 16 reported pursuing other degrees which included mathematics (6), math education (7), a physical science discipline (2), and marketing (1).

Student participants completed a 17-item online anonymous survey, which gathered information about degree, year in program, experiences with software and programming in various settings, and perceptions of industry-critical software tools. Responses were organized according to degree type and year in program to assist in identifying any trends in software experiences by sub-groups of students. The survey instrument for student participants is included in Appendix B.

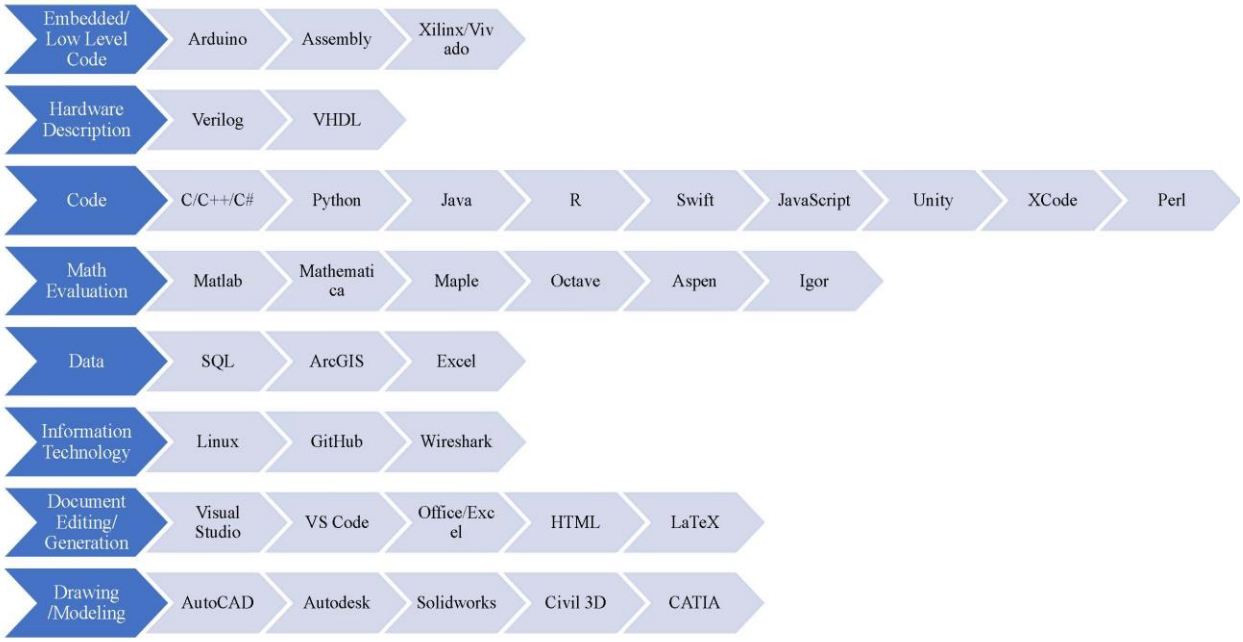

Figure 2 – Ontology of computing tools from engineering student survey responses.

Results

To what extent do engineering students encounter computing tools within their mathematics coursework that they perceive as industry-critical?

A generalized ontology of software/computing tools, displayed in Figure 1, was constructed from the various tools that student respondents (both engineer and non-engineer) provided regarding the survey item: *What software or programming tools do you think are important for you to know for your future career?* While this ontology does not completely exhaust every software or programming tool referenced by students, it captures a large percentage of them, especially those that were repeatedly mentioned. Tables 1, 2, and 3 illustrate the distribution of experience undergraduates within this sample reported with software tools from this ontology. Civil, environmental, chemical, mechanical, and materials science engineers are grouped in Table 1. Electrical and computer engineers as well as computer scientists are grouped in Table 2. Table 3 lists students in applied math, physical science, math, and math education.

The primary tool with which students were experienced in the Mathematical Evaluation category was MATLAB and the primary tools cited in the Code category were Python and C/C++. Almost all the students in the engineering curriculum report experience with MATLAB, however, only a small subset of engineering students consider it to be an industry-critical tool. Well under 10% of students in an engineering degree track reported engaging with a tool they listed as industry critical in a mathematics course, whereas over half of applied math students and about a third of all other degree types reported engaging with their industry-critical tools in upper-division math.

