IDENTIFYING COMPUTATIONAL PRACTICES IN INTRODUCTORY PHYSICS

By

Michael J. Obsniuk

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Physics — Doctor of Philosophy

2018

ABSTRACT

IDENTIFYING COMPUTATIONAL PRACTICES IN INTRODUCTORY PHYSICS

By

Michael J. Obsniuk

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Starting in the 1960's and continuing to present day, computers continue to revolutionize the way that we do and teach science. They play an increasingly integral role in everyday life. From their ability to analyze large sets of data to their ability to model chaotic physical systems, modern computers are powerful and useful tools. Given the implications this tool has on modern scientific research (e.g., modeling biological or social systems), it has become of national level importance[1].

Computation is important not only in terms of modern research in professional domains, but also as a pedagogical tool in educational domains. In order for an individual to develop professional computational skills, he must practice them – and there is no better place to practice than in an introductory physics course. Obviously, the particular skills an individual will need to be proficient in will depend on the particular job. However, focusing on the general practices that students should engage in while solving complex engineering and science problems is a reasonable place to start.

Accordingly, Secs. **??** and **??** will focus on computation's utility in both research and pedagogy, respectively. We will then introduce computational thinking in Sec. **??** and the computational practices that accompany it in Sec. **??**. With these introductory ideas, we can begin to buttress our study.

1.1 As a research tool

Computation is increasingly being referred to as the third leg of modern physics, along with experiment and theory. Its utility in solving complicated problems (e.g., predicting chaotic or non-linear motion) that cannot currently be solved analytically makes it indispensable in many field, from physics to chemistry to biology to engineering.

1.2 As a pedagogical tool

Sherwood developed and maintained VPython until it turned in Glowscript. VPython was expressly designed to engage students with computational modeling and the three-dimensional visualization that are characteristic to it.

Chabay et al. have studied computation involving VPython in introductory physics classrooms from a number of different lenses. These lenses range from investigating difficulties that students run into while computationally modeling, to investigating how to be incorporate computation into the classroom, to investigating how students are thinking computationally.

Although much work has been done around computation at the introductory level, there are still many unanswered or only partially answered questions. Answering these questions requires a deep understanding of the types of thinking and practices that students are engaging in while computing. Accordingly, we must introduce computational thinking and the accompanying computational practices.

1.3 Computational thinking

Computational thinking is a term that has become increasingly popular since its introduction in the early 1980s. This term, although frequently used today, is difficult to concisely explain given its many and varied definitions. Even within the fields of education and computer science, many different viewpoints exist on the topic, and the corresponding definitions are just as varied [7]. However, many of these definitions share one fundamental characteristic: solving complex problems through abstraction and analytic thinking with the aid of computer algorithms.

Seymour Papert first introduced computational thinking in terms of students actively constructing knowledge (constructionism) through the production of an artifact (i.e., a computer program). However, Papert does not initially attempt to define computational thinking. Rather, he comments that attempts to integrate computational thinking into everyday life have failed because of the insufficient definition of computational thinking. He optimistically claims that more attempts to define computational thinking will be made and eventually the "pieces will come together [11]." Papert would later go on to say that computational thinking involves "forging new ideas" that are both "accessible and powerful [12]."

Building on Papert, Jeanette Wing defines computational thinking in terms of taking advantage of the processing power of modern computers with the addition of human creativity. This echoes the core sentiments expressed by Papert: Using human creativity to forge new ideas that are computationally powerful. Wing is careful to remind readers that computational thinking is a fundamental skill for everyone, not just computer scientists [15].

Further elaboration by Alfred Aho points out that the process of finding the right tool for the right job is a clear indicator of computational thinking. Mathematical abstraction (mod-

eling) is at the heart of computational thinking, and be able to choose between competing abstractions (models) is of critical importance [2]. Aho points out that although there are many useful definitions of computational thinking within computer science, new domains of investigation (e.g., introductory physics) require definitions of their own.

Most recently, the Next Generation Science Standards (NGSS) laid out a framework for identifying computational thinking in K-12 settings. As early as the fifth grade students are expected to be able to think computationally. They describe computational thinking, at this level, in terms of analyzing data and comparing approaches. By the time students reach middle school, computational thinking advances to analyzing large data sets and generating explanations. Finally, in high school, computational thinking expands to constructing computational models and using them to answer questions [10]. Clearly, computational thinking is a complicated concept which requires substantial explanation.

