Examples of Plausible Projects for CS279

Prof. W. Kahan
University of Calif. at Berkeley

Notes for this course

A polished collection of notes for this course would have wide circulation and obvious value. Several hundred copies of the 1970's notes, now hopelessly out of date and available only on microfilm from NTIS, are believed to be in circulation

Proofs

Not all the proclamations in class will be proved; some of the proofs are too tricky for classroom use. However, they ought to be recorded somewhere, partly for archival reasons, and partly because tricky proofs have a large capture-cross section for mistakes. Another kind of proof is a (mostly) mechanized error analysis for a program that computes one of the elementary transcendental functions; see below.

Fast and Accurate Elementary Functions for IEEE 754

The math library distributed with 4.3 BSD Berkeley UNIX has been adopted on a wide variety of machines that conform to the floating-point standard. Its demands upon machine resources are very modest. A better library, faster and more accurate, could be built out of large tables, and would be preferable for machines with long pipelines and large bandwidths to memory. A team project. See also the topic of Proofs above.

New Computer Architectures

New machines appear (and sometimes disappear) so rapidly that keeping informed about them is a major effort. How well do they perform floating-point? Have they any obvious flaws? Each new family of machines (Data-Flow, Systolic, Hypercube, ...) provides a project opportunity.

PARANOIA

This program was written to discover at run-time how the computer's floating-point arithmetic behaves; what is its radix, range, precision, rounding procedures, ... ? The program is very ill documented, disorganized, and needs some extension; for instance, it does not do a good job in treating machines whose registers are wider than the variables normally stored in memory. A rewrite is overdue.

Binary-Decimal Conversion

Nobody knows how to test conversion programs, though very good conversion algorithms can be found in chapter 7 of J. T. Coonen's thesis. The outstanding problem is coping with all the diverse formats supported by diverse languages. A library of portable conversion programs (AtoF, ScanF, FtoA, ...) in C for UNIX would be very valuable regardless of how slow it was if it could serve as an impeccable accuracy standard.

Testing Conformity to IEEE 754

Although a tape of test cases exists, distributed by the University of California at Berkeley, nobody has provided yet a test of conformity to the floating-point standard as thorough as is really needed. A team project.

Confusing Benchmarks

Whetstones, Livermore Loops and many other commonly run benchmarks lend themselves to obfuscation; on the other hand, the LINPACK benchmarks collated by J. Dongarra are very informative but require tailoring to each unconventional machine's architecture before they can be relied upon. We need a wider range of benchmarks with diagnostic capabilities that would indicate the reasons for poor performance rather than merely average incommensurables. A team project.

Testing Elementary Functions

Z-S. Alex Liu has produced a marvelous program to test the accuracy of the most important elementary transcendental functions over the most important parts of their ranges. This program needs further work to cover more functions and to test argument reduction more extensively.

A Robust Equation Solver

Solving one real equation $f(z) = 0$ for its root z by numerical means alone is a common task provided for in the better math software libraries; but those libraries do not take full account of the abilities of machines that conform to IEEE 754/854. A better program, similar to what runs now inside HP-17, 18 and 28C calculators, would tolerate much poorer initial guesses and cope better with singularities.

Two not-so-Elementary Functions

The Error function and the Gamma function can be computed rather better than has been customary. And if an accurate program to compute the Error function and its inverse ran fast enough, then it would figure in the world's fastest numerical quadrature (integration) program.

Complex Arithmetic

Complex multiplication is surprisingly tricky because of its potential for exceptions that ought to be hidden if irrelevant, avoided if unnecessarily destructive. And the elementary transcendental functions are tricky too.