	Year in Program	Embedded/Low Level Code & Hardware Description	Algorithmic Code	Mathematical Evaluation	Data & Information Technology	Document Editing/Generation	Drawing/Modeling	Level of engagement with software in math settings	Discipline relevance of math software examples	Comments about software use in or out of classwork
Civil	Soph			x				none		
	Junior	x		x			x	little		
	Junior							none	mod	
	Junior		x	x			x	little	none	
	Junior			x				none		
	Junior			x				little	mod	
	Junior			x				little		
	Senior			x	x	x	x	none		
	Senior			x				little		
	Senior			x	x			mod		
	Senior		x	x			x	little		
	Senior		x	x			x	little	mod	
Env	Junior			x			x	mod	high	
ChemE	Junior							none		---> No because I don't use any software/programming tools
MSE	Senior			x				freq	mod	
ME	Soph		x		x			none		
	Soph				x		x	none		---> It would be very nice if more of then were free
	Soph		x	x	x			none	high	
	Soph		x	x	x	x		none		
	Soph			x			x	none	mod	---> I have only learned softwares in classes that explicitly teach the software
	Soph		x	x				little		
	Junior			x			x	little	none	
	Senior		x	x				none		---> I've only used Matlab extensively in my classes, with C appearing this semester. I am interested in pursuing software and programming tools outside of school and work, the dificulty is finding the time.
	Senior			x				freq		
	Senior	x		x		x	x	little		
	Senior		x					freq		

Table 1 - CE/Env/ChemE/MSE/ME software exposure by degree and year in program

To what extent is computing within mathematics education perceived as relevant to engineering students' engineering coursework and future careers?

Even if the tools engaged with in mathematics curriculum are not the precise tools engineering students anticipate encountering in their core engineering courses or in industry, the computing experiences gained in any academic setting are hopefully beneficial to their overall development as an engineer. This is most likely if the content of computing exercises in non-engineering courses are transferable to their engineering coursework.

	Year in Program	Embedded/Low Level Code & Hardware Description	Algorithmic Code	Mathematical Evaluation	Data & Information Technology	Document Editing/Generation	Drawing/Modeling	Level of engagement with software in math settings	Discipline relevance of math software examples	Comments about software use in or out of classwork
EE	Soph	x				x		none		
	Soph	x	x	x		x		none	high	
	Senior			x				freq	high	---> Excel is a good and easier tool
CompE	Soph	x	x	x		x		little		
	Junior			x		x		little		
	Junior	x		x	x	x		little		
CS	Soph		x	x				mod	none	
	Junior		x	x		x		little		
	Junior	x	x					none	none	
	Junior		x	x				none		
	Junior	x	x	x		x		high	high	---> They are helpful complament to the material. People should be familiar with the basics as they become more prevalent in our day to day
	Junior		x	x				freq	mod	
	Junior		x			x		none		
	Junior		x					none		
	Junior		x			x		none		
	Junior		x		x	x		freq	mod	---> I believe integrating the use of statistics software with some of the programming might be helpful for the future.
	Senior		x	x		x		mod		
	Senior		x					little		
	Senior		x	x				none		
	Senior							none	none	---> Yeah, [computation course] is a joke and there is no coding other than 1 extra credit assignment
	Senior		x	x				little		

Table 2 - EE/CompE/CS software exposure by degree and year in program

	Year in Program	Embedded/Low Level Code & Hardware Description	Algorithmic Code	Mathematical Evaluation	Data & Information Technology	Document Editing/Generation	Drawing/Modeling	Level of engagement with software in math settings	Discipline relevance of math software examples	Comments about software use in or out of classwork
Applied Math	Soph							none		
	Junior		x	x		x		freq	none	
	Junior			x		x		mod	mod	
	Senior		x	x		x		high	high	
	Senior		x	x		x		mod		
	Senior		x	x				freq	high	
	Senior		x	x				high	high	
	Senior		x	x		x		mod	none	---> Programming should be more integrated in classroom learning, vs the student being expected to teach themselves. This is only an issue in math courses.
Physical Science	Junior		x			x		mod		
	Junior							none		
Math	Junior		x	x				mod		
	Junior		x	x				high	high	
	Junior		x					none		
	Senior		x	x				little		
	Senior		x	x				little		
	Senior		x	x				mod		
Math ED	Fresh							none	mod	I wish I didn't have to use/learn any. I want to teach and its a scary technical skillset I don't plan on learning ever.
	Soph							none		
	Junior							none		--->
	Junior		x	x		x		freq	mod	
	Senior		x	x				freq	mod	
	Senior		x	x				high	none	
	Senior		x	x				mod	none	---> Why do teachers need to learn to code?
Marketing	Senior		x	x		x		little	mod	---> It would be nice to get some tutorials on IDEs occasionally...we usually have to figure those out ourselves.

