Training in "Computer Survival Skills" for students in the Biological, Mathematical, and Physical Sciences

John W. Heimaster

Draft of January 6, 2009

Our incoming students have grown up in a digital, multimedia world. They depend upon its services for much of their information and most of their entertainment. This experience has given them remarkable *computer literacy* in some areas, but they lack several basic skills needed for science education and science research. I propose that we provide our students a systematic introduction to the computational tools of the modern scientist, allowing us to enrich many of our courses and to prepare our students for the research laboratory.

The computing environment of the modern scientist extends far beyond the numerical mathematics of previous decades. We use powerful tools for symbolic and numerical mathematics (*Mathematica* and *Matlab*), document preparation (TEX), control of lab instruments (*LabView*), graphics, literature searching, management of large volumes of data, and many other tasks. We hesitate to use these same tools in our courses, since many of our students are not trained, and we do not wish to spend precious class time training them in each course. We could pose more realistic class problems and could receive more professional documents in our classes if our students were already trained. In our research labs we expect all necessary skills to flow from student to student, and we are continually surprised by the gaps in students' knowledge.

In an environment of limited available credit hours, it will be difficult to introduce any new content. This material must be presented very compactly, in a way that immediately improves the efficiency of other courses. OSU Engineering has been teaching *Matlab* in the Freshman Year (Engineering

183), and then using specific *Matlab* toolboxes for statistics, circuit analysis, etc., throughout their curricula.

For our purposes, I propose three few-credit-hour courses:

- An introductory course, taken very early in a student's career, emphasizing skills useful in many subsequent science courses.
- An advanced course, taken in the third or fourth year, emphasizing skills useful in research and skills too sophisticated for the introductory course.
- An optional course for incoming graduate students who did not obtain these skills at their previous institutions.

Such courses do not substitute for conventional courses in programming, numerical analysis, or computational science that are taken by many of our students. Rather, they are intended to introduce a set of "survival skills" that are not covered in such courses, and which students are often expected to learn *somehow.* We all learned (some of) these skills that way, and there is no excuse for inflicting the same learning process on today's students.

1 Introductory Course

This course should prepare the student to succeed in future science courses. It should be taught as early as possible, despite the students' limited knowledge of mathematics. At a minimum, it should:

- Introduce the many roles of computers in science.
- Acquaint the students with available software for their personally-owned computers. The major packages of interest are *Matlab*, licensed campuswide by the College of Engineering, and *Mathematica*, licensed campuswide by MAPS. These can be (and should be) installed on each student's personal computer(s).
- Provide basic training in *Matlab* and *Mathematica.*
- Discuss the rudiments of numerical analysis: errors and rounding, root finding, numerical integration, Monte Carlo methods.

- Discuss the program development process.
- Introduce document preparation with equations and figures.
- Introduce basic statistics.
- Discuss some exciting topics that the student has heard about: parallel computing, grid computing, large-scale data management.

In each case, it is more important that the students be aware of many topics and tools than that they master a few. This material should be team-taught by real practitioners.

2 Advanced Course

This course should complete the students' introduction to scientific computing, with emphasis on techniques for research. This course should:

- Discuss more advanced numerical analysis: linear algebra, differential equations, optimization, stability of solutions.
- Discuss TeX and LaTeX.
- Discuss the searching of scientific literature; the staff of the Science and Engineering Library might participate here.
- Discuss laboratory computers, data acquisition, and data preservation.
- Discuss more serious statistics: regression, confidence tests, and experimental design.
- Discuss the use of high-performance computing facilities.
- Demonstrate the use of presentation tools, including Internet videoconferencing and multimedia archiving.

This course *must* be team-taught, due to its breadth. It seems desirable to partition it into two pieces: common content for all our students (perhaps 80%) and discipline-specific content. For example, the specialized material might include the drawing of chemical structures, or might emphasize modelling for theorists and data mining for experimentalists.

3 Graduate Course

This course should combine much of the two undergraduate courses. Since the students' mathematics preparation is strong, some rearrangement of material is appropriate. This course should also be team-taught, with a discipline-specific component.

4 Implementation

Much of this material can be effectively learned outside of class, exploiting formats such as *Mathematica* notebooks. A collection of individual modules might be appropriate, with self-paced delivery. Classroom time could then be devoted to real examples, showing the relevance of these topics to the students' future careers.

5 Related Work

Several OSU departments teach courses that partially overlap the material discussed here. Physics teaches some introductory material in a required undergraduate lab course. Both Chemistry and Physics teach courses in the computational aspects of their respective disciplines. Mathematics teaches a one-quarter course in numerical methods, a one-quarter course in numerical solution of PDEs, and a three-quarter course in numerical methods in science. Computer Science and Engineering teaches a course in numerical analysis, a course in high-performance computing, and a course in parallel computing. These are generally much narrower and deeper than the courses contemplated here, are taken by a small minority of students, and are taken at the senior or graduate level.

A recent paper[1] asserts:

> To be able to invoke computational approaches when appropriate, physicists must be acquainted with at least one operating system, a versatile text editor, a spreadsheet program, an array/number processor, a computer algebra system, a visualization tool, a programming language that supports standard packages

[1]David M. Cook, Computation in undergraduate physics: The Lawrence approach, *American Journal of Physics*, Vol. 76, Nos. 4 and 5 (April/May 2008)

> and libraries, a program for circuit simulation, a program for data acquisition, a technical publishing system with the capability for easy inclusion of equations and figures, a drawing program for creating publication quality figures, and a presentation program. The curricula that train these physicists must include an exposure to at least some of these approaches and tools.

The paper describes a curriculum that distributes much of this material throughout a four-year curriculum, including a required course in Computational Mechanics and an elective course in Computational Physics (introduced "after several unsuccessful attempts to incorporate its topics as components of other upper-level courses").

6 Disclaimer

The writer's background is Physics and Engineering, which has strongly influenced the choice of course topics. A broad conversation is needed to design courses to serve our many disciplines.