Computational Pedagogical Content Knowledge (CPACK): Integrating Modeling and Simulation Technology into STEM Teacher Education

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

Abstract: Teaching with technology remains as a challenge for STEM teachers. Making judicious choices of when, what and how specific tools and pedagogies to use in the teaching of a topic can be improved with the help of curriculum inventories, training, and practice but as new and more capable technologies arrive, such resources and experience do not often transfer to new circumstances. Even within the realm of a particular technology, the choice of which tool would be better to teach a topic needs judicious thinking. This article presents a qualitative case study in which pre-service and in-service teachers are trained about not just the use but also basic operating principles of a technology in an attempt to enhance its integration into teaching in a more permanent, constructivist, and tool-independent way. The focus of our work is computational modeling and simulation technology (CMST). Based on pre/post activity surveys, focus group interviews, and artifacts, the results suggest that if teachers move beyond 'using' tools and learn the basic mathematical and computational principles of modeling and simulation, they can construct their own models, rather than using readymade ones, navigate between multiple tools that operate on the same principles, and as a result gain confidence to more judiciously decide under different circumstances as to what tools might better facilitate teaching of a topic.

Introduction

Challenges in technology education have brought education and computing communities together. Computer scientists' challenge is the current crisis of underproduction and underrepresentation of students. It has triggered a major K-12 outreach to help boost interest in computing by addressing learning difficulties of computing concepts and principles (Fincher & Petre 2005) and improve the teaching of these concepts in the context of applications (CC 2005). Educators' challenge is how to improve technological pedagogical content knowledge (TPACK) of teachers (Mishra & Koehler 2006). Teaching with technology is complex; not only does it often require customization but also the technologies themselves must be content specific and pedagogically suitable (Koehler & Mishra, 2008). 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). However, due to frequent changes in available tools and their increasing capacity; curriculum inventories and teacher PD content do not often transfer to new circumstances. While latest technologies offer more capacity, their optimum utilization may necessitate knowledge of underlying principles for easier transfer into new circumstances and better integration (Koehler & Mishra 2005, 2008; Niess 2005; Flick & Bell 2000).

Computational modeling and simulation technology (CMST) offers a way to potentially address some of the problems mentioned above. There is enough evidence in the literature about effectiveness of computer simulations as a tool (Bell & Smetana 2008; Wieman *et al.* 2008; Rutten, van Jolingen, van de Veen 2012; Smetana & Bell 2012; Maeng *et al.*, 2013) but difficulties remain regarding its current use, spread and integration into classroom instruction (Koehler & Mishra 2008). While readymade computer simulations can be considered as effective tools that allow learners to manipulate variables, test hypotheses or develop relationships between variables in the context of a topic, design-based modeling and simulation tools offer more features to also manipulate mathematical and computational accuracy of simulations and the laws of nature that are embedded in the model.

For the past decade we have been promoting the use and integration of design-based CMST tool (such as Interactive Physics, AgentSheets, Excel, STELLA, TI-84 graphing tools, and Geometer's Sketch Pad (GSP)) into secondary school instruction. Through support from the National Science Foundation's Math and Science Partnership (MSP) program, we offered a 3-tier (beginner, advanced, and expert) summer workshop from 2003 to 2010, serving 188 STEM teachers from a high-needs urban (Rochester City) school district (SD) and a high-achieving affluent suburban (Bright Central) SD. The beginner-level workshop was meant to teach them technology knowledge (TK); the advanced-level training targeted their technological content knowledge (TCK) development, involving use of these tools to construct curriculum modules and lesson plans in their subject areas; and the expert-level workshop focused on teaching them CMST principles while demonstrating how this particular technology interacts with specific instructional methods to improve their technological pedagogical content knowledge (TPCK, or TPACK). Despite its favorable impact, curricular, structural, and technology access barriers made it difficult to sustain its success, use, and spread with the same intensity after funding for the program ended in 2010. In an exit survey, many teachers complained about not having adequate number of available curriculum modules and not having enough time to gain tool proficiency in order to construct new models and simulations in their subject areas, or even understand readymade lesson plans that used such models.

