UNIVERSITY OF CALIFORNIA
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

**Prof. R. Fateman**

**Spring 2002**

# General Course Information: CS 282 Algebraic Algorithms

## Personnel

| | |
|---|---|
| **Instructor:** | Richard J. Fateman, 789 Soda |
| **Office Hours:** | immediately after class, WF 11-12 |
| | but I am in Soda Hall most days: send email to be sure. |
| **Phone, email:** | 642-1879, fateman@cs.berkeley.edu |
| **Teaching Assistants:** | none |
| **Information:** | See URL `http://www-inst.eecs.berkeley.edu/~cs282` (the class home page). |

## Catalog Description

Prerequisites: Some knowledge of modern algebra, some programming experience, interest in applied mathematics and/or computational aspects of pure mathematics. Suggested courses: Math 113 and CS 164 or graduate standing in mathematics or computer science, or permission of instructor. )

Theory and construction of symbolic algebraic computer programs. Polynomial arithmetic, GCD, factorization. Integration of elementary functions. Analytic approximation. Simplification. Design of computer systems and languages for symbolic mathematics. Programming Exercises. (3 units)

## Scheduling

Meeting time/place: 9:30-11 Wed, Friday, 320 Soda Hall. Some but not all of the lectures will be available on power-point slides. There will be notes handed out in class, and posted on the class web page.

## Grades, course assignments, homeworks, exam schedule

Your grade in this course will depend on three components:

1. **Problem Sets:** There will be 5 sets, plus a take-home final. Some will involve computer usage. (problem sets determine 50% of grade)

2. **Project:** We've attached a list of possible projects. Some projects require significant programming background; some require some familiarity with moderately advanced mathematics, (algebra, analysis, applications) and some require both. A few projects are primarily reading and analysis. In the past, some of the most interesting projects have emerged from student suggestions. In addition to a written report, an oral presentation will be required. In the past a number of CS 282 projects have grown to masters' projects and/or published papers. (Your project determines 40% of your grade)

3. **Class participation:** Class preparation: I hope to give you a chance to prepare and lecture on a topic during this semester. Other types of participation include reading the assignments, preparation for discussion, etc. (10% of grade)

## Computers:

You may have to spend some time becoming familiar with the computer facilities if you have no experience with our local systems.

Graduate students in EECS, CS, and maybe Mathematics will probably be able to use their normal accounts. Others can request 'named' EECS accounts by following the instructions posted on the walls of every Instructional lab. Students enrolled via TeleBears, should be already pre-approved for a 'named' account. Not enrolled? See me.

The principal pieces of software that most people will use are the computer algebra systems Macsyma, Mathematica, and Maple, although others (Axiom, Reduce, Pari, MuPAD, Macaulay) may be useful. Additionally, programs in Lisp, Scheme, C, C++ or Java may be of interest. (Full-source computer algebra systems in Lisp or Scheme include Mock-MMA, Weyl, JACAL, Maxima [a public-domain version of Macsyma].)

There appear to be licensed copies of Mathematica for all EECS machines, Some copies of Macsyma for UNIX and Windows NT, and a variety of systems running Maple. Some of these we expect you will access remotely from other systems supporting X-windows.

You may use computers that you own, since some of these programs run quite adequately on the current crop of high-end laptops, Pentium PC or Macintosh PowerPCs machines.

## Texts

**There is no required text,** but there are a number of possible reference books. This first list is given in approximate order of usefulness, but they are rather different and are not directly comparable

- Michael J. Wester, *Computer algebra systems : a practical guide,* Wiley, 1999. 436 p. ISBN:0471983535

