Teaching Computational Physics

by Steven E. Koonin

MODERN PHYSICS RESEARCH is concerned increasingly with complex systems composed of many interacting components — the many atoms making up a solid, the many stars in a galaxy, or the many values of a field in space-time describing an elementary particle. In most of these cases, although we might know the simple general laws governing the interactions between the components, it is difficult to predict or understand qualitatively new phenomena that can arise solely from the complexity of the problem. General insights in this regard are difficult, and the analytical pencil-and-paper approach that has served physics so well in the past quickly reaches its limits. Numerical simulations are therefore essential to further understanding, and computers and computing play a central role in much of modern research.

Using a computer to model physical systems is, at its best, more art than science. The successful computational physicist is not a blind number-cruncher, but rather exploits the numerical power of the computer through a mix of numerical analysis, analytical models, and programing to solve otherwise intractable problems. It is a skill that can be acquired and refined — knowing how to set up the simulation, what numerical methods to employ, how to implement them efficiently, when to trust the accuracy of the results.

Despite its importance, computational physics has largely been neglected in the standard university physics curriculum. In part, this is because it requires balanced integration of three commonly disjoint disciplines: physics, numerical analysis, and computer programing. The lack of computing hardware suitable for a teaching situation has also been a factor. Students usually acquire what skills they do have by working on specific thesis problems, and as a result their exposure is often far from complete.

This situation and my professional background in large-scale numerical simulations motivated me to begin teaching an advanced computational physics laboratory course at Caltech in the winter of 1983. My goal was to provide students with direct experience in modeling non-trivial physical systems and to impart to them the minimal set of techniques for dealing with the most common problems encountered in such work. The computer was to be viewed neither as a "black box" nor as an end in itself but rather as a tool for getting at the physics.

Another factor in my decision to develop a computational physics curriculum was the ready availability of hardware that could pro-

Computational Physics curriculum

Unit	*Numerical Methods*	*Example*	*Project*
1	Differentiation, quadrature, finding roots	Semiclassical quantization of molecular vibrations	Scattering by a central potential
2	Ordinary differential equations	Order and chaos in two-dimensional motion	Structure of white dwarf stars
3	Boundary value and eigenvalue problems	Stationary solutions of the one-dimensional Schrödinger equation	Atomic structure in the Hartree-Fock approximation
4	Special functions and Gaussian quadrature	Born and eikonal approximations to quantum scattering	Partial wave solution of quantum scattering
5	Matrix inversion and diagonalization	Determining nuclear charge densities	A schematic shell model
6	Elliptic partial differential equations	Laplace's equation in two dimensions	Steady-state hydrodynamics in two dimensions
7	Parabolic partial differential equations	The time-dependent Schrödinger equation	Self-organization in chemical reactions
8	Monte Carlo methods	The Ising model in two dimensions	Quantum Monte Carlo simulation of the hydrogen molecule

vide each student with an individual computing environment. Personal computers can be used easily and interactively through a variety of high-level languages, and they offer a numerical power sufficient for illustrating many research-level calculations. Moreover, the graphics facilities commonly found on such systems allow an easy but often startling insight into many problems. In short, I (and the students) could concentrate on the strategy of a calculation and the analysis of its results, rather than on the mechanics of using a computer.

How, then, was I to teach the art of computational physics? I quickly decided that the traditional lecture-cum-assignment approach was not optimal, as there is no teacher better than direct experience and computing is a very personal activity for most physicists. I therefore planned a situation where students would work through material on the computer that would teach by example and exercise. This format had the added benefit that students could work largely on their own, at their own pace, and at times of their choosing.

In defining the course, it was relatively easy to identify the broadly applicable numerical methods I wanted to cover, but choosing the physical situations in which to demonstrate these methods was more difficult. I attempted to satisfy simultaneously the criteria that the physics discussed be an "interesting" extension or enrichment of the usual quantum, statistical, or classical mechanics material, that the scale of the computation be appropriate to the numerical power of the hardware, and that the problem not be soluble analytically. In the end, I was able to find 16 such case studies. In more than half of them the student can compare calculated results with experiment or observations.

The curriculum I developed during the 1983 and 1984 academic years consists of eight units. The student begins each by reading a text section that gives a heuristic discussion of several related techniques for accomplishing a particular numerical task. Intuitive derivations of simple, general methods are emphasized, with appropriate references cited for rigorous proofs or more specialized techniques. Short, mathematically oriented exercises involving only a small amount of programing reinforce this material.

