

Math/CS 365, Spring 2004

Scientific Computation

Daniel Kaplan, Macalester College

Course Policies and Outline

Purpose

This course provides an intermediate level exposure to many of the important concepts, algorithms, and techniques of scientific computing. This course is still developing as we attempt to find the right balance of theory versus application, and classical topics versus trendy ones. As computer hardware and software have advanced, topics that were once essential practicalities have become specialized details of implementation. Difficult algorithms have been packaged in a way that makes them trivial to use. Problems whose size alone made them research topics can now be easily handled by standard hardware designed for game-playing and multimedia. Ironically, the fantastic developments in hardware and software have made theory a more important component of a course such as this so that users of modern software aren't misled by the sometimes obsolete techniques taught in theoretical courses.

The most important bit of theory for this course is linear algebra: matrices, solution of linear equations, transformations of vector spaces. Many of the important techniques of this course rely on concepts from linear algebra. Linear algebra provides the technical apparatus that allows us to translate concepts in that are simple and intuitive in one-dimensional functions into the less familiar world of multi-dimensional mathematics.

Topics

A tentative schedule of topics for the course is given below. Each topic will take one to two weeks. There will be a homework assignment associated with each topic. There will also be a set of short projects due roughly two weeks apart.

Topic 1 Numbers and numerical operators. Sensitivity and conditioning. Basic algorithms of optimization and solution with one unknown. Reading: Heath: Ch 1. Kaplan: Ch 14 & 15. Project 1: Floating Point Calculator.

Topic 2 Systems of linear equations. Reading: Heath: Ch 2. Kaplan: Ch 16. Press: Ch 2.

Topic 3 Least squares. Singular value decomposition. Reading: Heath: Ch 3. Press: Ch 2. Project 2: Image compression via singular values.

Topic 4 Interpolation. Reading: Heath: Ch 7. Kaplan: Ch 15. Press: Ch 3. Project 3: Image alignment (e.g., for panoramic photos)

Topic 5 Nonlinear equations. Reading: Heath Ch 5. Press: Ch 9.

Topic 6 Optimization. Reading: Heath Ch 6. Press: Ch 10.

Topic 7 Random numbers and simulations. Covariance and classification. Bootstrapping and ANOVA. Reading: Heath Ch 13. Kaplan: handout of draft chapter. Project 4: Signature verification.

Topic 8 Numerical integration and differentiation. Reading: Heath Ch 8.

Topic 9 Fourier Transforms. Reading: Heath Ch 12. Project 5: Speech recognition.

Topic 10 Tentatively: Dynamical programming and hidden markov models. Reading: Handouts.

Substantial deviations from this schedule are possible, depending on feedback from the students about areas of difficulty and areas of interest.

Grading

Your grade in the course will be based primarily on homework assignments, short project work, and a term project, as indicated in the following table. Homework assignments typically focus on theory or writing short programs that perform a very narrow task. Short projects involve slightly more open-ended applied problems described in detail in the project assignment. The term project will be a creative elaboration or extension of one of the short projects.

Homeworks	35%
Short projects	25%
Term Project	15%
Class Participation	10%
Final Exam	15%

The final exam will be open-book and open-computer.

Homework assignments and weekly projects are due by 11 PM on the indicated day. (That's right — almost midnight.) They may be marked down if they are late.

We will occasionally discuss homework and project solutions in class using student papers as examples of both successes and errors.

Text

The main text for this course is

Michael T. Heath, *Scientific Computing: An Introductory Survey* 2nd edition.

Exercises and problems will be drawn from the book. Another very useful book is

William H. Press *et al.*, *Numerical Recipes: The Art of Scientific Computing*, 2nd edition.

There are various flavors of this book (in C, Fortran, ...). It is available on line at http://www.numerical-recipes.com/nronline_switcher.html.

Finally, those of you who do not already know Matlab may want to get a copy of the textbook for CS 121. You can either borrow a copy from a student who has already had CS 121, or you can purchase it from whatever bookseller you prefer.

Daniel T. Kaplan, *Introduction to Scientific Computation and Programming*

MATLAB

The programming system we will use in this class is MATLAB, a commercial package that integrates programming and graphics and provides a vast array of operators for scientific programming.

You will need to learn Matlab early in the semester. It is a relatively simple package, but no package is so simple that you can learn it without putting in some effort.

MATLAB has been installed on the Windows and Linux machines in the Math/CS computer lab. A student version of MATLAB, that works on either Windows, Mac, or LINUX computers and costs about \$100, can be ordered via the bookstore and can be used without time limitation.

Problem Sets

There will be frequent problem sets, of three types:

Review Questions Heath has many short review questions at the end of each chapter. You should be prepared to answer any of these in class on the date indicated in the syllabus, but you do not need to write them up in advance.

Exercises These are theoretical or mathematical problems. You should do all of the assigned problems and be prepared to discuss them in class. Choose two of the problems and write them up beautifully; this is what you will hand in. "Beautifully" means that a reader who

doesn't know how to solve the problem should be able to read your solution and understand it and that the solution should be neatly and professionally done.

Computer Problems These are problems, often related to the exercises, that involve using the computer and, sometimes, writing a program or series of programs. Like the exercises, you should do all of them and choose two to write up beautifully.

In addition to handing in your two beautifully written exercises or computer problems, you should keep a portfolio of them and your projects that you will hand in at the end of the semester. You will get a small amount extra credit for choosing problems that are difficult or unpopular.

Each of the beautifully written exercises or computer problems should be written *entirely by yourself*, working alone. For the other problems, feel free to work with other students, doing the problems jointly.

I recognize that often you will work through all of the problems before selecting the two to write up. This is fine, even if you are working with other students. But your write-up should be done entirely independently, starting from scratch. See the next section for more details.

Working Together

One of the main resources you have available in your work is your fellow students. Studies have found that students who work together in science classes consistently do better than students who work alone. I wish, therefore, to provide an environment that encourages you to work with others. Feel free to ask other students for advice, criticism, suggestions, and so on.

One of the reasons that scientific programming has advanced so far since its inception about 50 years ago is that the packaging of algorithms as computer programs allows people to build on the work of others. (Systems such as MATLAB or MATHEMATICA are marvelous examples of the advantages of such packaging.) A scientific programming course that encourages you to write everything from scratch is performing a disservice. Nonetheless, a course such as this exists in a somewhat artificial environment: you will often be asked to write programs implementing well established algorithms and even, sometimes, to write a program that duplicates the output of another program. The purpose of this is to help you learn the details of algorithms as well as to give you experience programming in a context where the desired result is clear.

Working together and making use of other people's work are good things, but in some cases they border on cheating. To help you to distinguish between working together and cheating, here are some rules:

- All write-ups submitted for a grade should be written by you in your own words. When paraphrasing or quoting

another person's ideas be sure to provide an appropriate citation.

- Do not present other people's work as if it were your own. If you copy a significant fragment of computer code, either from a published source or another student, you should note the origin of the copied material both as a comment in your computer code and, when appropriate, in the write-up of your work.
- Asking another student to read, comment on, or proof-read your work is perfectly acceptable.
- On examinations, either in-class or take-home, there

should be absolutely no communication with others.

- Part of authoring a computer program involves debugging the program. Receiving extensive help in debugging is the same thing as receiving extensive help in authoring. It should be appropriately acknowledged in your write-up.
- Feel free to talk to me about whether an act is appropriate "working together" or inappropriate "copying." If you initiate such a conversation from your own initiative, the worst outcome is that you will be asked to repeat that part of the homework assignment or laboratory.