Structural and curricular barriers will be hard to overcome until teaching and learning of computational skills are fully implemented as a result of the new state and national student learning outcomes. However, to address training and inventory needs, we developed a comprehensive database of curriculum modules (see www.brockport.edu/cmst) using the tools listed above. The beginner-level in-service training was transferred into a methods course in natural sciences (NAS), namely NAS 401/501 CMST Tools, serving more than 300 hundred in-service and pre-service teachers from 20 school districts since 2008. No data was collected on how this methods course impacted both groups because of the high number of teachers and SDs, lack of funding to collect and analyze data, and the cumbersome process and protocols to follow up with a school district to collect data. A second methods course (NAS 402/502 CMST Principles) was developed two years ago to go beyond tool use and teach fundamental principles and pedagogical aspects of modeling and simulations.

Informed by findings and resources of our work in the past decade with inservice teachers from Rochester City SD (RCSD) and Brighton Central SD (BCSD) as well as the preservice teachers from our recent Robert Noyce Scholarship program, we are currently exploring *how a deep knowledge of the workings of CMST tools improves development of teachers' TPACK skills to judiciously know what, when, and how to use such tools in the teaching of a topic.* This work has also been inspired by a recent study (Maeng *et al.* 2013) to contribute to the ongoing TPACK research. In that study, authors investigated technology-enhanced inquiry instruction and TPACK preparation of 27 secondary school preservice science teachers in the context of a 2-year Master of Teaching program. While our preservice Noyce Scholars similarly took 3 courses (educational technology, NAS 401/501 & NAS 402/502 methods courses) and did a capstone project, there are some distinctions between our studies. Their investigation included general technologies, such as digital images, videos, online computer simulations, animations, probeware, spreadsheets, and so on; our focus is only on a particular technology ─ i.e., design-based computational modeling and simulation technology. The second distinction is about research methodology. While their research included classroom observations during student teaching, we had no data from classrooms of the preservice Noyce Scholars yet ─ this is part of future plans. What follows in the sections below is how CMST can be considered within the general TPACK framework; background knowledge about CMST principles and pedagogy; and a discussion of qualitative data collected from pre/post activity surveys and focus group interviews.

Computational Pedagogical Content Knowledge (CPACK)

Figure 1 illustrates interaction among CMST's multiple knowledge domains (mathematics, computer science, and sciences). This interaction has given rise to a new content domain as witnessed by establishment of many bachelor's, master's, and doctoral programs in computational science and engineering (CSE) in the past two decades (Swanson 2002; Yaşar *et al.* 2000; Yaşar & Landau 2003). Emerging of a new knowledge domain from interaction of multiple domains stands testimony to what has been advocated by Mishra and Koehler (2006) and other educators about the unique nature of TPACK when technology, pedagogy, and content closely interact. Adapting a saying by Aristotle to our case, we can perhaps say "the whole is different from its parts." The interaction of math, science, and computing technology has also given rise to a particular pedagogy as described in Yaşar & Maliekal 2014a-b, Yaşar *et al.* 2006a-b, and Yaşar 2004. This is even more interesting than the emergence of a content (CSE) domain, because pedagogy was not even among the constitutive domains of CMST to start with. In the context of the TPACK framework, CMST may be understood as Computational Pedagogical Content Knowledge (CPACK) as it only involves the computational technology whereas TPACK is more general and inclusive. CMST's link to K-12 teaching pedagogies was not obvious initially and it took a long time for it to evolve to this point as explained below. The declining number of high school students entering college STEM programs (Augustine 2007) as well as the prerequisite knowledge needed in such an interdisciplinary field led to K-12 outreach activities by CSE educators, including the authors. Close collaboration with teachers created clear benefits of using computer simulations, animations, and images coming from national labs, NASA, and the academic scientific computing community. Compounding this later was the increasing capacity, friendliness and easy-to-use feature of new software tools that made it possible to replicate in the classroom the inquiry process a scientist followed in his/her research practice. Scientists often start with a statement, a hypothesis, an experiment, or a computer model (a simplified version of reality). They, then, deductively dig further to gather evidence using theoretical scenarios, experimental trials, or simulations. Finally, putting the pieces of the puzzle together after much searching and sorting, they inductively arrive at some conclusions. In this two-way process (top-down🡇 and bottom-up🡅) as shown in Fig. 2, if the new conclusions formed inductively from detailed evidence are different from the initial assumption, understanding, or model, then one can say that a *conceptual change* has taken place (Carey 2000). In case of a study that uses a computer model, new findings provide feedback to modify the initial model. Since learning theories suggest that students learn science by practicing it the way scientists do (Bransford, Brown, and Cocking 2000; Donovan & Bransford 2005), then recent design-based modeling and simulation tools offers just that; not only by bringing the scientific inquiry process to the classroom but also representations of systems that are too big (e.g., solar system), too small (e.g., molecules), too dangerous (e.g., fire), or expansive to access physically.