Table 3 - Non-engineering track software exposure by degree and year in program

Tables 1, 2, and 3 show the extent to which students of the various engineering disciplines, as well as non-engineering students, felt the software-based examples in math courses were relevant to their future disciplinary coursework and industry. 35% of civil and mechanical engineers commented on the relevance of software examples, most of whom reported that software examples were moderately or even highly relevant to their coursework or future industry needs. 38% of EE/CompE/CS students reported on relevance of software examples, about two thirds of whom found examples moderately to highly relevant. Finally, half of non-engineering majors

reported on relevance of software examples, with the majority of those characterizing them as moderately to highly relevant.

To contextualize these responses further, it is necessary to notice: (1) many students reported no engagement with software in math contexts and thus had no reason to report on the level of relevance of examples; (2) we cannot infer the disciplinary relevance of math course computing experiences of 28% of the students, who reported engagement but failed to report relevance; and (3) there is not a direct connection between the level of hands-on experience a student reported with software in a math course and the level of relevance of that experience to their discipline. For example, CEs and MEs reported only minimal engagement with computing in math, but a comparatively strong level of relevance of examples used. Fewer students reported, conversely, a high level of engagement, but low relevance of examples. In broad terms, it appears that for this set of undergraduates, *if* they were given an opportunity to compute in a mathematics course, they often found the content of the examples, if not the tools themselves, to be relevant to their disciplinary studies.

To what extent does computation within mathematics education contribute to engineering students' proficiency with domain-specific computing paradigms in the engineering curriculum?

Much more work much be done to truly make any stong claims regarding our final research question, which is beyond the current scope of data collection and analysis of this study thus far. However, what can be observed about the mathematics faculty interviewed so far is a definite concern for students' depth of understanding of mathematical computing. Half of the mathematics instructors indicated a need for students to gain more skill with software. Specific skills included understanding and interpreting output, knowing what's going on under the hood, and recognizing possible software limitations and when to trust output. However, only half of those who discussed a need for these types of skill development in software use also indicated the need for software to be incorporated into mathematics education. Furthermore, math instructors were more likely to reference the restrictions against incorporating computing tools or software (such as overloaded curriculum, students' lack of prior exposure, etc.) than to reference any possible benefits of computing in the mathematics classroom.

Only one mathematics faculty member made any reference to the use of software or programming as a means of enhancing the learning of mathematics (for example, using technology to visualize three-dimensional objects or to numerically explore higher-dimensional relations). The only other capacity in which an instructional relationship between math and computing was mentioned was the insistence that students must learn the mathematical underpinnings of software packages.

With regard to student software perceptions, none of the freshman or sophomores from non-engineering degrees (3 students) reported having prior experience with software before the fall of 2019, whereas over 90% of sophomores in engineering degree tracks reported prior software experience. About 90% of *all* junior respondents enrolled in these upper-division math courses, whether in engineering tracks or not, reported having prior hands-on software experience of

some level. Furthermore, 27% of sophomore engineering students attributed part of their software experience to math courses, and 34% of junior engineering students and 39% of senior engineering students attributed part of their software experience to math courses. The remaining engineering students reported either having no exposure to computing in math settings or only being shown in-class demos with no hands-on projects or homework involving computing. Civil and mechanical were the only engineering degree types in which some students (2 MEs and 4 CEs) reported gaining their software and programming training *exclusively* from mathematics courses.

