CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2014; **26**:2316–2328
Published online 2 June 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3288

SPECIAL ISSUE PAPER

INSTANCES: incorporating computational scientific thinking advances into education and science courses[‡§]

Rubin Landau[1,*†], Greg Mulder[2], Raquell Holmes[3], Sofya Borinskaya[4], NamHwa Kang[5] and Cristian Bordeianu[6]

[1]*Physics Department, Oregon State University, Corvallis, OR, USA*
[2]*Physics Department, Linn-Benton Community College, Albany, OR, USA*
[3]*Improvscience, Boston, MA, USA*
[4]*Berlin Center for Cell Analysis and Modeling, UConn Health Center, Farmington, CT, USA*
[5]*Science and Math Education, Oregon State University, Corvallis, OR, USA*
[6]*Colegiul Militar Stefan cel Mare, Cimpulung Moldovenesc, Romania*

SUMMARY

The conceptual framework and initial steps taken by a project that aims to incorporate computational scientific thinking into the university-level classes taken by preservice and in-service teachers (education majors) are described. The project is called INSTANCES, an almost-acronym for incorporating computational scientific thinking advances into education and science courses, and is supported by National Science Foundation as part of their Transforming Undergraduate Education in Science, Technology, Engineering, and Mathematics program. The overall goal of the project is to provide an introduction to scientific thinking with computation. Mathematics, programming, algorithmic thinking, and computing accuracy are explicit elements of the science education curriculum, and they are included as integral elements in modules that walk teachers through various examples of computational scientific thinking. Brief descriptions of the modules and their use are presented. Copyright © 2014 John Wiley & Sons, Ltd.

Received 18 September 2013; Revised 23 March 2014; Accepted 24 March 2014

KEY WORDS: education; algorithms; measurement; documentation; experimentation; languages; theory

1. INTRODUCTION

The INSTANCES project strives to create science education materials that incorporate computation as an essential element [1]. In the words of the National Science Foundations (NSF), ‘computation [has been] accepted as the third pillar supporting innovation and discovery in science and engineering and is central to NSF’s future vision of the Cyberinfrastructure Framework for 21st Century Science

*Correspondence to: Physics Department, Oregon State University, Corvallis, 1085 Victoria Falls Drive, Redmond, OR 97756, USA.
†E-mail: rubinhlandau@gmail.com
‡Categories and Subject Descriptors.
D.1 [Programming Techniques], D.1.7 [Visual Programming], D.2.4 [Program Verification, model checking], D.3.3.2 [Python], G.1 [Numerical Analysis], G.2 [Discrete Mathematics], G.3 [Probability & Statistics], H.4.1 [Excel], I.6 [Simulation & Modeling], J [Computer Applications] J.2 [Physical Science & Engineering], J.3 [Life & Medical Sciences], K.3 [Computers & Education].
§Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author. Copyright is held by the owner/author(s). *XSEDE ’13*, Jul 22-25 2013, San Diego, CA, USA ACM 978-1-4503-2170-9/13/07. 10.1145/2484762.2484769.

and Engineering' [2]. An image of how the authors incorporate this modern approach of scientific problem solving is illustrated in Figure 1, which we use as a guide for our developments.

Although a decade ago the combination of computing, science, and applied mathematics known as computational science was rarely known beyond a few research universities, today K-12 organizations such as the Computer Science Teachers Association [3] and the National Science Teachers Association [4] recommend that secondary school classrooms teach simulation as a cornerstone of scientific inquiry.

In spite of these recommendations, and in spite of the fact that present-day science and engineering research and development deal with realistic problems of practical importance that require computation, college classes in traditional disciplines such as physics have yet to fully teach computing as part of their science [5]. In our experiences, too often it is assumed that a third party such as a computer science (CS) class will provide an understanding of how to use computers as part of the scientific problem-solving process. Consequently, and unfortunately, science and math preservice teachers (in addition to science students) are often not prepared properly for their work with computers in K-12 classrooms. For instance, when used in science teaching, computing often takes the form of using simulations in a classroom without learning to explain how the simulation works. Thus, the computer and simulation are treated as the proverbial 'black box'.

Our educational materials advocate what we call 'computational *scientific thinking*' (CST). This view assumes that science is a process requiring continual evaluation via comparison of theoretical predictions to data, visualization, and critical thought, with the computer being used in all of these steps (lower part of Figure 1). All too often, science is taught only as a collection of knowledge, without the requisite tools and process.

More explicitly, we view CST as meaning

- Using simulation, data processing, and visualizations to augment the scientific method's search for scientific truth.
- Using computation and abstractions to reveal relationships and mechanisms hidden within data.
- Appreciating how multiple disciplines are used to solve problems, and how these disciplines become more understandable when placed in the context of solving a problem.
- Being sufficiently capable and confident to examine the commands that create a computer application so as to assess, at least a general way, what it does.
- Understanding that there may be simplifying assumptions made in order to solve a problem in a mathematically 'exact' manner, whereas the 'approximate' numerical solution may require fewer assumptions and thus be a more accurate description of nature.
- Understanding how, with the proper tools, one may reveal simplicity underlying complex phenomena.

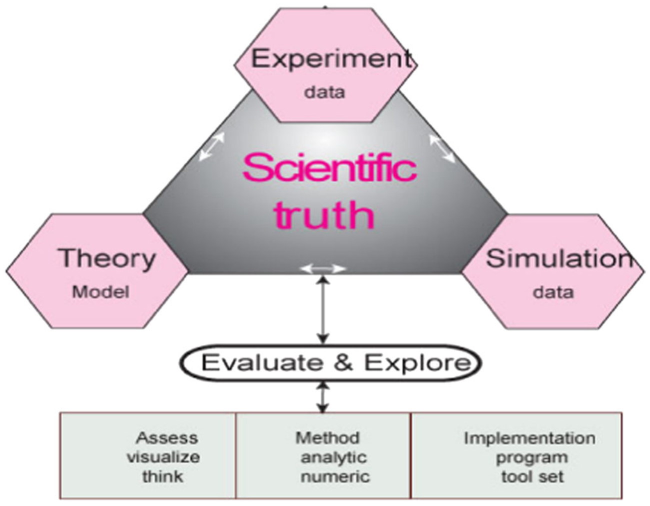


Figure 1. The computational scientific problem-solving model used as a basis for INSTANCES projects.