Recent advances in computer software have alleviated many of the hurdles that once stood in the way of model construction. Through advanced software tools, such as the Interactive Physics (IP), young students can construct a model (e.g., a pool table, a rollercoaster, or a spring experiment as seen in Fig. 3) and conduct simulations without having to initially know the mathematical, computational, and scientific principles embedded in the model. Actually, regardless of whether these models are created by students or introduced by teachers, this deductive approach engages students as it hides threatening details and boring aspect of science (Augustine 2007). Once a model is constructed by selecting from a drop-down menu and drawing appropriate connections, a series of controlled simulations can lead to changes to the model and seeing the consequences to the simulations. This process is more than just establishing a relationship between variables as it is done with online simulations. The more control the user wishes to have over the design and modification of the model, the more interest it might generate about the underlying principles and facts used in the model construction by the tool itself. A teaching method consisting of model construction and subsequent simulations might, then, create a dynamic cycle— one in which learners would be at the center of constructive learning to test a hypothesis or principles surrounding a model via controlled experiments, then analyze and evaluate results to go back to improving the model. Anyone who learns in this deductive/ inductive fashion (see Fig. 2) would, in fact, be practicing the craft of scientists (Hammond 2001). In so doing, learners may be guided to reconcile the difference between their own preconceptions and simulation results, resulting in deep, as opposed to shallow, learning. The efficacy of this on deepening content knowledge has been evidenced at a project-based afterschool program organized by authors (Yaşar *et al.* 2014). Experience by a group of 9th and 10th grade students from this program is published in NSF's MSPNET library (2005, 2006). Studying its effectiveness in a regular classroom will require additional research.

Mathematical and Computational Principles of CMST

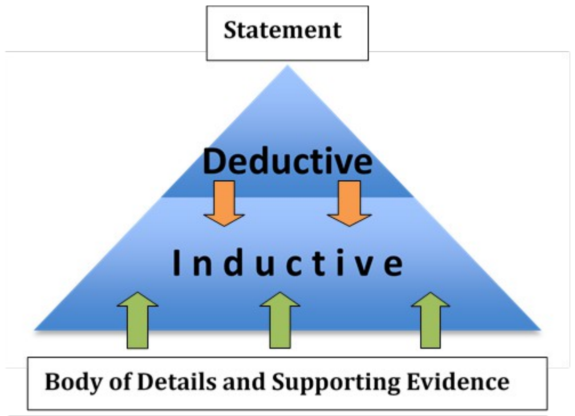
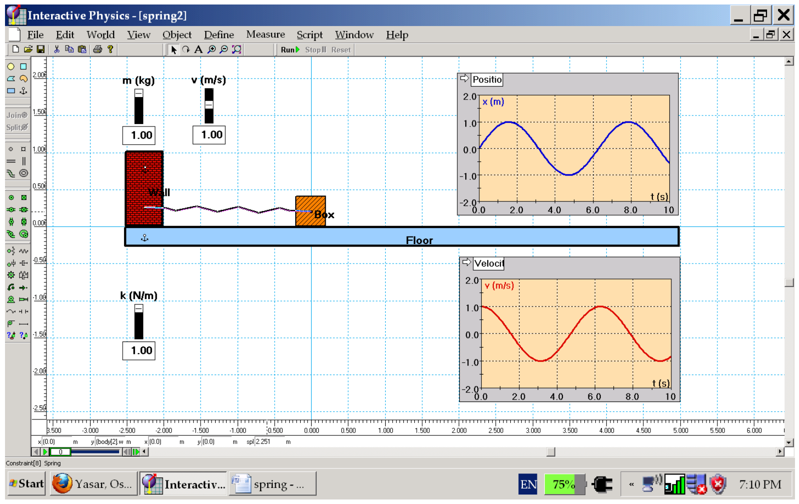


