Computational Science and Engineering Education

Joan Adler, Technion-IIT

July 2, 2019

Abstract

This volume presents approaches to computational education in different fields of science and engineering. Contributions include descriptions of early and current general courses at the senior undergraduate and early graduate levels, as well as some computational exercises for more specific courses. Computer science courses as such are intentionally excluded, because our emphasis is on applications of computers to various science and engineering subjects.

1 Introduction

Despite the enormous impact of computers in our lives, and the frequent use of computers in the administrative side of education, many university course syllabi have barely changed in many years. A depressing example is the $\sin x = x$ equivalence in elementary mechanics. This was important before students had easy access to computers, but led to confusion among the more rigorously inclined. Today, students may have a computing course requirement, but it may not be correlated with their basic courses and serious education towards good practice or relevant numerical algorithms for their field is lacking.

Justifications for the lack of computational aspects include:

- Too time-consuming and disruptive.
- No funds.
- This is how it has always been.

Such explanations may be related to lack of knowledge and therefore confidence on the part of the lecturers. While it is true that many students know more about numerical coding than their lecturers, and not everyone is comfortable with this, good preparation and support materials can help overcome this. For example, it is possible to incorporate the students' knowledge into class content; e.g. by giving them a challenge in advance to answer a specific question. This will be discussed further below.

In this volume emphasis is placed on a single or few courses in computational science/engineering added to a more standard degree program. One type of course is an introduction to High Performance Computing (HPC), such as those presented in a 2008 volume of Computers in Science and Engineering (CISE) [1]. Less emphasis is placed in this issue on computer science or degrees or certificates in computational science and engineering; although they can be excellent in and of themselves, the addition of significant expertise in one or two courses is the challenge we choose to address here. A leaning towards physics and materials science is a natural result of this author's research area.

A special issue of CISE in 2006 [2] addressed Computational Physics education, both of the additional course or two type and of the entire degree variety, one example of the latter being Rubin Landau's article in that issue. The present volume has a more general coverage, but the problems are similar.

2 Past and extant computational science and engineering courses and projects

This list cannot be all inclusive and apologies to authors of articles not included here. Descriptions of new projects not mentioned here would likely provide excellent material for articles in future issues of Computers in Science and Engineering. There are several well-funded iniatives towards syllabus development with integration of physics/chemistry material with numerical support, examples include a national initiative in Norway [3].

Early computational physics courses that have been described in the archival literature include Peter Bocherd's [4] class taught since about 1984 in laboratory format and Steven Gottlieb's [5] class since 1987, although his paper is dated much later. I have been teaching such a course since 1988 again with a manuscript published only in 2014 [6]. Although developed independently these have loose similarities, such as hands-on experience in class. The Computational Physics community and the present author in particular owe the late Peter Bocherd much appreciation for his leadership and encouragement of Computational Physics European and International Computational Physics collaboration. There are many, many more courses, both similar to the above as late undergraduate courses and more integrated or specialised, especially those given by the authors of the textbooks discussed in the next session.

3 Textbooks

Undergraduate courses work best with one (or more) associated textbooks. Again, there are too many of these to list, but the more widely used ones for Computational Physics include Koonin's books in several languages with the first in BASIC dating from his course in 1983 at Caltech and Gould and Tobochnik's early book in TRUE BASIC from 1988. Later outstanding books include those written by Gould, Tobochnik and Christian, and recent and several web based creations, see below. There are many more recent books of substantial merit, including but not limited to Giordano [7] from 1997 as well as several web collections in diverse languages (of the spoken, not computer flavour), notably Joaquin Marro's spanish material [8]. Titus Beu [9] has recently published an excellent book with python and C/C++ codes. Giordano's book has an interesting section on interdisciplinary topics and it's second edition, (with Nakanishi) has python and associated MATLAB exercises and solutions. As above, textbooks considered important ommissions from this brief list could form the basis for future contributions to "Computers in Science an Engineering.

4 Some background material

A short survey passed around at recent conferences and in the author's vicinity has had some interesting responses and covers preferences on algorithms, evaluation method(s) and correct coding practices. Algorithms of choice are somewhat dependant on the responders' research area, but include Monte Carlo (several types), Density Functional Theory, GW (Green's Function/ screened Coulomb interaction), numerical analysis, iterative procedures, ODE solutions, molecular dynamics numerical methods for PDEs, Metropolis, Wang-Landau, Verlet, particle dynamics, agent models, and percolation. All respondents except one consider vizualization important, and everyone considered correct coding practices important. The big variations were on evaluation practices. Again all but one person used projects, gave homework, had class lectures, but in each case the exception was a different individual. In addition to the textbooks mentioned above, only Allen and Tildesley's "Computer simulation of liquids" was mentioned more than once in the responses,

5 Language

This is an issue with many different "correct" answers and possibly no single best choice. It is also a function of time and place. Some of us prefer a compiled language (FORTRAN, FORTRAN90, c/c++) as the best transition to later HPC research and/or industrial programming. Others like "complete suites" such as MATLAB/MATHEMATICA or python, in all of which visualization is easily included as it is in TrueBASIC. One good policy is a mix in order to prepare the student for many possibilities. All three options have advantages and disadvantages. For compiled languages graphics need to be incorporated in addition, "complete suite" approaches tend to use a lot of CPU time, and mixes require a lot of learning. The latter two also require licenses and are not always accessible in the student's future.