- Joachim von Zur Gathen and Jurgen Gerhard. *Modern Computer Algebra* Cambridge University Press, 1999 753 p. ISBN:0521641764 This is a major work including illustrations, history, and detailed theoretical analysis of abstract computer algorithms for a variety of mathematical tasks. Unfortunately the selection of topics is not completely to our taste.
- K. O. Geddes, S. R. Czapor, G. Labahn: *Algorithms for Computer Algebra* Kluwer Academic, 1992. 585 p. ISBN:0792392590 A large compendium explaining in (sometimes grueling) detail essentially how the internal algorithms of a computer algebra system (in particular Maple) might work.
- B. Buchberger, G. E. Collins, and R. Loos (eds): *Computer Algebra*, (Springer-Verlag) 1st or 2nd edition, 1984 (paperback)
- Chee K. Yap, *Fundamental Problems in Algorithmic Algebra,* Oxford University Press, 1999. ISBN 0-19-512516-9. Yap has made a reasonably complete draft of this book available on-line free for educational use. `http://cs.nyu.edu/yap/book/` Yap is concerned with what he refers to as the fundamental computational problem of algebra, namely solving the polynomial equation $p(z) = 0$, and variations of it.
- J. Davenport, Y. Siret, E.Tournier: *Computer Algebra, Systems and Algorithms for Algebraic Computation,* Academic Press, 1993. This is slightly preferable to the earlier 1988 edition, however, it is out of print.
- Richard E. Zippel *Effective polynomial computation* Kluwer Academic Publishers, 1993. ISBN:0792393759 Zippel has pioneered work in sparse polynomial GCD algorithms.
- David A. Cox, David A., John Litte, Donal O'Shea *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra* Springer, 1997 536 p. (2nd ed) ISBN:0387946802 A standard mathematical approach to some of our material.
- D. E. Knuth, The Art of Computing Programming, vol 2: Seminumerical Algorithms Addison Wesley. A classic source for certain algorithms on polynomials, updated in the second edition.

Additional material consisting of notes, reprints from journals and conference proceeding, and programs will be distributed.

For work on your assignments and projects, and for additional information, there will also be volumes on reserve in the Engineering Library (Bechtel Center). These will include volumes of conference proceedings, manuals for computer systems (Axiom, Reduce, Macsyma, Maple, Mathematica), etc. Some of this material will also be available in the Soda Hall Reading Room.

Unfortunately, there is no entirely satisfactory text for this course. The book by von zur Gathen is encyclopedic in its scope, but takes a substantially different view of the subject matter. The Geddes book is large, extraordinarily detailed, and very expensive. The Davenport book is ideosyncratic, sometimes quite inaccurate with regard to systems issues, and the authors occasionally (without warning) assume the reader has a thorough knowledge of some advanced topic in algebra, number theory, or analysis. On the other hand, in most places it is far more readable, less formal and generally less intimidating than von zur Gathen.

There are a number of other texts covering some of the material in this course. Of particular interest may be these, listed alphabetically by author:

- A. Akritas: *Elements of Computer Algebra with Applications* (Wiley, 1989).
- E. Bach, J. Shallit: *Algorithmic Number Theory - Volume 1, Efficient Algorithms,* MIT Press, Cambridge, Mass., 1996.
- M. Bronstein: *Symbolic Integration I - Transcendental Functions,* Springer-Verlag, Berlin, 1997.
- P. Brgisser, M. Clausen, M. A. Shokrollahi: *Algebraic Complexity Theory,* Springer-Verlag, Berlin, 1997.
- M. Pohst, H. Zassenhaus: Algorithmic Algebraic Number Theory, Cambridge University Press, 1997.
- M. Mignotte, D. Stefanescu: *Polynomials - An Algorithmic Approach* Springer-Verlag, Singapore, 1999.
- B. Mishra: *Algorithmic Algebra* (Springer-Verlag, 1993).
- W. V. Vasconcelos: *Computational Methods in Commutative Algebra and Algebraic Geometry,* Springer-Verlag, Berlin, 1998.
- F. Winkler: *Polynomial Algorithms in Computer Algebra*, (Springer, 1996)

## Conference Proceedings

Much of the new work in this area appears first in conference proceedings. (see Wester's book for descriptions) Looking up these acronyms in Melvyl will get you to our library's collections of proceedings: ISSAC, AAECC, IMACS, DISCO, CADE, CASC, COCOA, AISC, MEGA, OpenMath workshops.

There are also these (paper) publications: *SIGSAM Bulletin Journal of Symbolic Computation.*

And these web sites `http://www.symbolicnet.org` CAIN: (European) Computer Algebra Information Network `http://www.riaca.win.tue.nl/CAN/`

There is a tension in the list of texts above concerning the choice of the central organizing principle for the subject: mathematical (a course in algebra with an emphasis on constructive methods), complexity (a course in complexity of algorithms), application programming (a course in problem solving by computer), systems building (constructing a practical toolkit for symbolic mathematical problem solving).

My own central emphasis is actually on the last of these topics, since without it there is no figure of merit for work in this field.

