Spring 2012

Numerical Techniques in Astronomy

AST 911 is a one-semester course in numerical techniques. Although the title adds, "in Astronomy," these methods are generally applicable to other areas of physics.

The above image is a simulation of a single mode Rayleigh-Taylor instability. The computation was performed with the FLASH code. From Calder et al., *ApJS* ***143****: 201 (2002).*

Course Objectives

After taking this course, the student will be knowledgable about algorithms for common numerical problems, including roots of algebraic equations, ordinary and partial differential equations, N-body systems, and computational fluid dynamics. In addition, the student will be adept at writing organized, modular code, and will be able to present results of a computation clearly and concisely. The course is "broad and shallow:" it covers a diverse set of topics, but doesn't drill deeply into any particular one. The intent is to introduce the student to families of numerical techniques.

Instructor

Assoc. Prof. Edward Brown
3258 Biomed Phys Sci
884-5620
ebrown@pa.msu.edu

Scheduled Course Time and Location

The course meets Fridays 12:40–2:30 in 1308 BPS.

Online Materials and Code Repository

Course information is posted on ANGEL. Codes and notes used in the course are maintained in an svn repository; see the course ANGEL page for instructions on how to access the repository. Each student will have a private directory in the repository (accessible only by that student and myself) for storing codes developed as part of the assigned projects. Packages for installing subversion clients on various platforms are available from http://subversion.apache.org. Sources for the course notes and plots will be available on the repository; the course notes are generated using XeLaTeX (http://tug.org/texlive/), and the plots are made using TIOGA (http://tioga.rubyforge.org).

Text

There is no official text for the course. The following texts are on reserve at the MSU Engineering Library. I have private copies of these texts that you may borrow for a limited time, along with books on programming, high-performance computing, and graphical design.

Texts on Reserve (in the Engineering Library)

1. Mathews and Walker, *Mathematical Methods of Physics*. New York: W. A. Benjamin (1970). QA401.M42 1970
2. LeVeque, *Finite Volume Methods for Hyperbolic Problems*, Cambridge University Press (2002). QA377 .L41566 2002
3. Metcalf et al., *FORTRAN 95/2003 Explained.* Oxford University Press (2004). QA76.73.F235 M48 2004

4. Press et al., *Numerical Recipes in FORTRAN*. Cambridge University Press (1992). QA297.N866 1992

Office Hours

There are no regularly scheduled office hours, but you are welcome to drop in, if my door is open, or to make an appointment.

Coursework

There will be frequent reading assignments consisting of short exercises, mostly analytical, as well as longer problem sets that require writing short programs. In addition, there will be a longer project that involves coding an algorithm, performing a calculation, writing a summary of the results, and presenting the results to the class. The write-up should be in a professional style[1] and should discuss the organization of the code and verification tests. There is no final exam.

Grading Policy

Problem Sets	60%
Project	40%
Total	100%

Computer Languages

This course emphasizes algorithms, not programming. Indeed, the trend in scientific programming is away from the exclusive use of any one language. You are therefore welcome to use any computing language you wish for the exercises and project. For simplicity, I chose a single language for all in-class examples and course notes, and that is FORTRAN. The specific compiler I'll use is GFORTRAN 4.6.2; packages for installing GFORTRAN on various platforms are at http://gcc.gnu.org/wiki/GFortran.

Why FORTRAN? First, FORTRAN has a 50-year history, and was the dominant language into the 1990s for scientific computation. Many numerical libraries and codes have been, and continue to be, written in FORTRAN, and any scientist should at least be acquainted with the language. Second, FORTRAN is a relatively simple language, and is hence easy to learn. Compared with C/C++, FORTRAN has a smaller set of intrinsic functions, uses pointers more sparingly, and is less laissez-faire with syntax. Third, FORTRAN is designed to generate efficient, highly optimized code. It therefore remains the language of choice for high-performance applications. Finally, modern FORTRAN has an impressive suite of intrinsic array functions, which allow operations to be expressed in a more natural and data parallel fashion. Incorporating this functionality into, e.g., a C++ code requires a digression into optimized classes for vectors and matrices, and this is too much overhead for the course. Although scripting languages such as PYTHON, when combined with compiled numerical libraries, are popular for many short computational tasks, I did not want to neglect completely the use of a compiled language such as FORTRAN.

[1] A meta-goal for this course is professional development, which means learning best-practices for the presentation of scientific results. Citations should follow standard convention; figures should be publication-quality and well-explained; the prose should be readable and at an appropriate level.