Figure 2: Deductive and Inductive Approaches

A core principle of scientific modeling and simulation is that differential equations are solved to predict a system's new behavior based on its old behavior and the change that took place. We do this because the nature often talks to us in the language of change, such as $dy/dx= 2x$, instead of revealing a direct relationship that we so desire, such as $y = x^2$. Mathematicians call dy/dx a differential equation, derivative, or simply the rate of change as it is known in secondary schools. One can often infer a direct relationship, $y=f(x)$, by summing up (or integrating) the incremental changes in y; i.e., $y_{new}= y_{old} + dy = y_{old} + 2x \cdot dx$ (the change in y depends on x and the change in x). We can manually do this summation until we cover the range of interest in x by updating it similarly as $x_{new}= x_{old} + dx$. It often can be done mathematically, too, via a simple rule: increase the power of x by 1 in the expression for dy and divide it by the new power – i.e., $y = \int dy = \int 2x \cdot dx = 2 \cdot (1/2) \cdot x^{1+1} = x^2$. However, the problem is that finding an analytical answer this way is not always possible; especially when there are multiple variables and higher-order derivatives (derivatives of derivatives) describing the change. Examples include climate change, spread of fire, population dynamics, and nuclear reactions, etc. What we do under those conditions is to do it manually (numerically), as mentioned above, by using a simple algebraic equation (*new = old + change*) for x and y over and over. A table of *new* data points (x_n, y_n) can be constructed using the *old* data points (x_{n-1}, y_{n-1}) as shown in Table 1 and 2 where we arbitrarily chose the initial conditions as $(0, 0)$, a range for x as ($0 \leq x \leq 5$), and a value for dx as 1.

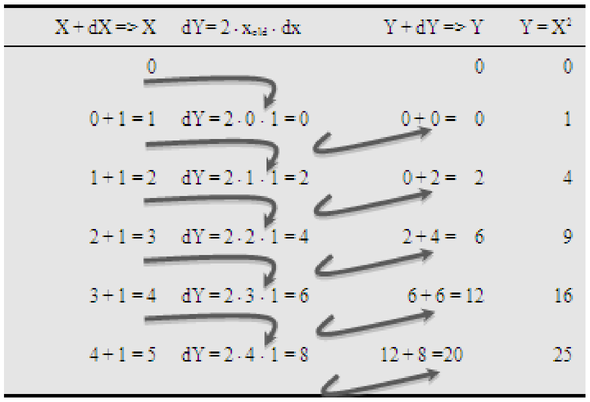


Numerical integration constitutes a major aspect of computational modeling and simulations. The mathematical method for updating and the choice for the step size (dx) affect the accuracy of computations. While tools such as IP provide a way to control these parameters, a user may not want to deviate from default methods and values. However, when doing it manually either by hand or a spreadsheet the user gets to have full control over them. In so doing, at least three important lessons are learned. For example, when results for different dx values are compared to the analytic solution ($y= x^2$) using a graph (see Fig. 4), a correlation between the step size (dx) and the accuracy of the results becomes obvious right away – i.e., *the smaller the dx, the more accurate the answer*. Secondly, the cost of accuracy become obvious as more accuracy means more data points. Furthermore, the need for automation becomes obvious when the number of data points increase. While a human can calculate a few data points by hand when dx is 1, or 0.5, for smaller dx values such as 0.1 or 0.05 one would have to move to Spreadsheets, such as Excel, to automate the calculation and to graph y= $f(x)$ curves. But for much smaller increments (dx), such as 0.0001, or 0.000001, spreadsheets would not be much help to handle and monitor such a large data set – a million steps would be needed in the above example for $dx = 0.000001$. This is where the need for computer programming becomes obvious.

Table 1. Steps to build an (X, Y) table based on initial conditions, known range of x, and chosen value for dx.

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.



X+dX=>X	dY=2·x_old·dx	Y+dY=>Y	Y=X²
0		0	0
0+1=1	dY=2·0·1=0	0+0= 0	1
1+1=2	dY=2·1·1=2	0+2= 2	4
2+1=3	dY=2·2·1=4	2+4= 6	9
3+1=4	dY=2·3·1=6	6+6=12	16
4+1=5	dY=2·4·1=8	12+8=20	25

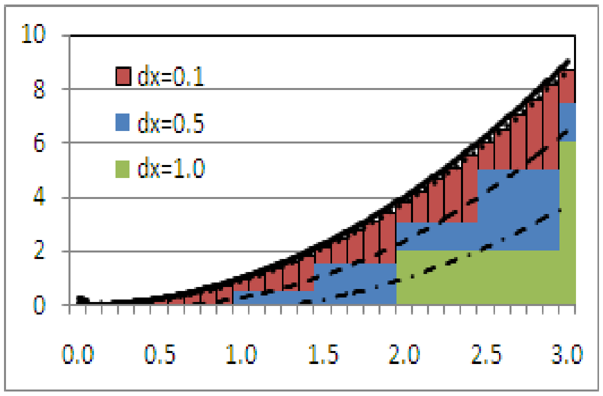


Figure 4: Numerical results (dashed lines) are compared to the analytic solution (solid line).

