# Putting computation on a par with experiments and theory in the undergraduate physics curriculum

Ruxandra M. Serbanescu, Paul J. Kushner, and Sabine Stanley
*Department of Physics, University of Toronto, Canada M5S 1A7*

(Received 31 January 2011; accepted 3 May 2011)

Computation is an indispensable tool for physicists, but incorporating computational physics into the undergraduate physics curriculum presents many challenges. How can instructors fit computational physics into already packed curricula, and how can some level of computational proficiency become second nature to today's physics students? We review how web-based learning resources have been used to integrate computational activities into redesigned lab and lecture courses at the University of Toronto. The methodology of incorporating computation into lecture and laboratory courses at the first and second years is presented. 
[DOI: 10.1119/1.3593296]

## I. INTRODUCTION

Methods of computational physics are implemented in many North American undergraduate physics curricula, motivated in part by a historical AIP survey,[1,2] which showed that physics majors are unprepared for the workplace because of a lack of basic computational skills. Beyond preparing students for the workforce, using computational methods in lecture courses has a number of pedagogical advantages: it develops analytical skills,[3] facilitates learning through visualization of systems and phenomena,[4] and enhances physics education.[3] Integrating computational methods into undergraduate physics is challenging.[5–8] At some universities, crowded curricula make the introduction of a new computational course for credit nearly impossible.[9] How can computational physics be implemented in a way that improves physics education and the undergraduate experience?

At the University of Toronto, we have integrated a significant computational physics component into our lecture- and laboratory-based physics courses at the first and second year levels. The effort is part of an across the board renewal of our undergraduate curriculum. The overall goal of this effort, involves what we call "Practical Physics," which aims to bring the undergraduate physics student closer to the everyday experience of professionals in the physical sciences.

Our students typically have little or no computational background and need to efficiently acquire computational skills for the required laboratories and lecture activities. We take advantage of open source, high level, easy to use programming tools, and the ubiquity of personal computers available for independent study. We describe how we motivated computation in the first and second year physics courses in Sec. II, discuss the details of the methodology in Sec. III, and the incorporation of computation in the lecture and lab courses in Secs. IV and V. We assess the impact of the computational curriculum in Sec. VI. Section VII concludes with a few cautionary notes and future plans. We welcome all members of the physics teaching community to try our tools and provide feedback on our efforts.

## II. MAKING COMPUTATION RELEVANT TO PHYSICS STUDENTS

Students are usually hampered by their lack of mathematical tools in introductory physics courses. The examples must be simple enough so that students can solve them analytically. Students obtain a numerical answer, but with little intuition about the physical processes involved. Recognizing these obstacles to learning became an initial driver for introducing computational activities at the introductory level.

There is no consensus about what computer language to use in computational physics. We decided to adopt PYTHON[10] and its modules such as VPYTHON, which are free, simple to use, and rich in functionality and scientific applications. VPYTHON is mostly used in the first-year physics courses for simulations and visualizations of physical phenomena.[4,11] PYTHON functionality is introduced in second-year classes for more advanced numerical work.[7,12]

For the 2008 academic year, we started a pilot program to introduce computational examples and activities in our first-year physics lecture and lab course in our specialist program. (The specialist program is equivalent to a U.S. honors program.) Through an elementary introduction to the numerical integration of dynamical equations, we eliminated some of the mathematical hurdles, which opened the door to studying more complicated and more interesting examples. We used computational examples as a teaching tool in lecture and outside of lectures with on-screen visualizations of the systems of interest. To see the benefits of this activity, the first exposure would need to be followed by a more intensive incorporation of computation at the second-year level.

The main part of the computational curriculum was introduced in the 2009–2010 academic year. We completely redesigned our second-year laboratory course, Practical Physics. The course begins with a module to build computational skills for the rest of the year. With its emphasis on independent and hands-on learning, this laboratory course provides an excellent focus for our computational curriculum. In our design, we were mindful of the emphasis physics education research places on model-based advancement of learning.[11,13] Our programming-based teaching aimed at linking real objects (processes and events) that occur in nature and are observed in experimental settings to theoretical objects[14] and at building new concepts by modeling.[15] Because all students in the specialist program are required to take Practical Physics starting in the first semester of their second year, this course provides a foundation for our core physics students.