The materials being developed will teach computation as part of scientific problem solving, and thus provide a contextual understanding of computation as well as concrete examples to use in the teachers' classrooms. Teaching computing, math and science in the context of scientific problem solving provides a more effective way for students to learn each subject, because the context makes the relations among, and the importance of, the different disciplines clear from the start [6, 7]. It also has been shown to increase the interest and motivation in science and computing of those students who are underrepresented in these disciplines [8].

The developed materials incorporate discussion and exercises aimed at giving the teachers a basic understanding of the scientific concepts and general numerical techniques used to solve problems. For example, the fact that chance plays an important role in many scientific phenomena is an important concept, while being able to use a computer to simulate probabilistic events provides a method to simulate nature. With this understanding, simulation changes from a black box solution to an essential element in science. Our approach contrasts with approaches that try to make science more accessible by using the computer to hide the math and thereby eliminate an essential component of problem solving. We want to provide materials that give teachers the capability to explain all aspects of the simulations they use in their science classes (something often not provided by a CS class).

2. THE MODULES

We are creating learning materials that can be used in undergraduate or graduate courses that include CST. The courses are designed to be online or blended, that is, combining online and face-to-face elements, in order for in-service teachers to take them while still teaching in schools. The materials include software that operates within various computing environments (*e.g*., Python, Excel, and Vensim), lesson plans, references, instructor's guide and background materials. The materials focus more on learning how to apply computing as part of the scientific process than on mastering a programming language. Although CS topics are presented, they are presented as requisite tools for the problem on hand.

Because we are not teaching programming from scratch, we provide simple source code for simulations that can be executed and modified by students. In order to permit broad adoption and portability of our materials in the absence of any standard scientific computing environment for science, we present simulations within three different environments:

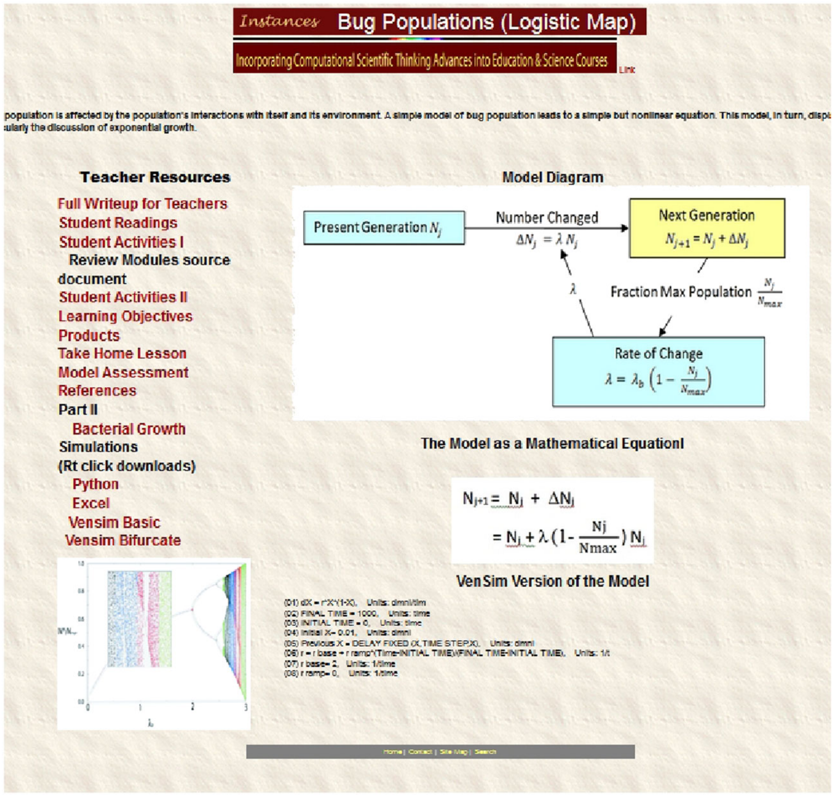
1. *Python with the Visual packages* ('*VPython*') [9]: A free, popular, and modern programming language in which executable code is obtained by processing a source code.
2. *Microsoft Excel*: A commercial spread sheet package already familiar to most students [10], and widely available.
3. *Vensim PLE*: A commercial agent-based modeling tool featuring visual programming [11]. The PLE version is available free for academic use.

We recommend the Python language as the first choice because of its simplicity, universality, scaling ability, built-in graphics, shallow learning curve, and ease of running. In addition, being able to read the source code of a program helps the user make the connection between the algorithm and the computation clearer.

After realizing how big a step it is to create or teach an entire course from scratch in one step, we have developed a collection of modules that could be used in already-existing Teacher Technology classes or be put together to form an entire class. This approach also permitted a formative evaluation in which we developed, assessed, and improved two modules and then used those modules as templates for further development.

2.1. Module format

A table of contents for all modules is maintained on the project's Web page [1]. A sample page for an individual module is given in Figure 2. Note that the left-hand side of the page contains a list of the



Instances Bug Populations (Logistic Map)

Incorporating Computational Scientific Thinking Advances into Education & Science Courses Link

population is affected by the population's interactions with itself and its environment. A simple model of bug population leads to a simple but nonlinear equation. This model, in turn, displ
ularly the discussion of exponential growth.