Experts in the field still have a ways to go when it comes to clearly defining computational thinking within physics education. However defined, though, this type of abstract and algorithmic thinking is pervasive – it extends beyond computer science into fields from geology to astronomy, and even beyond STEM [4]. It is becoming increasingly clear that "computational thinking is a fundamental skill for everyone, not just computer scientists [16]."

One such domain of investigation that could benefit from a clear definition of computational thinking is that of introductory physics. Many realistic physics problems require thinking abstractly and the computational power of computers (i.e., non-linear forces in Newton's second law). This combination of abstract and algorithmic thinking (computational thinking) is the "heart" of the "computational physics approach" to solving problems [1]. This approach relies heavily on computational thinking as it relates to a particular tool

and on both technical and physics computational skills. The AAPT defines 3 technical skills and 7 physics skills.

Many of the physics skills defined by the AAPT involve modeling and the modeling process. Computational modeling is a powerful problem solving tool. Research shows that computational modeling can be successfully incorporated into the (middle-divison) physics classroom [5]. There are a number of common mistakes students make when solving a Newtonian gravitational problem. We should encourage students to synthesize both analytic and computational skills [6, **?**]

Other research on (high school) computational modeling has focused on success. Students' ability to adapt to novel problems seems to rest on the ability to synthesize physics and computational skills [3]. The ability of students to understand the iterative process, that is characteristic of computation, plays a crucial role in the ability to construct novel computational force models.

To summarize, research shows that computational thinking is desirable in many fields and still lacks precise defintion. Also, it shows that adding computation and computational thinking to a course is difficult and there are many mistakes that need to be worried about by instructors. Additionally, it shows that the iteration that is characteristic of physics and computational thinking is linked to the ability to adapt to novel problems. Therefore, it is important to encourage computational thinking in introductory physics courses.

1.4 Computational physics practices

The efficient construction of domain specific knowledge by students has always been one of the ultimate goals of physics education research. The fundamental concepts, ideas, and

theories of introductory physics are the foundation for all inquiry in not just the field of physics, but in many related disciplines (e.g., chemistry and engineering). However, there is more to being a productive member of the scientific community than just amassing a collection of facts – it is important that this knowledge be applied in some practical way (e.g., utilizing Newton's second law to numerically predict the trajectory of a rocket). For this reason, the NGSS has broadly defined scientific practices as a combination of both knowledge and skill "to emphasize that engaging in scientific investigation requires not only skill but also knowledge that is specific to each practice [13]."

Given the recent interest in scientific practices, and computational thinking more specifically, a taxonomy of the computational practices indicative of computational thinking has been defined [14]. This taxonomy, comprised of twenty-two individual yet inter-related practices, fitting into four different categories, is meant to help guide instructors and researchers as they attempt to teach and better understand computational thinking in science classrooms. Each practice, according to the taxonomy, is defined broadly so as to be applicable to a wide range of science classrooms.

However, the broad definitions that make the taxonomy widely applicable also leave it relatively vague and difficult to apply to any particular situation. Reducing the vagueness and difficulty of applying this taxonomy to a specific domain of inquiry (i.e., introductory physics) is a challenging but important task. Having a taxonomy that is both precise and easy to apply will provide a solid foundation for instructors to generate/validate computational problems and for researchers to analyze the learning process. Accordingly, it is important that we identify, through direct observation, the set of computational practices that are common to computational introductory physics. This involves not only identifying the practices, but also the underlying knowledge and skills.

In Ch. **??** we explicate the prior research on computation and its results, as well as the theoretical and methodological underpinnings of the study. This includes the historical and recent results from Physics Education Research (PER) and Computer Science Education Research (CSER). In Ch. **??**, we describe the course from which our data has been collected – a calculus-based introductory physics course with a focus on engineering, working in groups, and computation. We also describe the types of computational problems students are working on in class. In Ch. **??**, we provide a motivation for not only the existence of the study, but also the theories and methods that we decided on using. Our theories and methodologies used depended highly on the type of data that we had and the type of research we were conducting. In Chs. **??**–**??**, we present the analysis and results of our current study with concluding remarks.

Chapter 2

Background

2.1 Historical and Recent comp. research

Chabay et. al.