In an after-school project initiated by the authors, several 9th graders from Brighton High School (NY) were able to replicate Interactive Physics simulations in Fig. 3 using Excel and later with Fortran programming language to compute the position ($x_{new} = x_{old} + dx$) and velocity ($v_{new} = v_{old} + dv$) as a function of time, where $dx = v \cdot dt$ and $dv = a \cdot dt$, and $a=F/m$ *(acceleration=Force/mass)*. Since change is caused by external forces, all they needed was a formula for the forces. As a result, computation of *change* motivated them to learn basic laws of physics. For example, the force applied by a spring onto an attached object in Fig. 3 is $F = - k \cdot x$, where k is the stiffness coefficient of the spring and x is the displacement of the object from the equilibrium position. Once they learned the process and understood these basic mathematical, computational, and scientific principles, they transferred their skills and knowledge to other circumstances. To simulate planetary motion all they had to know was the formula for the interplanetary gravitational force ($F=G{\cdot}M{\cdot}m/x^2$; where G is a Universal Constant, M and m are masses of the planets separated by distance x), which governs the orbital motion of a planet around the Sun. Below is a brief statement from their project to prove Kepler's Laws in physics using multiple tools.

We had not taken any physics courses and did not know the laws of universe that govern planetary motion. We first learned how to use Interactive Physics and used it to design the solar system with realistic data about distances and masses. We then transferred visuals images from IP to another tool (Geometer's Sketchpad) to measure angles, distances, and areas of triangles needed for the proofs. We learned how to manually construct the same simulation with Excel but realized later a need for more accurate results which motivated us to learn basics of another tool, computer programming! While it was initially frustrating to learn new tools, we quickly realized the opportunity in our hands. In the end, we did not make a discovery in physics like Kepler did, but we certainly discovered that physics was not that threatening or boring. We also discovered the role of mathematics and programming in physics. The foreboding nature of complicated physics was abolished and we all look forward to classes on these subjects.

Discussion of Results

The work reported here involves: 1) 26 preservice Noyce Scholars from our secondary math and science certification programs, and 2) 188 inservice teachers from Rochester City and Brighton Central SDs. Out of the 26 preservice teachers, only 12 had taken both NAS 401/501 CMST Tools and NAS 402/502 CMST Principles and Pedagogy course at the time of data collection: we will call them PT2. The remaining 14 (PT1) had only taken NAS 401/501 CMST Tools course. Of all 188 inservice teachers who attended MSP summer training, 110 teachers attended the training once which only involved CMST tools training with some experience with developing modules and lesson plans to use those modules; we will call them IT1. 78 inservice teachers attended two summer sessions with additional training on two tools of choice and development of modules and lesson plans; we will call them IT2. 43 inservice teachers attended three summer sessions that included those above plus some knowledge of CMST principles (excluding text-based computer programming) as well as a capstone project. Data collected on preservice teachers include pre/post activity surveys, interviews and artifacts (lesson plans, modeling and simulation examples, and class assignments) because of its relative recent start. The analysis of data on inservice teachers for this study also included classroom observations and student responses to technology. Details of other data sources and their analysis for inservice teachers can be found in Yaşar *et al.* (2014). Authors used a simple descriptive statistic to analyze Likert-scaled pre/post activity surveys and an inductive approach (Creswell 2009) to analyze participants' responses to the open-ended questions and interview transcripts. The transcripts were independently analyzed and coded around common themes by two researchers and an external evaluator. Data from artifacts were used to triangulate the results.

Inservice Teachers: Exit surveys showed that usage of the tools in the classroom was directly linked to the amount of training teachers received. All trained teachers reported that on a daily base they used laptops for presentations, graphing calculators for math instruction, and electronic smart boards for interactive lessons. 60% of group IT1 teachers reported occasional use of one or more CMST tools in their classrooms. On the other hand, 50% of IT2 teachers and 78% of IT3 teachers reported regular use of one or more tools. Additional data from IT3 demonstrate their TPACK skills (what, when, and how to use a tool). While both math and science teachers took the same training and were exposed to the same tools, usage profiles and purpose of usage per subject area reflect judicious thinking: Math (6% drill, 12% demo, 18% modeling and simulation; and 64% other purposes), Science: (0% drill, 8% demo; 87% modeling and simulation; and 23% other purposes). Usage profiles also changed per grade level as shown in the table below. Another indicator that demonstrates their TPACK is that 60% stated that they occasionally search for free tools and use them at least one time to teach a topic. Once someone found a tool, it was quickly propagated to the whole group (see table, Java, GIS/GPS, Flash, Project Interactivate (PI, www.shodor.og) and PhET (Wieman 2008).