Teacher Resources

Full Writeup for Teachers
Student Readings
Student Activities I
Review Modules source document
Student Activities II
Learning Objectives
Products
Take Home Lesson
Model Assessment
References
Part II
Bacterial Growth
Simulations (Rt click downloads)
Python
Excel
Vensim Basic
Vensim Bifurcate

Model Diagram

Present Generation N_j
Number Changed $\Delta N_j = \lambda N_j$
Next Generation $N_{j+1} = N_j + \Delta N_j$
λ
Fraction Max Population $\frac{N_j}{N_{max}}$
Rate of Change $\lambda = \lambda_b \left(1 - \frac{N_j}{N_{max}}\right)$

The Model as a Mathematical Equation!

$$N_{j+1} = N_j + \Delta N_j$$
$$= N_j + \lambda\left(1 - \frac{N_j}{N_{max}}\right) N_j$$

VenSim Version of the Model

```
(01) dX = r*X*(1-X),    Units: dmnl/tim
(02) FINAL TIME = 1000,    Units: time
(03) INITIAL TIME = 0,    Units: time
(04) Initial X= 0.01,    Units: dmnl
(05) Previous X = DELAY FIXED (X,TIME STEP,X),    Units: dmnl
(06) r = r base + r ramp*(Time-INITIAL TIME)/(FINAL TIME-INITIAL TIME),    Units: 1/t
(07) r base= 2,    Units: 1/time
(08) r ramp= 0,    Units: 1/time
```

Home | Contact | Site Map | Search

Figure 2. A sample Web page for a module.

resources available for teachers, including the simulations as well as background materials that may, or may not, be provided to the students (instructor's choice). Aside from the source codes, the materials are all PDF files. As shown in Figure 2, there are often some sample visualizations, some of the mathematical equations used in the model, and some sample coding.

2.2. *Relation to national standards*

A full course assembled from the modules would address several Science Teaching Standards in the National Science Standards [12] and in the National Council of Teachers of Mathematics [13]. The first and foremost teaching standard in science is for teachers to 'plan an inquiry-based science program for their students' ([12], p.30). This standard can best be achieved by having teachers understand the scientific inquiry method, which, as indicated in Figure 1, is the template used for the modules. The aim is to engage students in inquiry and to develop an understanding of scientific concepts, an appreciation of 'how we know' what we know in science, an understanding of the nature of science, some skills necessary to become independent inquirers about the natural world, and the dispositions to use the skills, abilities, and attitudes associated with science ([12], pp. 104–107). The developed materials address these teaching and content standards for K-12 students by including:

- The nature of scientific inquiry as shown in the use of experiment, theory, mathematics, and simulations.
- How a natural system is viewed as the integration of data, theory, algorithms, and software.
- How simulation, visualization, data analysis, and abstraction serve the search for mechanisms, relationships, and scientific principles hidden within data.
- The meaning and importance of accuracy, precision, reproducibility, verification, and validation.
- Why a scientifically 'correct' answer may contain uncertainties and indeterminacies.
- The value of understanding how simulations work rather than viewing them as black boxes.
- How CST promotes deeper thinking.

A specific K-12 mathematics standard we address is to ‘Understand numbers, ways of representing numbers, relationships among numbers, and number systems; understand patterns, relations, and functions; use mathematical models to represent and understand quantitative relationships; use visualization, spatial reasoning, and geometric modeling to solve problems’ [12]. The developed materials address this standard by teaching:

- The basics of doing finite precision mathematical operations on a computer; division, subtraction, rate of change, differentiation, and integration.
- The experimental determination of machine precision.
- Understanding how computers can do algebra or attain ‘infinite’ precision.
- Why a mathematically ‘exact’ solution may not be as ‘correct’ as an approximate solution.
- The key problem-solving skill of decomposing a complex problem into tractable parts.

Basic CS concepts that will let the teachers look inside the black box include the following:

- Computing concepts such as logic, iteration, and abstraction.
- The basics of programming and how equations and algorithms differ.
- Trial and error searching; solvable problems without analytic solutions.

In order for teachers to be able to use these topics in a classroom (service related issues), we try to cover the following:

- How each concept relates directly to classroom teaching.
- How CST is related to practical applications such as population dynamics, lifetimes of radioactive wastes, and diffusion.
- How multiple disciplines enter into the solutions of realistic problems.

2.3. *Module topics*

The following modules are available (although not finalized):

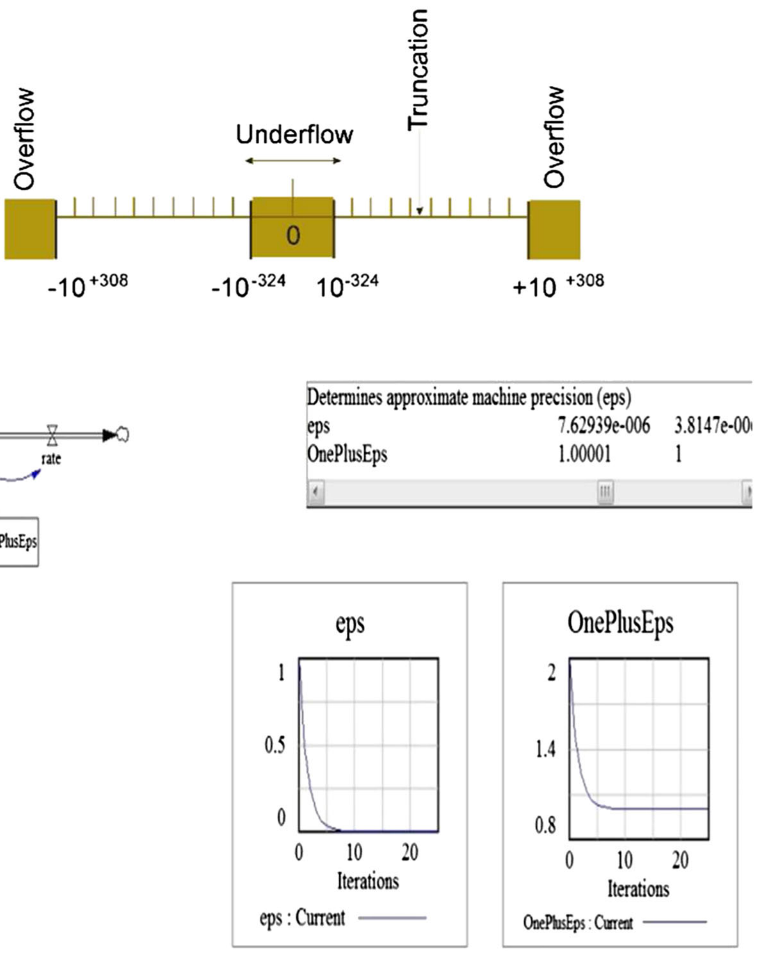
1. Computer Precision
2. Spontaneous Decay
3. Biological Growth
4. Bug Population Dynamics
5. Random Numbers
6. Random Walks
7. Stone Throwing
8. Predator-Prey Models
9. Projectiles with Drag (TBA)
10. Trial and Error search