## Extended Course Description

Text: Material from journals, conference proceedings, on-line sources.

Selections from the following topics:

I- Symbolic Mathematical Computer Systems

http://www.riaca.win.tue.nl/CAN Introduction to symbolic mathematical systems (examples from Macsyma, Maple, Mathematica, Theorist, etc.) Discussion of interactive problem solving; differences between symbolic and numeric systems, approaches to languages design.

Simplification: the needs of users vs. the system designers. Display of expressions in two dimensions (mathematical typesetting).

Data structures: modularity, selectable data structures, programming from abstractions (data, mathematics), choice of implementation languages.

II- Symbolic Representations and Algorithms

1. Integers, Fractions, Approximations
2. Algebra of polynomials, rational functions and power series
3. Normal forms and algebraic representations: Sparse, Dense, Blackbox
4. Homomorphisms and Chinese remainder algorithms
5. Newton's iteration and the Hensel constructions
6. Efficient Calculation of polynomial GCD's
7. Solving systems of equations: resultants
8. Gröbner bases for polynomial ideals
9. Factorization of polynomials
10. Integration of rational polynomials
11. Introduction to the Risch integration algorithm

III- Applications (selections from these topics)

Sparse (symbolic) matrix calculations

Summation in closed form. Operator calculus. Solving recurrences. Functional Equations.

Solution of ordinary differential equations in closed form. Manipulation of programs as data: symbolic program execution. Generation of programs in other languages (e.g. Fortran).

Perturbation expansions.

The use of parallel computation in a symbolic setting.

Applications from engineering, physics, mathematics.

## CS282 Projects — Spring, 2002

Some projects will involve programming with existing algebraic manipulation systems. Others may be written "from scratch". Maple, Mathematica, Macsyma, Reduce, MuPad each have a "user" programming language. For some projects you may find programming in the "implementation" language preferable to the "user" language. (this is Lisp for Macsyma and Reduce, C for several other systems). Most systems have hooks for programming in C or Fortran, as well, and this may be appropriate for efficient coding for Maple or Mathematica.

1. Design an algorithm to locate and determine the nature of singularities in algebraic expressions. Find a useful representation for computing with Riemann sheets, branch cuts, etc. (ref: Harlan Seymour, Adam Dingle.)

2. Consider the Schwartz-Christoffel transform as a symbolic method; compare to the numerical version (Trefethen).

3. Interval arithmetic has been implemented in several computer algebra systems (and also as extensions to conventional systems). Explore the literature and implement good routines for polynomial evaluation, equation solving (Newton iteration, bisection), elementary transcendental functions, etc.

4. Implement a system for computing with continued fractions and lazy evaluation. (Gosper's CF paper),

5. Explore the options for computing with piecewise-defined functions, via signum, absolute value, etc. For example, integrate wrt $x$ from 0 to 1, the function "if $x > 0.5$ then 0 else 1".

6. Continue work with the implementation of a calculus of symbolic divided differences — useful for error and stability analysis. (Kahan/Fateman)

7. Refine the interface for a particular numeric/symbolic/graphical interface. In addition to the usual suspects, consider Mathcad, Matlab, PDEase. How can symbolic solution of integral or differential equations be hooked up to a symbolic system? Can web-based (html, java, javascript,...) access be made plausible? (cf. WebEQN, TechExplorer etc). Some prior expertise in graphics (etc) programming would be required. This is not the time to jump in as a newbie.

8. Figure out how to "assumptions" or "provisos" or "guards" to an existing computer algebra system in a constructive way, e.g. when simplifying $x^0$ to 1, add a proviso that $x \neq 0$. Provisos emerge from "knowledge" in reference books.

9. Carefully analyze and benchmark one or more significant algorithms as implemented in one or more systems. Identify performance problems of space and/or time, and remedy if possible. For example replace slow Lisp code with faster Lisp or C code. (constant factor time/space improvement); find a better algorithm or data structure (better than constant factor improvement). Polynomial factorization is a plausible one. Coming up with a world-beating Gröbner basis algorithm would be an excellent objective.

10. Review and improve upon the facilities for exact or approximate solution of a well-defined class of problems (e.g. integral equations, perturbed algebraic equations, trigonometric identities problems).

11. Provide solved problems and exercises for an algebra system, using classical material in engineering, math, physics, etc. for demonstration and education purposes.

