Computational Pedagogy Approach to STEM Teaching and Learning

Osman Yaşar and Jose Maliekal
The College at Brockport, State University of New York
Brockport, NY 14420, United States
oyasar@brockport.edu

Abstract: This article describes computational pedagogy, an approach to teaching principles of computing aided by modeling and simulation. Besides fostering abstraction skills in students, this approach motivates them to learn programming and supports both the deductive and inductive forms of instruction. Specifically, the article reports how computational pedagogy was implemented in the context of a professional development program for teachers and what impact that professional development program had on STEM teaching and learning in secondary schools.

Introduction

Proliferation of digital technology has led to the inclusion of computational thinking and modeling skills in the latest K-12 Learning Standards in the United States (2013). Teaching with technology is complex, however. Not only does it often require customization (Koehler & Mishra, 2008) but the technologies themselves must be content specific and pedagogically suitable also. Teachers need help, through professional development (PD), to deploy appropriate technologies in the classroom, stay up-to-date with emerging technologies, and assess efficacies of different pedagogical approaches (Loucks-Horsley et al., 2010). The conceptual framework of technological pedagogical content knowledge (TPACK) posits that understanding of three knowledge domains – content, pedagogy, and technology – and interplays among them (Fig. 1) are vital for the effective deployment of technology in the classroom (Koehler & Mishra, 2008). The conceptual framework of TPACK and the computational approach to problem solving (henceforth referred to as computational-STEM) pioneered by scientists in national laboratories and universities more than five decades and practiced widely today (Yaşar & Landau, 2003) have much in common, as we elaborate below.

To start with, both TPACK and computational-STEM, require its practitioners to acquire skills in multiple disciplinary areas, a complex task. That interdependency among multiple knowledge domains give rise to many contextually bound variables only increases this complexity. As noted in the Computing Curricula (2005) and field experiences (Yaşar et al., 2003-2014), balancing the breadth and depth of multiple fields in an interdisciplinary course or training program requires not only increased preparation time for the instructor but also frequent recalibration based on feedback from the audience. Undergraduates, especially freshmen with inadequate preparation, often experience frustration and withdrawal, as do pre-service teachers, who are required to complete certification coursework on top of the requirements of their academic major. Even in-service teachers undergoing professional development are not immune from this predicament.

Thanks to the TPACK framework, there is now the opportunity for computing and education experts to speak a common language. Computing community has been championing for ‘contextualized computing education’ in introductory college courses and through the teaching of ‘computational thinking’ skills in AP high school courses (The College Board, 2011). Informed by the experience of conducting a PD, this article advances the idea that the goals of teacher preparation and secondary STEM education can be improved by orienting the attention of the higher education and K-12 towards each other in a way that focuses on developing abstraction skills of students, crosscutting practices, and computational pedagogies.

Figure 1: The TPACK framework (Koehler and Mishra, 2008).
The Role of Abstraction Skills in Computing Education

While ‘attention to details’ is important to master any skill, human capacity to store information is limited. The most effective strategy to improve memory performance and information retrieval for problem solving is organizing disparate pieces of information into meaningful units (How People Learn, 2000). The process of abstraction can help by simplifying, categorizing, and registering key information and knowledge for quicker retrieval and processing. Abstraction is an inductive process, whereby details are filtered out and focus is placed on general patterns, thereby allowing one to assign priority and importance to the newly acquired information. Researchers still do not know how humans make strong generalizations and construct powerful abstractions from sparse, noisy, and ambiguous data (Tenenbaum, 2011), but it aids us in our personal and professional lives.

Shielding oneself from low-level details is key to abstract thinking. Abstraction enables computer scientists to write large and complex codes (such as operating systems, compilers, and networking) efficiently by distributing the complexity of the problem at hand into seemingly independent layers and protocols, thereby hiding the details of how each layer accomplishes specific tasks. Dijkstra, an early pioneer in computing science (CS), regarded abstraction as the most vital activity of a competent programmer (Armoni, 2013). Recently, Wing (2006) described computational thinking, a skill now recommended for early grades, as “using abstraction and decomposition when attacking a large complex task or designing a large complex system.” Since abstraction skills are so essential, CS educators strive to improve it beyond what is innate through teaching, training, and practice. However, as noted by Armoni (2013), teaching abstraction in the context of programming is challenging. Four levels of abstraction have been identified. From low to high, they are: (1) execution-level, (2) program-level, (3) object-level, and (4) problem-level. The ability to oscillate between different levels of abstraction is often what distinguishes a good programmer from a poor programmer. Even advanced undergraduates barely move beyond level 2 (language-specific) or 3 (algorithm-specific) abstraction (Armoni, 2013). While reaching level 4 is important to transform appropriate algorithmic and programming skills into different application contexts, the teaching of programming itself does not seem to accomplish this. As a result, a problem solving approach has now been recommended to teach computing concepts in the context of applications (Computing Curricula, 2005). Below, we propose a pedagogical approach to improve problem-level abstraction skills using computational modeling and simulation technology (CMST).