In the sections to follow, we present samples from each of the modules (Figures 3–12).

2.3.1. Computer precision. This foundational module examines and experiments with the manner in which computers store numbers. CST concepts here include the following: (1) understanding that computers are finite and therefore have limits; (2) being cognizant that uncertainties in floating-point calculations are unavoidable; (3) understanding how it is possible to work within the limits of a computer to obtain meaningful results; and (4) developing a feel for the range of numbers that may be used to describe natural phenomena.

2.3.2. Spontaneous decay. This module aims to give students understanding of some key aspects of spontaneous decay of radioactive nuclei as well as an example of how to simulate it. The meaning of and the mathematics of exponential decay and growth are discussed. Emphasized is the meaning of ‘spontaneous’ decay and that the simulation, being a probabilistic process, is a more accurate description of nature than the traditional exponential function.

Computational scientific thinking concepts here include the following: (1) eliminating some of simplifying assumptions that ignore the probabilistic aspect of the process but are necessary to obtain an analytic solution; (2) use of the computer and its random number generator to truly



Determines approximate machine precision (eps)		
eps	7.62939e-006	3.8147e-00
OnePlusEps	1.00001	1

Figure 3. *Top*: A schematic (not to scale) representing the cause of the different types of errors that may occur in floating-point calculations. *Bottom*: A Vensim model that determines machine precision experimentally.

simulate the natural process; and (3) the use of a recursion relation that repeats until no nuclei are left in contrast to the convention direct solution in one step.

2.3.3. Biological growth. This module extends the spontaneous decay module by examining the predictions of exponential growth for populations and applies it to Wolffia plant growth. This allows us to take the same concept into two domain applications: physics and biology. Our goal is to have science teachers, whose backgrounds in scientific disciplines vary, be able to see the relationship of the mathematical models to their own or multiple areas of expertise.

2.3.4. Bug population dynamics. This module extends the discussions of growth and decay to systems that exhibit nonlinear behaviors. We aim to have students understand some key properties of nonlinear systems, discrete mappings, and displaying data, in particular, how to identify bifurcations and chaos.

Computational scientific thinking concepts include the following: (1) developing abstract representations of growth properties rather than focusing on individual members; (2) devising and exploring a model whose only solution is numerical; (3) learning that the computer can be used as an experimental lab to see what is happening in nature; and (4) discovering that simple mathematical models can explain very complicated behaviors.

2.3.5. Random numbers. This module addresses the meaning of chance and randomness and the technique that computers use to generate pseudorandom numbers. Although this may seem like just mathematics, many processes in nature and all scientific measurements contain elements of chance.

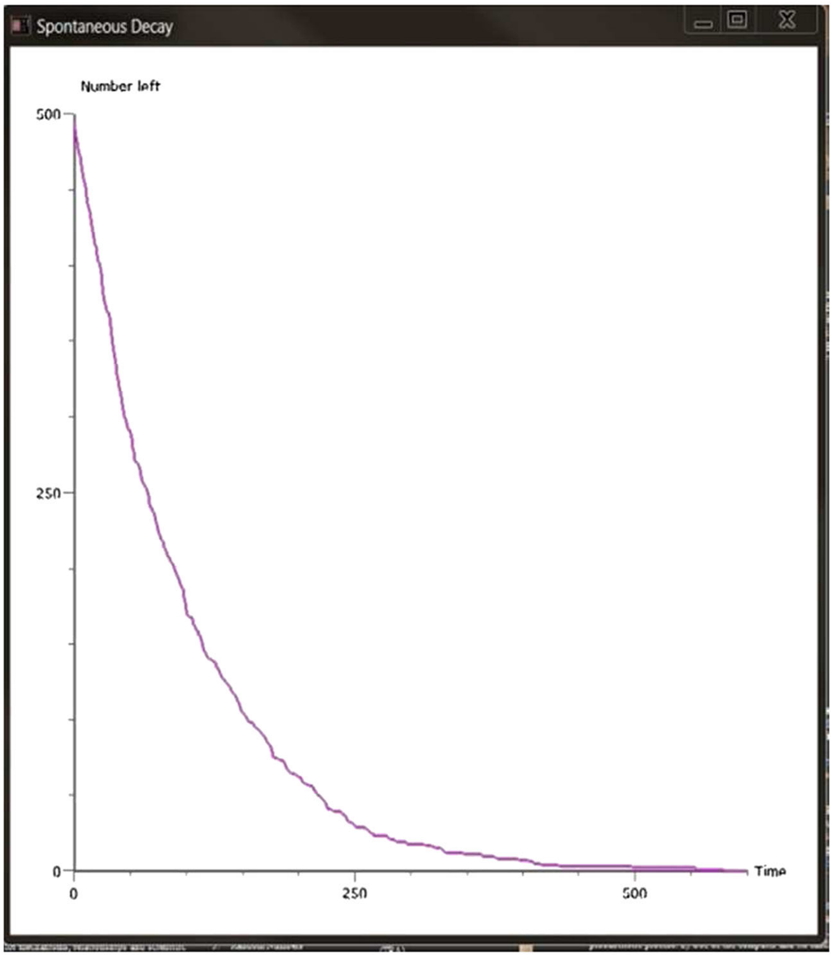


Figure 4. The output from a Python simulation of radioactive decay. Note that the decays look exponential at the small times when the number of nuclei is large, but then exhibits its inherently stochastic nature (the bumps) as the number gets smaller at larger times.

