Reviews

EDITED BY JOSEPH KONHAUSER
Department of Mathematics, Macalester College, St. Paul, MN 55015

Numerical Recipes, The Art of Scientific Computing. By W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. Cambridge University Press, 1986. xi + 817 pp.

LAWRENCE F. SHAMPINE
Mathematics Department, Southern Methodist University, Dallas, TX 75275

Numerical analysis is concerned with the numerical solution of mathematical problems. This simple description already provides a reason why the subject is hard to teach. It is necessary to understand the mathematical task before taking up its numerical solution. Obviously, then, the mathematical training of the students limits the tasks that might be taken up in a course. It is usual in an introductory course to study a few mathematical problems that are sufficiently basic to be familiar to most students. Tasks that require more mathematical training or sophistication are left to topics courses. Unfortunately, students may have a need to solve mathematical problems that they understand poorly, if at all. Besides this difficulty with the diversity of tasks, there is a difficulty with the diversity of methods used to attack them. Although there are basic principles common to the numerical treatment of problems as different as the computation of eigenvalues and the numerical solution of ordinary differential equations (ODEs), the methods brought to bear on one task may have little in common with those required for another. As a consequence, an introductory course is likely to be perceived as a collection of unrelated problems and methods. To a considerable extent this perception is correct, for the student sees a body of theory only when treating a topic in depth.

The variety of reasons for studying numerical analysis complicates the teaching of the subject. Scientists and engineers often want to learn no more than how to prepare their problems for a library routine and how to interpret the results. Despite the readability of the documentation, the broad applicability of the codes, the reliability of the methods, and the protection afforded by the best mathematical software, this is simply not enough. The scientist is quite likely to encounter problems that can be solved with library codes only if he understands what the codes are doing, and why. The problems that arise in practice are often too complicated to be solved directly with library codes. To devise ways to solve them, an understanding of the principles of numerical analysis is essential. The tasks are mathematical, and in the main the methods for their solution are, too. Numerical analysis is, then, a branch of applied mathematics and so an appropriate subject for a mathematician to study. Computer scientists are often required to take a course in numerical computation as a cultural matter. Perhaps this sounds odd, but it is entirely appropriate. Numerical computation is an important part of computing, and it has surprisingly little in common with mainstream computer science.

American Math Monthly
vol 94 #9 11/87

Corresponding to the variety of potential students is a variety of backgrounds. Scientists and engineers may have a shallow understanding of some of the mathematics necessary for a survey course. The mathematician is likely to have an adequate training in this regard, although he may not be familiar with why one wants to solve such problems, but is not nearly so likely as a scientist or engineer to be familiar with programming and the use of computers. The computer scientist is well equipped for a study of numerical analysis in many respects, but may not be comfortable with the mathematics of the continuous processes that are the principal object of numerical analysis—the reason for the word "analysis" in the name.

Proper treatment of a topic in numerical analysis involves the reason for solving the problem, the mathematics of the problem and a way to approximate its solution, a programming language, and certain aspects of computation such as computer arithmetic, data manipulation, and documentation. Because the potential students have quite different backgrounds, and interests, in these matters, it is difficult to provide a well-rounded treatment of a topic that is palatable to all. Several ways of responding to the difficulties are seen. One is to skip over some aspects of a complete treatment. This is dangerous because it encourages "amateurism." If the complexity of sound numerical analysis is not made clear, the student is encouraged to think that it is an easy matter to produce code with the qualities of mathematical software enumerated earlier. The result is unsound and inferior work. Another way to reduce the material to be developed is to assume that all the students have the same background. A well-rounded treatment is possible, even for the full range of students, if the number of topics and methods taken up is kept quite small. The disadvantage of this approach is that the teacher must be very sure that the few tasks taken up and, especially, the few methods for their solution that are treated are the important ones.

An item of mathematical software is complex and involves art as well as science. There are few publications that seek to explain in detail even one kind of mathematical software, and they are appropriate only to topics courses. Some introductory texts direct students to "black boxes" in libraries and concentrate on how to use the software and interpret the results. Others include state-of-the-art codes, but discuss only the general principles of the codes; they work with "grey boxes." A third approach is to explain in detail an implementation that is of high quality but that does not include all of the art and the subtleties of the best codes. Each of these approaches involves substantial codes, so the author adopting one must deal with the issue of making the codes accessible to the students. Nowadays numerical analysis texts are expected to contain codes. Providing codes and explaining them is so difficult that some authors choose to provide only fragments of code, perhaps in a pseudo-language, that make specific points about how software is constructed. By providing mere translations of mathematical algorithms into some computer language, it is possible to satisfy the expectation of codes without having to face up to the difficulties.

