Middle Science Computing Integration with Preservice Teachers

Lauren Margulieux
Georgia State University
USA
lmargulieux@gsu.edu

Aman Yadav
Michigan State University
USA
ayadav@msu.edu

Abstract: We explored how preservice teachers in a middle school science methods course learned and applied computational thinking (CT) concepts and activities during a month-long intervention. In the intervention, preservice teachers learned about CT concepts through an hour-long lecture in their methods class, practiced a computing-integration activity for electromagnetic waves, and prepared and implemented a lesson plan based on the activity in student teaching. The intervention was in the early stages of design, and, therefore, the research is exploratory with primarily qualitative data. The data were collected at multiple points throughout the month to measure the development of knowledge and attitudes about CT and computing integration. We found that preservice teachers had little knowledge of computing before the intervention that gradually evolved into a deep understanding that they wanted to apply to computing-integrated activities science and other subjects. Though they had high levels of uncertainty after initial instruction and practicing the computing-integration activity, they found the student teaching experience rewarding and motivating to including computing in their future teaching practice.

Introduction

Integrating computing into required courses is a highly effective method for equitably achieving computational literacy and using computers as tools for learning (DeLyser, Goode, Guzdial, Kafai, Yadav, 2018; Kale et al., 2018). To integrate computing, teachers of non-computing subjects need to know fundamental computing concepts and practices, often referred to as computational thinking (CT), and methods for integrating computing into their teaching practice within their subject area. For an example in science, teachers can use various methods to help their students visualize phenomena to better grasp concepts (Gilbert, 2005), and each method has different affordances. When learning about waves, a string-and-paper model (see Figure 1) allows students to work with tactile materials to visualize the wave and label the different parts. A drawback is that it is a static model—once you have glued the string, it cannot be manipulated. In contrast, a digital model built in Pencil Code (see Figure 2) provides a dynamic visualization of a wave that allows students to change the values of different wave properties, like wavelength, and explore how it affects the wave. Furthermore, the teacher can use the model to teach scientific practices by asking students predict what the wave will look like if the wavelength or amplitude is increased or decreased, and students test their predictions by entering values into the model. This example shows how computing tools can provide teachers and their students to explore scientific phenomena in ways that are not possible otherwise.

The Pencil Code model also affords an opportunity to teach students about computing. If given the pre-built model in Figure 2, students can learn about computing concepts such as defining and using variables, defining and calling functions, and using loops by exploring the different pieces of code and how they contribute to the model. The common barrier to using a tool like Pencil-Code is that most science educators do not have a background in using computing tools, concepts, and practices. Although some of computing concepts and practices are similar to those in science (or math or literacy or art), most preservice teachers (PSTs) self-report knowing little about computing (Qian et al., 2018) and having little confidence that they could integrate computing in their classroom (Yadav et al., 2014). The goal of this work was to explore how to prepare PSTs to integrate computing tools and activities in their teaching. This goal faces a couple of challenges, 1) PST preparation programs are already overloaded with the pedagogical knowledge that PSTs
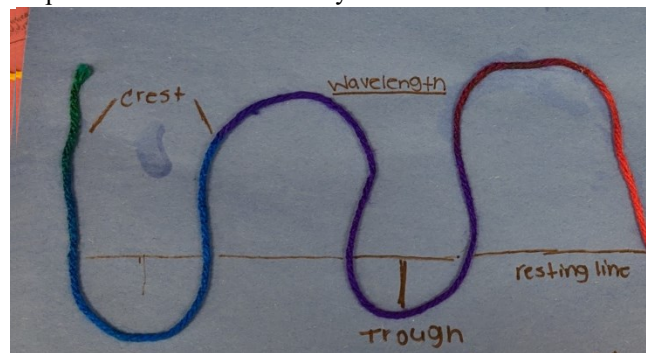

Figure 1. String-and-paper model of wave students made.

must learn to be successful in the classroom, so adding computing and activities places more stress on PSTs, and 2) PSTs also must learn the content knowledge in their area, so adding computing content knowledge when many of them have none (Qian et al., 2018) adds additional stress. Therefore, to sustainably integrate computing with PST preparation programs in other subjects, the integration must build upon PSTs existing knowledge of their subject and how to teach it.

The guiding questions for this work were:

1. What knowledge of CT concepts do PSTs have after an hour of instruction on CT concepts, an hour of preparing a computing-integrated activity, and an hour of student teaching using that activity?
2. What similarities do PSTs recognize between their subject area concepts and teaching practice and computational thinking concepts and computing-integrated activities?