Computational Pedagogy

Because humans use abstraction (inductive: from details to general) for effective information storage, a reverse process (deductive: from general to details) is often used to retrieve information and use it to interpret novel situations. Historically, teachers have used the deductive approach to instruct students (See Fig. 2). In recent years, the inductive approach has been introduced in many forms. Inquiry-based teaching is one such form. It enables the learner to arrive at his/her own authentic conclusions based on facts and experience. Both deductive and inductive approaches have their own pros and cons. But if the two are used together, the learner may be led to oscillate between multiple levels of abstraction as desired in computing education (Armoni, 2013).

Figure 2: Illustration of the informational organization and the resulting deductive/inductive instructional pedagogies.

Figure 3: Illustration of Optimal Flow in learning (Csikszentmihalyi, 1990).
An activity that can facilitate deductive and inductive reasoning is ‘computational modeling and simulation’ of scientific phenomena. Modeling utilizes abstraction by simplifying the physical reality. Such a (inductive) simplification helps application scientists to eliminate less critical information from consideration and focus on what is critical. In education, computational modeling and simulation can support learning by enabling the learner to grasp important facts surrounding a STEM topic in the following way. First, use modeling to introduce a topic deductively at its elemental level (high level of abstraction) and then guide the learner to delve deeper into details through simulations and hypothetical situations to facilitate rediscovery of general principles inductively. Together, modeling and simulation process, therefore, encompass deductive and inductive pedagogies. Clearly, they provide a dynamic medium for the learner to conduct scientific experiments in a friendly, playful, predictive, eventful, and interactive way. Oscillating between deductive and inductive (♣↑) approaches in a stepwise and non-threatening way is consistent with the pedagogical framework of flow (Csikszentmihalyi, 1990) and scaffolding strategy of balancing skills with challenges as shown in Fig. 3. Since instruction consisting of computational modeling and simulation embodies an inductive strategy (for practitioner and learner) and a deductive strategy (for teacher and learner), activities associated with it can put the learner at the center of a constructivist learning experience and utilize both bottom-up and top-down approaches in an iterative way (NSTA, 2008).

Computational pedagogy carries many other characteristics of the constructivist approach (Grabinger, 1995), including inquiry-based, generative, cooperative, and interactive learning as well as project and team based instruction. Using models and simulations, learners become actively engaged in “doing,” rather than passively “receiving” knowledge. Advanced software tools, such as Interactive Physics (IP) and AgentSheets (AS), can be used to teach about a scientific topic via a series of student-controlled experiments without having the student to know the mathematical and scientific details of the phenomenon under study.

Using existing models that are posted on the Web, instructors can introduce a topic via games and simulations without burdening students with complicated STEM and computing principles first. Students may modify an existing model, or create one from scratch. Tools such as IP and AS can be used to create many engaging games and science experimentations. Such tools also come with features (i.e., buttons for controlling the run-time and accuracy) that would allow the learner to explore some of the underlying principles of computational modeling. After an initial experimentation with modeling in the context of a game or science topic, students can be introduced to a core principle, as outlined below, that can eventually (and quickly) lead them to understand several aspects of computational thinking skills (Wing, 2006), including the virtue of decomposing a problem into smaller chunks, the tradeoff between computational cost and accuracy, and the need to use a programming language in order to study complex problems or to handle situations with increasing number of data points (due to decomposing the problem into much smaller chunks).

A Core Principle of Computational Modeling and Simulation

A core principle of scientific modeling and simulation is that differential equations are solved to predict a system’s behavior. Analytically, this is done via mathematical integration (\( \int f \, dx \)). But an analytical answer is not always available, especially when there are multiple variables and higher-order derivatives describing \( dx \) (change in \( y \)). We, then, employ numerical integration that can be taught at early grades as follows.