We brought computational approaches to lab experiments by teaching numerical methods and applications to data analysis. We emphasized the applicability of computational tools in a wide class of problems employing different scientific techniques (see Sec. IV).

  

We coordinated the Practical Physics computational curriculum with our second-year lecture course on classical mechanics. The links between lab and lecture courses advance the development and organization of students' thinking about physical systems and phenomena. In 2010–2011, we are implementing further activities in other second-year courses, and our long-term goal is to have computational physics naturally integrated throughout the undergraduate curriculum.

## III. DEVELOPMENT AND METHODOLOGY

A significant challenge in the second-year lecture courses is that many students come from outside the specialist program and are not required to take the Practical Physics course. To provide all students an appropriate foundation in computation for the lecture courses, we have developed a web-based learning resource. The wiki consists of an extensive four part PYTHON tutorial with many exercises and examples, a reference guide with useful commands and concepts, a section on using VPYTHON for the first-year course, and sections on some of the many PYTHON modules to be used in second-year and upper-level courses.[16] The aim of the tutorial is to teach programming concepts and skills using mostly physics motivated examples.

The tutorial uses the University of Toronto Physics PYTHON Distribution, which provides a local version of freely available PYTHON packages. The distribution includes the VISUAL, NUMPY, SCIPY, and MATPLOTLIB modules of PYTHON as well as IDLE, an integrated development environment. The distribution is meant for home or departmental computer installation on a variety of operating systems.

The wiki aims to get students comfortable using computers to model physical systems and for working with data in the laboratory, rather than to teach expert programming skills. Our expectations are that students who work through the tutorials will be able to independently write simple PYTHON scripts, test and debug them, and document them. Understanding how to create efficient, well constructed, and reusable code is a long-term goal that requires additional time and coursework. The tutorials provide the basic skills in the first semester. The four part tutorial typically involves 20–30 h of independent study over 2 weeks. The tutorials are not marked for credit but are supplemented by in-class problems in the lecture and Practical Physics courses. Mastery of the material is demonstrated by its application to laboratory and lecture coursework.

Another key part of the computational initiative is to enhance the students' ability to acquire data on a computer for subsequent analysis. For this purpose, we designed and produced a data acquisition device[17] (see Fig. 1) with digital and analog inputs/outputs, USB connectivity, and $1.2 \times 10^6$ samples/s capture rate. Its core is a National Instruments USB board attached to a in-house designed connector board. Computer-based lab data acquisition consisted of experimental setups interfaced with computers through data acquisition devices. The sensors (rotational, light, and Geiger) were purchased from PASCO.

## IV. COMPUTATION IN PRACTICAL PHYSICS

Computation was first implemented into the Practical Physics class in the 2009/2010 academic year. The class consisted of 96 students in two sections with 6 lab hours/week over 12 weeks. Student backgrounds were heterogeneous: 40% came from outside the physics major. Their computational experience was limited: 75% had not taken a computer science course. To provide the required background, the computational component of the syllabus included 1.5 weeks of wiki-based PYTHON instruction, 6 weeks of guided work on computation/lab problems, and 4.5 weeks of experimental work comprising 4–5 independent experiments. We set up a PYTHON Clinic consisting of a help email address and 4 h/week of direct help provided by two advanced undergraduates highly proficient in PYTHON.

The computational/lab problems were based on simple experimental setups: four problems used a simple pendulum, one was based on modeling transient voltages and one on modeling radioactive decays. Guide sheets provided some insight, but left the main task to be discovered by students. We emphasized code reuse between labs: new applications were based on previous ones. Peer instruction was an integral part of the computational learning environment and students worked in pairs.[18] There were four teaching assistants per section of 48 students, canvassing the room while answering questions. Student teams were free to engage in discussions and exchange ideas with other groups. Laboratory work and PYTHON code was submitted weekly as one report per group. Equal credit was given to students for the writeup, but the overall marks for individual students were also partially determined by an evaluation from the supervising teaching assistant.