Expand on computational research here.

2.2 Methods for my research

Thematic analysis.

2.3 Framework

AAPT and Weintrop.

Provide a table for the AAPT and Weintrop practices.

Chapter 3

Context

It is important to understand the course from which we have collected our data to better understand the results of our study. That course – called Projects and Practices in Physics (P^3) – is based on a social constructivist theory of learning and a flipped/problem-based pedagogy. In other words, students familiarize themselves with relevant material before coming to class, where they will work in small groups to actively and socially construct knowledge while solving complex analytical and computational physics and engineering problems. The course has intentionally been designed to encourage computational thinking wherever possible. Specifically, computational thinking has been incorporated into the notes, pre- and post-class homework, in-class feedback and assessments, and a selection of the in-class problems.

3.1 Course design

Each week in P^3, students are expected to do a number of things. They must read and complete the pre-class homework based on information gathered from the pre-class notes. They must work in small groups on two related analytical problems or a mixture of one related analytical and one related computational problem during the two two-hour weekly meetings. For the computational problem, that means designing and constructing a correct computational force model in a small group and answering follow-up questions. There are

also post-class homework questions based on information gathered from the pre-class notes and the in-class problems. This all occurs while students simultaneously prepare for the following week.

3.2 Introduction to VPython

Given that the vast majority of students enter P^3 with little to no prior programming experience, we need to ensure that they are prepared to handle computational problems early in the semester. One way that we can achieve this is by requiring students to engage with the fundamental programming ideas before coming to class through pre-class homework and notes. These notes and homework questions highlight the fundamental physical and programming ideas specific to VPython and the computational problems that will be delivered in class.

For example, consider the portion of the course notes shown in Fig. **??**. These notes are made available to the students at the beginning of the semester. The content is meant to provide students with a basic understanding of the utility of VPython and a list of common errors.

Figure 3.1: Page of on-line notes introducing computional physics. The picture is just a placeholder.

3.3 Pre-class work

Similarly, consider the pre-class homework questions shown in Fig. **??** that are made available at the beginning of the third week of the course. This question is meant to demonstrate that

there are multiple correct ways that a unit vector can be constructed in code. Given the nature of the corresponding week's computational problem (see Sec. **??**), we expect students to be able to draw on and take advantage of this knowledge when faced with a related albeit more complicated problem.

Figure 3.2: Pre-class homework question focusing on the different ways that the magnitude of a vector can be constructed in VPython code.

3.4 In-class tasks

Figure 3.3: A schedule for the semester. The images is just a placeholder.

There are a number of in-class computational problems spread out throughout the semester (see Fig. **??**). The first few computational problems focus on different force models and the resulting linear motion of objects. The last few computational problems focus on extended objects and their rotation. While solving these problems, groups are expected to engage in a number of computational practices that the problems have been designed around:

P1. developing and using models,

P2. planning and carrying out investigations,

P3. analyzing and interpreting data,

P4. using mathematics and computational thinking,

P6. constructing explanations,

P7. and engaging in argument from evidence.

P8. and obtaining, evaluating, and communicating information.

One of the scientific practices used heavily on both analytic and computation days is that of (P1) developing and using models. Whether those models be mathematical or computational, we expect students to not only work together in groups to develop the model, but also to utilize that model in further investigations. This type of scientific practice (P1) and the associated learning goals[8] were further used to generate the type of in-class project that this study focuses on.

3.4.1 Analytic problem

3.4.2 Computational problem

This study focuses on the third and most complicated computational problem delivered to the students, shown in both Figs. **??** and **??**. In order to focus our analysis on specific challenges students demonstrate while working this problem, we developed a framework for analyzing this problem by performing an expert task analysis on the problem. A task analysis consists of breaking a complex "task" (e.g., modeling a gravitational system) down into related "sub-tasks" (e.g., determine the direction of the force). We aim to identify somewhat more manageable steps that must be taken in order to complete the overall task [**?**].

The task analysis of this problem was initially constructed by a single content expert. After the first iteration it was presented to additional experts. Through this discussion, it became clear that the construction of the position dependent Newtonian gravitational force in code is a multi-step procedure involving a number of different sub-tasks. The task analysis was iteratively refined through this process until all experts agreed that the sub-

tasks were sufficiently described to be useful in video analysis. At the same time, targeted pre-class homework problems were developed to help scaffold students overcoming what we perceived to be challenges based on the task analysis. In doing so, we attempted to place (most) students in the Zone of Proximal Development (ZPD)[**?**]. For example, the pre-class homework problems shown in Fig. **??** were developed to facilitate student understanding of the unit vector of a separation vector between two objects prior to working the computational problem.