Computing Integration and CT in K-12

Since Wing (2006) discussed CT as the essence of computer science, computational literacy work has focused on bringing CT to K-12 learners. Most of this work has been within in-service teacher professional development exploring how they can use CT in their classrooms. Research on in-service teacher professional development has suggested that teachers have often have conceptions of CT that do not align with computer science education researchers' conceptions (Sands, Yadav, & Good, 2018). For example, Sands, Yadav, and Good (2018) surveyed 74 elementary and secondary teachers and found that while some of teachers' views of CT aligned with the literature (e.g., using heuristics or algorithms and solving problems), teachers also had ideas about CT that did not (e.g., knowing how to use a computer and using technology in teaching). At the preservice level, there has been some work examining the influence of introducing CT within teacher education courses on

Figure 2. Pencil Code model of a wave that was used as a template during student teaching.

pre-service teachers' knowledge and attitude towards computing. For example, Mouza et al. (2017) explored how a re-designed educational technology courses around CT affected preservice teachers' knowledge of CT. They found that introducing CT in the courses significantly influenced preservice teachers' definition of CT, as well as knowledge and beliefs about CT in their future classrooms. They measured how preservice teachers applied their knowledge of CT concepts, computing tools, and practices to design and implement content-specific lessons. They found that while some students were able to use their knowledge to support student development of CT, many had difficulty in clearly identifying and labeling CT concepts and practices supported by their lessons. In particular, they tended to focus on general uses of technology to solve problems or do mathematics/calculations. A number of researchers have found similar results that brief introduction of CT ideas can help shift preservice teachers understanding of CT ideas and how they see its relevance in their future classrooms (Yadav et al., 2014).

Exploring CT concepts as part of a year-long program or re-designed educational technology course is perhaps the most desirable way from computer science education researchers' perspective to introduce these concepts (DeLyser et al., 2018), but many PST preparation programs cannot accommodate additional extended instruction. Some universities have even incorporated educational technology instruction into methods courses to
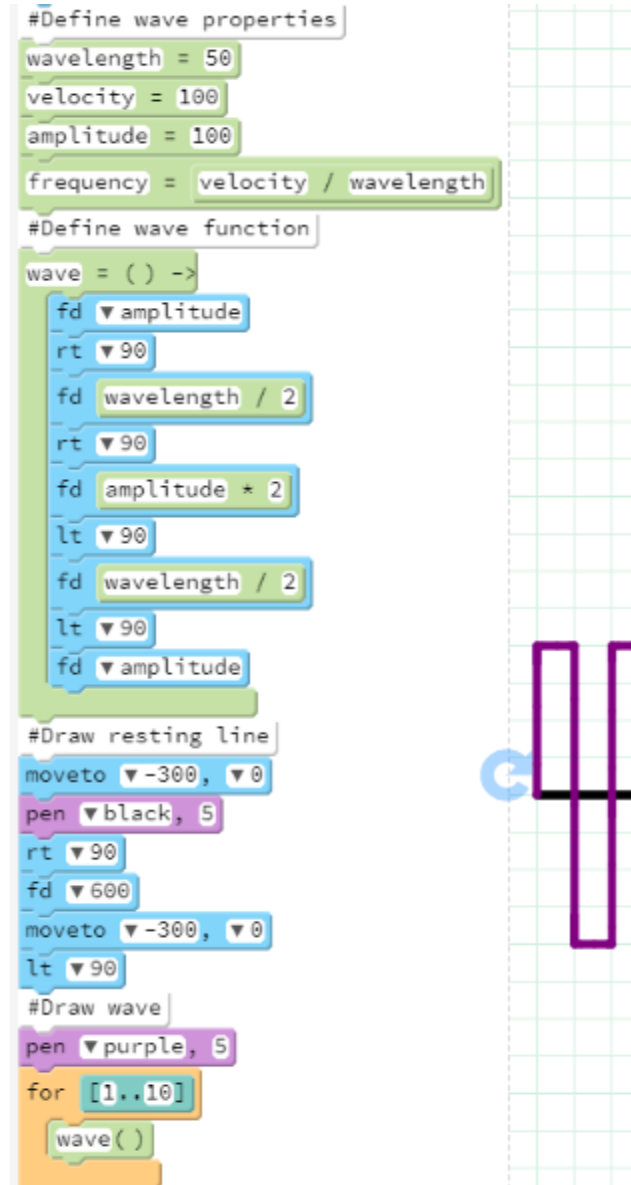
reduce the number of courses PSTs must take. Therefore, this study explores a shorter-term, month-long, computing integration intervention in a middle school science methods course. This paper describes how PSTs' knowledge of CT and computing integration developed over the month, and how they aligned these new concepts with their prior knowledge and teaching practice.