Subject/Grade	Daily	Weekly	Bi-weekly	Special Projects
Math, 7-8th	Laptop, smartboard	Power Point, PI, TI tools, GSP, Excel, Flash	AgentSheets	Interactive Physics, Stella, GIS/GPS, Java
Math 9-12th	Laptop, smartboard, TI graphing calculators	Power Point, PI	Excel, Flash	Interactive Physics, Stella, GIS/GPS, Java
Science 7-8th	Laptop, smartboard, Power Point	AgentSheets, Excel, PI	Interactive Physics, GIS/GPS, Flash, Java	Stella, GSP
Science 9-12th	Laptop, smartboard	Flash, Excel, Power Point	Interactive Physics, TI tools, GPS, Java	Stella, AgentSheets, GIS, PhET

Furthermore, 94% of IT3 teachers agreed that training made them more effective in the classroom; 87% agreed that it strengthened their pedagogical skills; 73% agreed that it strengthened their content knowledge; 100% agreed or strongly agreed that training strengthened their skills related to modeling and simulation; 86% reported that they continue to use the hardware, software and curriculum inventory made available through project in their classrooms; and 80% believed that their participation served to build their school leadership skills. As far as assessing the impact on student learning using data on achievement scores see Yaşar *et al.* (2014), but here we will briefly mention how students in different subject areas responded. 90% of teachers who did use modeling and simulation tools and smart boards agreed that it increased student engagement and made STEM concepts more comprehensible. Student reaction to modeling was found to be 97% favorable in math and 77% in science classes. Observed improvement in problem solving skills was reported by 72% of math and 31% of science teachers. 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%). Finally, students in higher grade levels found modeling and simulations more engaging in both math classes (grades 7-8: 77% vs. grades 9-12: 90%) and science classes (grades 7-8: 75% vs. grades 9-12: 85%). It was even found helpful to non-traditional (special education) learners; again the higher the grade level the higher the engagement: math classes (grades 7-8: %76 vs. grades 9-12: 100%) and science classes (grades 7-8: 75% vs. grades 9-12: 85%).

Pre-service Teachers: The data for preservice Scholars include surveys, open-ended questions, and focus group interviews. We examined their evolution from stage 1 (1st summer-NAS 401/501 CMST Tools) to stage 2 (2nd summer- NAS 402/502 CMST Principles and Pedagogy). Just like the inservice teachers, responses changed with the amount and content of training. The data for stage 1 include groups PT1 and PT2 while data for stage 2 includes only PT2. At stage 1: 65% strongly liked and 35% somehow liked learning new tools; 57% said they definitely want to learn CMST principles and pedagogy while 43% said OK; 100 % liked project-based learning; 10 % reacted unfavorably to group work (because we mixed them with interns who were not at the same par); 20% wanted more time and help for lesson development. Although there was no classroom data to show how judiciously they would choose what, when, and how to use CMST tools, their inclinations towards some tools were obvious as 50% found Interactive Physics difficult to use and understand and 100 % loved AgentSheets and Excel. While Interactive Physics is generally very intuitive and easy to use, the threatening aspect of physics overtook them (because they did not take it in high school!). This definitely changed at stage 2 as they figured out how it worked. We saw them going from not knowing much about computer simulations to wanting to learn more about mathematical modeling and computer programming. Some indicated a desire to teach computer science once they start a teaching job. Below are some common themes from each stage in the form of statements from students.

Themes emerging from interviews at the end of 1st (intro level) summer coursework: it was really beneficial to work with someone in a different content area • I feel like I had the ideal situation as a math major. I was paired with a biology student • it was nice taking a specific science subject and trying to see how you can fit mathematics into that subject but, you know, I would also like to have the reverse situation where we take mathematics and see what real life applications, what science applications we can put to this, instead of the other way around • we struggled with equations and the variables as far as how it relates to our content area but we worked to figure out how to bring it to high school level and I think that I normally would have just been like, "Okay. Well we will just do something and forget about the math for right now." So it was really beneficial to see that transition into simplicity • I will absolutely use these tools, if it is available, to model systems that cannot be recreated safely in a classroom, such as a zombie apocalypse… definitely too dangerous for a classroom, since it might end humanity. It will also engage the students and increase learning, and the "hands on" approach also dramatically increases quality of learning • I honestly had no idea how to use Excel and I feel it was really beneficial to learn it. I could use Excel with students using equations and graph the relationships and then really apply mathematics to real life.