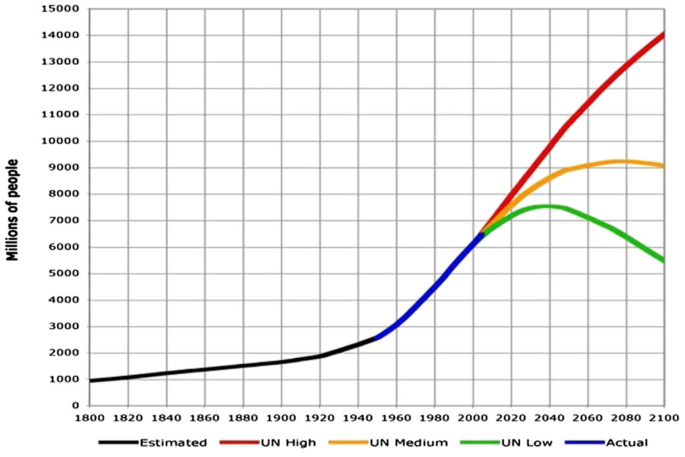
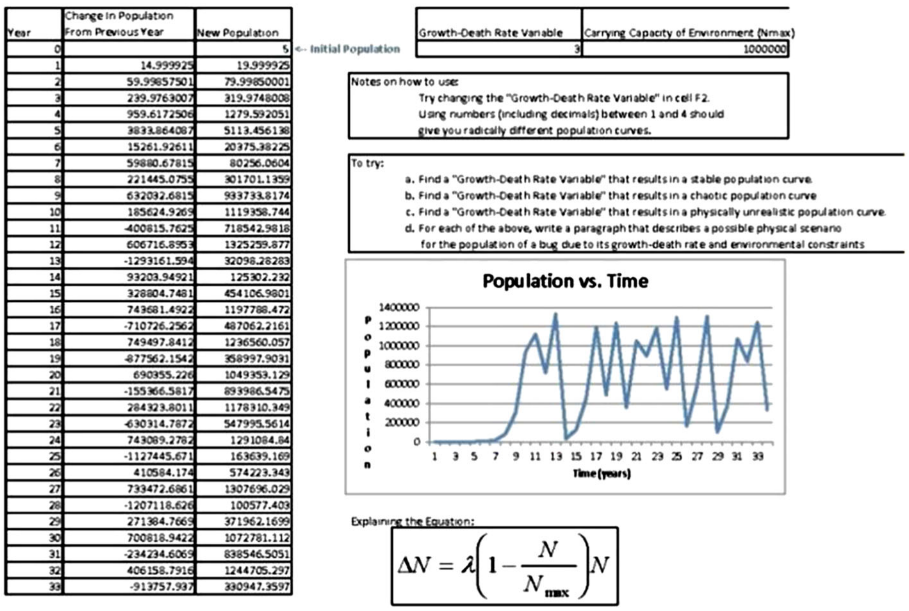


Figure 5. Three projections of the world's population growth to the year 2100. Although often called 'exponential' growth, they may not be.

CST concepts here include using a deterministic computer to generate chance, developing an intuition about what randomness looks like, and then testing for randomness.



Year	Change In Population From Previous Year	New Population
0		5
1	14.999925	19.999925
2	59.99857501	79.99850001
3	239.9763007	319.9748008
4	959.6172506	1279.592051
5	3833.864087	5113.456138
6	15261.92611	20375.38225
7	59880.67815	80256.0604
8	221445.0755	301701.1359
9	632032.6815	933733.8174
10	185624.9269	1119358.744
11	-400815.7625	718542.9818
12	606716.8953	1325259.877
13	-1293161.594	32098.28283
14	93203.94921	125302.232
15	328804.7481	454106.9801
16	743681.4922	1197788.472
17	-710726.2562	487062.2161
18	749497.8412	1236560.057
19	-877562.1542	358997.9031
20	690355.226	1049353.129
21	-155366.5817	893986.5475
22	284323.8011	1178310.349
23	-630314.7872	547995.5614
24	743089.2782	1291084.84
25	-1127445.671	163639.169
26	410584.174	574223.343
27	733472.6861	1307696.029
28	-1207118.626	100577.403
29	271384.7669	371962.1699
30	700818.9422	1072781.112
31	-234234.6069	838546.5051
32	406158.7916	1244705.297
33	-913757.937	330947.3597

<-- Initial Population

Growth-Death Rate Variable	Carrying Capacity of Environment (Nmax)
3	1000000

Notes on how to use:
Try changing the "Growth-Death Rate Variable" in cell F2.
Using numbers (including decimals) between 1 and 4 should
give you radically different population curves.

To try:
a. Find a "Growth-Death Rate Variable" that results in a stable population curve.
b. Find a "Growth-Death Rate Variable" that results in a chaotic population curve
c. Find a "Growth-Death Rate Variable" that results in a physically unrealistic population curve.
d. For each of the above, write a paragraph that describes a possible physical scenario
for the population of a bug due to its growth-death rate and environmental constraints

Explaining the Equation:

$$\boxed{\Delta N = \lambda\left(1 - \frac{N}{N_{\max}}\right)N}$$

Figure 6. The Excel worksheet for the logistic map.

