J. S. can be something of a knockout if his themes get hold of you. And his influence on what followed was (you may say) substantial.

REFERENCES


DAVID WILLIAMS
that of 'integers'. 'Integer arithmetic' is exact, but 'real arithmetic' almost
never is; thus in it, \((5/7) \times (7/5) \neq 1\). For this reason, one cannot test 'real'
_vectors (called 'arrays') for linear dependence on computers. Hence one
cannot be sure in general whether a given matrix is singular, or just has a very
large 'condition number'.

By way of compensation, many computers can be programmed to solve
accurately in seconds 150 simultaneous equations in as many unknowns, at a
cost of a dollar or so—and can find all the eigenvalues and eigenvectors of
typical 80 \times 80 matrices to eight significant decimal digits with equal facility.
Truly enormous advances have been made in understanding computer
arithmetic since von Neumann and his collaborators began to analyze proba­
babilistically how many binary digits (or 'bits') had to be allocated to each 'real'
number, in order to achieve eight digit accuracy in the final result. (In
practice, the answer seems to be from 32 to 64.)

In improving its mathematical tools, Computer Science has received indis­
pensable advice from its older half-sibling, Numerical Analysis, another child
of Mathematics produced by earlier liaisons, but which has kept in much
closer touch with its senior parent.

Due primarily to this advice, many ingenious tools for minimizing and
maximizing functions are also available, as are programs for solving
APPROXIMATELY a large variety of initial and boundary value problems
intractable by the methods of Classical Analysis.

Thanks to the development of libraries of ‘portable’ Fortran software,
usable on many makes of computer, the tools I have mentioned are becoming
easier and easier for us mathematicians to use. Therefore it seems high time
for us to become better informed about how at least some of our computa­
tional problems can be solved automatically.

Each of the three books under review should help to make this information
available to mathematicians. But here their similarity ends: each is aimed at a
different audience, and each has its own special philosophical, mathematical,
and technical flavor.

Henrici's book is written for the “creative student” of any age, especially
for gifted amateurs having a taste for constructive complex analysis as
presented in his two-volume _Applied and computational complex analysis_. His
new book shows by example how one can execute many algorithms of
'constructive analysis' by a programmable pocket computer with an 'assem­
bler' language of about 6 words and having 60 words of storage. One can
factor quartic polynomials, exponentiate and find reciprocals of power series,
compute Bessel functions, and locate zeros of the Riemann zeta function on
the critical line. The creative student of constructive analysis can learn much
about good programming and good mathematics by trying to write and test
programs for other algorithms in a similar style.

Nash, on the other hand, concentrates on algorithms for linear algebra and
function minimization, scrupulously avoiding commitment to a particular
programming language or size of computer. He presents his algorithms in
English, leaving to the student the burden of translating them into debugged
Fortran, Algol, Pascal, APL, or Basic. Although his exposition is very lucid
and informative, it leaves the reader with a curious sense of being a spectator and not a participant.

The LINPACK user's guide is at the opposite extreme. At its core are 129 pages 'listing' 50 subroutines. These are written in standard Fortran and are also available on magnetic tape, hence easily executable at most computing centers. The Guide's main purpose is to document these programs. Written for computing professionals, it has the same sparkle as a manual explaining to expert repair mechanics the workings of an automobile engine.

Yet it is an important and meaty document, giving an authoritative picture of current mathematical software technology. Its programs have been exhaustively tested at a number of computing centers on a variety of machines, and can be certified as optimal (in the present state of the art) for solving many problems of linear algebra. Many millions of dollars of computing and programming time will be saved by using them! (The best direct and iterative methods for solving discretizations of linear elliptic boundary value problems are not included, however.)

Though the three books differ in most respects, they share a weakness: none of them points out the limitations of the methods that it explains. Thus the LINPACK user's guide nowhere mentions the fact that the rank of a general matrix cannot be computed in 'real arithmetic'; Nash does not comment on the possibility that his minimization algorithms fail for some functions (e.g. for \(2x^2 - \pi x - 5 \exp[-(10^3 x)^2]\)). Even Henrici, whose skillfully written programs would surely have delighted Euler or Gauss, fails to observe that the coefficients of his power series are not expressed on a computer as rational numbers, hence cannot easily be recognized as generating functions.

By making allowances for this minor weakness, the curious reader who browses through these books can acquire a realistic picture of the state of numerical mathematics today.

GARRETT BIRKHOFF


At the beginning of the twentieth century, mathematics had been greatly enlarged by the ideas of a number of giants, including Riemann, Cantor, Poincaré and Hilbert. Considerable effort was then expended on understanding and developing the whole new areas of mathematics which had been created. And this plus the general trend toward axiomatization meant that a parochial view largely dominated mathematics. In the last couple of decades, however, that has changed and some of the most exciting developments have come from the interaction, often in unexpected ways, of different parts of