### A. Sensitivity of modeling to numerical method

A typical problem was to measure and model a simple pendulum undergoing oscillations at various angles with damping. Students modeled the data from a small angle pendulum, comparing the forward Euler and Euler-Cromer[19] methods of numerical integration. The forward Euler algorithm updates the angle using derivative information from only the beginning of the interval $\Delta t$. The numerical solution has a spiral orbit in $(\theta, \omega)$ phase space. Students were given just the beginning of the code and asked to discuss the angle, phase, and energy plots. The failure of the algorithm is obvious, but only a few students were able to relate the method's breakdown to violation of energy conservation. The


Fig. 1. (Color online) Data acquisition device used (details available on request).

symplectic Euler-Cromer algorithm combines the forward and backward Euler methods into a stable numerical integration tool. The problem emphasized that any numerical algorithm has to be critically examined before using it in a conservative system. In Fig. 2, we present the plots for the amplitude using the forward Euler and Euler-Cromer algorithms.

### B. Using PYTHON packages to fit data

A series of experimental/computational problems focused on the statistics of data analysis. The problems were preceded by lab lectures on linear regression methods.[20] PYTHON and SCIPY packages are very rich in scientific modules. One of them is `leastsq`, which uses the Levenberg-Marquardt algorithm.[21] This module was used by students to build a data fitting program with statistical output. The experimental setup involved a large angle pendulum with damping. The data set (10,000 data points, measured over 20 min of damped oscillations) was retrieved from the data acquisition device. Students were given the framework of the fitting program and taught how to output $\chi^2$ and the convergence test. In a special lab lecture, covariance was introduced, and the leastsq input and output arguments were analyzed. Students were taught how to use the covariance matrix elements to output the errors in the calculated parameters. This problem aimed to help students understand the output of a typical data fitting program including $\chi^2$, degrees of freedom, root mean squares of residuals, reduced $\chi^2$, and fitted parameters that minimized the residuals (within 68% confidence interval).

In the radioactive decay study, students retrieved data from a Ba-Cs isotope generator using the data acquisition device. They were asked to analyze the decay and provide the best model by choosing between a simple and a double exponential, based on the statistical output of the fitting program.

The low, random count from a red Fiesta dinner plate was also analyzed. Fiesta plates are known for their safe but easily detected levels of uranium oxide in the glaze.[22,23] The model used in our problem is based on the histogram module and the normal distribution function from SCIPY. Students wrote the program and used the normal distribution to do a qualitative fit to the histogram profile. Figure 3 shows the Fiesta plate analysis output. Extra credit was given to the small number of students who were able to fit the histogram with the Poisson probability distribution function.

### C. Using programming as an everyday lab tool in independent lab experiments

In the last 5 weeks of the Practical Physics course, students were required to complete 4–5 experiments from a group of 14. Each of these experiments included a programming task, using either numerical integration or data analysis with statistical output. For example, Fig. 4 shows a result from a polarization of light experiment in which students study polarization by transmission or reflection. The light intensity $I_p$ acquired from the data acquisition device is plotted with MATPLOTLIB as a function of the angle between the polarizers and compared with Malus's law[24] given by $I_p = I_0 \cos^2 \theta$.

While doing their chosen experiments, students became comfortable with programming. They often performed a quick data fit, examined the statistical output, and eventually went back to redo parts of the experiment.

