sult of a systematic approach to new methods and new approximations, should be an indispensable reference work for all those engaged in scientific computation.

J. W. W.

1. Cecil Hastings, Jr., Jeanne T. Hayward & James P. Wong, Jr., *Approximations for Digital Computers*, Princeton Univ. Press, Princeton, N. J., 1955. (See *MTAC*, v. 9, 1955, pp. 121–123, RMT **56.**)
2. A. Fletcher, J. C. P. Miller, L. Rosenhead & L. J. Comrie, *An Index of Mathematical Tables*, second edition, Addison-Wesley Publishing Co., Inc., Reading, Mass., 1962. (See *Math. Comp.*, v. 17, 1963, pp. 302–303, RMT **33.**)
3. L. A. Lyusternik, O. A. Chervonenkis & A. R. Yanpol'skii, *Handbook for Computing Elementary Functions*, Pergamon Press, New York, 1965. (See *Math. Comp.*, v. 20, 1966, pp. 452–453, RMT **64.**)
4. Charles L. Lawson, *Bibliography of Recent Publications in Approximation Theory with Emphasis on Computer Applications*, Technical Memorandum No. 201, Jet Propulsion Laboratory, Pasadena, Calif., 16 August 1968.

17[2.10].—Arne P. Olson, "Gaussian quadratures for $\int_1^\infty \exp(-x)f(x)dx/x^m$ and $\int_1^\infty g(x)dx/x^m$," tables appearing in the microfiche section of this issue.

Abscissas and weights of N-point Gaussian quadrature formulas for the integrals in the title are given in Table II to 16 significant figures for $N = 2(1)10$, $M = 0(1)10$. Table I contains 24S values of the moments $a_{n,l} = \int_1^\infty \exp(-t)dt/t^{n-l}$ for $n - l = 10(-1)-19$. As a possible application of Table II, the author mentions the evaluation of the series

$$\sum_{k=0}^{\infty} E_m(t + kh) = \int_1^\infty \frac{\exp(-xt)dx}{[1 - \exp(-hx)]x^m} \quad (E_m \text{ the exponential integral}),$$

which occurs in the calculation of neutron collision rates in infinite slab geometry.

W. G.

18[2.45, 12].—Donald E. Knuth, *The Art of Programming*, Vol. I: *Fundamental Algorithms*, Addison-Wesley Publishing Co., Reading, Mass., 1968, xxi + 634 pp., 25 cm. Price $19.50.

Most people think of mathematics as being a very complicated subject. On the contrary, if we define the complexity of a problem as the amount of information required to describe the problem and its solution, it is apparent that mathematics can deal only with simple problems. Further, if we consider the universe of all problems, almost all of them are too complex for solution by mathematical methods, which rely heavily on abstraction and conceptualization, which themselves reflect the limitations of the brain. Thus, while it is reasonable to expect a mathematical solution to the four-color problem, it is not reasonable to expect a mathematical solution to the problem of language translation, which can be stated only in terms of massive amounts of information from grammars and dictionaries, and whose solution will probably require an algorithm of comparable size. The importance of the computer is that it permits the consideration of such irreducibly complex problems.

It is not surprising therefore, that what looks like the most authoritative work on computer programming is entitled "The Art of Computer Programming". Programming, which is concerned with algorithms for solving complex problems, is itself a complex problem, and we would be mistaken if we expected it to have a neat and tidy theory. There is simply too much detailed information associated with computer programs for them to be analyzed completely and axiomatically, particularly by human intelligence. Certain aspects of certain algorithms are amenable to theoretical treatment, but the great bulk of programming decisions are made by various combinations of experience, taste, habit, intuition, philosophy and guesswork.

This book, by one of the best-known workers in the field, is the first volume of a seven-volume set. The complete contents of the set are:

Volume 1: Fundamental Algorithms
 Chapter 1: Basic Concepts
 Chapter 2: Information Structures
Volume 2: Seminumerical Algorithms
 Chapter 3: Random Numbers
 Chapter 4: Arithmetic
Volume 3: Sorting and Searching
 Chapter 5: Sorting Techniques
 Chapter 6: Searching Techniques
Volume 4: Combinational Algorithms
 Chapter 7: Combinatorial Searching
 Chapter 8: Recursion
Volume 5: Syntactic Algorithms
 Chapter 9: Lexical Scanning
 Chapter 10: Parsing Techniques
Volume 6: Theory of Languages
 Chapter 11: Mathematical Linguistics
Volume 7: Compilers
 Chapter 12: Programming Language Translation

The scope of this set is so large that we must regard it as an attempt at 'the' reference work on programming. The provisional schedule is for one volume to be published each year.

Chapter 1 starts by introducing the notion of an algorithm (pp. 1–9), and then continues with the basic mathematics necessary for comparing the efficiencies of algorithms (pp. 10–119). This is concerned mainly with finite and infinite series.

The next section (pp. 120–181) is devoted to a hypothetical computer called MIX which is described in detail, and which the remainder of the book uses to illustrate programming examples. The author's decision to use machine language for most of his examples was not made lightly, and the author defends it persuasively.

