Chapter 8

Designing numerical libraries in C

The purpose of this chapter is to have a bit of a look "under the hood" to see how a library of routines in C can (and we believe, should) be built up. The philosophy here is to make use of the features of C to make programs more flexible and easier to write (and debug), while not sacrificing too much efficiency. There are other ways of designing numerical libraries, but this has been found to be a useful and flexible way of designing numerical libraries in C.

8.1 Numerical programming in C

Numerical and scientific programming has been traditionally associated with Fortran. Indeed, a great deal of software has been written in Fortran, in spite of its well known defects (lack of good data structures, lack of strong typing, reliance on "GOTO", poor lexical characteristics, clumsy input/output). This has led to the "historical" defense of Fortran: "There is so much already written in Fortran that we have to program in Fortran."

However, more sophisticated algorithms need more sophisticated data structures and more structured programs. Sparse matrix data structures and operations on them are one example of this. C is one of a number of languages that easily support such structuring. As well, C is a very flexible language, especially as regards memory management. While it is often argued that C is "merely a systems programming language", several aspects of C seem to indicate otherwise. For example, C has both single and double precision. Sometimes the argument is made that C is not suitable for numerical programming because single precision numbers are automatically converted to double precision whenever they are passed as arguments or used in expressions. This is no longer true in ANSI C. Even with the older C convention, the main drawbacks are the time spent converting between double and single precision numbers. Operations done entirely in double precision are immune to this inefficiency. It is, in any case, a better state of affairs than not having double or extended precision numbers as is the case with Pascal or the original version of Modula-2. Also, the standard Unix^{TM} mathematics library has not only the standard functions (exp, log, and the trigonometric