Discussion and Conclusions

While a primary motivation of this ongoing study is to characterize the extent to which math courses taken by engineering students are contributing to their overall software preparedness, an additional line of questioning emerged in response to the feedback received from faculty. Mathematics instructors frequently communicated frustration with students' lack of prior software/programming experience upon entering upper-division mathematics, citing this as a major restriction on the incorporation of computing into the mathematics classroom. They expressed further concern that even if computer science majors had prior programming experience, it was unlikely that other majors would be entering with analogous coding and software exposure, making the incorporation of software into the math classroom only slightly more feasible, as the vast majority of students would require specialized help in rudimentary programming. Student survey data, however, revealed that 71% of sophomores and 86% of juniors entering engineering-related upper-division math (such as differential equations), whether in a computer science track or not, professed prior hands-on programming or mathematical software experience to a least a moderate extent. Further data gathering and examination is necessary to determine the cause of this disparity between math faculty perception of general lack of computing preparedness and students' claim of computing experience.

Limitations. Derivation and classification of themes was bolstered by the representation of perspectives of multiple faculty from two institutions, but represent only the topical concerns of a subset of the mathematics faculty and may have excluded differing perspectives. We hope to address this limitation through additional rounds of data gathering and analysis. Similarly, while perspectives of undergraduate students are represented from multiple degrees and degree stages within multiple institutions, all students were enrolled in upper-division mathematics courses, which thus assumes that such mathematics courses sufficiently capture a reasonable cross-section of engineering disciplines.

Relatedly, it must be acknowledged that participants in the online survey may not be representative of all students at a given degree-level (i.e. sophomores), because they are specifically drawn from a subset of students who are enrolled in upper-division mathematics. Similarly, the participants are not held as strictly representative of all students in a given degree path (i.e. mechanical engineering). However, because enrollment in differential equations is a nearly universal requirement for those in engineering paths, and because the sample captured students enrolled in differential equations at a cross-section of time-points in their degree trajectories, the results are felt to be a fair reflection of the level of software exposure for

multiple degree paths as they enter differential equations specifically, and upper-division math courses more generally.

It is not possible to characterize the prior and current software exposure of students who did not respond to the survey, therefore it is not possible to say that the percentages revealed by this analysis are truly reflective of the percentages of mathematical software and programming exposure among engineering (and non-engineering) students in general. When the study was introduced, students were told that the aim was to discover how software is involved in their degree program. Thus, students who do not consider themselves to have any software knowledge may have elected not to participate. Thus, the percentages of students entering upper-division mathematics with prior software knowledge found in this study is most safely treated as an *over-estimate* rather than an under-estimate.

Future Work

The overall sampling for this study, even spread across two universities, suffers from limitations of convenience sampling within a predefined geographical region. Further study would benefit from investigating the perceptions and realities of upper-division student software preparedness in a variety of geographical regions and in a variety of institutional structures.

A number of mathematics faculty referred to viewing engineering students and computer science students as having different computing skills and different computing needs from, presumably, *all* other students. Multiple instructors voiced a need to incorporate computing into differential equations, in particular. Because there were a non-trivial number of math education majors present in the class, the question is raised: what effects would a computing 'overhaul' of differential equations have on those students? Would there be any explicit benefits or detriments to students of other degree paths if computing were incorporated into this typically junior-level course? Further investigation is necessary to determine the feasibility and impacts of incorporating computing into a single course for an institution with structure similar to those of the study.

Finally, goals of future work include the closer examination of whether students' underperformance in mathematical software or programming implementations in core engineering course work is attributable to either poor conceptual understanding of mathematics or to poor understanding of programming principles.

Acknowledgments

Special thanks to Aaron Crandall, a clinical associate professor of computer science at Washington State University, for his thoughtful assistance in outlining an ontology of computing tools reported by participants in this study, especially those closely associated with specialized engineering endeavors involving embedded systems, web development, and 3D drawing solutions.

References

[1] Brown, J. S., Collins, A. and Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher, 18*, 32-42.

[2] Magana, A. J., Falk, M. L., Vieira, C. and Reese, M. J. (2016). A case study of undergraduate engineering students' computational literacy and self-beliefs about computing in the context of authentic practices. *Computers in Human Behavior, 61*, 427-442.

[3] Pennell, S., Avitabile, P. and White, J. (2009). An engineering-oriented approach to the introductory differential equations course. *PRIMUS*, 19 (1), 88-99. DOI: 10.1080/10511970701474111.