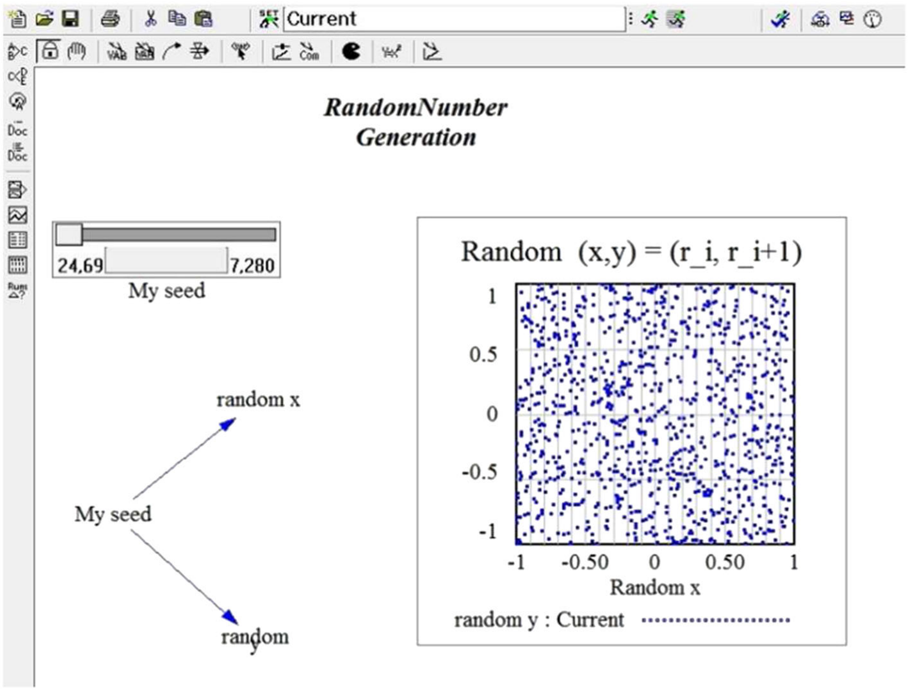


Figure 7. A Vensim simulation that produces random numbers and produces a scatter plot of random (*x*,*y*) values. Although the plot does not prove ‘randomness’, a discernible pattern or clustering in the plot would disprove randomness.

2.3.6. Random walks. This module explores how to use random numbers to simulate a random walk. Such walks are used as models for phenomena such as the path traced by a molecule as it travels in a liquid or a gas, the variations in the market price of a stock, the trail followed by a grazing animal, or the accumulation of numerical uncertainty in a many-step calculation. This is another example of how a computer can simulate nature and thereby permit it to be used as a virtual experimental laboratory.

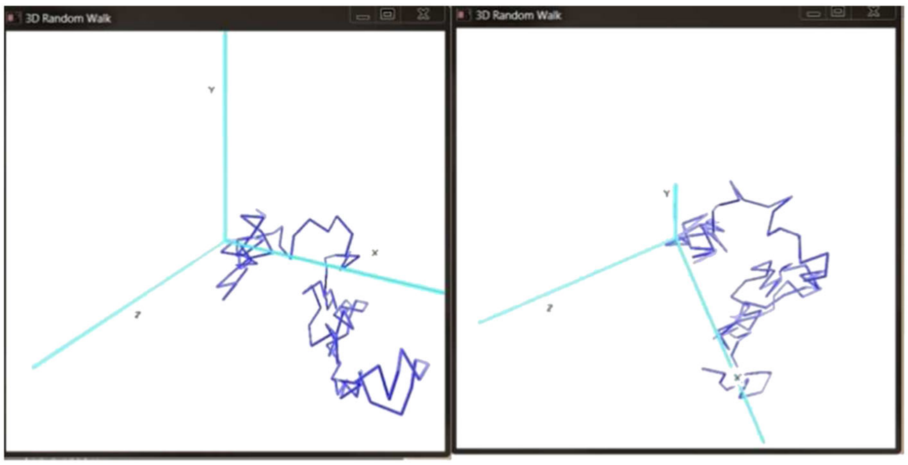


Figure 8. Views from two different viewing angles of the same three-dimensional random walk (Python output).

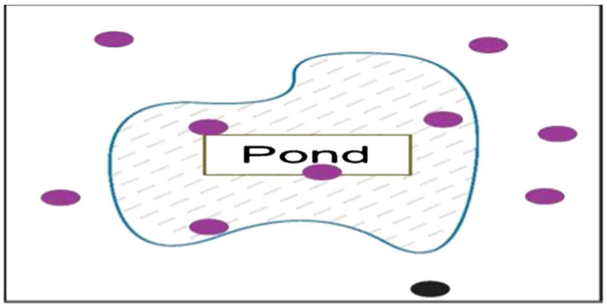


Figure 9. A pond of odd shape whose area is to be determined. A box of known area is drawn around the pond, and the ground within the box is cleared of stones. Then handfuls of stones are thrown randomly and uniformly up into the air, and the number of splashes in the pond is counted, as well as the number of stones that have landed on the ground within the box.

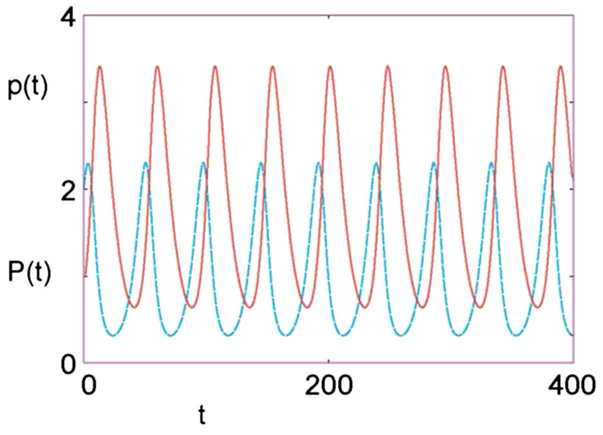


Figure 10. The time dependences of the populations of prey p(t) (solid curve) and of predator P(t) (dashed curve). Here, one population continually lags the other (Python output).