To predict a system’s new situation, be it tomorrow’s weather or spread of a disease, we need to know the old situation (initial conditions) and a formula to compute the change. To comprehend this via a simple case, students can apply the “new = old + change” methodology in conjunction with the formula \( dy = 2x \cdot dx \) (change in \( y \)) to find a numerical approximation to the expression \( y = x^2 \), provided that the initial condition is known. To do this, a table of \( x \) and \( y \) may be constructed using initial data starting from \((0,0)\) for different choices of increment in \( x \) (\( dx = 1, 0.5, 0.1 \), and so on). Table 1 illustrates the steps to construct such a table. Table 2 demonstrates the numerical steps involved for \( dx = 1 \) and \( 0 \leq x \leq 5 \). When the numerical results for different \( dx \) values (Fig. 4) are compared to the analytic solution (\( y = x^2 \)), students could be led to discover the correlation between the step size \( (dx) \) and the accuracy of the results: the smaller the \( dx \), the more accurate the answer. While a human can calculate a few data points by hand when \( dx \) is 1, or 0.5, the need for automation (and accuracy) becomes obvious for smaller \( dx \) values such as 0.1 or 0.05. Spreadsheets, such as Excel, may be used to automate the calculation and graph \( y = f(x) \) curves. But for much smaller increments \( (dx) \), such as 0.001, 0.0001, or 0.0000001, students will discover that spreadsheets are not of much help. The need for finer and faster automation, via computer programming, becomes evident.
In an after-school project run by the first author, several 9th graders from Brighton (NY) were able to produce the harmonic motion in Fig. 5, using Excel to compute the position \(x_{\text{new}} = x_{\text{old}} + dx\) and velocity \(v_{\text{new}} = v_{\text{old}} + dv\) of a spring-driven object at times separated by \(dt\). Here, \(dx = v \cdot dt\) and \(dv = a \cdot dt\); where \(a\) (acceleration) is Force/mass, or \(- (k \cdot x)/m\), with \(k\) being the stiffness coefficient of the spring and \(x\) is the displacement of the box from the equilibrium position. Although small time intervals (integration step, \(dt\)) were needed to dampen the numerical error in Excel (Fig. 6), students enjoyed having more control over the outcome in comparison to simulations with IP. To further control the numerical error in modeling, students inquired about other tools and, as a result, learned basic structures of a programming language (Python) to obtain more accurate results using time intervals several orders of magnitude smaller than what was practically doable with Excel. Once these students grasped the principles of modeling and computation, they started to look under the hood and understand how higher-level tools such as IP worked. The following year, these students enrolled in programming and physics courses and, as part of the same after-school program, they modeled two-dimensional orbital motion with Excel (see Fig. 7), using the gravitational force \(F = G \cdot M \cdot m / r^2\) to compute change in position \((x, y)\) and velocities \((v_x, v_y)\) as follows:

\[
\begin{align*}
x_{\text{new}} &= x_{\text{old}} + v_x \cdot dt; \\
v_{x,\text{new}} &= v_{x,\text{old}} + a_x \cdot dt; \\
 a_x &= (x/r) \cdot a, \\
y_{\text{new}} &= y_{\text{old}} + v_y \cdot dt; \\
v_{y,\text{new}} &= v_{y,\text{old}} + a_y \cdot dt; \\
 a_y &= (y/r) \cdot a, \\
r^2 &= x^2 + y^2; \\
v^2 &= (v_x)^2 + (v_y)^2; \\
a^2 &= (a_x)^2 + (a_y)^2.
\end{align*}
\]

While these high school students were exceptions, their (deductive + inductive) learning cycle does show a path that can be promoted in the classroom. Linking computing to science through the computation of change provides a motivation for science majors to learn programming and for computing majors to learn more about science. The motivational and deductive/inductive cycle of such an approach can be used to broaden participation in computing and sciences, as witnessed in our case study below.