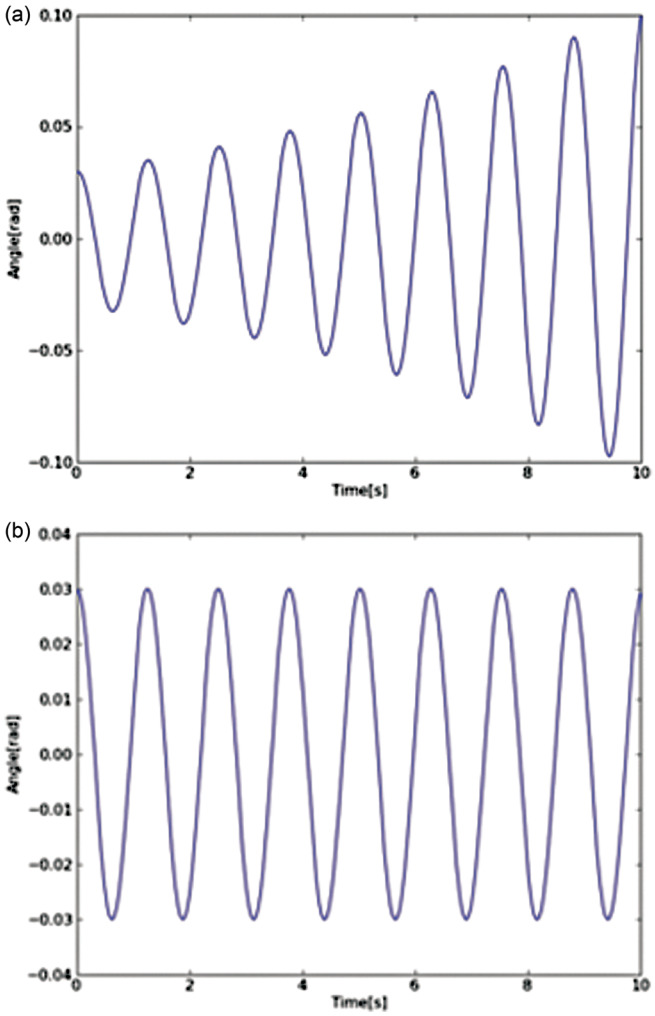


Fig. 2. (Color online) Methods of numerical integration. Pendulum amplitude plots: (a) Forward Euler method, and (b) symplectic method.

## V. COMPUTATION IN LECTURE COURSES

### A. First-year course

The first-year physics honors course includes a 12 week unit on mechanics. This course is the one in which we introduce computation in a way that assumes no prior computational background. Students were given a series of computational problems using VPYTHON in weekly tutorials. These problems involved a physical concept related to the material being studied concurrently in the lectures. Example problems involved free fall, collisions, rigid body rotation, the physics of drag, conservation of energy, and angular momentum conservation.

Students used simple codes to model systems, visualized the resulting motion, and produced plots to quantify the solutions. They worked in groups of 3–4 and then individually handed in follow-up questions to the tutorial problem with their problem sets.

*Example: Visualizing the physics of drag*. Students wrote simple programs to compare the motion of balls in free fall, one-dimensional motion under gravity and air drag, projectile motion, and two-dimensional motion under gravity and air drag. Numerical integration problems were based on using Newton's laws to determine the acceleration of an

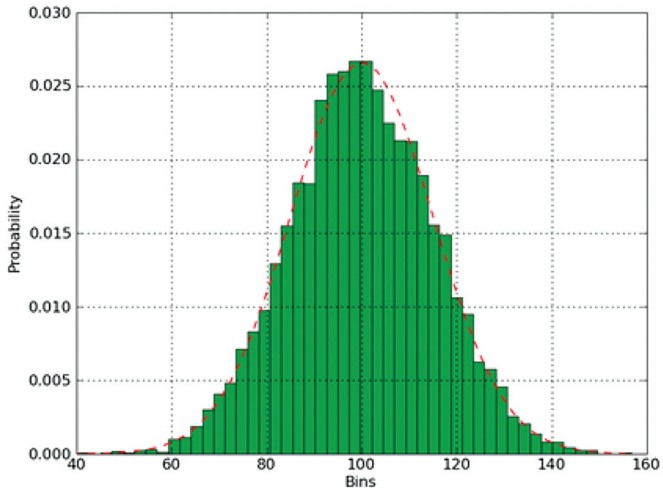


Fig. 3. (Color online) Fiesta plate radioactive count modeled by a histogram. The dotted line represents the normal distribution function.

object, integrating the acceleration to determine the velocity, and using another integration to determine the position. Numerical integration was kept as simple as possible and made to look similar to analytic formulas.