Themes emerging from interviews at the end of 2nd (advanced) summer coursework: I think a lot of strength comes from creating a model yourself. You have to really understand a system or you understand a system by making a model. You may go in like, I kind of get this, and as you make it, you realize the rules and interdependencies; so that's an excellent chance for education • I'm a chemistry major and we don't really get a lot of exposure to computer science principles, so it seemed like a very innovative idea to be able to bring in computer science concepts, which are extremely relevant nowadays to more physical science stuff, which I have more experience with • I really liked learning computer programming with Scratch and Processing. I think it's really useful to look at parts of different models and just see the mathematics and the science behind it. I thought it was really good exposure to those things and kind of connect everything together • learning programming has been phenomenal, because I think it allows the students to see what's going on a little bit better; plus you can see what other people have done. You can look at their code and being able to do that actually made it a little easier, and you're able to see what goes on, and Scratch is actually so simple that a nine-year-old can really do it • I went home and showed my daughter Scratch. Within 5 minutes she created a program. And, that really showed me, you know that my students can do it too • There's a vast source of good programming material, raw material in the natural sciences that I really now need to explore on my own. That's a wonderful prospect that I've got all that now additional resource to tap into for good programming.

Conclusion

There are several barriers blocking full utilization of new technologies and pedagogies in teaching. Increasing emphasis on high-stakes testing directly impacts what and, more importantly how it is taught. Therefore, an effective, research-based pedagogy may not find its way into the classroom setting. Another barrier is that not all technologies, pedagogies, and content can be integrated, and teachers need long-term assistance in developing knowledge of such intricate interactions. This study focuses on a specific case of TPACK — namely computational pedagogical content knowledge (CPACK) — based on the use of CMST tools as described in this paper. The study provides evidence that integrating CMST into teaching and learning situations engages both teachers and students because of its constructivist approaches and authentic experiences that mimic how scientific inquiry and science processes are done. It also argues that CMST has a significant warrant to be considered as a knowledge domain in its own right rather than just a teaching tool. Making this kind of fundamental shift will help teachers more judiciously decide how to integrate CPACK directly into the experiences of their classroom. Future work associated with this manuscript will include collection and analysis of further data from observations of classroom instruction by teachers involved in this study.

References

1. Augustine, N. (2007). Is America Falling Off the Flat Earth? Washington, D.C.: The National Academic Press.
2. Bell, L. R and Smetana, L. K. (2008). Using Computer Simulations to Enhance Science Teaching and Learning. In Technology in the Secondary Science Classroom (Eds. Bell et al.). Washington, DC: NSTA Press.
3. Bransford, J., Brown, A. and Cocking, R. (2000). How People Learn. National Academy Press, Wash., D.C.
4. Carey, Suzan. (2000). Science Education as Conceptual Change. *Journal of Applied Deve Psych*, 21(1): 13-19.
5. Creswell, J. W. (2012). Educational Research. 4th Edition. Pearson Education, Inc.
6. Computing Curriculum (CC). 2005. ACM, AIS, and IEEE Computer Society. http://www.acm.org.