2.3.7. Integration by stone throwing. This module explores a computational approach that determines the area of an arbitrary shape (integration) by simulating a physical experiment in which stones are thrown randomly at a pond. The CST concepts here include the following: (1) using random

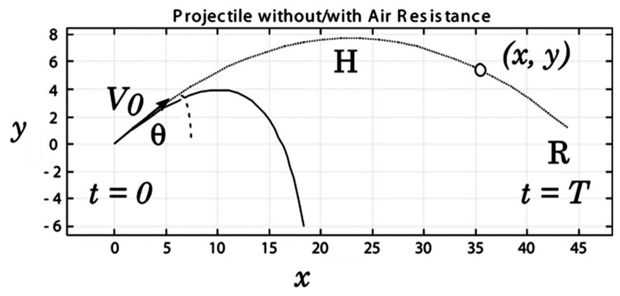


Figure 11. The trajectories of a projectile without (upper curve) and with (lower) air resistance. Drag is seen to reduce the range *R* and to produce a non-parabolic trajectory with a vertical drop at the end.

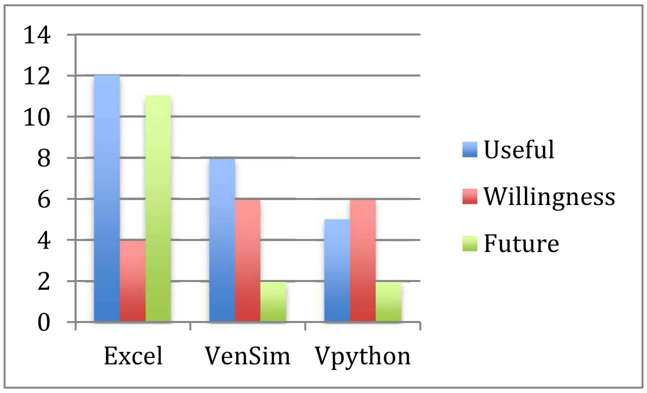


Figure 12. Preservice teacher perceptions after module use. The graph illustrates the number of people who stated that a given software environment was useful for learning computational science (‘Useful’) whether they would be willing to learn more about an application (‘Willingness’) and whether they believe that the software would be useful in future classrooms (‘Future’). All but one student had no prior experience with VPython or Vensim.

numbers to do nonstatistical mathematics like evaluating areas; (2) how using a very simple algorithm is sometimes balanced by requiring a very large amount of computer time (3) testing a new method first by using it on a problem whose answer is known; (4) how, in determining an area, the ratio of two statistical counts can predict a value for a universal constant of nature, namely π; (5) how sometimes an indirect technique is the best or only way to calculate a quantity; (6) seeing the relation between computer-generated random numbers and physical events; and (7) how experiments including chance can be conducted on a computer.

2.3.8. Predator-prey models. This module extends the module on bug population dynamics to include a second population that interacts with (‘eats’) the first population. CST concepts here include (1) new behaviors occurring when there are two species interacting with each other beyond that modeled by the logistics map and (2) that once we understand the one-variable problem, the two-variable problem becomes easier to attack.

2.3.9. Projectiles with drag. By including air resistance, this module adds realism to the analytic treatment of projectile motion traditionally studied in physics classes and shows how to solve numerically equations of motion. CST concepts here include (1) the ease with which the effect of realistic friction, which is very difficult to include analytically, can be included simply in the numerical treatment; and (2) how the more realistic treatment that includes friction resembles the actual observation of projectiles so much more than the idealized, parabolic trajectory.

3. FORMATIVE EVALUATION

We tested and assessed the *Exponential Decay and Growth* and *Bug Population Dynamics* modules [1] in a preservice teacher education class, Science and Math Ed 413, at Oregon State University. This was a week-long test and was followed by a student survey. We describe here the survey, the initial findings, and the subsequent revisions.

3.1. Course context

Science and Math Ed 413, *The Nature of Science and Science Education* is an undergraduate course for preservice teachers (PTs) that typically includes computational modeling as part of the course. Through collaboration with the instructor, Ron Gray, we were able to integrate the modules into the class and survey the students afterwards. Approximately 20 PTs were enrolled in the class, and because the course had already covered some computational modeling, the PTs could compare the INSTANCES approach to what they had already experienced.

We employed a qualitative survey with open-ended questions designed to obtain as much information as possible from the PTs. The questions, given in the succeeding text, focused on their experience using the modules in learning about the use of computational tools for solving scientific problems, and the value of the provided resources (materials, tools).

(1) In what ways do you think the computational tools we've been examining in these classes are useful to you or generally worthy of learning about? What did you know about computations in science before you had the series of classes? What are the new ideas you gained from the series of classes?
(2) Of the options presented, circle the one(s) that you want to learn about more if covered in a course.
(3) Of the options presented, which do you see as being most likely to be used in teaching science, and why do you see it as most useful?
(4) Can you tell us why you didn't choose the others in question 3?
(5) Can you think of a topic or two from your science or math area that would be appropriate for the use of computational tools in a classroom? Please explain why you think the tool(s) would work.
(6) From a science point of view, how well did the computational technology elucidate the science of the situation for
(i) Excel for exponential decay; (ii) VPython for exponential growth; and (iii) Vensim for predator-prey?

(7) Did you read the INSTANCES background material?
(7a) If the answer to (7) is 'yes', then (i) How easy was the reading? (ii) How useful was the reading? (iii) Include any suggestions you have to making the reading either easier to read or more useful.
(7b) If the answer to (7) is 'no', then please give your reason for not doing the reading. Also, please provide some feedback on what would make you do the reading.

3.1.1. Initial feedback findings. Fourteen of the PTs completed and returned the survey. Of the 14 respondents, only one reported having taken a Computer Science course and only one reported having taken a class in programming for the biological sciences. The remainder had some experience using Excel, but not Python or Vensim. Thus, the majority of PTs did not have significant experience with the type of classroom computing that the modules assume. A major issue stated by the participants was that the amount of time needed to learn a given application was too long (the instructor gave the students the entire module and not a reduced student version). It is not surprising that, of the three tools we use in the modules, the PTs' preference was for the more familiar Excel, and not new software.

