the consistency of the model. It also has to be remarked that no exact result is known about the distribution of the global maximum of a Gaussian random function. This means that, unless the number of function evaluations is fixed in advance, the question of how to evaluate the error of an approximation to the global optimum cannot be answered satisfactorily.

An outline of the book is as follows. Chapter 1 contains a discussion of the main advantages of the Bayesian approach. Chapter 2 presents a general definition of Bayesian methods of global optimization. Chapter 3 contains an axiomatic justification of the Bayesian approach. In Chapter 4 the Gaussian class of prior random functions is derived from the conditions of homogeneity, independence of partial differences, and continuity of sample functions. Chapter 5 provides the expressions for the one-step approximation of the dynamic programming equations. This chapter also discusses the replacement of the Kolmogorov consistency conditions by the weaker condition of the risk function continuity. Chapter 6 discusses methods to reduce the dimensionality of global optimization problems. In Chapter 7 the Bayesian approach is applied to find local optima of objective functions with noise. In Chapter 8 a number of real-life applications is described. Chapter 9 provides a description of the portable FORTRAN package which is contained in the book.

Although the Bayesian approach to global optimization, in my opinion, did not yet yield efficient algorithms which are fully theoretically justified, this new book shows that the approach is very appealing, and that the approximations work well. Also, the book contains all the relevant theorems, proofs, and computer programs. Hence, although the book is not very clearly written, and contains very many typos (7 in the preface), it can serve well for investigators who want to pursue the approach. In addition, the programs of the methods, which are based on approximations, can be used by practitioners to solve real-life problems.

C. G. E. BOENDER

Econometric Institute
Erasmus University Rotterdam
3000 DR Rotterdam, The Netherlands

1. H. J. Kushner, *A versatile stochastic model of a function of unknown and time varying form*, J. Math. Anal. Appl. **5** (1962), 150–167.

27[33–04, 65D20].—UNITED LABORATORIES, INC., *Mathematical Function Library for Microsoft-FORTRAN*, Wiley, New York, 1989, xvii + 341 pp., 25 $\frac{1}{2}$ cm., loose leaflets in 3-hole-punched binder, including three $5\frac{1}{4}''$ diskettes. Price $295.00.

In 1964 the National Bureau of Standards (now the National Institute of Standards and Technology) issued a massive handbook of formulas, graphs and numerical tables of the elementary mathematical functions and the so-called

higher transcendental functions or special functions of mathematical physics [6]. This N.B.S. Handbook immediately filled, and continues to fill, a tremendous need in scientific work; according to the Science Citation Index (published by the Institute for Scientific Information, Inc., Philadelphia, PA), the current rate at which it is cited in the mathematical and scientific literature is of the order of 1,300 entries per year.

The need for the numerical tables of the elementary functions in the N.B.S. Handbook has now largely disappeared: library routines for generating exponentials, logarithms, and trigonometric and hyperbolic functions are available in all major scientific software libraries as well as being required by the FORTRAN Standard [1]. Also, these routines are incorporated in hand-held calculators designed for scientific calculations. The loose-leaf manual and accompanying diskettes under review, which we shall refer to as the "UL Library", may be regarded as an attempt to replace the numerical tables of the higher transcendental functions supplied in the N.B.S. Handbook by a comprehensive software package. The functions treated include Bessel and related functions, hypergeometric and confluent hypergeometric functions, elliptic functions and integrals, exponential integral and related functions, error function and related functions, Gamma and incomplete Gamma functions, orthogonal polynomials, probability functions and random number generators. This list is not quite isomorphic with the list of functions tabulated in the N.B.S. Handbook; among the omissions are parabolic cylinder functions, Mathieu functions, spheroidal wave functions and the Riemann Zeta function. The UL Library is designed to be used on Personal Computers of IBM type and equipped with a Microsoft FORTRAN 77 Compiler. For efficiency, a numeric co-processor is recommended.[1] The operating precision is IEEE double precision (53 bits in the floating-point mantissa), but the accuracy of the computed function values is generally less, sometimes well below single precision.

The project is an extremely ambitious enterprise, especially as it appears that all of the programs for the library routines have been constructed *ab initio*. For such a project to be completed successfully its authors need to have a thorough knowledge of, and experience in, several areas of classical and numerical analysis, including analytic properties of the higher transcendental functions, asymptotic analysis, approximation theory, and error and stability analyses. How well have the present authors (who are nameless) succeeded?

A comprehensive answer to the question just posed would necessitate a tremendous amount of numerical testing. We concentrated our testing on just a few functions with which we have had previous software experience, namely Airy, Bessel, hypergeometric, confluent hypergeometric and Legendre functions. Usually the UL Library performed in accordance with the specifications for each routine. However, we studied the documentations in detail, looking for major

[1]The library diskettes provide FORTRAN programs, which could be compiled and used on a Personal Computer without a co-processor, as well as coprocessor assembly code for each subroutine. For the purposes of this review, attention is restricted to the co-processor assembly code.

ways in which the algorithms used might go astray. Unfortunately, we were successful.