Middle School Science Methods Computing Integration

To design the computing integration, the first author, a computing education research, met with two faculty, science education researchers, to discuss opportunities to integrate computing into a middle school science methods course. They agreed on the following design specifications: 1) use an interactive scientific model of a phenomenon to help visualize it, and 2) use a programming language that could be used for activities in multiple subjects because middle-level PSTs specialize in at least two subjects. The first author recommended a block-based language so that PSTs would not need to learn programming syntax in the 1.75 hours they had for instruction. The faculty chose Pencil Code to meet the design specifications and because it is block-based with the option to translate to text-based CoffeeScript or JavaScript in case students were going to gain experience with it across multiple subjects.

During student teaching the PSTs would be teaching electromagnetic waves. Therefore, the first author created an activity using Pencil Code for modeling electromagnetic waves. With this activity in mind, she developed an hour-long guest lecture about CT concepts and practices based on second author's previous work with PSTs and definitions of CT from the literature (Ajo, 2012; Armoni, 2016; Barr & Stephenson, 2011; Brennan & Resnick, 2012; Cuny, Snyder, & Wing, 2011; Denning, 2017; Grover & Pea, 2013; Weintrop et al., 2016; Wing, 2006, 2008). CT was defined as answering three questions: (i) Should I get a computer to help me solve this problem? (ii) How would I get a computer to solve this problem? (iii) Does the computer solve the problem accurately and efficiently?

The CT instruction was given in class to the PSTs before introducing the electromagnetic wave computing-integration activity and was organized into six main components. It included motivation for CT and computing integration and five CT concepts: Abstraction, Algorithms, Automation, Deconstruction, and Debugging. Four of six components included reflection questions to prompt PSTs to relate the CT concepts to their prior science content and pedagogical knowledge. The reflection questions were analogous to, "Think of an example of these concepts/practices in your field," or "How would you use these concepts/practices in your teaching practice?" The think-pair-share paradigm was used so that PSTs thought about their answer individually first, discussed answers at their table, and shared their answers with the class.

1. Abstraction - definition and rock cycle example; extended discussion with paper airplane example; reflection activity
2. Algorithms - definition; create a paper airplane activity; reflection activity
3. Automation - definition and password manager example; trade-offs in automation; paper airplane example; reflection activity
4. Deconstruction - definition and breakfast example; revisit paper airplane activity; reflection activity
5. Debugging - definition and rubber duck debugging; fixing sound example; review of lesson

This instruction took about an hour of class time (originally allotted 45 minutes) with a ratio of about two-thirds of time on lecture and one-third on discussion and activities.

After the CT instruction, the course instructor and first author guided the PSTs through a review of the relevant science concepts for waves, in which the PSTs defined the components and drew waves on whiteboards. Before introducing Pencil Code, PSTs were asked to write pseudocode (i.e., commands in plain English) for drawing a wave. Then, the first author guided the PSTs to create Pencil Code accounts and

Figure 3. Simplified Pencil Code model used to introduce computing integration activity in class.

demonstrated a five-minute tutorial focused on the menus and blocks that PSTs would use for their model. After the tutorial, she walked students through a simplified wave model with only wavelength and not amplitude (see Figure 3). After walking through the simplified model, PSTs recreated the model in small groups. For homework, PSTs were asked to expand on their models so that they included amplitude (see Figure 2). PSTs were given an image of what the final output should look like and told that they should not use arcs as they did in the simplified model. This expansion upon the activity was intended to be completed in class, but time did not permit. The activity took about 45 minutes.

Both of the Pencil Code models (Figures 2 and 3) were designed to include variables. The variables supported easy manipulation of values in the model and, at the course instructor's and the school's request, the use of equations to connect to math concepts. The models also included functions and for loops, at the first author's request. A more elegant program might include wavelength and amplitude as parameters of the wave function, but using variables contributed to math learning goals and simplified the concept of functions by not using parameters. Instead functions were described as a tool for abstraction that could be defined once and called when needed.

After the classroom instruction, PSTs had two weeks to complete an assignment (described in section on data collection sources). Two weeks after this assignment was due, PSTs participated in student teaching in an 8th grade science class in an [blind city] Public Schools secondary school. The lesson plan for student teaching started with the 8th grade students recalling what they know about waves in groups of four. Then the PSTs guided the students to create Pencil Code accounts and draw their name to test and use different features. Next, the PSTs showed students the complete Pencil Code model (see Figure 2) and asked them to explain how it worked, make predictions for how changing values would affect the output, and test their predictions with the model. Last, students were asked to create their own wave model based on their knowledge of waves and Pencil Code.

A critical difference between the learning goals for the PSTs and the learning goals for the 8th graders was the amount of computing they learned. For the PSTs, learning computing concepts and practices so that they could design and implement lesson plans with computing integration was a key learning goal. For the students, however, given that we had only an hour with them, our learning goals were to focus on science concepts, using the model as a tool, and to expose the students to computing. Despite different learning goals, both groups showed interest in learning more about computing, as discussed in the results section.