Preservice teachers provided explicit feedback on ways of improving the modules. This included an increased focus on hands-on activities for the PTs and their students, a decrease in the amount of reading materials needed before beginning the modules (the students were given the full, instructor's package rather than just the student version). The PTs noted the extra work involved in learning both the scientific topics and the computational approaches. Both were new for some of the PTs. Learning how to present examples in their own classrooms was yet a further challenge.

3.1.2. Module evolution. Based on the feedback of the PTs, we modified the modules. In addition to a single linear narrative encompassing all elements of the curricula, we added sections that are tailored for different activities or audiences. Teacher materials are designed for the teacher who would take the materials to their classroom. It is a single narrative containing all elements. Exercises that may be found throughout the narrative are aggregated into activities that can be accessed independently of the reading materials. Student readings provide the basic information required to understand the scientific problem, the mathematics, and the computation. We also added some sections to help the PTs learn about the computing environments.

4. CONCLUSIONS

The main challenge we faced within the INSTANCES group was to agree upon a level of mathematics that might be accessible to many users without shortchanging the math as an essential element in CST. Our attempt to address this challenge was to provide background materials for the teachers containing the math and then leave it up to the teachers to decide what to present to their students.

Conclusions regarding the effectiveness of the developed materials for the intended audience are somewhat premature because the first class derived from the materials was not to be taught until the summer of 2013. However, this class was canceled when a co-PI scheduled to teach it unexpectedly left the country. We are still looking for someone to test our modules in an actual class and welcome contacts. In any case, we plan to incorporate them into collections of similar materials.

From the qualitative feedback provided by teachers in the first course in which we piloted the *Bugs* and *Decay* modules, we know that a focus on computation in the scientific process and content is important to the students. The main challenges faced by the preservice teachers were (1) the need to balance the amount of information conveyed in reading materials with hands-on investigations and (2) dealing with the disparity of computing experience that both teachers and students bring to class.

Clearly, we cannot just give examples in Python and Vensim without preparing the students in these languages. The inclusion of Excel examples in the materials is important in order not to exclude those students who are uncomfortable with Python or Vensim, yet an effort should be made to go beyond Excel in the actual practice of science. Our hope is that those students not experienced with Python and Vensim may yet use these tools as the course progresses. In any case, the exact tools used are not as important as having teachers learn how to incorporate mathematical modeling and computing concepts in their science classes, and thereby answer the NSF's challenge to teach science as it is currently being performed: a unity of theory, experimentation, and simulation.

ACKNOWLEDGEMENTS

This work has been supported predominantly by the National Science Foundation TUES Award 1043298-DUE as part of their Transforming Undergraduate Education in Science, Technology, Engineering, and Mathematics program. We thank them.

REFERENCES

1. Incorporating computational scientific thinking advances into education & science courses. Web page, (Available from: http://science.oregonstate.edu/INSTANCES/) [Accessed on May 21, 2014].
2. NSF 10-015, dear colleague letter: *cyberinfrastructure framework for 21st century science and engineering* (*CF21*). (Available from: http://www.nsf.gov/pubs/2010/nsf10015/nsf10015.jsp) [Accessed on May 21, 2014].
3. Phillips P. Computer Science Teachers Association (CSTA), Association for Computing Machinery, "Highlighted Resources", *Computational Thinking*: *A Problem-Solving Tool for Every Classroom*, 2008. (Available from: http://csta.acm.org/Resources/sub/HighlightedResources.html) [Accessed on May 21, 2014].
4. Bell LR, Gess-Newsome J, Luft J. Technology in the Secondary Science Classroom, Washington, DC: NSTA Press, 2008. Free electronic copies are available to the members of the National Science Teachers Association at http://www.nsta.org [Accessed on May 21, 2014].
5. Graduate education in physics, A Conference to Discuss the Status and Future of Graduate Education, American Center for Physics, College park, MD, (Available from: http://www.aps.org/programs/education/graduate/upload/2008-APS-Graduate-Education-Conference-Report_v0213.pdf) [Accessed on May 21, 2014].

6. Yasar O. Computational science education: standards, learning outcomes and assessment, Computational Science-ICCS 2001, Int. Conf., V.N. Alexandrov et al., eds., Lecture Notes in Comput. Sci 2073, Springer-Verlag, Berlin, 2001; 1159–1169.
7. Yasar O, Landau R. Elements of computational science education. *SIAM Review* 2003; **45**:787–805. (Available from: http://www.physics.oregonstate.edu/~rubin/) [Accessed on May 21, 2014].
8. aLearning through evaluation, adaptation and dissemination (LEAD) center, University of Wisconsin, (Available from: http://www.ntlf.com/html/lib/suppmat/72lead.htm); bCuny J, Aspray W. Recruitment and retention of women graduate students in computer science and engineering, Computing Research Association's Committee on the Status of Women in Computing Research, 2001. (Available from: http://www.cra.org/Activities/craw/projects/best_practices.php; http://www.cra.org/reports/r&rwomen.pdf); cRosser SV. Teaching the Majority. Teachers College Press: New York, NY, 1995.
9. VPython (downloads include packages). (Available from: http://www.vpython.org/) [Accessed on May 21, 2014].
10. Microsoft Excel. (Available from: http://office.microsoft.com/en-us/excel/) [Accessed on May 21, 2014].
11. Vensim. (Available from: http://vensim.com/) [Accessed on May 21, 2014].
12. Principles and Standards for School Mathematics, National Council of Teachers of Mathematics, 2000. (Available from: http://www.nctm.org/standards/content.aspx?id=16909) [Accessed on May 21, 2014].
13. National Science Education Standards, NRC, National Academy Press, 1996. (Available from: http://www.nap.edu/openbook.php?record_id=4962) [Accessed on May 21, 2014].