

convex) vector spaces will probably gain very little, as is the case with most highly specialized texts. But the manuscript is very carefully written, rather well organized, and for those interested in or working with Baire-like spaces (in the non-technical sense), it will be very valuable.

## REFERENCES

- [BP] J. Bonet and P. Pérez Carreras, *Barrelled locally convex spaces*, North-Holland Math. Stud., vol. 131, North-Holland, Amsterdam, 1987.
- [DMS] J. Diestel, S. A. Morris, and S. A. Saxon, *Varieties of linear topological spaces*, Trans. Amer. Math. Soc. **172** (1972), 207–230.
- [H] J. Horvath, *Topological vector spaces and distributions*, Addison-Wesley, Reading, MA, 1966.
- [S] H. H. Schaefer, *Topological vector spaces*, Springer-Verlag, Berlin and New York, 1986.
- [V] M. Valdivia, *Topics in locally convex spaces*, North-Holland Math. Stud., vol. 67, North-Holland, Amsterdam, 1982.

H. H. SCHAEFER  
FLORIDA ATLANTIC UNIVERSITY

BULLETIN (New Series) OF THE  
AMERICAN MATHEMATICAL SOCIETY  
Volume 32, Number 3, July 1995  
©1995 American Mathematical Society

*Polynomial and matrix computations*, by Dario Bini and Victor Pan. Birkhäuser, Basel and Boston, MA, 1994, xvi + 415 pp., \$64.50. ISBN 0-8176-3786-9

Since the appearance of the first working computers, the treatment of matrices and polynomials has been a mainstay of Numerical Analysis. For a wide class of problems, like solving systems of linear equations and finding eigenvalues and singular values of matrices, a long development process has led to algorithms that obtain results that are as accurate as data in floating point will warrant. These algorithms also use the time and space of the computer as efficiently as any practical user ever dares to ask. In addition, experience from those computations has influenced the way computers are designed today—to mention only two: the IEEE standard for floating point computation and the LINPACK benchmarks used in wide circles to evaluate performance of computers for scientific computation.

Starting slightly later, but now pursued even more vigorously, researchers in Computing Science have developed a theory of algorithms, for instance the entertaining and thought-provoking elaborations in the monumental work of Knuth [5]. Here the emphasis is more on the very elementary operations expressed in Boolean algebra and on combinatorial questions such as: What is the very smallest circuit that can deliver a prescribed result for given data and in what number of operations? In this setting we talk about space and time complexity.

In recent years there has been a fruitful migration of ideas and tools between these fields.

The book under review has its home in the Computing Science speciality of computational complexity but gives enough leads into numerical algorithms and applications to be readable by the present reviewer, who is an old hand at matrix computation.

One basic idea in the book is to describe and use different representations for a polynomial, then study first how one representation is transformed into another, and then how different manipulations can be done using the representation that is most convenient just for that manipulation.

A polynomial  $p$  of degree  $n$  can be given as an  $n+1$  vector  $c$  of coefficients,

$$p(x) = c_0 + c_1x + c_2x^2 + \cdots + c_nx^n,$$

or as another  $n+1$  vector of its values in a sequence of points (Lagrange interpolation),

$$p(x) = L(x) \sum_{i=0}^n r_i w_i / (x - x_i),$$

where  $L(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$  and  $p(x_i) = r_i$ . (This is the notation used in the book; I would never give out an expression that may evaluate as  $0/0$ , but for  $x = x_i$  we already know the values.) If the points  $x_h$  happen to be the  $K$ th roots of unity ( $x_h = \omega^h$ ,  $\omega^K = 1$ ), the coefficients  $c_k$  can be obtained from the values  $r_h = p(\omega^h)$  by the discrete Fourier transform. Using the fast Fourier transform (FFT), we can jump from one representation to the other. The FFT algorithm evaluates a polynomial of degree  $K = 2^k$  in those  $K$  points using  $K \log_2(K)$  multiplications, much faster than the  $K^2$  multiplications needed in the standard algorithm. This way we can multiply two polynomials by first evaluating both of them on a set of points using the FFT, then multiplying those values together, and at last recovering the coefficients with the inverse FFT.

I hope that this simple example gives a hint of the techniques that this text intends to make familiar. Other representations of a polynomial are the set of zeros, the power sums of the zeros, and we may even in some important cases let a matrix represent its characteristic polynomial.

I recently got a matrix of order  $n = 480$  obtained from the reformulation of a set of five second-degree equations in four variables describing the positioning of a robot arm; see [6]. It had been recast as a polynomial in one variable using the Kronecker substitution (described on page 62 in this book) and then expanded as an eigenvalue problem. It is routine nowadays to find the eigenvalues of a  $480 \times 480$  matrix using standard numerical software. Handling polynomials of this order by other means still demands great care; integer coefficients get huge, and series expansions get enough terms to fill the memory of even a large workstation.

Now and then, the text touches on questions of computer arithmetic. One can look at long numbers as polynomials evaluated at a power of 2; for the binary integer  $[u_{t-1}u_{t-2} \cdots u_0]$  with  $t$  bits we get  $u = \sum_{i=0}^{t-1} u_i 2^i$ . A special consideration is given to modular arithmetic. There, results are always kept in range by subtraction of multiples of the modulus, a specified large prime number. Large numbers are handled by keeping the values for several moduli. Several small numbers can be packed into one long computer word: just evaluate a polynomial with small integer coefficients at a larger power of 2 and see how

the coefficients line up in sequence; this technique is called binary segmentation in the book.

