Course Description and Motivation

This is a course targeted toward Junior and Senior level undergraduate majors in the Physical Sciences (Physics, Chemistry, Biology, Geology, ...). The purpose of the course is to provide these students with a coherent picture of the role of computers in the sciences as well as the practical skills they will need if they choose to pursue graduate study or take jobs in the private sector. The role of computers in science is, of course, an enormous topic with many highly specialized niches. This course will focus on what one might call a “base level of knowledge” which most scientists will be expected to have. It also will provide a good starting point from which a more advanced course or self-study may follow. It is anticipated that the course will initially be offered as a “Special Topics” course through the Department of Physics and Astronomy, and eventually be offered as a 300 level course.

Students will not be required to know computer programming before the course, however they should have had at least 2 semesters (3 preferably) in calculus. This will be a very practical course in the sense that students will learn skills which they can immediately apply in their other courses or professional endeavors. For some topics, designing proper computational tools requires an understanding of the mathematics behind them. So basic theory in optimization (linear regression), numerical derivatives, and numerical integration will be covered.

There are many excellent computer languages on which to base such a course, and the list changes year by year. I believe the Python programming language is an excellent tool for both quickly learning programming concepts and developing truly useful programs. Python is gaining popularity in the physics and biology communities because of it’s rapid development cycle, extensibility, and portability. Some of it’s features are: full object orientation, an intuitive syntax, a large and openly available scientific code base, it is open source and cross platform (even when graphics and windowing are incorporated), and it is a scripting language so “experimentation” is quick and painless. Also, there is a library called “Pylab” which provides a syntax very similar to the popular commercial package Matlab. I generally will use only open source and cross platform tools so students will be able to load these packages on their machines.
Topics to be covered: (time estimates in parentheses)

- An introduction to the Python programming language and fundamental programming concepts. (4 weeks)
  1. Data structures: integers, floating point numbers, lists/arrays, functions
  2. Flow control: while and for loops
  3. Conditional statements: if-then-else and while
  4. Input/Output: file I/O, user I/O, text formatting
  5. Python graphics and plotting with *pylab* and *visual* modules

- Graphical representation of data. (2 weeks)
  1. Several tools for creating graphs of data such as python, gnuplot, and spreadsheet-style tools such as Origin.
  2. What makes a “publication quality” graph.
  3. More intuitive and informative ways to represent data such as scaling and normalization and multiple data sets.
  4. Various advantages and disadvantages to the available electronic file formats (bitmap versus vector).

- Linear and non-linear regression methods.
  We’ll develop the theory for linear regression and write python programs to perform it. We will also explore packages for fitting non-linear data and develop an understanding of the basic concepts and possible “pitfalls” associated with non-linear fits. (1 week)

- Numerical differentiation methods. (2 weeks)

- Numerical integration methods. (2 weeks)

- Numerical precision and error issues such as rounding errors, machine precision, algorithmic, and discretization errors. (1 week)

- Roots of polynomials and transcendental functions. (1 week)

- A tour of various capabilities of Mathematica including syntax, symbolic algebra and calculus, functions, and graphics (including Vetor Field plots). (1 week)
Location and Format

This course must be taught in a computer lab so students can immediately try what the instructor is doing. I suggest a ratio of no more than 2 students per computer so no student is too far removed from the task at hand. The course will be taught in the Astronomy Lab in the basement of Lewis Hall which has 12 newly upgraded Macintosh computers. All of the tools required for this course have been installed on these machines. Another option is a Linux based operating system which includes all the required packages, is tailored to scientific users, and is entirely run from a DVD (this works fine on the newer Macs). The local hard disks are mounted as read-only so the original configuration of the workstations can not be altered. Students can either access a network storage location for their files or keep all files on a USB key which they would keep with them. The Linux DVD would be given to them so they can stay in a familiar environment for their work outside of class. This course would ideally occupy one full semester, but could reasonably be offered in the intersession format.

Evaluation

Students will complete weekly homework projects related to the current topics in the course. Most tasks will be done on the computer, however some will require written out answers and some mathematics. There will be two larger scale projects in which students will choose from a set of topics designed to incorporate a wide range of the concepts covered in this course with the specifics of the problem relevant to their particular field of study. They will have to produce working code and a 2-3 page written description of relevant issues which had to be solved, the capabilities and limitations of their tools, and suggestions for future improvements.