Students were also taught the importance of a small time step size $\Delta t$ and demonstrated that the numerical integration can give incorrect results if $\Delta t$ is too large. This problem anticipates some of what they learned in Practical Physics. The most difficult initial concept was the difference between analytical and numerical solutions. For example, in the free-fall case, we asked students to use VPYTHON to plot the analytical and the numerical solutions. Many students initially did not understand the difference between analytic and numerical integration. We stressed this concept throughout and demonstrated many problems without analytical solutions.

### B. Second-year lecture course

Starting in the 2009/2010 academic year, computational physics was implemented through the second-year lecture course on classical mechanics. It is featured in lectures, problem sets, a midterm test, and a dedicated computational assignment. In 2009, about a third of the students in the class were simultaneously enrolled in Practical Physics. These students benefited from having their learning reinforced from the two courses (see Sec. VI). Problem sets included computational and plotting problems that required learning about half the material in the wiki tutorials over a 2 week period.

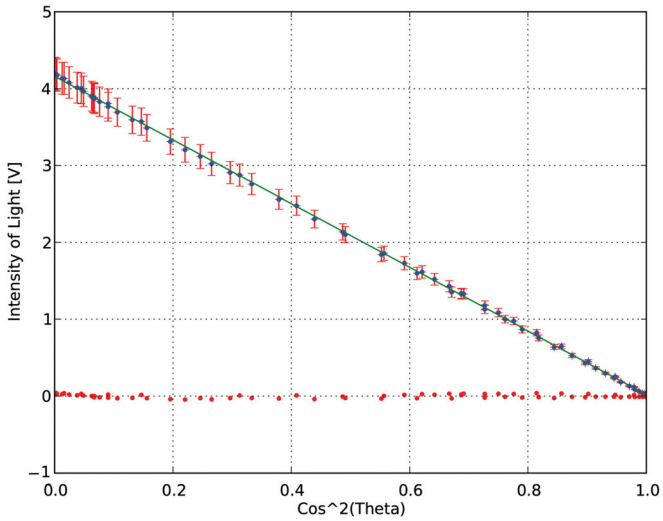


Fig. 4. (Color online) Polarization of light experiment. Data fitting with SCIPY. Baseline dots are residuals.

As an example, students compared particle motion in a harmonic potential $V(x) = kx^2/2$ to motion in a Morse potential with $V(x) = a\,[1 - \exp\,(-(x-b)/c)]^2$. Students were asked to find the stable equilibria of these potentials. They then used numerical integration with NUMPY and visualization with VPYTHON to show that the period in the harmonic potential is amplitude independent and the period in the Morse potential is amplitude dependent if the amplitude of the motion becomes sufficiently large.

The computational assignment explored nonlinear and chaotic dynamics of a damped driven pendulum[25] and a constrained spring system (see Fig. 5).

Students explored the period doubling route to chaos and the sensitive dependence on initial conditions, using tools such as phase-space plotting and Poincaré sections. The course emphasized the power of computers in demonstrating dynamics through visualization and in exploring apparently simple systems with complex behavior.

## VI. EVALUATING THE COMPUTATIONAL CURRICULUM

We analyzed student reactions to the new curriculum using surveys conducted in 2009 and 2010 and began testing our hypothesis that integrated computational physics learning across labs and lecture courses improves both.

The 2009/2010 Practical Physics class was surveyed halfway through the semester: the 22 question survey was voluntary and anonymous. Sixty-eight surveys out of 96 students enrolled were completed; answers were given on 1 to 5 scales. The survey was repeated in 2010/2011 (65 surveys completed out of enrollment of 115 were processed). The main survey results are presented in Table I.

The Practical Physics course filled an instructional gap by connecting several pedagogical tools: we taught experimentation, programming, modeling, and data fitting. As one student put it: "I took this course… because I wanted to do something other than just reading textbooks and listening to lectures. So far it has been fun, although quite difficult." In many other comments, students acknowledged the challenge: "It has been an incredible stressful 3 h, but I liked the integration of PYTHON/PYLAB into the experimental part of this course."

