APC/AST 523: Numerical Algorithms for Scientific Computing

Lectures: MW 11-12:20, Fine 314.

Homepage: `http://www.astro.princeton.edu/∼jstone/APC523`

Professors:	**Jim Stone**	**Robert Lupton**
Office:	125 Peyton Hall	134 Peyton Hall
e-mail:	`jstone@astro.princeton.edu`	`rhl@astro.princeton.edu`

Text: None required, but several recommended, including:
Numerical Recipes, by Press, Flannery, Teukolsky, & Vetterling
Introduction to Numerical Analysis, by Neumaier
A First Course in the Numerical Analysis of Differential Equations, by Iserles

Introduction:

The ever-increasing performance of computer hardware is revolutionizing research in the sciences and engineering by enabling numerical solutions to complex problems. However, this revolution is not only being driven by advances in hardware, but also by the development of more accurate, efficient, and reliable algorithms. APC/AST 523 (Numerical Analysis in Scientific Computing) will give students a broad introduction into the basic principles of numerical analysis that are the foundation for modern scientific computing.

Computational science is inherently multidisciplinary. In this course fundamental topics in computer science, mathematics, and numerical analysis, as well as the application domains themselves will be covered. While a solid grounding in the fundamental mathematics underlying algorithms is crucial (and will be covered), **this course is meant to be a practical, hands-on introduction to scientific computation.**

Note that the primary focus of this course is *algorithms*. While we will discuss some aspects of software engineering in the first two weeks, the bulk of the course covers topics in numerical analysis. Topics in software engineering are covered in APC524 (Software Engineering for Scientific Computing), which is taught every fall in alternate succession to this course.

Prerequisites

The level of the course is aimed at beginning graduate (or advanced undergraduate) students in the physical sciences, engineering, and mathematical sciences. A solid background in mathematics is required, including linear algebra, calculus, and preferably differential equations (Princeton courses which cover this material are MATH 240, MATH 241, and MATH 246). Taking this course at the advanced undergraduate level will require permission of the instructor. No astronomy background is required; the physical systems which motivate the computational methods will be discussed in the lectures.

No formal coursework in computer science is required, however previous programming experience **is a must**. Students should know at least one compiled computer language, preferably either C/C++ or FORTRAN. Scripting languages, such as Python are also suitable for much of the course work. However, in the term projects where performance may be an issue, students must be experienced enough with Python to be able to bind modules written in compiled languages (using, e.g., Swig) when necessary. The advantages and disadvantages of different languages will be discussed in the course (including using packages such as MatLab for scientific computation!).

A working knowledge of the UNIX/LINUX operating system is required. Students who have never used UNIX/LINUX, but are "computer literate", i.e. who have considerable experience in programming computers using other operating systems, should not have trouble learning UNIX/LINUX as part of this course. *However, students who have never programmed before will probably have difficulty with this course; such students must speak to the instructor before taking this course for credit.*

Students will have access to a variety of HPC systems managed by the Princeton Institute for Computational Science and Engineering (PICSciE) including both LINUX and GPU clusters. No previous experience on parallel systems is required. For more information about research computing at Princeton in general, see `http://www.princeton.edu/researchcomputing`

Topics

The course will begin with a brief discussion of selected topics from computer science, such as basic computer architecture, structured programming, performance optimization, and parallel programming for both distributed and shared memory systems (using MPI and OpenMP respectively). This material will be presented in more of a tutorial style, motivated by the philosophy that it is impossible to understand how to implement algorithms which are efficient on modern processors without some knowledge of how such processors work. Discussion will be brief since these topics are covered in more detail in APC524 (Software Engineering for Scientific Computing).

The course will then cover basic topics in numerical analysis, such as arithmetic with finite-precision numbers, truncation error, and convergence and stability of algorithms.

The bulk of the course will focus on an overview of a wide range of numerical methods to solve scientific problems. This includes numerical methods for the solution of systems of linear and nonlinear equations, and ordinary and partial differential equations. Since modern scientific instruments (such as astronomical detectors) generate terabytes of data, numerical methods for the analysis and modeling of large data sets will also be discussed. In each case, examples (usually drawn from astrophysics) will be used to motivate the methods.

Tentative Schedule

- **Week 1:** Introduction; history and modern examples. Programming languages. Structured programming, principles of software engineering. Performance profiling and optimization.

- **Week 2:** Basics of computer architecture, including parallel processors. Programming on parallel systems: OpenMP and MPI. Introduction to Numerical Analysis: stability, convergence, consistency. Arithmetic with finite precision numbers.

- **Week 3:** Numerical linear algebra. LAPACK, NAGlib, etc. Sparse matrix solvers.
- **Week 4:** ODEs I: initial value problems. RK methods, adaptive stepsizes, higher-order methods.
- **Week 5:** Analysis of large data sets. Minimization. FFTs.
- **Week 6:** ODEs II: boundary value problems. Shooting methods, relaxation methods. Application: stellar structure.
- **Week 7**: Example applications from guest lecturers.
- **Week 8:** N-body methods: Stability and accuracy of time-integration methods, symplectic methods. Force evaluation for large-N. GRAPE systems. Barnes-Hut and KD tree algorithms.
- **Week 9**: PDEs I: solving elliptic equations. ADI, multigrid. Application: solving Poisson equation for self-gravity.
- **Week 10**: PDEs II: solving parabolic equations. Stability and implicit methods.
- **Week 11**: PDEs III: solving hyperbolic equations. Stability of FTCS. Lax-Wendroff. Systems of hyperbolic conservation laws. Euler equations. Smooth particle hydrodynamics (SPH).
- **Week 12**: PDEs III: hyperbolic equations (cont.). Shock capturing. Godunov schemes. Application: gas dynamics in one-dimension.

Grading

There will be no exams in this course. The grade will be determined by a set of homework assignments, each of which will require some programming. Most of the course credit will associated with a term project, which will also involve substantial programming, and will substitute for the final exam. A list of suitable projects will be distributed later in the semester, however students can also suggest their own projects related to their own research (for example, parallelizing and optimizing an application code). Each student's project must be approved by the instructors.

Students are discouraged (but not forbidden!) from auditing this course: the subject matter can really only be learned by writing and running code.