

2.3 Numerical analysis

Anyone writing numerical code needs *some* awareness of numerical analysis, to avoid burning up CPU time with a hideously inappropriate and inaccurate algorithm.

By far the most accessible introduction to numerical methods is Numerical Recipes¹ [nr], which comes in different editions, containing code in C, Fortran and Fortran 90. Use the second edition: the first has a significant number of bugs.

A large part of the book's popularity stems from the fact that its authors are scientists rather than numerical analysts, so that they are more concerned with producing reliable results than with a point of view which sometimes appears to see efficiency, unshakable robustness and algorithmic elegance as ends in themselves. For further discussion, and some caveats, see *Section 2.3.1*.

My advice is to use *Numerical Recipes* for your numerical programming until it runs out of steam on your particular problem. Follow the references in there, or look at the booklist² in the on-line Numerical Analysis FAQ. Also (perhaps unexpectedly), I suggest you take a look at the Usenet newsgroup `comp.lang.fortran`, even if you don't actually use Fortran. This is one of those happy few Usenet newsgroups with a high signal-to-noise ratio, and listening in on this can be profitable when the conversation turns to general numerical analysis. A similar resource is the JISCMail `comp-fortran-90`³ list.

To support more specialised numerical computing, refer to the libraries section below, *Section 4.2*.

2.3.1 Numerical Recipes

Numerical Recipes⁴ [nr] does not claim to be a numerical analysis textbook, and it makes a point of noting that its authors are (astro-)physicists and engineers rather than analysts, and so share the motivations and impatience of the book's intended audience. The declared premise of the NR authors is that you will come to grief one way or the other if you use numerical routines you do not understand. They attempt to give you enough mathematical detail that you understand the routines they present, in enough depth that you can diagnose problems when they occur, and make more sophisticated choices about replacements when the NR routines run out of steam. Problems *will* occur because the routines are not written to be bullet-proof, and if you use them thoughtlessly you can break them without much difficulty. Also, the routines will likely prove inadequate if you have a very demanding application which needs a more efficient or more specialised routine than the ones available here.

That is, the NR library is not filled with magic bullets, and if you try to use its contents as such, the only thing you'll shoot is your foot. However, NAG and SLATEC don't supply magic bullets either (though 'any sufficiently advanced library is indistinguishable from magic to the unsophisticated programmer', as Arthur C Clarke didn't quite say). You might use the NR

¹<http://www.nr.com>

²<http://www.mathcom.com/corpdire/techinfo.mdir/scifaq/q165.html>

³<http://www.jiscmail.ac.uk/lists/COMP-FORTRAN-90.html>

⁴<http://www.nr.com>

routines successfully for years, but if and when they fail you, you'll use the more sophisticated substitute all the better because you'll understand *why* the simpler original was inadequate.

It is a consequence of the book's aims that the routines will not necessarily be the most intricately and obscurely efficient. Efficiency matters a lot if you are running some huge hydro code taking months of Cray time, but I would claim that if your code takes less than a week of wall-time to run, then the efficiency gains from using opaque library routines is simply not worth the cost in debugging time. No matter how robustly the library routine is written, you will be able to abuse it – you will manage to break it somehow – and when that happens your only recourse is to work through pages of Fortran IV trying to find the overflowing total, or the case you hadn't realised was marginal.

If you need very high efficiency, then take a course on numerical analysis and spend time understanding the subtleties of the library routines. Otherwise, read NR carefully (there are sometimes important caveats in the text), customise the routines to your problem, cross check the results you obtain, and keep alert. If you have an aversion to documentation, use NAG or another library: using the NR routines as a black box is a numerical recipe for disaster.

Despite NR's popularity, it has its critics. These are typically that the NR routines do not use the most efficient modern algorithms, and that they sometimes go wrong in borderline situations. While this may be true (and will be more true of the first edition than the second), it is largely a consequence of NR's declared intention of being accessible and intelligible. Without making the point explicit, as far as I can see, the NR authors privilege intelligibility over high efficiency, and practicality over unabusable robustness; I don't believe it's entirely fair, therefore, to criticise them for not being ultra-efficient and bulletproof. There is a collection of criticisms of the books by W Van Snyder at JPL, at <http://math.jpl.nasa.gov/nr/>, along with some suggestions for alternatives; there is a rebuttal from the NR authors at <http://www.nr.com/bug-rebutt.html>.

From the NR home pages, you can browse and print out chapters from the books, but you can't download the source code. Do remember that the NR code is copyrighted and *not* free; this may affect your ability to redistribute code which uses it.