AAPT Topical Conference Report

Computational Physics for Upper Level Courses

Davidson College, July 27-28 2007

Recommendations

In the twenty-first century physicists are faced with an important educational decision: Do we continue to teach our subject and our students the way we have in the past or do we modernize our teaching approach to include new, interesting, and cutting-edge topics. One broadly applicable way to modernize the teaching of the subject is by including computation throughout the undergraduate physics curriculum. In the past, computation has been viewed as a secondary, non-essential tool. However, currently computation is, in many cases, the primary (or sole) means to solve new and interesting cutting-edge problems in physics and the other sciences. There are numerous reasons to include computation in the undergraduate physics curriculum:

1. The solution of physics problems requires three separate and important subdisciplines: experimental, theoretical, and computational. Education should also contain a balance of the three.
2. Many new areas of science, such as non-liner dynamics and lattice gauge field, rely on computation for their understanding.
3. The inclusion of computation brings with it many pedagogical features such as interactive engagement that are deemed positive from the results of physics education research.
4. Research experience is increasingly important for our students' careers (graduate school and employment) and computational projects can be very similar to research experience.
5. Studies show that our graduates lack computational skills.
6. The use of computation can be a bridge across curricular boundaries (providing a natural vehicle for multidisciplinary work).

To this end, we put forward the following recommendations based on the discussions at the Computational Physics for Upper Level Courses Topical Conference:

1. Undergraduate physics students should learn computational methods throughout the curriculum. To this end we point to four basic modalities:
    a. **Non-compiled, low-level programming based on high-level packages.** The idea here is to have easy-to-use tools that allow for quick prototyping. The focus is on the model and not on programming or algorithms. Examples include problem solving environments such as: VPython, Easy Java Simulations, STELLA, Maple, Mathematica, MatLab, etc.
    b. **Compiled, high-level programming based on libraries.** The idea here is to have tools that allow for a full programming environment. The focus is specifically on programming and algorithms. Examples include: C, Java, Fortran, etc.
    c. **Symbolic mathematical manipulation programs, also called computer algebra systems (CAS).** The use of these tools allow for quick manipulation of mathematical expressions. Examples include: Mathematica, Matlab, MathCad, and Maple. (These tools are also problem solving environments.)
    d. **Data analysis and acquisition software.** Given the increased sophistication of experiments, it is important for students to learn about computer-based data

acquisition and high-level analysis of data. Examples include: Excel, DataStudio, and LabView.

2. New textbooks should be developed that integrate computation along with content and problem solving.
3. The pedagogical effectiveness of computational tools and methods should be assessed. While the generic features of the integration of computational physics into the curriculum appears to follow the recommendations of physics education research, direct assessment of courses that use these techniques has yet to be carried out.

The rationale behind this set of recommendations can be summarized in a recent post to the AAPT Committee on Physics Undergraduate Education list-server: Computation should be (but typically isn't) a major component of the curriculum. It is currently the case that a physics major may graduate never having seen any computational physics at all. Yet physics is no longer theory and experiment but rather theory and experiment and computation, and the interplay among all three. A curriculum in which computation is absent or plays a minor role is inauthentic to the contemporary discipline. This is one of the most striking examples of our failure to update the curriculum.