Method

To address our research questions and evaluate the effect on PSTs of the CT instruction and computing-integration activity in the classroom and in student teaching, we collected qualitative and quantitative data about the process and outcomes of the intervention. We used a within-subjects experimental design, exploring the effect of the intervention at four points: before instruction, immediately after instruction, two weeks after instruction following an assignment, and after student teaching. The focus of analysis for all data is on the PSTs.

PSTs, Instructors, and Institutions

The middle science methods course had 12 PSTs who participated in the study. Not all PSTs completed all data collection sources, so the sample size is listed for each source. Participants included six PSTs who were white, three who were African American, and three who were Asian American. There were seven women and five men. Most of the PSTs wanted to specialize in science education, but at least three did not. The two faculty instructors were both assistant professors with different educational backgrounds. The course instructor specializes in science education and teacher preparation and had little background in computing, and the first author specializes in computing education and had little background in science education or teacher preparation. They combined their areas of expertise using a co-teaching model with the PSTs, and both of them were present for all components of the intervention. For student teaching, we partnered with a secondary school in [blind city] Public Schools. The school is for female students only and serves grades 6 through 12. We worked with an eighth-grade science course.

Data Collection Sources and Analysis

Throughout the project, we collected data through four major sources: CT survey, pre-student-teaching assignment, post-student-teaching reflection, and field notes for classroom instruction and student teaching. The CT survey was based on the survey from Yadav et al. (2014). It was completed before the classroom instruction (pre-survey), at the end of classroom instruction (post-survey), and at the end of student teaching (post-post-survey).

Eleven PSTs completed the survey all three times. The survey had a quantitative component that asked PSTs to rate 1) their familiarity with CT, 2) how easily CT can be integrated into other subjects, 3) how comfortable they would be integrating CT, and 4) their general comfort with using computers. The qualitative component of the survey asked PSTs to explain/define 1) CT, 2) how they might implement CT in their class, 3) barriers that they might face implementing CT, and 4) list three things that someone who knows computing could do.

The pre-student-teaching assignment was completed by 12 PSTs two weeks after classroom instruction. The assignment had several components: 1) reflection on classroom instruction, 2) explain CT and the five concepts discussed in class, 3) reading reflections on articles about CT written by Wing (2006) and Grover and Pea (2013), 4) expand the simplified model in Pencil Code to include amplitude, 5) match the computing-integration activity to standards for 8th grade science, 8th grade math, ISTE teacher, ISTE student, and social justice, and 6) write a 90-minute lesson plan for using the computing-integration activity.

The post-student-teaching reflection was completed by 11 PSTs, and it was due 2 weeks after student teaching. It asked four questions: 1) What parts of the activity went well and did not go well?, 2) What would you change about the lesson plan based on your experience?, 3) What are the tradeoffs in using a computing-integrated lesson?, and 4) If you had a pre-built resource, like the existing code that modeled the wave and how it connected to science concepts, how likely and comfortable would you be using a computing-integrated activity in the future?

To supplement data collection, the first author took field notes during the classroom instruction, which had 11 PSTs, and the student teaching, which had 11 PSTs. Because she was engaged in instruction, these are not systematic field notes from an impartial observer. Instead they were to record the topics discussed in the classroom instruction and anything of note (either good or bad) during the classroom instruction and student teaching.

To analyze quantitative data and quantitative coding, we used only descriptive statistics. The sample size does not support inferential statistics nor were inferential generalizations a goal of this exploratory study. Most data analyses focused on qualitative data using content analysis (Hsieh & Shannon, 2005) using NVivo 12 software. Content analysis allowed themes to emerge from the data by iteratively coding the data to explore different interpretations. Each component of the data collection sources was coded, and one component could be coded into multiple nodes. The initial, exploratory nodes that we started with were to code which research question the data addressed. During the first round of analysis, we classified components into the research question nodes and made additional nodes for high-level themes within each research question. During the second round, we classified components within each research question into the high-level themes and made additional nodes for sub-themes. During the third round, we classified components within the themes into sub-themes and did not recognize additional thematic nodes. All content was scored by two raters to establish inter-rater reliability, which was 84% agreement.

For the first research question, we added additional nodes for CT definitions, explanation of CT concepts, and expressions of uncertainty. For the second research question, we had additional nodes for effective practices for teaching CT to preservice teachers, recognition of alignment with standards (science, math, ISTE teacher and student, and social justice), applications of computing integration in science, and applications of computing integration to other subjects. All content was scored by two raters with 84% agreement.