Step (Sub-Task)	**Associated Code**
Construct separation vector between interacting objects	`sep = obj2.pos - obj1.pos`
Construct the unit vector	`usep = sep/mag(sep)`
Construct the net force vector	`Fnet = -G*m1*m2*usep /mag(sep)**2`
Integrate the net force over time into momentum	`obj.p = obj.p + Fnet*dt`

Table 3.1: Some of the necessary steps that must be taken when constructing a Newtonian gravitational force in code. Each step is associated with the construction/modification of a line of code.

With an agreed upon task analysis, we further focused our video analysis on the sub-tasks that were closely related to the construction of the position dependent Newtonian gravitational force (see Table. 3.1). In order to identify when groups were actually engaging in these sub-tasks, we analyzed groups on two levels: their *speech*, including any and all words or utterances, and the associated *visuals*, including any drawings, writings, lines of code, gestures, or simulation visualizations. Theses two levels were then used to better understand how students overcome the challenges associated with the problem.

Figure 3.4: In-class problem focusing on a non-constant net force model. The picture is just a placeholder at this point.

3.4.2.1 MWPs

3.4.2.2 TQs

3.4.3 Feedback/Assessment

3.4.4 Post-class work

Chapter 4

Motivation

Aside from a general interest in introductory computational physics, it is important to understand the underlying motivation(s) for the current study.

The process of identifying an interesting computational practice, described in Sec. 4.1, was the earliest motivation for this study. We found that it was extremely difficult to define and identify the particular practice of what we named "physics debugging." Not only did the practice need to be clearly defined, it also needed to be clearly identified in the data. This required a lot of in-depth qualitative analysis and inter-rater reliability, motivating our use of the Weintrop framework and the qualitative methods of Clarke in the current study.

Additionally, as described in Sec. 4.2, we found that it was very difficult to understand the qualitatively different ways in which students experienced computational introductory physics. This difficulty motivated a task analysis with a focus on identifying practices that the students were engaging in through in-class observation, as opposed to their experiences through out-of-class interviews.

4.1 Debugging

In this section, we present a case study of a group of students immersed in this P^3 environment solving a computational problem. This problem requires the translation of a number of fundamental physics principles into computer code. Our analysis consists of qualitative ob-

servations in an attempt to describe, rather than generalize, the computational interactions, debugging strategies, and learning opportunities unique to this novel environment.

We focus this case study on the interactions between group and computer to begin to understand the ways in which computation can influence learning. Particularly, we are interested in the interactions occurring simultaneously with social exchanges of fundamental physics principles (FPPs) specific to the present task (e.g., discussing $d\mathbf{r} = \mathbf{v}\,dt$ on a motion task) and the display of desirable problem solving strategies (e.g., divide-and-conquer). These group-computer interactions vary in form, from the more active process of sifting through lines of code, to the more passive process of observing a three-dimensional visual display.

One previously defined computational interaction that reinforces desirable strategies,[**?**] borrowing from computer science education research, is the process of debugging. Computer science defines debugging as a process that comes after testing *syntactically* correct code where programmers "find out exactly where the error is and how to fix it."[9] Given the generic nature of the application of computation in computer science environments (e.g., data sorting, poker statistics, or "Hello, World!" tasks), we expect to see unique strategies specific to a computational *physics* environment. Thus, we extend this notion of computer science debugging into a physics context to help uncover the strategies employed while groups of students debug *fundamentally* correct code that produces unexpected physical results.

4.1.1 Analysis

In Fall 2014, P^3 was run at Michigan State University in the Physics Department. It was this first semester where we collected *in situ* data using three sets of video camera, microphone, and laptop with screencasting software to document three different groups each week.

