PHYS 3210: Computational Methods in Physics (winter)

Calendar description: This course introduces the student to methods of solving mathematically difficult or tedious problems using the computer. After a brief introduction to Unix and Fortran or C, the course focuses on some of the algorithms most useful to a physicist, including root-finding, spline fitting, Richardson extrapolation, Romberg integration, Runga-Kutta and Monte Carlo methods. Student complete a term project which applies learned algorithms to problems in computational (astro)physics.

Contents

Overview

The "Methods (200) Stream" is specifically designed to provide the mathematics and computing skills needed in concurrent and subsequent physics courses. The 200-stream courses are not intended to teach mathematics and computing for their own sakes, but for the purpose of relieving the instructors of the physics courses from having to teach mathematics in support of their curriculum. Thus, the curriculum of the 200-stream courses should be sensitive and altered to suit the mathematical and computational needs of the dependent physics courses.

Prerequisites

PHYS 3200 Mathematical Methods in Physics I

This is a weak prerequisite, and is here to provide the mathematical sophistication that is useful in appreciating the difficulty in solving certain problems by hand, and thus

appreciate better the need for a computer.

PHYS 3500 Quantum Mechanics I

This is also a weak prerequisite, and is listed to ensure the student has a certain level of sophistication in physics to comprehend the nature of the problems being solved with computational methods. Many, but not all, of these problems are drawn from Quantum Mechanics and many, but by no means all, of the computational physics problems that can be assigned as term projects involve the direct numerical integration of Schrödinger's equation. Still, most students could get through this course with little more than the physics learned in PHYS 1101 (University Physics II).

Dependent courses

While there are no courses that list this course as a prerequisite, instructors of fourth year courses are encouraged to assign their students homework assignments that may very well require the use of the computer, either to use existing computer and graphics packages or to write a program to solve a particular problem. Further, many students will need the computing skills learned in this course to perform their duties as a summer RA after their third year, and/or to perform their honours thesis research in the fourth year.

Student Outcomes

Students completing PHYS 3210 should begin to master the following skills:

1. Be able to log onto a unix-based platform and freely use existing software packages necessary to solve a problem in physics or graph and analyse the results.
2. Create a working and comprehensible (by a human) computer program capable of solving significant physical problems without undue encumbrance by the details of the computing OS or language;
3. Create a type-set scientific document (e.g., LaTeX).
4. Gain familiarity in the most commonly used algorithms in solving physics problems on the computer, such as root finding, interpolation (spline fitting), Richardson extrapolation, Romberg integration, Runga-Kutta, *etc.*

Typical problems students completing PHYS 3210 should be able to solve:

1. Find the energy levels in a finite square well
2. Given a sparse data set, fit the data to a smooth and continuous curve
3. Given *any* n^{th} order ODE, find and graph its numerical solution to within a prescribed tolerance.
4. Solve a truly non-linear problem in physics with the computer.

Curriculum

1. introduction

   - basic binary logic/truth tables, *etc.*
   - UNIX
   - FORTRAN, C^{++}, or both

2. computational algorithms useful for a physicist

   - root finding
   - interpolations (spline fitting)
   - approximations (Richardson extrapolation)
   - Romberg integration
   - n^{th} order differential equations (Runga-Kutta)
   - Monte Carlo

3. non-linear computational physics term project (in lieu of final exam) such as:

   - time-dependent Schrödinger equation (*e.g.*, time-evolution of a wave packet encountering a finite barrier)
   - time-independent Schödinger equation (*e.g.*, energy levels of carbon)
   - investigate the Kelvin-Helmholtz instability in 2- and 3-D (*e.g.*, using ZEUS-3D)
   - N-body simulations (stability of solar system, Toomre and Toomre problem, *etc.*)
   - chaotic systems, Monte Carlo methods
   - solving Poisson's equation

Suggested texts

In the past, both DeVries' text and Numerical Recipes (details for both listed below) have been used as formal texts for this course. In truth, only Numerical Recipes will endure in the students' library as an invaluable resource, and thus if there is to be a required text for this course, Numerical Recipes is recommended. The other texts, all available within the department and some at the library, are useful for ideas and specific algorithms for term projects, if that is how the instructor wishes to assess the course.

Numerical Recipes by Press *et al.* (ISBN 0-521-43064-X, FORTRAN; 0-521-35465-X, C)

- offers lucid and complete explanations for each algorithm, with exhaustive references for additional mathematical background as needed

- no computational physicist's library is complete without this text

- recommended as the required text for this course

A First Course in Computational Physics by DeVries (ISBN 0-471-54869-3)

- used for two years as the main text for this course, it was discontinued as many bugs in the algebra and algorithms made its use difficult.
- written in simple and easy to understand language, it covers most of the algorithms normally used by physicists
- may be too low a level for third year students

Computational Physics by Koonin (ISBN 0-201-12279-0)

- reference text for projects

Numerical Methods for Physics by Garcia (ISBN 0-13-151986-7)

- reference text for projects

Computational Physics by Giordano (ISBN 0-13-367723-0)

- reference text for projects

Computing in Advanced Undergraduate Physics, edited by David Cook (Dept. of Physics, Lawrence University, Appleton, WI 54912, no ISBN)

- reference text for projects

In addition, Clarke's Primers on FORTRAN, Unix, LaTeX, gnuplot, and DBX (debugger) are available for use.

Notes to the instructor

1. (This note is primarily directed to the department chair.) At one time, PHYS 3210 was cycled with PHYS 3400 (Electrodynamics), but it was found that under this arrangement, half the students didn't get any computational experience until the second semester of their fourth year, too late to be any use for their third year summer RA and for their honours thesis research. Thus, regardless of numbers and the fact that no fourth year course specifically lists this course as a prerequisite, this course needs to be taught every year and taken by third-year students in their second semester. Note that Electrodynamics is better cycled with PHYS 3350 (Thermal Physics).

2. This is a rather difficult course to teach because one gets students of all abilities, from those who have never used a computer other than to check their e-mail, to those whose skills may surpass those of the instructor. There may be a temptation to open up a unix, FORTRAN, and/or C text, and plod through it, but this is largely a waste of time. The advise that 'any competent physicist can pick up unix and FORTRAN over a weekend' is actually quite practical. Computer-phobic students (and there are

several still) will be forced into it by the first assignment and will come quickly up to speed, while those whose skills surpass the instructors won't get bored out of their minds.

Time is much better spent in developing the mathematical algorithms (many such as Richardson extrapolation, spline fitting, and even aspects of root finding are quite surprising even to the most sophisticated students), and to bring the students to the point where they can tackle a *bona fide* non-linear problem in computational physics, which is one of the most important skills to be conveyed by this course.

3. It may be useful for students to learn more than one computing language which could be taught formally in class, or simply left as an assignment to submit a program in a second language. Currently, the only computer language primer available in the department is Clarke's FORTRAN Primer, and an equivalent primer for a second language might have to be found or developed if the goal of bi- or multi-lingual students were to be set.

4. The instructor is encouraged to add their assignments and computational projects they may develop to a growing pool for use by future instructors of this course. To be useful, these assignments and projects should come with a working solution program(s), clear instructions on how to use the programs, and specific references where appropriate.

5. The instructor is encouraged to solicit ideas for projects from other faculty, especially those teaching third and fourth year courses.

6. If the instructor is interested in incorporating a lab designed to construct of a computer/calculator from various circuits and components, he or she is encouraged to discuss this with the instructor of PHYS 3600 and/or PHYS 4600 (Experimental Physics I and II).