[4] Burton, L., Falk, L. and Jarner, S. (2004). "Too Much, Too Seldom." *International Journal of Mathematical Education in Science and Technology*, *35*, 219-226.

[5] Saldaña, J. and Omasta, M. (2018). *Qualitative Research: Analyzing Life.* (Chapters 4 and 5: analytic coding of transcribed interviews.) Los Angeles: Sage.

Appendix A: Mathematics Faculty Semi-Structured Interview Protocol

What mathematics courses that may support the engineering curriculum have you taught in the past five years?

Which engineering disciplines are being pursued by students who take your classes?

What programming languages do you personally use for your work or research?

What programming languages do you use in your classes, either as a requirement or as a demonstration?

Which, if any, mathematical modeling software do you personally use for your work or research?

Which, if any, mathematical modeling software do you use in your classes, either as a requirement or as a demonstration?

What languages and software do you feel are most crucial for engineering students' industry preparedness?

Other thoughts about mathematical and computational tool learning for (engineering) students?

Appendix B: Student Survey Instrument

What is your academic major?

What year of your college degree are you currently in?

In what mathematics course(s) are you currently enrolled? (Course number OR title)

Think of your PREVIOUS MATH CLASSES. What software or programming tools, if any, have you used in any MATH courses before beginning this semester?

- NO software or programming in previous math classes
- Matlab
- Mathematica
- Python
- Other software or programming tools (list as many as necessary)

In previous math classes that used programming or software, did you do homework assignments or projects USING the programming or software?

- Not at all (no assignments used software or programming)
- Somewhat (a few assignments used software or programming)
- Frequently (most assignments used software or programming)
- None of these is quite right. COMMENT below:

Think about PREVIOUS NON-MATH CLASSES. What software or programming tools, if any, have you used in any NON-MATH courses BEFORE beginning this semester?

- NO software or programming used in previous non-math classes
- I used the following software/programming tools in previous non-math classes: (list as many as necessary)

In previous non-math classes that used programming or software, did you do homework assignments or projects using the programming or software?

- Not at all (no assignments used software or programming)
- Somewhat (a few assignments used software or programming)
- Frequently (most assignments used software or programming)
- None of these is quite right. COMMENT below:

Think about software or programming use OUTSIDE OF UNIVERSITY COURSEWORK. Were there any non-academic sources of software or programming learning for you before beginning this semester?

NO previous non-academic software or programming tool learning.

I learned the following software or programming tools at or for WORK: (list as many as necessary)

I learned the following software or programming tools ON MY OWN: (list as many as necessary)

I learned the following software or programming tools in some OTHER setting: (list as many as necessary)

Think of your CURRENT MATH CLASSES. What software or programming tools do you or your instructors use in your current math course(s)?

- Python
- Mathematica
- Matlab
- Other software or programming tool(s): (list as many as necessary)
- NO software or programming used in current math course(s).

In your current math classes that use programming or software, do you do homework assignments or projects using the programming or software?

- Not at all (no assignments use software or programming)
- Somewhat (a few assignments use software or programming)
- Frequently (most assignments use software or programming)
- None of these is quite right. COMMENT below:

To what extent has the software used in your current math course(s) been explicitly connected to the math content of your course(s)?

- 1 - Software use has no connection to math content
- 2 - Software is occasionally connected to math content
- 3 - Almost all software use is explicitly tied to math content

To what extent have the demonstrations or assignments using software or programming in these classes been based on examples that are relevant to your future studies and/or future career?

- 1 - Examples using software have no relevance to my future studies or career
- 2 - Examples using software are occasionally relevant to my future studies or career
- 3 - Examples using software are highly relevant to my future studies or career

Are there any software or programming tools that you are currently using in a NON-MATH course?

- I currently use the following software/programming tools in NON-MATH course(s): (list as many as necessary)
- NO software or programming used in my current non-math courses.

Are there any software or programming tools that you are currently using that are not related to university coursework?

- NO software or programming that is unrelated to coursework.
- I use the following software/programming tools that are unrelated to coursework:

What software or programming tools do you think are important for you to know for your future career?

Do you have any additional comments about the use of software or programming tools in ANY of your CLASSES?

Do you have any additional comments about the use of software or programming tools NOT connected to any of your classes?