7. de Jong, T., & van Joolinger, W. R. (1998). Scientific Discovery Learning with Computer Simulations of Conceptual Domains. *Review of Educational Research*, 68(2), 179-201.
8. Donovan, S. and Bransford, J. D. (2005). How Students Learn. The National Academies Press, Wash, D.C.
9. Fincher, S. and Petre, M. (2005). Computer Science Education Research. Taylor and Francis e-Library.
10. Flick, L. and Bell, R. L. (2000). Preparing tomorrow's science teachers to use technology: guidelines for Science educators. *Contemp Issues Technol Teach Educ* 1: 39-60.
11. Hammond, L-D., Austin, K., Orcutt, S. and Rosso, J. (2001). How People Learn: Introduction to Learning Theories. Stanford University. http://web.stanford.edu/class/ed269/hplintrochapter.pdf.
12. Koehler, M., Shin, T. S., Mishra, P. (2011). How Do We Measure TPACK? Let Me Count the Ways. In R. Ronau, C. Rakes, and M. Niess (Ed.). Education Technology, Teacher Knowledge, and Classroom Impact.
13. Koehler, M. J., and Mishra, P. (2008). Introducing TPCK, in Handbook of Technological Pedagogical Content Knowledge (TPCK) for Educators, Routledge Press, New York & London.
14. Koehler, M. J., and Mishra, P. (2005). What happens when teachers design educational technology? The development of technological pedagogical content knowledge. *J Educ Computing Research*, 32(2), 131-152.
15. Loucks-Horsley, S., Stiles, K. E., Mundry, S., Love, N., Hewson. (2010). Designing professional development for teachers of science and mathematics. Third Edition, Thousand Oaks, CA: Corwin Press.
16. Maeng, J. L., Mulvey, B. K., Smetana, L. K., and Bell, R. L. (2013). Preservice Teachers' TPACK: Using Technology to Support Inquiry Instruction. *J. Science Education Technology*, DOI 10.1007/s10956-013-9434-z
17. Mishra, P., Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for integrating technology in teacher knowledge. *Teachers College Record*, 108 (6), 1017-1054.
18. MSPNET (2005). Mathematical and Computational Tools to Observe Kepler's Laws of Motion. Yaşar, P., Kashyap, S., and Roxanne, R. NSF MSPNET Library, http://hub.mspnet.org/index.cfm/14566.
19. MSPNET (2006). Limitations of the Accuracy of Numerical Integration and Simulation Technology. Yaşar, P., Kashyap, S., and Taylor, C. NSF MSPNET Library. http://hub.mspnet.org/index.cfm/14568.
20. Niess, M. (2005). Preparing teachers to teach science and mathematics with technology: Developing a technology pedagogical content knowledge. Teaching and *Teacher Education*, 21, 509-523.
21. Rutten, N., van Joolingen, R., and van der Veen. (2012). The Learning Effects of Computer Simulations in Science Education. *Computer & Education*, 58; 136-153.
22. Smetana, L. K. and Bell, R. L. (2012). Computer Simulations to Support Science Instruction and Learning: A critical review of the literature. *Int. J. Science Education*, 34 (9); 1337-1370.
23. Swanson, Charles. (2002). A Survey of Computational Science Education. *http://cssvc.ecsu.edu/krell/Computational%20Science%20Education%20Survey%20Paper.htm*
24. Wieman, C., Adams, W., Perkins, K. (2008). PhET: Simulations That Enhance Learning. *Science*, 332; 682-3.
25. Yaşar, O. (2014). A Pedagogical Approach to Teaching Computing Principles in the Context of Modeling and Simulations. *J. Computing Teachers*, Winter Issue.
26. Yaşar, O. and J. Maliekal. Computational Pedagogy. (2014a). *IEEE J. Comp. in Sci & Eng*, 16 (3), 78-88.
27. Yaşar, O. and Maliekal, J. (2014b). Computational Pedagogy Approach to STEM Teaching and Learning. In M. Searson & M. Ochoa (Eds.), *Proc of Soc for Info Tech Conf 2014* (pp. 131-139). Chesapeake, VA: AACE.
28. Yaşar, O., J. Maliekal, L. little, and P. Veronesi. (2104). An Interdisciplinary Approach to Professional Development for Math, Science, and Technology Teachers. *Journal of Comp in Math and Sci Teaching*, 33 (3).
29. Yaşar, O. (2013a). Computational Math, Science, and Technology (C-MST) Approach to General Education Courses. *J. Computational Science Education*, 4 (1), 2-10.
30. Yaşar, O. (2013b). Teaching Science through Computation. *J. Sci. Tech. and Soc.*, Vol. 1 (1), 9-18.
31. Yaşar, O., Maliekal, J., Little, L. and Jones, D. (2006a). A Computational Technology Approach to Education. *IEEE Comp. in Sci. & Eng.*, **8** (3), 2006, pp. 76-81.
32. Yaşar, O., Little, L. J., Tuzun, R., Rajasethupathy, K., Maliekal, J. and Tahar, M. (2006b). CMST: A Strategy to Improve STEM Workforce & Pedagogy to Improve STEM Education. *Lect Notes in Comp Sci*, 3992, 169-176.
33. Yaşar, O. (2004). CMST Pedagogical Approach to Math & Science Education. *Lect Notes in Comp Sci*, Vol. 3045, 807-816.
34. Yaşar, O. and Landau, R. (2003). Elements of Computational Science & Engineering Education, *SIAM Review*, 45; 787-805.
35. Yaşar, O., Rajasethupathy, K., Tuzun, R., McCoy, A. and Harkin, J. (2000). A New Perspective on Computational Science Education, *IEEE J. Comp. in Sci & Eng*, **5** (2).