Results

The results section is organized around the research question and the qualitative nodes identified through content analysis. The qualitative nodes are supplemented with quantitative data when applicable. **RQ1: What knowledge of computational thinking concepts do PSTs have after an hour of instruction on CT concepts, an hour of preparing a computing-integrated activity, and an hour of student teaching?**

Evolution of CT Definitions

We examined PSTs' definitions of CT across our measures to examine how their understanding evolved as they engaged with CT. The first definitions they gave were on the CT survey before classroom instruction. On the pre-survey, 9 of 12 PSTs said that they definitely had not heard of CT before or might have heard of it. The others had heard of it once but were not familiar with it. Overall, the PSTs gave a wide range of guesses for the definition of CT with the most common responses being

- Using technology (4 PSTs) - "Incorporating computer skills and technology into classroom settings and lessons"
- Problem solving or thought process (5 PSTs) - "Possibly a logical, methodological type of thought process"

On the post-survey at the end of class, PSTs gave more technically correct but also closely mirrored the instruction,

- Listing concepts discussed in class (7 PSTs) - "Computational thinking is applying automations, algorithms, decomposition, and debugging to solve problems"
- Using computers to solve problems (4 PSTs) - "Problem solving methods through computers"

After two weeks, the PSTs completed the pre-student-teaching assignment in which they elaborated on their CT definition. The definitions still focused on the five CT concepts (Abstraction, Algorithms, Automation, Deconstruction, and Debugging) discussed in class, but they expanded to include applicability of CT across many subjects while using the computer as a tool. Three PSTs discussed applying CT without computers, and three implied that CT should be applied only with computers. For this reason and others, the definitions at this stage were not completely accurate, but the errors seemed to represent a simplistic understanding of the concept rather than problematic misconceptions. Seven PSTs gave an answer like,

"CT is a problem solving method that emulates computational patterns and behaviors such as how a computer would solve the problem. CT is broken into five parts to make the process easier to understand and make it easier to walk through to find a solution to the problem. CT is usually implemented when creating and executing computer programs however it has been found useful across disciplines including math, science, and the humanities. CT is also good on eliminating whether or not a computer would be useful to solve the problem or not. It encourages problem solving and increases digital literacy."

Four other PSTs repeated the three questions used to define CT in class with minimal, but thoughtful, additions.

"Thinking computationally basically means answering three questions: Should a computer help me solve this problem? How would I get a computer to solve this problem? Does the computer solve the problem accurately and efficiently?. It is useful because the steps involved with CT help students to 'keep working" or "keep trying" to solve a problem."

After student teaching, the PSTs completed the CT survey again and the post-student-teaching reflection. In both data collection sources, they provided definitions of CT that were less reliant on using computers or listing CT concepts. Over the month, the PSTs understanding seemed to have evolved to subject-independent process for solving problems that could be enhanced with the use of computers. The common themes were

- Decomposition (4 PSTs) - "CT allows students to break things down and understand better of the little things that make up the bigger things."
- Problem solving and thinking systematically (3 PSTs) - "A step-by-step process of learning and educating in which you approach problems and information from an analytical point of view"
- Thinking like a computer to use a computer for problem solving (5 PSTs) - "You can use computational thinking in every subject to show how computers can be used to do more complex problems, or repeated sets of data."

Explanations of CT Concepts

In the pre-student-teaching assignment, PSTs were also asked to explain the five CT concepts introduced in class. About half of explanations demonstrated accurate but shallow processing of the material, e.g., repeating definitions and examples from class. However, 5 of 12 PSTs included examples from math, literature, community events, and science, suggesting that they connected the new concepts to their existing knowledge and experiences. These include "Algorithms is like a step by step process. An example of this is like a lab procedure, where you give specific instructions on what to do," and "Abstraction – identifying important details and opportunities for generalization (ex: analyzing a reading passage with notes and highlights)." All definitions and examples that PSTs gave were not specific to computing, even for the concept of debugging, "Debugging is the part of computational thinking I think we can all benefit from and utilize the most. In my opinion, this is a version of checking your work. 'Rubber duck debugging' is explaining to an actual rubber duck what you're doing, and it helps you to uncover discrepancies. I do this every time I write a paper. I will ask someone to listen to me read it aloud so that I can catch any errors that my mind may have overlooked after having stared at my computer for so long." The connections that PSTs recognized between CT and other subjects are discussed more in the results of RQ2.

Expressions of Uncertainty

PSTs expressed doubts in their ability to implement a computing-integration activity during the intervention but gain confidence by the end. From the pre-survey to the post-survey to the post-post-survey on the question, "I am confident that I could integrate computational thinking into my future classroom," with a scale of 1 - Not at all to 5 - Completely, confidence grew on average from 3.3 to 4.3 to 4.7. In contrast, in response to the question, "Can computational thinking be integrated into non-computing classes?" with a scale of 1 - Not practically to 4 - Easily, scores increased overall, but not linearly. Nine PSTs who had given a rating of three (somewhat easily)

increased to four (easily), but the two PSTs who had given a rating of two (not easily) decreased to one (not practically). On the final post-post-test after student teaching, however, all but one PST gave the highest rating, including those who had said CT could not practically be integrated.