It is clear that a knowledge of machine language is necessary in order to describe assemblers or compilers, so a specific language must be introduced. However, it is not clear that there is any advantage in using machine language to specify or discuss algorithms, apart from the ability to determine their efficiency, and even that is getting more difficult with the increased parallelism in today's hardware. The most dangerous aspect of the use of machine language is that it perpetuates the view that 'real' programming is done in machine language, and that those who use Fortran or Snobol do not really know what is going on. This misapprehension is the one that is giving the computing world the most trouble—many otherwise competent programmers believe that their talent for writing clever machine-language code is a virtue rather than a vice. The real truth is that the Fortran or Snobol programmer wishes to have a more powerful grasp of his problem, and his decision represents a desire to ignore details, rather than an inability to learn machine language. It seems difficult to believe that the casual user wanting to find out about data-structures will take the trouble to learn MIX, so some of the expertise contained in this book will remain hidden. It seems unlikely that many instructors will implement a MIX interpreter for use by their classes, but will continue teaching the machine language of whatever machine is accessible to them, and using higher-level languages for discussion of specific algorithms.

The next section (pp. 182–227) introduces some fundamental programming techniques, including subroutines, coroutines, a MIX interpreter written in MIX, and input/output. Of these, the section on coroutines is perhaps the most interesting, concerning itself with routines each of which thinks it is calling the others as subroutines. This is an unfamiliar notion to many programmers and is given clear treatment. This section provides a quite sophisticated introduction to machine-language programming.

Chapter 2 (pp. 228–464), is entitled 'Information Structures' and gives an excellent survey of the sort of data-structures found in programming. Section 2.2 on linear lists (pp. 234–304) describes stacks, queues, dequeues, (double-ended queues!), sequential allocation, linked allocation, circular lists, doubly linked lists, arrays and orthogonal lists. The treatment is well organized and comprehensive; a surprising omission is the gimmick of using the exclusive-or of the right and left pointers to save space in a double-linked list.

Section 2.3 (pp. 305–422) on trees is perhaps the most important section in the book. There are subsections on traversal, binary trees, other representations of trees, mathematical properties of trees, lists and garbage collection. The orientation towards trees rather than lists, the more usual data-structure, is presumably made to allow the treatment to be linked naturally with the theory of graphs and trees. This is probably the first adequate treatment of this area in the literature. The author points out, quite rightly, that such techniques can be used to advantage in many problems, and are not restricted to explicit 'list-processing' languages.

The chapter ends with sections on multi-linked structures (pp. 423–434), dynamic storage allocation (pp. 435–455) and a history and bibliography (pp. 456–464). As the author points out, there is little known about storage allocation, and one is left with a strong feeling that this treatment is far from complete.

It has in the past often been impossible to find books appropriate for course-work in Computer Science beyond the first undergraduate semester or two, except in the more mathematical areas. Nearly all graduate courses have had to be based on papers in journals and the instructor's own work. This book could provide for some time an excellent basis for a course on data-structures, and may serve as additional reading in other courses. The many excellent exercises should provide valuable practice for the student (and the teacher). A large part of the book (pp. 465–606) is devoted to answers to the exercises. These range in difficulty from trivial to unsolved problems of significant research interest.

This is surely a 'must' for those who aim to be experts in the field of programming. The mind boggles at the fact that in a book written with such style and accuracy there can be a reference to section 12.3.4, which, if the length of this volume is any indication, must be fully 3000 pages away.

MALCOLM C. HARRISON

Courant Institute of Mathematical Sciences
New York University
New York, New York 10012

19[3].—JOAN R. WESTLAKE, *A Handbook of Numerical Matrix Inversion and Solution of Linear Equations,* John Wiley & Sons, Inc., New York, 1968, viii + 171 pp., 23 cm. Price $10.95.

This is, as the title indicates, a handbook. A two-page introduction is followed by brief descriptions of the standard direct and iterative methods for a total of 85 pages, with no theory and (at this stage) no evaluation. This is followed by chapters on measures of condition (four pages), measures of error (five pages), scaling (two pages), operational counts (four pages), comments and comparisons (14 pages), including some test results. In the appendix are a glossary (nine pages), a collection of basic theorems (ten pages), and a set of test matrices, and finally a list of references (126 items), a table of symbols, and an index. No programs are given. Each method is briefly but clearly described and the selection is quite reasonable. It should be a useful and convenient reference for the purpose intended.

A. S. H.

20[3].—ANDRÉ KORGANOFF & MONICA PAVEL-PARVU, *Éléments de la Théorie des Matrices Carrées et Rectangles en Analyse Numérique,* Dunod, Paris, 1967, xx + 441 pp., 25 cm. Price 98 F.

This is the second volume in a series entitled "Méthodes de calcul numérique," of which the first, *Algèbre non linéaire,* appeared in 1961 as a collection of papers on the subject, edited by the senior author of this volume. The first volume provides a fairly elementary but rigorous development of methods available at that time for solving nonlinear equations and systems of equations, the most sophisticated chapter being the first on error analysis.

The present volume, by contrast, treats only a limited aspect of the subject, but treats it in considerable depth and at a rather high level of sophistication. Primarily it is concerned with the Moore-Penrose generalized inverse, a subject which, along with still more general "generalized inverses," has recently led to a rapidly expanding literature. Presumably the eigenvalue problem will be the subject of a later volume.

The book is divided into three "parts." Of the three chapters in this part, the first provides a survey of certain notions from functional analysis, and the remaining two continue in a similar vein with the theory of norms as the guiding principle, these having been introduced already in the first chapter.

In Part 2, the first chapter starts out quite generally to discuss, for a given matrix a, the most general solutions of the equations $e_g\, a = a$ and $a\, e_d = a$, thence the most general solutions of $a_g'\, a = e_d$, and $a\, a_d' = e_g$, and proceeds to impose