These general observations explain why popular numerical analysis texts present the subject in very different ways. Let us now try to fit *Numerical Recipes* into the spectrum of possibilities. The book is encyclopedic in scope. Only a small fraction

of the volume could be covered in a single course. The breadth of topics requires a team, and this book has four authors. In only one instance did I note a reference in the book to a paper by one of the authors. From this, the affiliations of the authors, and the tenor of their remarks, I take them to be people who do a lot of numerical computing, as opposed to people interested in developing new methods or in developing mathematical software. A complete review of the book would also require a team. My own background includes training as a numerical analyst, two decades' experience in industry and universities, many papers published about the numerical solution of ODEs, and considerable experience with the writing and testing of mathematical software. Despite all this, I am not in a position to comment about large portions of this book in any way. Whether the quality of chapters I studied is representative, I do not know.

The chapter on the initial value problem for ODEs treats the most common methods. Little is said about the preparation of problems for their solution with the codes provided. The treatment of Runge-Kutta methods is limited to the classical four stage, fourth-order formula with local error estimate by doubling and local extrapolation. The authors seem pretty enthusiastic about the method. It is remarkable that they do not even mention the embedded formulas that are the basis of the codes found in all the well-known libraries. Most attention is devoted to extrapolation of the modified midpoint rule. The authors do not explain what is "modified," and so Gragg's important contribution to the method is never acknowledged. The scheme they describe is obsolete; current practice is described in the nice survey of Deuflhard [1]. The authors extol the virtues of rational extrapolation in this chapter, but they use polynomial extrapolation in the quadrature chapter. The authors have a light-hearted style that is very agreeable, but it sometimes falls flat. For example, in connection with predictor-corrector methods they say, "Let us here dispose of two silly ideas." The trouble is that their explanations are silly. Plainly they neither understand nor appreciate the powerful variable order, variable step multistep codes that are so widely used. This is odd in view of their treatment of stiff problems; the authors direct the reader to Gear's implementation of the backward differentiation formulas, which is just such a code. This chapter describes numerical methods for ODEs from the viewpoint of 1970. If the authors had consulted an expert in the subject or read one of the good survey articles available, I think they would have assessed the methods rather differently and presented more modern versions of the methods.

The quadrature chapter focuses on extrapolation methods. Examination of popular libraries, collections like QUADPACK, and the published advice of experts, such as the appropriate chapter in [2], shows the great importance of adaptive methods. No reference is made to these methods. The chapter on linear equations highlights Gauss-Jordan elimination with complete pivoting. Not many experts would think it a good idea to invert a matrix every time a linear system is solved. The authors do provide a Crout reduction code with partial pivoting. In connection with residual correction, the authors say just that it is a good idea to calculate the residual in double precision (if available); this is inadequate. Indeed, the TURBO Pascal version does use single precision without a warning of the implications. The

Jacobi method is highlighted as a way to compute the eigenvalues of symmetric matrices. This is despite a description of more modern methods and codes along the lines of EISPACK. The authors favor these less popular methods because they are robust and "simple." Presenting "simple" methods is of some pedagogical value, but that does not seem to be one of the aims of the book. I would rather present the methods that people really use rather than those that are easy to explain to the student.

The book is subtitled, *The Art of Scientific Computing*. This suggests interesting examples and illustrative computations. However, I noticed only one interesting example, the computation of spheroidal harmonics as solutions of boundary value problems for ODEs. I expected that a companion volume labeled "Example Book" would contain some substantial examples with a discussion of their significance, a discussion of how to prepare the problem for solution with the codes provided, and an interpretation of the results. The examples are of the most straightforward kind with practically no discussion at all. They do provide a partial check that the codes are performing properly. My guess is that few people would find an example book and its accompanying diskette to be worth the cost.

In order to present a great many topics, the authors devote little space to any one method. The treatment of each method is adequate to appreciate the basic idea, and references are provided to the literature. Generally the treatment is too superficial for a textbook. Further, the lack of examples, illustrative computations, and exercises make the book unsuitable for the classroom. The presentation of the codes is quite good in every respect. I have tried out a number of the codes on an IBM PC AT with two FORTRAN compilers and one Pascal compiler. There were no problems, with the single exception that one FORTRAN compiler complained, correctly, about the absence of EXTERNAL statements when the name of a subroutine was passed through several levels of subroutine calls. The codes themselves are of better than average quality for a survey book. However, they are far from being mathematical software. Even in the cases of a high quality code taken from the literature, the documentation expected of mathematical software is absent. Although the authors repeatedly express their distaste for "black boxes," they do refer the reader to such codes in a number of instances. With few exceptions, the reader would be well advised to turn to reputable sources of mathematical software rather than to the codes given in this book. The advice offered does not always correspond to the methods advocated by leading practitioners and implemented in leading libraries, c.f. [2].

In my view, *Numerical Recipes* is best described as a reference book providing a quick introduction to the solution of a mathematical task and a lead-in to more thorough treatments.

REFERENCES

1. P. Deuflhard, Recent progress in extrapolation methods for ordinary differential equations, *SIAM Review*, 27 (1985) 505–535.
2. *Sources and Development of Mathematical Software*, W. R. Cowell, editor, Prentice-Hall, Englewood Cliffs, NJ, 1984.