During the classroom instruction based on PSTs' responses to reflection questions during instruction and other discussion, they seemed to easily grasp the five CT concepts in relation to their prior knowledge about science education. However, when PSTs were asked to explain to a computer or write pseudocode for how they might make a paper airplane or draw a wave, they quickly got stuck, asking for help after about 30 seconds into the activity. By the end of the activity, the most advanced responses were akin to, "Draw a wave with the width based on the wavelength and height based on amplitude." Perhaps these activities needed more scaffolding given that most of the PSTs had never programmed before. However, the PSTs immediately were active in making programs when we introduced them to Pencil Code, with many starting to draw on their own before the tutorial. Thus, a better approach to the instruction might be to start with a short programming activity, even before PSTs know any concepts, to give them a better sense of what programming is and what the possibilities for pseudocode might be. Another feasible alternative based on our results is that the pseudocode activity is unnecessary.

Despite being immediately active in Pencil Code and accurately recreating the simplified model in class, PSTs performed poorly on the assignment to extend the Pencil Code model. All PSTs used the `for` loop in their extended model, but most did not use variables or a function. One PST with prior programming experience created the model with all specified features. Variables were a main affordance of the model that allowed learners to easily manipulate the value of different wave components. From these results, we argue that a code writing activity, even in a block-based language, is too advanced for this type of introduction to CT. Instead, we should have focused on preparing the PSTs to remix and explore a pre-built model, which they did well during the classroom instruction.

Based on the first authors' field notes, in the hour of preparation before student teaching, many PSTs were uncertain and nervous about computing-integrated activity. They were motivated, however, by wanting to provide the best, hands-on learning activities for their students and multiple methods for learning. In the original design, PSTs were supposed to use their own extended model during student teaching, but many PSTs were uncomfortable with this. Instead, they were given the full model to use (see Figure 2). This change immediately relieved anxiety as the PSTs shifted attention from explaining the model they had created to explaining the scientific phenomenon. In the post-student-teaching reflection, most PSTs said they would be excited to use a computing-integration activity in their class as long as someone else had already developed the program or algorithm.

Though the focus of the paper is not on the students taught during student teaching, the positive reactions of the students had a motivating impact on the PSTs. None of the students had used Pencil Code before. Like the PSTs, students immediately began using a lot of different types of blocks when they logged into Pencil Code. Then using the program in Figure 2, the PSTs explained how the wave model worked, and students would change with values and run the code. In one exchange, a student said, "It's a lot of testing, that's the cool thing about this." Then students created their own models and compared them. During the comparisons, students said, "How did you do that?!" "I didn't know I could do this," and "There's different parts of science, and [computing] is one of them." In the post-student-teaching reflections, most PSTs mentioned the excitement of the students as a motivation to include a computing-integration activity in their future classroom, such as, "We asked the students if they'd want to do something like this in the future and all their hands shot up!"

RQ2: What similarities do PSTs recognize between their subject area concepts and teaching practice and computational thinking concepts and computing-integrated activities?

Effective Practices for Teaching CT to PSTs

In the pre-student-teaching assignment, we asked PSTs to reflect on the CT instruction and discuss which parts resonated most with them. The purpose of this question was to identify parts of the instruction PSTs found most interesting or applicable to replicate or expand in the future. Almost every PST gave a different response to this question, though, suggesting that PSTs connected most with different aspects of the instruction.

A general theme for some PSTs was using computers or CT as a tool for teaching and learning. For one PST, CT was a new way of problem solving, "I learned how to problem solve using different methods, especially in ways that a computer would execute. The part of the presentation that resonated with me the most is the psychology of the rubber duck [debugging]. When we hit a roadblock in problem-solving, we can explain our problems and it helps to decode our goal." Another emphasized "teaching" the computer as a way to learn, "It really resonated with me when Dr. [blind] told us that teaching is a good way to learn...I can most certainly attest to teaching being a great way to learn, and computational thinking is just that: explaining your processing down to the most finite details so

you know exactly what's going on." Another saw the computer as a tool to understand student thought processes, "It also helps me as a teacher because I can ask a student how they went through it and got the answers that they did. Being able to reflect on each step and figure out the student got from A to Z is very useful to student and teacher."

Other PSTs focused on the content they could teach with CT. This could be teaching students about computing concepts, "I never learned how to code as a student, so therefore I think teaching my students will be a great experience," or teaching students about science with computing, "It was exciting to learn a new way to relate scientific ideas to my future students with technology they may not be introduced to yet."

