29 [3].—M. LUNELLI, Editor, *Una Biblioteca de Sottoprogrammi dell'Algebra Lineare*, Franco Angeli, Editore, Milano, 1972, 429 pp. Price: Lit. 12000.—.

A collection of Fortran programs derived from *The Algol Collection in Linear Algebra* by Wilkinson and Reinsch.* The first half of the book introduces and discusses the methods and relevant perturbation theory—all in Italian.

B. P.

The following review has been reprinted from SIAM Review, Vol. 14, No. 4, October 1972, p. 658, with the permission of Arthur Wouk, editor of Book Reviews of *SIAM Review*.

30 [3].—J. H. WILKINSON & C. REINSCH, *Handbook for Automatic Computation*. Vol. II, *Linear Algebra*, Springer-Verlag, New York, 1971, ix + 439 pp., 24 cm. Price $20.80.

Those with a strong interest in numerical linear algebra will already be familiar with some of the algorithms given in this book. In this review I shall try to address an imaginary SIAM member who is not very interested in the subject but who wishes to know when something important has happened, which topics are receiving most attention, and which of them are dead.

This important reference book presents 82 procedures written in an official subset of the language ALGOL 60 to perform a variety of well-defined tasks in solving linear systems of equations or in finding eigenvalues and eigenvectors. With each algorithm there is a brief discussion of its scope, the relevant theory, special features, numerical properties and test results. This collection represents continuous efforts by acknowledged experts over more than ten years. The algorithms have been pre-published individually in Numerische Mathematik and thus have been subject to public scrutiny and usage. In a real sense this anthology defines the state of the art in this domain, although Wilkinson hastens to say that he is not claiming that these programs are the last work on the subject.

It is only proper to hand out bouquets to the authors for creating this landmark and for setting such high standards of performance and documentation. One of the pleasant aspects of the effort has been the friendly cooperation on an international scale, a contrast with the intensely individual and competitive atmosphere in the world of mathematics.

The appearance of this book raises a number of interesting questions.

Why has it taken nearly fifteen years to implement decent programs in a subject which was finished off by the beginning of the century and has become a standard part of all undergraduate training in the physical sciences and engineering? It is one thing to learn that you cannot just say Newton's method when the subject of polynomial zeros is raised. It is quite another matter to provide a zero finder which will cope with most eventualities, never lie, and not be too clumsy. When a mathematician

* See following review.

asks to be told the deepest result or the key theorem in numerical linear algebra I feel that he is imposing the wrong framework. The subject is the design and analysis of algorithms which must use noisy arithmetic. This is part of computer science. It is a new field and we are still not sure how to teach it or how to talk about algorithms.

Consider, for example, Rutishauser's chapter presenting an implementation for Jacobi's method for diagonalizing a real symmetric matrix by finding principal axes in successive plane sections. He reorganizes the traditional formulas for effecting a plane rotation so that the new diagonal elements differ from the old ones by a multiple of the tangent of the angle of rotation. This permits an elegant improvement over the traditional version: each diagonal element is updated just once in each "sweep" through the matrix. This is valuable algorithmically but trivial mathematically. This ruse had escaped previous investigations including an intensive one by Murray, Goldstine and von Neumann [2]. Admittedly they were considering only so-called fixed-point calculations, but the trick is no less applicable in that case. Perhaps the most useful function of error analysis is its power to make us seek and judge alternative expressions, all equivalent in exact arithmetic, of the quantities of interest to us. When, if ever, would you write $(x - 0.5) - 0.5$ instead of $x - 1.0$?

In a sense this book does finish off an important part of numerical linear algebra. Frankly, I cannot imagine major improvements to many of these programs. The advance from very little, in 1956, to fast, compact, reliable routines in 1971 is so great that subsequent improvements must look pale in comparison. Of course there is work for a few specialists. We do not really know how to scale our equations. We do not have a rapid, well justified test for neglecting subdiagonal elements of Hessenberg matrices. We do not have standard routines for effecting a posteriori error analyses. Many important tasks still lack reliable algorithms. However, the pioneering days are over. You can no longer present a bad method and rest assured that no one else's routine works either.

My reader may have heard occasional reports on the war between ALGOL and FORTRAN, the two most popular languages for writing numerical programs. The first volume in this Handbook Series was in two parts. Rutishauser gave a lucid *Description of Algol 60* and Grau et al. discussed the *Translation of Algol* (into machine language) and presented a compiler to do it. With this underpinning, the algorithms do constitute unambiguous descriptions of computing processes. They are abstract in the sense that the basic operations of arithmetic and the set of numbers in which variables can take values are not prescribed and may be given different interpretations by different machines. In particular, they may be interpreted with exact arithmetic on the set of extended real numbers. This is the context in which the theories of the various methods are studied.

FORTRAN would not have been so elegant for the purposes of definition, but it is important to say that for nearly all numerical routines a standard ASA FORTRAN version also provides unambiguous definition of the algorithm.

Those who do not have access to good ALGOL compilers will be pleased to hear of the NATS project. Stanford University and the University of Texas at Austin are collaborating with Argonne National Laboratory to produce FORTRAN versions of the eigenvalue programs in this book, helpful documentation and elaborate testing. The procedure for obtaining codes may be obtained from NATS Project,

Applied Mathematics Division, Argonne National Laboratory, Argonne, Illinois 60439. FORTRAN translations of some of these handbook algorithms are already available in a rental package from IMSL, Suite 510, 6200 Hillcroft, Houston, Texas.

