Matrix Computations - Math 221 - Fall 2009 - T Th 12:30 - 2pm in 81 Evans Hall
Professor: Jim Demmel, 831 Evans Hall and 564 Soda Hall, 643-5386, demmel@cs.berkeley.edu.
Class Home Page: www.cs.berkeley.edu/~demmel/ma221

Recommended Texts:


Prerequisites: Good knowledge of linear algebra, programming experience, numerical sophistication at level of Ma 128ab or equivalent.

Other Reading

1. Numerical Linear Algebra, L. N. Trefether and D. Bau, SIAM, 1997. Also aimed at a first year graduate audience, but has a more pure mathematical flavor than the main text.


8. Perturbation Theory for Linear Operators, T. Kato, Prentice Hall. Comprehensive account of analytic perturbation theory for eigenvalues and eigenvectors; chapter 2 covers the finite dimensional case, which is the subject of this course.


16. **MGNet**, or Multigrid Net, is a web page (www.mgnet.org) with pointers to books, software, and tutorial material.

**Computer Resources:** Since this is a graduate course, I will assume that students have access to computer accounts already; if this is not the case please contact me. Also, I will assume that students have access to Matlab (or its public domain look-alike, Octave). Again, please contact me if this is not the case.

**Grading:** Grades will be based on weekly homework, as well as programs and a final project. Homework or programs turned in late will receive only half credit. You may work together on homework, but it should be turned in individually. It is all right to discuss programs with one another, but work should be done individually. Final projects should be done individually.

Programs will be of two kinds, Fortran, C or C++ (your choice), and Matlab. Assignments to be written in Fortran/C/C++ will use subroutines from libraries like LAPACK or CLAPACK. Matlab software related to the course is available on the class homepage. You may find it convenient to do mixed language programming (e.g. calling a Fortran routine from C or C++, or a C routine from Matlab); this is up to you.

Final project proposals are due Oct 15 in class: design a project related to the course material that will require about the same effort as 4 homework assignments, and write up a 1-page summary of what you plan to do, why it is related to numerical linear algebra, and how it relates to any larger scientific goals you have. The final project writeup (5-10 pages) is due Dec 14.

**Syllabus:** The standard problems whose numerical solution we will study are systems of linear equations, least squares problems, eigenvalue problems, and singular value problems. Techniques for dense and sparse problems will be covered; it is impossible to cover these areas comprehensively, but students should still come to appreciate many state-of-the-art techniques and recognize when to consider applying them. We will also learn basic principles applicable to a variety of numerical problems, and apply them to the three standard problems. These principles include (1) matrix factorizations, (2) perturbation theory and condition numbers (3) effects of roundoff error on algorithms, including properties of floating point arithmetic (4) analyzing the speed of an algorithm, (5) choosing the best (fastest and/or most accurate) algorithm based on the mathematical structure of your problem, and (6) engineering numerical software.
There will be one big change from my last offering of this course: the computers on which everyone runs these (and other) algorithms are changing dramatically. The first change is that they are all becoming parallel: the number of parallel processors, even in your laptop, is doubling every few years, so all algorithms that you want to run faster need to run in parallel. The second change is that the majority of the cost of running many algorithms has shifted from doing arithmetic (adds, subtracts, multiplies, etc) to moving data within the computer system (for example, between main memory and cache, or between parallel processors connected over a network). This means that when we design and study algorithms, we need to understand how much data they move, not just how much arithmetic they do. These changes, which effect all computing (not just matrix computations!), will be reflected in the course. In particular, a number of new algorithms have been invented recently that move much less data than old algorithms (while still giving the right answer, as far as we know, though there are many open problems).

In addition to discussing established solution techniques, other open problems will also be presented.