The last general theme was PSTs appreciated the use of examples that were personally meaningful to them. The examples could be relevant to them as a science teacher, "What really resonated with me was the way she made it so accessible. She introduced an entirely new concept by making it relate to the things that are most important to us; reaching our future students!" or to everyday life, "She use real world issue to help us understand the concepts."

Recognition of Alignment with Standards

As part of the pre-student-teaching assignments, PSTs connected the computing-integration activity to standards for 8th grade science in Next Generation Science Standards (NGSS), 8th grade math in Mathematics [blind state] Standards of Excellence, ISTE Standards for Teachers and Students, and Social Justice Standards from Teaching Tolerance Anti-bias Framework. For 8th grade science and math, PSTs identified the content standards, Waves & Media in science and Solving Systems of Equations in math. For the NGSS crosscutting concepts, they recognized many possible options, including cause and effect (6 PSTs listed this standard), structure and function (4), and patterns (4). In addition, for the NGSS practices, they recognized develop and use models (8), planning and carrying out investigations (3), engaging in argument from evidence (1), and asking questions and defining problems (1). This variation in identifying standards suggests that PSTs could apply the computing-integration activity in their classroom to flexibly achieve NGSS crosscutting concepts and science and engineering practices.

PSTs similarly identified several applicable standards for ISTE teachers, ISTE students, and social justice. The program was updating to the new ISTE standards during the project semester, so their responses are based on the previous version. For the teacher standards, they identified design and develop digital age learning experiences and assessments (10 PSTs listed this standard), facilitate and inspire student learning and creativity (8), and model digital age work and learning (4). For the student standards, PSTs identified creativity and innovation (9), communication and collaboration (4), research and information fluency (3), critical thinking, problem solving, and decision making (4), digital citizenship (1), and technology operations and concepts (3). They also identified social justice standards for Identity, #1 (2) and #4 (2); Diversity, #6 (5), #9 (2), and #10 (2); Justice, #14 (3); and Action, #17 (2) and #20 (2). Because the PSTs thought that the computing-integration activity aligned with several standards, they can apply the activity to flexibly achieve these standards in their classroom.

Applications of Computing Integration in Science and Other Subjects

During the CT instruction in class, PSTs were asked reflection questions after introducing a set of CT concepts related to the five main concepts to connect the new information to their prior knowledge. When asked to share with their table after independent reflection, each table had lively discussions that were recorded through field notes. The point of these reflection items was not to create an exhaustive list of overlap between science and CT, but to help PSTs recognize that they already knew something about the concepts that were being discussed and to build upon that knowledge to learn about integrating computing in their field. Below are the responses that groups shared with the whole class. For example, when asked, "What problems in your field need to be decomposed?" PSTs gave examples including cycles (like the rock cycle so you can focus on how one part happens), graphing, and the scientific method.

- How would you apply defining parameters, conditionals, and test cases to teaching? – Come up with what-if scenarios for teaching (e.g., if computers aren't working, if you were doing an activity outside but there was bad weather); Set parameters for projects (e.g., size of bottle for making bottle rockets); Come up with rubrics for projects to set parameters and expectations
- How do mental models, logical thinking, iterative design, or sequencing apply to your field? – Scientific method relates to all of them (mental model of how experiment will go, logical thinking for hypotheses/reasoning, iterative design when things don't go as planned, and a specific sequence of steps); Iterative design for working on experiments and projects; Logical thinking is related to reasoning in science

- What do you wish you could automate in your field? – Grading; Running experiments repeatedly; Data collection – especially over a long time
- What problems in your field need to be decomposed? – Cycles, like the rock cycle so you can focus on how one part happens; Graphing; Scientific method

PSTs made connections to science teaching in their pre-student-teaching assignments and post-student-teaching reflections. Five of their ideas centered on data collection, analysis, and visualization, "I would love to incorporate this in my future classroom by making a lesson plan around making a graph in class together; showing them how to make science and technology fun." Four focused on promoting CT in science without include an activity that used programming, such as "In science computational thinking has a place. Most spacecrafts operate automatically. I could have a lesson that explains how they operate along with the computational thinking that went into its development." Two others emphasized that the computing integrated activity allowed them to deepen students' knowledge of topics, "The lesson presented helped to build upon students' understanding of their existing knowledge, and by manipulating the waves in pencil code, they were able to learn more, such as what frequency is, and what the different variables do."

In the pre-student-teaching assignment, five PSTs provided novel examples of CT in other subjects These examples tended to focus on using CT as a process for solving problems.