From the subset of this data containing computational problems, we *purposefully sampled* a particularly interesting group in terms of their computational interactions, as identified by their instructor. That is, we chose our case study not based on generalizability, but rather on the group's receptive and engaging nature with the project as an *extreme case*.[**?**]

The project that the selected group worked on for this study consists of creating a computational model to simulate the geosynchronous orbit of a satellite around Earth. In order to generate a simulation that produced the desired output, the group had to incorporate a position dependent Newtonian gravitational force and the update of momentum, using realistic numerical values. The appropriate numerical values are Googleable, though instructors encouraged groups to solve for them analytically.

This study focuses on one group in the fourth week of class (the fourth computational problem seen) consisting of four individuals: Students A, B, C, and D. The group had primary interaction with one assigned instructor. Broadly, we see a 50/50 split on gender, with one ESL international student. Student A had the most programming experience out of the group. It is through the audiovisual and screencast documentation of this group's interaction with each other and with the technology available that we began our analysis.

To focus in on the group's successful physics debugging occurring over the 2 h class period, we needed to identify phases in time when the group had recognized and resolved a physics bug. These two phases in time, *bug recognition* and *bug resolution* are the necessary limits on either side of the process of *physics debugging*, as represented in Fig. **??**. We identified these two bounding phases at around 60(5) min into the problem, and further examined the process of debugging in-between. That is, we focused on the crucial moments surrounding the final modifications that took the code from producing unexpected output to expected output.

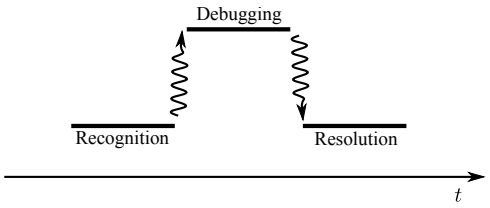

Figure 4.1: The debugging process necessarily corresponds to a phase beset on either side by the phases of recognition and resolution. Note the absence of a vertical scale, as the vertical separation merely acts to distinguish phases.

4.1.1.1 Recognition

4.1.1.2 Resolution

4.2 Phenomenography

A description of the phenomenography study.

4.2.1 Protocol

A description of the development of the protocol.

4.2.2 Analysis

Since there are no real results, we can just describe how we analyzed the interview data.

Chapter 5

Analysis

5.1 Specific methods

5.2 Findings

Recall that, according to the framework, each category of practice can be broken down into a number of individual practices. Each individual practice, further, can be identified in terms of a number of fundamental characteristics. Depending on the particular situation, some categories show up in the data more often than others. For example, we expect to see fewer systems thinking practices and more computational modeling practices. Further, within any category, some individual practices show up in the data more than others. For example, within the data practices, we expect to see fewer instances of collecting data and more instances of creating data.

Most importantly, within any individual practice, there is a broad set of defining characteristics. Ideally, each characteristic would be present in some form when its corresponding practice has been identified. However, each characteristic can be organized in terms of being either necessary or sufficient for the cause. That is, we need not necessarily observe every characteristic in order to identify a practice according to the framework.

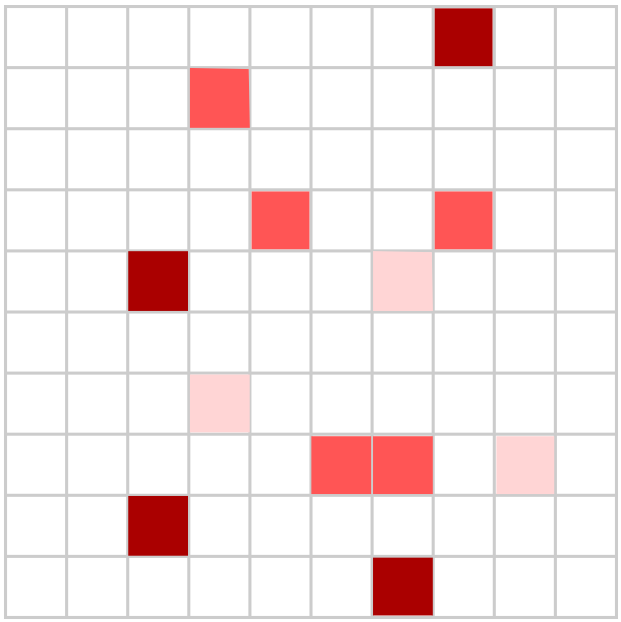

Figure 5.1: A "density plot" for all computational practices and all groups.

