

APC523/AST523: Scientific Computation in Astrophysics

Lectures: TuTh 1:30-2:50, Jadwin A07.

Homepage: <http://www.astro.princeton.edu/~jstone/AST523>

Professors: **Jim Stone**

Office: 125 Peyton Hall

e-mail: jstone@astro.princeton.edu

Robert Lupton

134 Peyton Hall

rlh@astro.princeton.edu

Text: None required, but several recommended, including:

Computer Architecture: A Quantitative Approach, by Hennessey & Patterson

Introduction to Numerical Analysis, by Neumaier

Numerical Recipes, by Press, Flannery, Teukolsky, & Vetterling

Computer Simulation Using Particles, by Hockney & Eastwood

An extensive set of lecture notes with further references will be handed out in class.

Introduction:

The ever-increasing performance of computer hardware and improvements to the accuracy of numerical algorithms are revolutionizing scientific research in many disciplines, but perhaps none more so than astronomy. APC523/AST523 (Scientific Computation in Astrophysics) will give students a broad introduction into the most important computational techniques being employed in modern research in astrophysics. Computational science is inherently multidisciplinary; in this course fundamental topics in Computer Science, Mathematics, Numerical Analysis, and Astrophysics will be covered. The underlying theme will be focused on developing the tools and skills necessary for solving application problems in astrophysics and other physical sciences. Thus, the course is suitable for students in disciplines other than Astrophysics who would like an introduction into modern techniques of Scientific Computation.

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.

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 goal is to show that algorithms that work are always built on sound mathematical principles. These concepts will be demonstrated through a discussion of the most successful numerical methods for the solution of systems of linear and nonlinear equations, and ODEs using examples drawn from astrophysics.

Since modern astronomical detectors generate terabytes of data, the analysis and modeling of large data sets is of prime importance. The emerging challenges in astronomical data

analysis, and the techniques that are necessary to meet these challenges, will be covered next using examples drawn directly from modern sky surveys.

The bulk of the course will focus on the application of numerical methods to solve problems in astrophysics. The use of ODE solvers, combined with solvers for systems of linear and nonlinear equations, to study stellar structure and evolution will be discussed. Numerical methods for the N-body problem, including direct N-body methods, particle-mesh methods, and tree codes for force evaluation, as well as symplectic methods for very long time integration of the few-body problem will be described. Finally, basic methods for astrophysical gas dynamics, and the combination of these methods with N-body codes, to study cosmology and structure formation in the early Universe, will be introduced.

The value added of this course is its breadth, and its demonstration of concepts through actual applications drawn from astrophysics. We will discuss only those topics in Computer Science and Numerical Analysis that are most important for actual applications. Students will therefore see first-hand the relationship between the application science (astrophysics), the mathematics, the numerical analysis, and the computer science issues that underlie the solution of problems using scientific computing. The course is meant to be a practical, hands-on introduction to scientific computation.

Tentative Schedule;

A partial list of topics to be covered in the course, and a tentative schedule for a 12-week term, is given below:

- **Week 1:** Introduction. History and modern examples. Introduction to computer architecture: hierarchical memory, vectorization and pipelines. Parallel architectures, linux clusters. Models for parallel programming: MPI versus OpenMP. Scientific Visualization, with examples drawn from sm
- **Week 2:** Structured programming, principles of software engineering. Languages: object-oriented languages, scripting languages. Performance and parallelization of algorithms implemented in different languages. Performance monitoring and profiling. Introduction to Numerical Analysis: stability, convergence, consistency. Arithmetic with finite precision numbers.
- **Week 3:** Numerical linear algebra. LAPACK, NAGlib, etc. Thomas algorithm for tri-diagonal systems. Sparse matrix solvers. Application: level population calculations for modeling spectra. Solving systems of nonlinear equations, root finding methods. Application: equilibrium chemistry in ISM; nuclear reaction networks.
- **Week 4:** Analysis of astronomical data: image formation, noise, sky estimation, object detection, convolutions and FFTs, photometry and model fitting.
- **Week 5:** ODEs I: initial value problems. RK methods, adaptive stepsizes, higher-order methods. Bulirsch-Stoer. Application: structure of steady C-type shocks. Stiff systems, implicit methods and stability. Application: time-dependent chemistry in ISM.

- **Week 6:** ODEs II: boundary value problems. Shooting methods, relaxation methods. Application: stellar structure, with detailed discussion of Henyey scheme, EZ-code.
- **Week 7:** N-body I: review of stellar dynamics and equations of motion. Stability and accuracy of time-integration methods: leap-frog, Hermite methods, symplectic methods. Application: long-time integration of planetary orbits.
- **Week 8:** N-body II: force evaluation for large- N . GRAPE systems. Barnes-Hut tree algorithm. KD-tree algorithm. Fast multipole methods. Applications: dynamics of globular clusters, merging galaxies.
- **Week 9:** PDEs I: solving elliptic equations. ADI, multigrid. Application: solving Poisson equation for self-gravity. N-body III: particle-mesh methods. PIC codes. Applications: plasma dynamics, bar instability in disk galaxies.
- **Week 10:** PDEs II: solving parabolic and hyperbolic equations. Stability of FTCS. Lax-Wendroff. Systems of hyperbolic conservation laws. Euler equations. Smooth particle hydrodynamics (SPH).
- **Week 11:** PDEs III: hyperbolic equations (cont.). Shock capturing. Godunov schemes. Application: gas dynamics in one-dimension: evolution of SNR.
- **Week 12:** Hybrid codes combining more than one algorithm. Application: cosmology (gas dynamics plus dark matter). Current state-of-the-art in various application areas, and future prospects.

It is vital that students appreciate the physics as well as the mathematics of the astrophysical applications discussed in the course. Whenever a new application is introduced, the physics of the problem will be reviewed, and the governing equations will be derived from basic principles. Students will be given references and suggested readings to explore the both the physical and mathematical background of each problem in more detail.

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.

Due to the rapid pace and broad subject matter, this course is primarily intended for graduate students in the physical sciences, PACM, and Engineering. However, it can also be taken for credit at the advanced undergraduate level with permission of the instructor. Some of the homework assignments, and the term projects, will be less challenging for undergraduate students.

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

Prerequisites

No astronomy background is required; the astrophysical systems which motivate the computational methods will be discussed in enough detail to give students a solid understanding of why the solutions are important and interesting. 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 formal coursework in Computer Science is required, however previous programming experience is a must. Students must know at least one compiled computer language, preferably either C or FORTRAN. Scripting languages, such as PERL or Python will be discussed in the course, but are not suitable for projects requiring, e.g., parallel algorithms on distributed memory systems. Students are welcome to use object oriented languages such as C++ or Java as long as their limitations in certain environments are understood (again, usually related to programming on parallel systems). The advantages and disadvantages of different languages will be discussed in the course (including using packages such as IDL 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 be given access to parallel systems on campus for completing homework and the term project. Students will be given a handout describing usage policies, and details of how to access these systems, in the lectures. No previous experience on parallel systems will be required.