After reading the text section, the student works through the remaining two sections of the unit — the example and the project. Each of these is a brief exposition of a particular physical situation and how it is to be modeled, using the numerical techniques taught in that unit together with a set of exercises for exploiting and understanding the associated program. The example and project differ only in that I expected the students to use the canned program for the former, while perhaps writing their own program from scratch for the latter.

Each of the programs I developed functions in several capacities — as an easy-to-use demonstration of the physics, as a laboratory for exploring changes in the numerical algorithms or parameters, and, in the case of the projects, as a model for the student's own program. Thus, more than simply working correctly, the programs had to be easily read and understood. Simple organization, full documentation, and a structured programing style were essential. As much of each program tends to be input/output (I/O) and "bookkeeping," the few important numerical sections had to be called out clearly and I often had to sacrifice elegance in coding and speed of execution for the sake of intelligibility. More significantly, my ambitions were restrained frequently by a desire to keep the calculation and graphics displays simple. Despite these constraints, with some thought and care I was able to work to my satisfaction within the format described.

The choice of language invariably invokes strong feelings among scientists who use computers. Any language is, after all, only a means of expressing the concepts underlying a program, and the important ideas in the curriculum are relevant no matter what language I decided to use. However, some language had to be chosen to implement the programs, and I finally settled on BASIC. The BASIC language has many well-known deficiencies, foremost among them being a lack of local subroutine variables and an awkwardness in expressing structured code. Nevertheless, these are more than balanced by the simplicity of the language and the widespread fluency in it; BASIC's ready availability on the microcomputers I was using; the existence of both BASIC interpreters convenient for writing and debugging programs as well as of compilers for producing rapidly executing finished programs; and the powerful graphics and I/O statements in this language. Virtually all of the students were familiar with some other high-level language and so could learn BASIC "on the fly" while taking the course. Several students elected to write their projects in other languages.

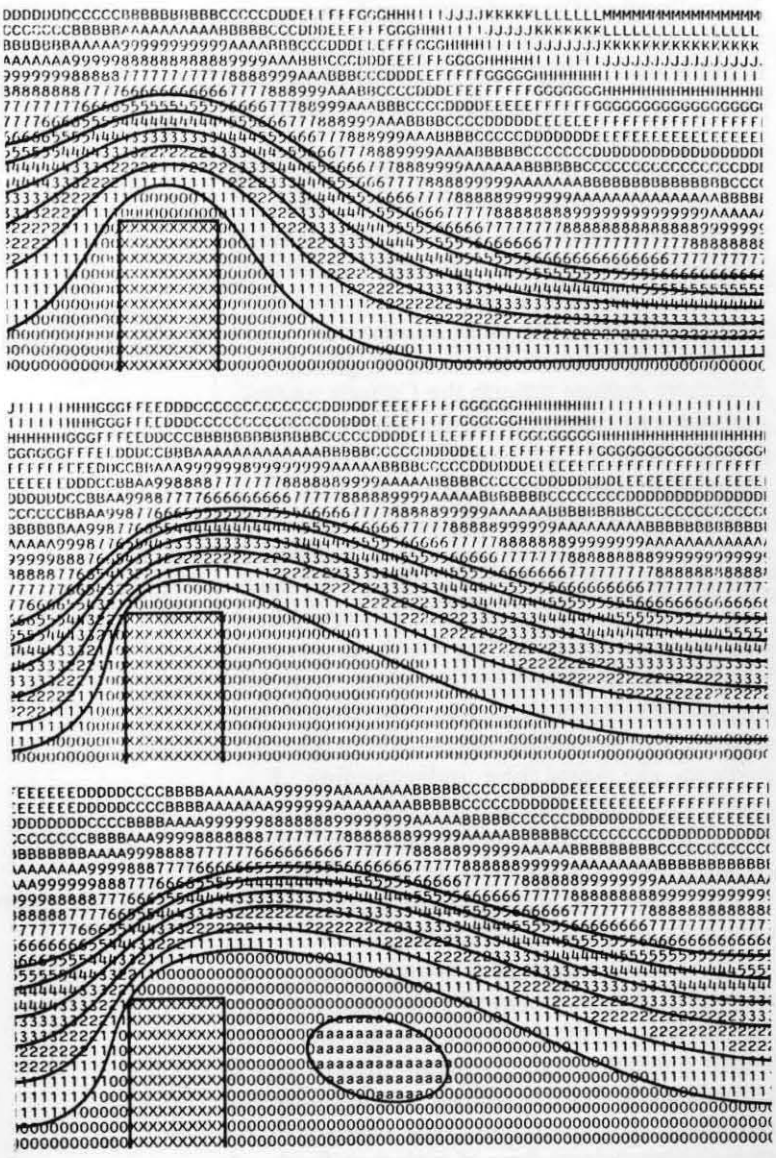
I taught the course in a laboratory format to junior and senior physics majors. All of the students had taken (or were taking) the conventional courses in classical, statistical, and quantum mechanics, and so were familiar with many of the physics concepts involved. Moreover, there is enough of a computer culture among the Caltech undergraduates so that most of them were quite familiar with the hardware before beginning the course; those who were not became proficient after several hours of individual instruction. As mentioned above, students worked through the material largely on their own, although a teaching assistant and I held weekly office hours during which we were available for help and consultation. Individual half-hour interviews upon the completion of each unit served to monitor the students' progress and assess their understanding. I found that typical students, working six or seven hours per week, could complete three or four units in a ten-week term, perhaps writing their own codes for two of the projects and using my codes for the others.