**Table 1.** An algebraic scheme to build an \((X, Y)\) table.

| \(X\) | \(X_1 = 0\) | \(X_2 = X_1 + dx\) | \(X_n = X_{n-1} + dx\) |
| \(Y\) | \(Y_1 = 0\) | \(Y_2 = Y_1 + 2 \cdot X_1 \cdot dx\) | \(Y_n = Y_{n-1} + 2 \cdot X_{n-1} \cdot dx\) |

**Table 2.** Hands-on illustration of the algebraic scheme.

<p>| (X) | (dX = \cdot X) | (Y) | (dY = \cdot Y) | (Y = Y) |</p>
<table>
<thead>
<tr>
<th>(X)</th>
<th>(Y)</th>
<th>(X)</th>
<th>(Y)</th>
<th>(X)</th>
<th>(Y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>25</td>
</tr>
</tbody>
</table>

**Figure 4:** Numerical results (dashed lines) are compared to the analytic solution (solid line).

**Figure 5:** Simple harmonic motion with Interactive Physics.

**Figure 6:** Excel computations of position and velocity for the same IP simulation shown in Fig. 5.
Table 3: Brief descriptions of the tools used in training.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
</table>

K-12 Impact of the Computational Pedagogy

To address the issue of low enrollments by reaching out directly to pre-college level, faculties from the College at Brockport designed an introductory-level CMST institute in 2003 for teachers in local school districts, particularly the urban Rochester City and suburban Brighton Central SDs. External evaluators were hired to conduct the evaluation efforts. Driven by the needs of schoolteachers and the nature of interdisciplinary education, the program and its subsequent modifications (Yaşar, 2014) resulted in an effective PD (Loucks-Horsley et al., 2010).

Supported by the National Science Foundation, the PD program offered incentives for teachers, including stipends, laptops, advanced graphing calculators, and yearlong coaching support. In return, teachers were expected to develop and implement lesson plans and modeling examples using the tools they learned (Table 3); prepare a student team to enter a project-based Challenge competition at the College; and offer turnkey training to a colleague who could not attend summer training. Due to an overwhelming interest from teachers, different levels of training were also added. Tables 4 shows the number of in-service teachers who benefited from the summer institute and Table 5 shows enrollments in a credit-bearing course for upper-level undergraduate and beginner-level graduate students that evolved from the summer institute. In-service teachers who attended the summer training were selected from partnering school districts that supported the initiative and gathered data to study the impact of the PD.

Table 4. Enrollments (In-service Teachers) at CMST Summer Institute (2003-2010)

<table>
<thead>
<tr>
<th>Subject</th>
<th>1st Year Teachers (Intro)</th>
<th>2nd Year Teachers (Advanced)</th>
<th>3rd Year Teachers (Expert)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Urban</td>
<td>Suburb</td>
<td>Total</td>
<td>Urban</td>
</tr>
<tr>
<td>Math</td>
<td>96</td>
<td>14</td>
<td>110</td>
<td>42</td>
</tr>
<tr>
<td>Science</td>
<td>38</td>
<td>15</td>
<td>53</td>
<td>17</td>
</tr>
<tr>
<td>Tech</td>
<td>7</td>
<td>3</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Spec. Ed</td>
<td>14</td>
<td>1</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>Subtotal</td>
<td>155</td>
<td>33</td>
<td>188</td>
<td>66</td>
</tr>
</tbody>
</table>

Note: The urban group includes Rochester City and the suburban group includes Brighton Central and 9 other districts.

Table 5. Number of Pre-service Students Receiving CMST Training During Fall and Spring semesters

<table>
<thead>
<tr>
<th>Course</th>
<th>2003</th>
<th>2004</th>
<th>2005</th>
<th>2006</th>
<th>2007</th>
<th>2008</th>
<th>2009</th>
<th>2010</th>
<th>2011</th>
<th>2012</th>
<th>2013</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAS 401/501</td>
<td>3</td>
<td>9</td>
<td>10</td>
<td>41</td>
<td>50</td>
<td>34</td>
<td>34</td>
<td>15</td>
<td>11</td>
<td>13</td>
<td>58</td>
<td>251</td>
</tr>
</tbody>
</table>

Partnersing districts reported high gains in teacher quality and student achievement during the initiative (2003-2010). It is hard to correlate district-level gains directly to our initiative as multiple factors are usually at play. However, we triangulated emerging patterns through quantitative data, surveys, and classroom observations. About

Figure 7: Orbital tracking of the Earth using Excel.
95% of beginner-level attendees completed the training. While such retention speaks highly of the engaging aspect of training, it was not without a challenge. During the first summer, many teachers were ready to drop out at the end of the 1st week. By the end of the 3rd week, they overcame the fear of new technology, of learning materials outside of their subject areas, and of teaming up with teachers from other subjects for developing modeling examples and lesson plans. Exit interviews indicated 100% satisfaction, with 57% rating it ‘very beneficial’ and 43% ‘beneficial.’