5.2.1 Assessing computational models

The most important computational modeling practice indicative of computational thinking is that of assessing computational models. This practice shows up 100 % of the time. Assessing computational models seems to be crucial to the process of designing and constructing computational models. Given that computational models take a lot of work and iteration to get to an acceptable level, it is no surprise that assessing shows up so frequently.

There are three characteristics that are necessary to be observed in an excerpt to warrant classification of "assessing computational models." The first characteristic that needs to be observed is a "model." There are several models that show up in the data. The second characteristic that needs to be observed is a "phenomenon." For the most part, the phenomena are related to the simulation of the trajectory of the geostationary satellite and its various physical interpretations (e.g., a central attractive force or a circular orbit). The third characteristic that needs to be observed is a "comparion" between the model and phenomenon.

The comparisons that we see are ultimately varied, given that they depend on not only the model but also the phenomenon.

a	b	c
c	d	100 %

Figure 5.2: A table showing rough statistics for assessing computational models.

5.2.1.1 Models

The two big models that we have observed in the data are a position dependent Newtonian gravitational force in terms of a separation vector and a centripetal force that depends on the polar angle of the satellite. Each model shows up 100 % and 100 % of the time, respectively.

5.2.1.1.1 Group A (gold star)

5.2.1.1.2 Group B (near miss)

5.2.1.2 Phenomena

The phenomena that we have observed in the data are almost always centered around the simulation of the trajectory of the satellite. The physical interpretations of the phenomena are, not surprisingly, closely related to the topics covered in the couse notes and the topics questioned in the weekly homework. These interpretations range from the crude (e.g., a circular orbit) to the sophisticated (e.g. a centripetal force and its properties). This variety of phenomena show up 100 % of the time.

5.2.1.2.1 Group C

5.2.1.2.2 Group D

5.2.1.3 Comparisons

The comparisons that we have observed in the data are quite varied. Given that the comparison is between the models and phenomena, we expect this characteristic to be the most varied.

5.2.1.3.1 Group E

5.2.1.3.2 Group F

Chapter 6

Discussion

Chapter 7

Concluding Remarks

APPENDICES

Appendix A

Additional excerpts

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Undergraduate Curriculum Task Force AAPT. Recommendations for computational physics in the undergraduate physics curriculum. Technical report, AAPT, 2016.

[2] Alfred Aho. Computation and computational thinking. *The Computer Journal*, 2012.

[3] John Aiken, Marcos Caballero, Scott Douglas, John Burk, Erin Scanlon, Brian Thoms, and Michael Schatz. Understanding student computational thinking with computational modeling. In *PERC Proceedings*, 2012.

[4] Alan Bundy. Computational thinking is pervasive. *Journal of Scientific and Practical Computing*, 2007.

[5] Marcos Caballero, Matthew Kohlmyer, and Michael Schatz. Fostering computational thinking in introductory mechanics. In *PERC Proceedings*, 2011.

[6] Marcos Caballero and Steven Pollock. A model for incorporating computation without changing the course: An example from middle-division classical mechanics. *Americal Journal of Physics*, 2014.

[7] Shuchi Grover and Roy Pea. Computational thinking in k-12: A review of the state of the field. *Educational Resarcher*, 2013.

[8] Paul Irving, Michael Obsniuk, and Marcos Caballero. P^3: A practice focused learning environment. *European Journal of Physics*, 2017.

[9] Renee McCauley, Sue Fitzgerald, Gary Lewandowski, Laurie Murphy, Beth Simon, Lunda Thomas, and Carol Zander. Debugging: a review of the literature from ane ducational perspective. *Computer Science Education*, 2008.

[10] Committee on a Conceptual Framework for New K-12 Science Education Standards. *A Framework for K-12 Science Education: Practices, Crosscutting Concepts, and Core Ideas*. The National Academies Press, 2012.

[11] Seymour Papert. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, 1981.

[12] Seymour Papert. An exploration in the space of mathematics educations. *Technology, Knowledge, and Learning*, 1996.

[13] NGSS Lead States. *Next generation science standards: for states, by states*. The national Academies Press, 2013.

[14] David Weintrop, Elham Behesthi, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. Defining computational thinking for mathematics and science classrooms. *Journal of Science Education Technology*, 2015.

[15] Jeanette Wing. Computational thinking and thinking about computing. *The Royal Society*, 2008.

[16] Jeannette Wing. Computational thinking. *Communications of the ACM*, 2006.