A brief discussion of a couple of the examples and projects will give some feeling for the level of the material and the style in which it is presented. Project 6 illustrates techniques for solving elliptic partial differential equations by considering steady-state (time-independent) flow of a viscous fluid about an obstacle; for example, the flow of a stream around a rock. The mathematical description of the flow is based on continuity (fluid is neither created nor destroyed) and the response of each bit of fluid to the pressure and viscous forces acting on it. When the flow is two-dimensional (coordinates x and y), these two physical principles can be embodied in the coupled non-linear elliptic equations,

$$\left[\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right]\psi(x,y) = \zeta(x,y);$$

$$\nu\left[\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right]\zeta(x,y) = \left[\frac{\partial\psi}{\partial y}\frac{\partial\zeta}{\partial x}-\frac{\partial\psi}{\partial x}\frac{\partial\zeta}{\partial y}\right]$$

Here, ψ is the stream function specifiying the direction of flow at each point, ζ is the



Printer output shows the flow of a viscous fluid around a rectangular plate at various velocities. Streamlines have been superimposed on the computer-generated pattern. The direction of flow is from the left, and each pattern is reflection-symmetric about the lower edge. Note particularly the hydraulic "jump" above the plate, the laminar flow at low velocity (top), the increasing separation of the flow from the rear of the plate at higher velocities (middle), and the vortex behind the plate at the highest velocity (bottom).

fluid's vorticity, and ν is the kinematic viscosity. Boundary conditions on ψ and ζ (for example, the fluid cannot flow into the object's surfaces) complete the specification on the problem.

These equations cannot be solved analytically, but are amenable to a numerical treatment on the computer through discretization. The resulting large number of non-linear algebraic equations can be solved through a relaxation process, in which initial guesses for the stream function and vorticity are refined iteratively.

Following a brief derivation and discussion of the equations, students are guided through a series of steps culminating in a program to solve the flow about a rectangular plate at various speeds. Results can be displayed as character plots, as shown in the figure above, and the drag and pressure forces calculated can be compared with values measured in the laboratory. Here again, the computer is used to simulate situations for which analytical solutions are impossible and for which intuition is difficult to develop.

Example 5 illustrates the use of a computer in a different way — the analysis of experimental data by least-squares fitting. The physical situation here is the scattering of high-energy electrons (approximately 200 million electron volts or more) from atomic nuclei, a topic currently of high research interest in nuclear physics. Because the electrons interact with the nucleus through the Coulomb force, the cross sections for such scattering are sensitive to the distribution of electrical charge (that is, the structure) of the nucleus. Indeed, the measured cross sections can be "inverted" in a model-independent way to obtain the nuclear charge density.

My program for this problem analyzes actual experimental data to infer the charge densities for nuclei of calcium, nickel, and lead. An iterative, nonlinear least-squares fit procedure is used to adjust the charge density to the measured cross sections. The density and fit to the data are displayed as the iterations proceed. In running and understanding this program, the student is asked to check the accuracy of the fits obtained, to extract information about the nuclei from them and compare with simple nuclear models, and to explore alternative fitting strategies.

In the course of developing and teaching this curriculum, I have been impressed by several unexpected advantages in formalizing the instruction of advanced students in using the computer as a tool for doing physics. To write a program simulating a given physical situation, the student must understand the physics in a different (and complementary) way than is needed for an analytical approach. Programs also bring a flexibility and vividness of presentation that is difficult to obtain otherwise. Moreover, simulating systems brings a sense of exploration and surprise to the learning process, as understanding how results change as parameters or algorithms change often leads to greater insights. Finally, because complex situations can be presented, students are exposed in detail to research-level problems at an earlier stage of their career. In these ways, I expect that computer-based education in physics at this level will supplement, rather than supplant, the traditional mode of lecture instruction. □