The data suggested that after their first year training, teachers did not immediately feel fully prepared to put the training into practice. About 35% felt prepared to apply modeling in the classroom in the following year, 40% felt ‘probably prepared’, 15% felt unsure, and 10% did not feel prepared at all. It was not until their third year of using new pedagogy and tools that the average teacher felt confident and comfortable. While only 60% of the beginner-level teachers reported occasional use of modeling tools in their grades 7-12 classrooms, 78% of expert-level teachers reported that they used these tools regularly. In general, teachers recognized that new tools made it easier to construct a simulation, but they reported not having enough training to construct new ‘models and simulations’ in their subject areas, or even understand lesson plans consisting of readymade models. Some teachers did not have adequate math and computing background to fully understand the mechanics of modeling to develop authentic simulations. As a result, they could not feel ownership of the newly acquired knowledge to go beyond rote memorization of tool usage by the end of the 1st year. These problems were addressed through equipment support for classrooms, extending the length of summer institute, and adding more training for interested teachers (Table 4).

In a 2010 survey of 40 active teachers, 94% agreed that the training made them more effective in the classroom; 87% agreed that it strengthened their pedagogical skills; 73% agreed that it strengthened their content knowledge; and 80% believed that their participation served to build leadership skills. Some teachers moved up quickly to key administrative positions, including the posts of principals and curriculum directors. In examining the job retention rate of participating teachers, 93% of those from Rochester City were still employed in their district at the end of 5 years. This ratio fell to 73% at the end of 7th year. At the same time, 79% of suburban teachers from Brighton were retained after 5 years; and only 50% after 7 years. Both districts publicly credited the CMST program for contributing to recruitment of new teachers and delaying retirements of veteran teachers. While the retention rate at the City exceeded the national average (50% within 5 years), the mobility of teachers within such a large district is common. This both strengthened and diluted the integration of computational approach into the curriculum. While urban teachers were initially more eager than their suburban counterparts to learn (and acquire) new technologies, the suburban teachers were more interested in technological and pedagogical content training. However, the urban profile changed over the span of five years to a point of resembling the suburban teachers.

Most teachers agreed that using the modeling and simulation tools in their classrooms significantly increased student engagement. In the 2010 survey conducted by external evaluators, 90% of teachers agreed that computational inquiry made math and science concepts more comprehensible to students. Student reaction to modeling (versus traditional techniques) was found to be 97% favorable in math and 77% in science classes. While science classes utilized technology less due to limited access and lack of science-related modeling examples, in instances where it was utilized, a deeper understanding of science topics was achieved, compared to math topics (83% vs. 76%). Majority of teachers agreed that students responded differently to technology. They reported that while male students showed more interest in playing with technology and plowing through the details with less regard to the big picture, female students initially seemed reluctant and timid but excelled when details (curriculum) were put into context of real-world problems and projects.

Alongside the self-reporting by teachers, student achievement data were analyzed in the participating districts via report cards and standardized test scores. Improved performance in math and the graduation rate with Regents diploma clearly emerged as the greatest gains in both districts during the initiative. The percent of students receiving a Regents diploma increased significantly from the baseline (RCSD: 21% → 59%, BCSD: 84% → 95%). In particular, the initiative exposed urban students to college experiences and opportunities, leading to increased interest (78% → 83%) in both 2-year and 4-year college programs over the period examined. The Rochester City School District’s passing rate (>65/100) in NY State Grade-8 Math exam increased from 10% to 33%. Its passing rate in NY Regents Math-A exam (Grade 11-12) increased from 13% to 67%. Since majority (90%) of secondary school math teachers participated in the professional development, a cultural change was evident at all levels of math curriculum. Passing rate in sciences, however, improved less significantly – Physics (3% → 22%) and Chemistry (9% → 27%) – which reflected the teacher participation levels in those areas. Limited access to computer labs, skepticism about use of technology, and lack of lesson plans and curricular modules may have discouraged.
science teachers at the early stages of the initiative to invest in training. At the same time, there were no efforts to link what was happening within the initiative with district science efforts at large. At Brighton, passing rates improved in both mathematics (Math-A: 51% → 99%) and sciences (Physics: 52% → 78%), reflecting, perhaps, a balanced participation of district’s math and physics teachers in the initiative. The number of students taking General Physics increased from 50% to ~100%. The number of students taking AP Physics also doubled.