Of all those with an interest in some numerical method, only a small number wish to carry their investigations as far as a reliable program in FORTRAN or ALGOL. Of these only a few worry about the suitability of the particular arithmetic systems which will in practice interpret the algorithms. For this review it suffices to stress the fact that this level does exist, that sometimes hardware characteristics should determine portions of the algorithm. It surprises many people to learn that the execution of *if* $a \neq b$ *then* $d = c/(b - a)$ can cause the computation to be aborted because of an attempted division by zero! This can occur if the test for a zero divisor is not identical to a test for equality. It is as well to remember such possibilities when there is a discussion about proving that algorithms do what they claim to do.

For certain simple tasks, such as solving a quadratic or iterating on residuals, very precise statements could be made about certain algorithms provided that some quantities could be computed in twice the precision of the rest of the computation. Intentionally, ALGOL was not made to describe such calculations. Consequently the comments and notes in which this information is conveyed form an important part of these procedures.

A great many matrix programs have been written over the last fifteen years. Hopefully many of these will now enjoy a well earned retirement and the new algorithms will become standard equipment. The only other collection of comparable quality of which I know is by Dekker and Hoffman in Holland [1]. They present fewer routines, a brief general discussion for each section, and comments opposite each page of ALGOL.

The engineering, or at least the nonmathematical aspects of the study of algorithms become apparent when we compare the Dekker-Hoffman procedure (D) with the Wilkinson-Martin procedure (W) for reducing an equilibrated real matrix A to Hessenberg form H ($h_{i,j} = 0$, $i > j + 1$). Even though this is a straightforward task, the routines are surprisingly different. (1) D reduces only the full matrix while W can reduce a principal submatrix. (2) D begins by computing $||A||_\infty$ whereas W does not use this quantity. (3) To find the maximal subdiagonal element in column $j - 1$, W finds the minimal i ($\geq j$) such that $|a_{i,j-1}| = \max|a_{k,j-1}|$, $k \geq j$, and then compares i with j. On the other hand, D finds $s = \max(\text{tol}, \max|a_{k,j-1}|, k > j)$ and then either annihilates these $a_{k,j-1}$ if $s = \text{tol}$ or else compares s with $|a_{j,j-1}|$. Here $\text{tol} = ||A||_\infty \times \epsilon$, where ϵ is the largest number such that the computed value of $1 + \epsilon$ is 1. The former case is flagged for purposes of later transformations by setting to zero the index which otherwise records an interchange at this step. Thus D effectively says that rounding errors of up to a value of tol will be committed at some places in the reduction, and it is reasonable to apply such a tolerance uniformly throughout. W, on the other hand, does not introduce such machine dependence where it is not necessary. The only tests are on (machine) zero. (4) Another difference between the procedures is that W uses subscripts such as $j - 1$ quite freely, even inside *for* loops, whereas D is careful to define $j1 = j - 1$ and uses $j1$ inside loops. Of course, an intelligent ALGOL compiler will spot that a quantity $j - 1$ may only need to be computed once for a whole loop and act accordingly. D's approach uses more variables but does not have to rely on much optimization in the compiler.

How are we to learn to distinguish between important and unimportant programming details if such things are not discussed somewhere? How this should be done I am not quite sure. The study of Volumes I and II of the *Handbook for Automatic Computation* might be a good way to begin.

B. P.

1. T. J. DEKKER AND W. HOFFMAN, *Algol 60 Procedures in Numerical Algebra, Parts I and II*, Mathematisch Centrum, Amsterdam, Holland, 1968.
2. H. H. GOLDSTINE, F. J. MURRAY AND J. VON NEUMANN, "The Jacobi method for real symmetric matrices," *J. Assoc. Comput. Mach.*, v. 6, 1959, pp. 59–96.

31 [4].—C. WILLIAM GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1971, xvii + 253 pp., 24 cm. Price $12.95.

"I have tried to gather together methods, mathematics and implementations and to provide guidelines for their use on problems." The author has succeeded admirably in this effort. With a careful selection of illustrative examples, he presents clear discussions of the reasons that various algorithms perform as they do. In each case, he begins with a concrete description of the numerical method and ends with a definite mathematical analysis of the procedure. The reader is masterfully guided through the regions of stability for each method. He explains how to choose an appropriate method (step size and order) for solving the initial value problem; and in particular, discusses the treatment of stiff equations, gives a brief development for handling singular perturbation or singular implicit equations, and shows how to solve for certain parameters that may appear as unknowns in a given system of differential equations. The author only describes those techniques that he has found to be of the most utility; in this way the book is kept slim and its subject matter alive. Three FORTRAN subroutines for the numerical solution of differential equations are listed. As indicated in the preface, the author hoped to repay his debt to society by setting his "thoughts on paper so that the useful among them might benefit others." In this connection, the reviewer believes that Gear's debt has been repaid many times.

E. I.

32 [7].—ALFRED H. MORRIS, JR., *Table of the Riemann Zeta Function for Integer Arguments*, Naval Weapons Laboratory, Dahlgren, Virginia, ms. of 3 pp. +2 computer sheets deposited in the UMT file.

The Riemann zeta function, $\zeta(n)$, is herein tabulated to 70D for $n = 2(1)90$. Confidence in the complete reliability of the tabular entries is inspired by the accompanying description of the details of the underlying calculations, which were carried to 80D.

This carefully prepared tabulation constitutes a valuable supplement to the corresponding 50D table of Lienard [1] and the 41S table of $\zeta(x) - 1$ of McLellan [2].

J. W. W.