12. Implement a simple interactive and inherently symbolic "language" for a problem domain such as applications in circuit analysis (simple LRC stuff), mechanical linkages, or design problems in other engineering disciplines.

13. Extend and generalize 2-D parsing techniques for recognition of mathematical expressions such as those in an integral table. (R. Anderson, N. Mitchell; T. Tokuyasu Lisp).

14. Reproduce in a modern symbolic language the ODE approximation techniques of A. C. Norman (Proc. ACM 72) and/or Van de Riet (ABC Algol) , Corliss/Yang ACM Trans. Math Software. This sets up Taylor series for initial and boundary value problems. Compare to purely numerical codes.

15. Study the complexity of programs produced by "algorithm differentiation".

16. Compare pattern matching features in Mathematica and Reduce: Study the complexity of (say) a Lisp-based pattern matcher with the semantics of Mathematica, and see how it runs compared to C.

17. Implement (yet another) but better heuristic program for solution of systems of algebraic equations. (Davenport, recent papers on Gröbner reduction).

18. Collect and implement algorithms for efficient representation and manipulation of symbolic linear algebra: sparse / diagonal / block matrices; canonical forms, eigenvalues, etc. (Symbolically perturbed matrices, perhaps). Canonical forms, (Smith, Jordan, Kronecker, Rational, Schur, LUP, QR, SVD). Consider the difference between sparse matrices and dense matrices filled with "sparse" polynomials, and the differences between exact and approximate entries. In the limiting case of all numeric data, there is substantial code.d

19. Symbolic systems have often pushed the limits of large-memory computing systems. Can/should such systems be file-based rather than address-space intensive? Celestial mechanics programs have usually taken this file-based approach. Investigate the system FORM.

20. Explore the user model implicit in systems such as Derive, Theorist, Milo, Mathscribe and the PowerMac Graphing Calculator. How can these be made more appealing for serious algebraic manipulation?

21. Examine techniques for parallel execution of algebraic manipulation tasks. These can include

(a) basic polynomial multiplication, division
(b) solution of algebraic systems
(c) polynomial factorization
(d) Gröbner basis computations
(e) FFT-related computations on polynomials
(f) modular/CRA methods
(g) polyalgorithms for (e.g., GCD, algsys)
(h) matching, rule-transformation
(i) determinant, resultant
(j) transformation of lists to arrays to files and back: alternatives to the huge virtual memory model.
(k) Linear algebra (Linpack for symbols?)

Several models of parallelism may be appropriate.

22. Extend existing work on computation with asymptotic series, perturbation analysis. For example see book by M. van Dyke, or work on Taylor-Green vortex.

23. Study representation and algorithms for expansion in orthogonal polynomials (e.g. Chebyshev); Fourier series; (refs: de Bruijn ? Fox?)

24. Write a program which determines, to the extent possible, whether an expression in one real variable is a constant, (or for what ranges of that variable it is constant). Using this program, do a good job of the zero-equivalence algorithm of Richardson. (compare to program by Joel Moses.)

25. Table lookup of ODEs: how can we characterized known solutions so they can be correlated with questions? perhaps related to the next problem.

26. Collect and evaluate (poke holes in...) techniques for ODE solving in various systems. Extend existing techniques.

27. Examine techniques for determining (heuristically), the radius of convergence of Taylor series based on some finite set of coefficients. Also program usual infinite series tests (ratio, root, d'Alembert, comparison).

28. Implement and test a collection of "black-box representation" facilities for basic arithmetic and more advanced algorithms. (E. Kaltofen)

29. Propose a project based on your own interests.

Student Survey

```
Last (family) name:

First name (and/or preferred form of address):

Electronic mail address if any:

year:(G/Undergrad)

Major department:

Credit (for grade) or audit?
(Note: department policy discourages auditing: better to register
for credit S/F).

Previous experience with symbolic mathematical computing
(e.g. have you used Macsyma, Maple, Mathematica, Derive, Reduce, Axiom, Theorist)?

Previous programming experience
(e.g. Pascal, C, C++, Fortran, Lisp, Prolog, Java, etc. ) ...how much?

Previous mathematics courses
(Have you taken undergrad / graduate courses in algebra? complex variables?
applied math/ engineering?)

Special interests (continue on reverse):
```