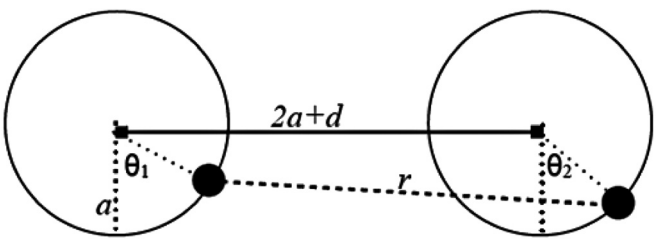


Fig. 5. System of two beads constrained to move in a horizontal plane on circular wires and joined by a spring with a restoring tension of magnitude $k|r - (2a + d)|$. This system has two degree of freedom and is strongly nonlinear in the angles $\theta_1$ and $\theta_2$. This system exhibits chaotic behavior and is impossible to treat analytically but can be readily explored on the computer.

Students also appreciated the usefulness of learning programming to other lecture courses. They often mentioned the second-year Classical Mechanics course, which used the same computational tools as Practical Physics. Table II presents the mean grades from Practical Physics and Classical Mechanics and the mean grades of the group who took both courses simultaneously. Although there is a difference in the overall grade of this group compared to the overall grade of each of the courses, we need to find better quantitative tools to test our hypothesis that integrated learning improves the learning of physics.

We paid special attention to the feedback given by students who were not enrolled in the physics honors or physics major programs. This group consisted of junior and senior students who acknowledged more difficulties than the majority of the Practical Physics class.

We expected peer instruction to work much better than it did in Practical Physics. We were surprised at students' requests to re-arrange about a third of the initial partnerships in the first 2 weeks of classes. Most requests came from groups with a large disparity in mathematical background.[26,27] From our observations, we understood that peer instruction appeared most beneficial for students with poor initial backgrounds.

The excitement and class enthusiasm gradually increased during the Practical Physics course. According to a student, "I was excited to learn programming and I'm glad I did it. There was a steep learning curve, but it was very useful and interesting. I expected bad marks and boring."

Practical Physics is largely based on online learning resources. We were interested in which online resource was mostly used by our students. We were not surprised to find that Wikipedia access was more popular than our own wiki. The last question of the survey asked: "Did you look forward to coming to the lab?". Students could answer: yes, no, not sure, or sometimes. In 2009 and 2010, only 21% of the answers have been no.

Table I. Practical Physics survey; $\sigma$ represents the standard deviation. Scale A: 1 = very poor, 2 = poor, 3 = neutral, 4 = good, 5 = very good; scale B: 1 = very difficult, 2 = difficult, 3 = average, 4 = easy, 5 = very easy; scale C: 1 = it did not work for me at all (it prevented me from learning all the time), 2 = it did not work, 3 = neutral, 3 = neutral, 4 = it worked well 5 = it worked very well.

| Survey Questions | 2009 Mean ($\sigma$) | 2010 Mean ($\sigma$) |
|---|---|---|
| Usefulness of computational problems: rate the practical experience acquired, connection to other physics courses (Scale A) | 3.80 (0.81) | 3.90 (0.70) |
| Computational exercises overall rating (Scale A) | 3.85 (0.78) | 3.96 (0.89) |
| Difficulty of learning PYTHON in this course format (scale B) | 3.25 (0.98) | 3.08 (1.17) |
| It is known that peer instruction enhances physics learning. We are not sure about is efficacy in learning programming in a physics lab context. Please give a rating of how peer instruction worked for you in this course (scale C) | 3.85 (0.89) | 4.06 (0.98) |

Table II. Grade correlation between the Practical Physics and Classical Mechanics courses (data from 2009 and 2010).

| Course | Entire class Mean ($\sigma$) | Cross-listed students only Mean ($\sigma$) |
|---|---|---|
| Practical Physics | 70 (9.3) | 83 (6.9) |
| Classical Mechanics | 66 (8.9) | 70 (8.9) |

## VII. DISCUSSION AND CONCLUSION

During the first phase of introducing computation into the undergraduate physics curriculum, our goal has been to facilitate a deeper understanding of physical phenomena using simple programming problems, showing students the usefulness of computational skills.