Conclusions

While new pedagogical strategies have emerged in recent years, their deployment have been varied, in part because not all content can be taught with new pedagogies, and in part, because not all technology tools are pedagogically suitable. Teachers need help to identify and deploy effective technologies. The Computational-STEM approach described in this article can at once improve the teaching of math and science concepts in high schools and contribute to the education of a new cadre of programmers with higher abstraction skills and broader problem solving skills.

Providing professional development (PD) to STEM teachers using the computational modeling and simulation technology (CMST) approach is relatively new. This nascent approach carries with it both a new vision and forward momentum to excite masses due to its motivational, evolutionary, strategic, and integrated nature. Its dependence on fast-changing technology tools lead to frequent evaluations, feedback gathering, and mid-course corrections, all of which are desirable attributes of an effective PD (Loucks-Horsley et al., 2010). Furthermore, as we have witnessed, it facilitates partnerships among various domains: subject areas (math, science, technology, and pedagogy), communities (urban and suburban school districts), and institutions of higher education and K-12. While the two school districts in our initiative were very different, the partnership motivated urban and suburban teachers to learn from each other and to make progress by embracing best practices and collegial interaction. Students participated in joint activities and projects and embraced opportunities to visit the college campus. Performance gap between urban and suburban students was reduced. Teacher aptitudes improved in both districts. In general, affluent suburban school districts tend to be small; their budgets and culture embrace rigorous academic curriculum, technological infrastructure, and smooth progression. Large urban districts are often financially stressed. As a result, they face a myriad of problems, ranging from insufficient technology readiness to frequent changes in staff assignments (within buildings). Understandably, implementing CMST approach may take longer. The implementation process may require more adaptive strategies on the part of teachers, coaches and staff. Yet, the impact in an urban setting could be higher if and when a cultural change takes place; and that is what we observed.

Evidence from our PD program that used the CMST approach shows high teacher retention rate, improved instruction, and high gains in student achievement. Surveys conducted among hundreds of participating teachers showed the following. Nearly 94% of those who participated in the survey agreed that the PD program made them more effective in the classroom; 87% agreed that it strengthened their pedagogical skills; 73% agreed that it strengthened their content knowledge; and 80% believed that their participation in it served to build leadership skills. A few examples of teacher statements that support the above quantitative analysis are given below.

“The CMST Institute has changed my teaching in a positive way. It stopped many questions students used to ask, which now allow more time for instruction.”... “The way computational pedagogy changed me was how I introduced new topics. My lectures to the class have become explorations of the concepts. The role of students has changed from passive to active and my role has also transformed from being the expert to become the guide during the explorations.”... “With computational technology, I have reached deeper to a portion of students that normally turn off science.”... “Learning the CMST pedagogy and tools are really two different things. I don't think that I really understood the pedagogy during my first summer institute. It took me some time to get the idea that computational science was using computer tools to solve real world problems and simulate real world phenomenon. I then knew that the goal was not necessarily to use the tools as display tools, but to get students to use them to create models and explore underlying science concepts”... “CMST style teaching has had a dramatic effect upon the students in my first year after the summer institute. The use of the computational lessons and video clips has engaged a section of students who would normally tune out school. I had a greater number of students this year who went through the entire year and passed compared to last year.”... “Technology provides great visual enhancement in the classroom and it captures student attention. When I worked on Interactive Physics with my students, it reinforced the concepts they had learned in the science club. An example is the motion of particles in liquids and gases. Presenting concepts
in a different way helps some students.”...“The institute allowed me to gain a more in-depth knowledge of the problems faced by the teachers working in other content areas. This experience has allowed me to step out of the isolation of my content area and place myself in the shoes of teachers in other content areas. My experience with other participants and with new technologies gives me further ammunition to push for greater integration of math, science, and technology with concrete examples of how this can be done in the classroom.”

For those interested in pursuing similar initiatives, details of the PD based on the CMST approach can be found in Yaşar et al. (2014). We continue to offer training to in-service and pre-service teachers. A database consisting of modeling examples and lesson plans can be found at www.brockport.edu/cmst.

Acknowledgement: Support was provided by the NSF Grants #0226962, #0942569, and #1136332.

References