"CT implements a thorough process through which we can learn and create. It is a specific way to solve problems and work towards a solution in ways in which a computer may. It integrates computational skills into non-computing arenas that helps students to come to their own meaningful explanations of concepts. It's very useful when you need to create and/or use an algorithm for a project such as analyzing poetry or finding patterns that follow a mathematical rule."

In the post-student-teaching reflection, PSTs discussed their interest in applying CT and computing integrated activities to other subjects that they might also teach. The subjects include math, "I did like the PencilCode to an extent and when playing around with it outside of class, I came up with a fun possible activity with constructing right triangles in PencilCode and practicing the Pythagorean Theorem," English, "I would also like to utilize it into any of the English classes that I teach by using it for reading comprehension and analysis. I also like the entire process feels very organic. It's not a wild approach because a lot of people already follow these steps without thinking about it or without giving a name to it. As a teacher who wants to teach a discussion based class I think this is also useful in generating conversations in the classroom about the material," and history, "I would definitely be able to use this in science but also in history if I choose, I can do it with science with countless data sets that we may come up with in the course of our learning and I can use it in History to show the route of a group of people, or the death toll from a war."

Discussion

These results have implications for future implementations on CT within teacher education. First, activities took longer than expected, especially for the CT general instruction in class when we engaged in activities and reported back to the group. Second, the computing aspects of the CT general instruction needed to be scaffolded more, or perhaps the programming language needs to be introduced earlier to give students a better idea of what a programming language is. Third, lower-level explanations of the programming language, even though it is block-based, is needed. For example, one PST did not recognize that the block for dividing (_ / _) was for division, so we should not assume that the purpose of blocks are intuitive. Fourth, PSTs were uncertain about their ability to debug. We could include common mistakes made in the activities, like misspelling wavelength, during the introduction to the computing-integrated activity and ask PSTs to debug them to gain practice and confidence at debugging.

Overall, the integration was successful for teaching PSTs fundamental CT concepts and preparing them to teach a computing-integrated activity. CT and computing-integrated activities, despite being unfamiliar to most PSTs and anxiety-inducing for some, were embraced enthusiastically by all but one PST by the end of the month. Though the intervention took a month, it actually was one week of class time in a single course, one substantial assignment, and one student teaching experience. The PSTs engaged in several other important and unrelated class and preparation activities during the intervention, which perhaps helped them to make the many connections to other subjects that they did. One major limitation of the research is that we have not followed up with the PSTs after their final reflections, and we do not know whether they continue to be interested in computing-integration activities or will use them in their classrooms. Based on the enthusiasm from some PSTs and having already used them in a classroom, we expect that they will continue to use them, "I am just excited to be able to learn it inside and out to

use this program and others like it to bridge the gap between known information and newly acquired information for my students."

Acknowledgments

References

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832-835.

Armoni, M. (2016). Computer science, computational thinking, programming, coding: the anomalies of transitivity in K-12 computer science education. *ACM Inroads, 7*(4), 24-27.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community?. *ACM Inroads, 2*(1), 48-54.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the AERA*, Vancouver, Canada (Vol. 1, p. 25).

Cuny, J., Snyder, L., & Wing, J. M. (2011). Computational thinking—what and why?. https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

DeLyser, L. A., Goode, J., Guzdial, M., Kafai, Y., & Yadav, A. (2018). Priming the computer science teacher pump: Integrating computer science education into schools of education. Report published by CSforAll (pp. 1-62).

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33-39.

Gilbert, J. K. (2005). Visualization: A metacognitive skill in science and science education. In *Visualization in Science Education* (pp. 9-27). Springer, Dordrecht.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38-43.

Hsieh, H., & Shannon, S. E. (2005). Three approaches to qualitative content analysis. *Qualitative Health Research, 15*(9), 1277–1288.

Kale, U., Akcaoglu, M., Cullen, T., Goh, D., Devine, L., Calvert, N., & Grise, K. (2018). Computational what? Relating computational thinking to teaching. *TechTrends, 62*, 574-584. https://doi.org/10.1007/s11528-018-0290-9

Mouza, C., Nandakumar, R., Yilmaz Ozden, S., & Karchmer-Klein, R. (2017). A longitudinal examination of preservice teachers' Technological Pedagogical Content Knowledge in the context of undergraduate teacher education. *Action in Teacher Education, 39*(2), 153-171.

Sands, P., Yadav, A., & Good, J. (2018). Computational Thinking in K-12: In-service teacher perceptions of computational thinking. In M. S Khine. (Ed.). *Computational Thinking in the STEM Disciplines* (pp. 151-164). Springer.

Qian, Y., Hambrusch, S., Yadav, A., & Gretter, S. (2018). Who needs what: Recommendations for designing effective online professional development for computer science teachers. *Journal of Research on Technology in Education, 50*(2), 164-181.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127-147.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717-3725.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education, 14*(1), 5-16.