In the first-year course, students appreciated the three-dimensional visualization capabilities of VPYTHON. They developed intuition about the physical concept by seeing physical systems evolve along with quantitative output of the motion. Some students also appreciated the chance to develop skills that would help them in future courses and research. This appreciation was not unanimous, and feedback at the end of the course showed that there are still students who would prefer to investigate problems only analytically.

By learning how to use some of the PYTHON scientific modules, our students finish the second year with a deeper understanding of lab tools such as data and error analysis and numerical integration. At the end of Practical Physics, they can do a data analysis with statistical output on any lab experiment. They do so with tools that are independent of commercial programs and with techniques that can be easily applied to other commercial and non-commercial software. The hands-on experience of using high-level numerical packages was particularly beneficial. It was remarkable to overhear students' informal discussions of error covariance statistics in lab results a mere 2 months after being introduced to computational methods and computer-assisted error analysis.

Computational physics has been shown to be a useful tool in enhancing the learning of physics concepts.[3,28] However, we observed mixed results from peer instruction in Practical Physics, contrary to observations from computer science courses.[29,30] Further, studies on this topic are intended.

Our efforts comprised many components: setting the new Practical Physics course, creating a wiki, putting together our PYTHON distribution package and enhancing existing documentation by writing tutorials, homework, and lecture examples in PYTHON. Critical to our efforts was a collegial help team that worked well together, shared issues and problems encountered, many creative ideas, and provided our students the help and support they required.

In the next stage of our project, we shall implement PYTHON packages (tutorial problems, computational assignments, and lecture demonstrations) in two additional second-year lecture courses. We shall also encourage the dissemination of computational methods (not necessarily limited to PYTHON) throughout the upper division curriculum. In addition, we have begun introducing the same software into our graduate curriculum and graduate research activities. Pedagogical issues related to computational physics will be the object of another study. We invite readers to try out our tools.[31]

## ACKNOWLEDGMENTS

The authors thank the anonymous referees whose comments improved this manuscript and our ideas about computational physics education, Ms. April Seeley for processing the midterm survey in Practical Physics and The University of Toronto Team: David Bailey, Charles Dyer, David Harrison, Stephen Morris, Larry Avramidis, Phil Scolieri, Sergei Sagatov, Andrew Martin. They also thank Prof. Roger Fearick, University of Cape Town, South Africa, for exchanging ideas about numerical integration problems and Prof. Bruce Sherwood, University of North Carolina, for giving us access to the compPADRE content in 2007. This project received funding from the University of Toronto Curriculum Development Fund.