A large part of the book is devoted to matrices, with a special emphasis on dense structured matrices such as Toeplitz (each NW-SE diagonal constant), Hankel (each SW-NE diagonal constant), Vandermonde (values of the  $n$  first powers at  $n$  points), and generalized Hilbert (reciprocals of differences between two sets of  $n$  values). Displacement rank is the natural notion of complexity for these types of matrices and works much like the ordinary rank for standard matrices.

We are told when one matrix algorithm can be expressed as a sequence of others. One of the fancier ways of solving a linear system  $Ax = b$  is to take a vector  $v$  and compute its Krylov sequence,

$$K(A, v, m) = \{v, Av, A^2v, \dots, A^{m-1}v\}$$

in the order,

$$v, Av, A^2[v, Av], A^4[v, Av, A^2v, A^3v], \dots,$$

successively squaring the matrix. Then a linear dependence in this sequence is computed by means of a Padé expansion; the coefficients of this give the minimal polynomial,

$$m(A)v = 0.$$

With a little bit of luck, one gets what is called in the book a Las Vegas randomized algorithm,  $m(\lambda) = c_A(\lambda)$ , the characteristic polynomial with the eigenvalues as roots. We refrain from computing these, since it is not a rational operation, but note that  $c_A(A) = 0$  by the Cayley-Hamilton theorem, and so

$$A^n + c_{n-1}A^{n-1} + \dots + c_0I = 0,$$

yielding the inverse from the sum inside the parentheses in the factorization,

$$A(A^{n-1} + c_{n-1}A^{n-2} + \dots + c_1I) + c_0I = 0,$$

if only  $c_0$ , which is the determinant of  $A$ , is nonzero. Multiplying this inverse with the right-hand side  $b$  yields the solution  $x$ . The complexity of this algorithm on a parallel random access machine (PRAM) is dominated by terms in  $n^2 \log n \log \log n$  and  $P(n) \log^2 n$ , where  $P(n)$  is the cost of multiplying two matrices together. This is claimed to be of the lowest order seen up to now, taking into account that there are algorithms for matrix multiplication of lower order than  $P(n) = n^3$ .

A numerical analyst would never dream of solving a linear system this way. The conventional wisdom is to use Gaussian elimination with pivoting to avoid division by zero. Then we solve a system with a dense matrix in one-third of the time it takes to multiply two dense matrices together. But, admittedly, it demands  $n^3/3$  operations for a full matrix.

I think this contrast illustrates both the strength and weakness of this book. It is not intended for those seeking algorithms for practical problem solving but gives many ideas on how different computational problems hang together. However, I, who have the texts on the numerical side of this subject area (see [8, 3, 7, 4]) as my ready references, soon feel tempted to delve into the computing science texts [1, 5] or hard-core mathematics such as [2].

Let me make some comments on the exposition. I cannot stand the common practice of using acronyms to denote different variants of an algorithm (or anything else for that matter). In this book the authors use abbreviations instead, and though it was unwieldy at first, I somehow got used to it. The polynomial procedure for solving a system that I just described is denoted by

$$(RECUR \cdot SPAN, M \cdot VECTOR, KRYLOV) \succeq MIN \cdot POL.$$

The capitalized words look awkward, but some of it depends on a misuse of  $\text{\TeX}$  mathematics style; changing the style into text yields

$$(RECUR.SPAN, M.VECTOR, KRYLOV) \succeq MIN.POL,$$

which looks slightly better to me. English is not the native tongue of the authors, nor of this reviewer, and it would have been helpful if the text had been scrutinized for language by the publisher. It is difficult for a Russian who never uses articles (either *the* or *a*) to get them in the right place.

#### REFERENCES

- [1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*, M.I.T. Press, Cambridge, MA, and McGraw-Hill, New York, 1989.
- [2] I. Gelfand, M. Kapranov, and A. Zelevinsky, *Discriminants, resultants, and multidimensional determinants*, Birkhäuser, Basel and Boston, MA, 1994.
- [3] J. A. George and J. W. H. Liu, *Computer solution of large sparse positive definite systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
- [4] G. H. Golub and C. F. Van Loan, *Matrix computations* (second ed.), Johns Hopkins Univ. Press, Baltimore, MD, 1989.
- [5] D. E. Knuth, *The art of computer programming II: Seminumerical algorithms* (second ed.), Addison-Wesley, Reading, MA, 1981.
- [6] D. Manocha and J. F. Canny, *Multipolynomial resultant algorithms*, *J. Symbolic Comput.* **15** (1993), 99–122.
- [7] B. N. Parlett, *The symmetric eigenvalue problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
- [8] J. H. Wilkinson, *The algebraic eigenvalue problem*, Clarendon Press, Oxford, 1965.

AXEL RUHE

CHALMERS UNIVERSITY OF TECHNOLOGY AND  
THE UNIVERSITY OF GÖTEBORG

*E-mail address:* ruhe@cs.chalmers.se

*Invariant potential theory in the unit ball of  $\mathbb{C}^n$* , by Manfred Stoll. London Math. Soc. Lecture Note Ser., vol. 199, Cambridge University Press, London and New York, 1994, x + 173 pp., \$29.95. ISBN 0-521-46830-2

To one who first met potential theory almost half a century ago it comes as a surprise that central topics such as “equilibrium distribution” and “energy integral” are never mentioned in this book. Even “capacity” seems to occur in