The compiled options require a seperate visualization package; our experience has been that the academic use PGPLOT package [10] fills this role admirably. Its subroutine library structure gives a very natural start to OpenGL coding down the line. My students over 25 years have added LINUX compatible graphics to both Koonin's and Gould and Tobochnik's FORTRAN codes, downloadable from links in [6]

6 Miscellaneous tips

Although a lot of numerical examples are desirable both in a Computational Physics class and also in general physics lectures, even lecturers who agree may experience difficulties. As mentioned above lecturers may lack confidence in their expertise relative to their students. There are approaches that can mitigate this even for lecturers concerned about students who "know" more than they do, or even think that they do.

For example, in my Modern Physics class, the week before revising wave addition (prior to the Bragg scattering topic) I challenged providing a code to add two sine/cosine waves. From a hundred student course I got 15 answers, including the obvious matlab/mathematica/excel and several web function interfaces I was not previously aware of. I learnt, they learnt, and the positive reinforcement to the contributing students calmed down the smart alecks.

Concerns about time requirements even in Computational Physics class can be mitigated by starting with ready codes, then moving to students needing to make minor adjustments and only when some confidence develops be left to create own code. This is straightforward with a little thought.

The last tip is again directed at the smart aleks and helps overcome the middle objection of the no funds aspect. One can always encourage students who think a problem is "too easy" to provide an additional solution in a languages of their choice, especially if its one the lecturer wishes to teach. Some bonus points in a homework exercise extending the ready FORTRAN with MPI code to e.g. OpenCL provides zero-cost examples and happy students who deserve good grades.

The first common objection, about the time-consuming aspects is quite preventable with good preparation. I cannot think of any solution to the "this is how it has always been done" objection beyond "so what".

7 Less HPC oriented options

Many of the older courses and books described above, have an implicit end-use of HPC orientation, as indeed do I. However, the "Complete Suite" approach, using publically avaliable tools such as Jupyter [11] with Physics examples can also be helpful, although many of these have less of a physics orientation than the texts mentioned above.

Figure 1: Author photograph

8 Acknowledgements

I thank my mentors for support over the years, especially Amnon Aharony who organised the first Computational Physics course that I taught while still a postdoctorant. Peter Bocherds encouraged my involvement in International Computational Physics activities. More recently discussions with some of the authors in this issue helped clar ify matters and special mention should be made of Jacob Fosso Tande for proposing the inclusion of the less HPC oriented frameworks.

9 About the author

Joan Adler is a still active but recently retired senior research associate at the Technion - IIT, Haifa, Israel. Her research interests include simulation and visualization of atomistic and electronic properties of materials and computational physics education. Adler received a PhD in physics from the University of New South Wales. Her students and postdoctoral fellows developed the AViz visualization code for atomistic and electronic systems. Adler is a past president of the Israel Physical Society and recent vice-chair of the IUPAP C20 Commission. Contact her at phr76ja@technion.ac.il. She is a member of the editorial boards of “Computers in Physics” and “Computing in Science and Enginering”.

References

[1] S. Lathrop and T. Murphy, ”High-Performance Computing Edcation” in Computing in Science and Engineering, vol. 10, no. 05, pp. 9-11, 2008.

[2] https://www.computer.org/csdl/magazine/cs/2006/05

[3] https://www.mn.uio.no/ccse/english/

[4] P. H. Borcherds 1986 Phys. Educ. 21 238

[5] http://physics.indiana.edu/ sg/p609.html

[6] J. Adler, Physics Procedia, 53 , 2-6 (2014).

[7] N. J. Giordano, “Computational physics”, Prentice-Hall, NJ, 1997.

[8] http://ergodic.ugr.es/cphys/

[9] T. A. Beu, “Introduction to numerical programming”, CRC Press - Taylor and Francis, Boca Raton, 2015.

[10] http://www.astro.caltech.edu/ tjp/pgplot/

[11] https://jupyter.org/

https://www.mn.uio.no/english/about/collaboration/cse/national-group/computing-in-science-education.pdf