[1]N. Chonacky and D. Winch, "Integrating computation into the undergraduate curriculum: A vision and guidelines for future developments," Am. J. Phys. **76**(4/5), 327–333 (2008).
[2]R. G. Fuller, "Numerical computations in U.S. undergraduate physics courses," Comput. Sci. Eng. **8**(5), 16–21 (2006).
[3]R. H. Landau, "Computational physics education: why, what and how," Comput. Phys. Commun. **177**, 191–196 (2007).
[4]R. Chabay and B. Sherwood, "Computational physics in the introductory calculus-based course," Am. J. Phys. **76**(4/5), 307–313 (2008).
[5]H. Gould and J. Tobochnik, "Integrating computation into the physics curriculum," in *ICCS 2001*, edited by V. N. Alexandrov et al (Springer-Verlag, Berlin/Heidelberg, 2001).
[6]K. R. Ross, "An incremental approach to computational physics education," Comput. Sci. Eng. **8**(5), 44–50 (2006).
[7]M. Johnston, "Implementing curricular change," Comput. Sci. Eng. **8**(5), 32–37 (2006).
[8]J. Tobochnik and H. Gould, "Teaching computational physics to undergraduates," in *Annual Reviews of Computational Physics IX*, edited by D. Stauffer (World Scientific, Singapore, 2001).
[9]R. L. Spencer, "Teaching computational physics as a laboratory sequence," Am. J. Phys. **73**(2), 151–153 (2005).
[10]P. H. Borcherds, "PYTHON: a language for computational physics," Comput. Phys. Commun. **177**, 199–201 (2007).
[11]A. Buffler, S. Pillay, F. Lubben, and R. Fearick, "A model-based view of physics for computational activities in the introductory physics course," Am. J. Phys. **76**(4/5), 431–437 (2008).
[12]T. Timberlake and J. E. Hasburn, "Computation in classical mechanics," Am. J. Phys. **76**(4/5), 334–339 (2008).
[13]P. Brna and D. Duncan, "The analogical model-based physics system: a workbench to investigate issues in how to support learning by analogy in physics," in *Lecture Notes in Computer Science*, edited by G. Goos, J. Hartmanis and J. van Leeuwen (Springer-Verlag, Berlin, 1996), Vol. 1108, pp. 331–339.
[14]M. R. Matthews, "Models in science and in science education: an introduction," Sci. Educ. **16**, 647–652 (2007).
[15]J. Starmer and C. F. Starmer, "The joy of learning: main ideas, scaffolding and thinking: building new concepts by modeling," <www.cs.duke.edu/~cfs/pubs/datamodel.pdf>.
[16]<compwiki.physics.utoronto.ca>.
[17]L. Avramidis, "LabDAQ improves physics education," National Instruments, <sine.ni.com/cs/app/doc/p/id/cs-11331>.
[18]R. Beichner, "An introduction to physics education research," www.compadre.org/Repository/document/ServeFile.cfm?ID=8806\&DOCID=1147.
[19]R. W. Hamming, *Numerical Methods for Scientists and Engineers*, 2nd ed. (Dover Publications, NY, 1973), Chap. 22.
[20]R. M. Serbanescu, "Notes on error analysis," <www.physics.utoronto.ca/~phy225h/web-pages/Notes_Error.pdf>.
[21]P. R. Bevington and D. K. Robinson, *Data Reduction and Error Analysis for the Physical Sciences*, 3rd ed. (McGraw Hill, Boston, 2003).
[22]Fiesta Plate consumer information, <www.orau.org/ptp/collection/consumer\%20products/fiesta.htm>.
[23]"Fiesta plate physics," <hyperphysics.phy-astr.gsu.edu/Hbase/Nuclear/nucbuy.html>.
[24]R. D. Knight, *Physics for Scientists and Engineers with Modern Physics* (Pearson Education, San Francisco, 2004), Chap. 34.
[25]G. L. Baker and J. P. Gollub. *Chaotic Dynamics: An Introduction* (Cambridge University Press, Cambridge, 1996).
[26]R. M. Serbanescu, P. Kushner, and S. Stanley, "Integrating computation into undergraduate physics labs and lecture courses: the University of Toronto experience," Canadian Association of Physics Congress, Toronto, June 7–11, 2010.
[27]R. M. Serbanescu and P. Kushner, "Setting a new lab course: from drafts to running it for the first time," Int. J. Arts Sci. **3**(9), 203–207 (2010).
[28]R. H. Landau, "Computational physics: A better model for physics education?," Comput. Sci. Eng. **8**(5), 22–30 (2006).
[29]G. D. Chase and E. J. Okie, "Combining cooperative learning and peer instruction in introductory computer science," ACM SIGCSE Bull. **32**(1), 372–376 (2000).
[30]T. G. Gill and Q. Hu, "Peer-to-peer instruction in a programming course," Decision Sci. J. Innovative Educ. **4**(2), 315–352 (2006).
[31]See Ref. 16, <www.physics.utoronto.ca/\~phy225h/web-pages/New_Practicals224_324.htm>, and <www.physics.utoronto.ca/~phy225h/web-pages/Python_exercises.htm>.