The first, and very serious, type of failure arise when the UL Library delivers completely wrong answers without any error message. This occurs with the routine BESJR in §8.8, which is designed to generate the Bessel function $J_\nu(x)$ for real values of the argument x and order ν. For example, with $x = 9\pi$ and $\nu = 2\frac{1}{2}, 4\frac{1}{2}, 6\frac{1}{2}, 8\frac{1}{2}, 10\frac{1}{2}$, BESJR yields the following values of $J_\nu(x)$ to ten decimal places:

$$-0.00670\ 05817,\ 0.03764\ 71379,\ -0.05418\ 00781,$$
$$0.31114\ 25305,\ -0.06567\ 89923.$$

These should be compared with the correct values[2]

$$0.01592\ 10880,\quad -0.05237\ 32560,\quad 0.10316\ 94874,$$
$$-0.14727\ 18330,\quad 0.14363\ 19977.$$

Not only are the numerical values totally incorrect, all the signs are wrong, too. Similar gross errors occur for $\nu = 22\frac{1}{2}(2)50\frac{1}{2}$, for example. On the other hand, BESJR generates correct values (within the prescribed error tolerance) for the intermediate values $\nu = 12\frac{1}{2}(2)20\frac{1}{2}$, and also for $\nu = \frac{1}{2}$ and $1\frac{1}{2}(2)49\frac{1}{2}$, x again being 9π. The reason for the errors appears to be that J. C. Miller's backward recurrence algorithm has been used, with the trial values normalized on the value of a single Bessel function, in fact $J_{1/2}(9\pi)$. Since $J_{1/2}(9\pi)$ is zero, this procedure is bound to lead to meaningless answers. Yet this cannot be the entire explanation, otherwise all values in the range $\nu = 21\frac{1}{2}(1)50\frac{1}{2}$ would be incorrect and not merely alternate ones. Thus the documentation must be in error, too. And there may be further inaccuracies here. For example, according to the documentation the value for $\nu = 2\frac{1}{2}$ is computed from the power series expansion of $J_\nu(x)$: either the Miller algorithm was used instead, or, perhaps less likely, the power series was summed incorrectly.

Similar errors occur for numerous other values of ν, both integer and non-integer, that we tested. For example, with $x = 8.6537279129\ldots$ (the third positive zero of $J_0(x)$), BESJR generates accurate values for $\nu = 0, 1, 2, 3$ and $7(1)50$, but grossly inaccurate values for $\nu = 4, 5, 6$. Furthermore, erroneous values of $J_\nu(x)$ are generated when the value of x is merely moderately close to one of the critical values, the magnitudes of the errors being inversely proportional to the distances of x from the critical value.

A companion routine to BESJR is BESYR (§8.10), which is designed to generate the Bessel function $Y_\nu(x)$ for real values of x and ν. Since one of the algorithms used in BESYR draws upon values of $J_\nu(x)$, we expected—and found—some difficulties. For example, BESYR computed $Y_\nu(9\pi)$ with wrong signs and incorrect numerical values for $\nu = -50\frac{1}{2}(2)-22\frac{1}{2}$ and $-10\frac{1}{2}(2)-2\frac{1}{2}$,

[2]Obtained by use of D. E. Amos' package [2]. See also comments made below on validation.

while the results were correct at intermediate values of ν. (This similarity of behavior with BESJR reflects the identity $Y_\nu(x) = (-1)^n J_{-\nu}(x)$ when $\nu = n - \frac{1}{2}$, n being an integer.)

A second type of failure is the generation of results which, while not completely inaccurate, contain errors greatly in excess of the accuracy claimed in the documentation. One example is provided by the routine BESYR mentioned in the preceding paragraph. The documentation claims at least 12-digit accuracy when the order ν is an integer. In itself this claim is careless because it takes no account of the inevitable loss of relative precision in the neighborhoods of zeros. But the situation is actually much worse. With $\nu = 119$ and 120, for example, we found that in the range $x = 848(0.1)852$ the accuracy often fell to 8 or 9 digits, even well away from the zeros of $Y_{119}(x)$ and $Y_{120}(x)$. Another example is provided by the routines AIRYA and AIRYAD (§§8.1 and 8.3) for the Airy function $\mathrm{Ai}(x)$ and its first derivative. In both routines at least 13-digit accuracy is claimed when $-10^{10} \le x \le 100$. But we found many values of x in the (zero-free) range 5.2 to 5.8 for which they yielded only 8 or 9 correct digits.

The third type of failure arises when the UL Library generates an inaccurate value, but the user is warned by an error message such as "unable to compute the ... function with acceptable accuracy" or "numeric overflow in the ... function". However, if these failures occur too frequently, then there will be huge gaps in the effective ranges claimed for the variables. Such is the case with the routines CHGFU (§9.2) for generating the confluent hypergeometric function $U(a, b, x)$, and HPRGMT (§20.1) for generating the hypergeometric function $F(a, b; c; x)$.

CHGFU employs two algorithms. The first is evaluation of the asymptotic expansion of $U(a, b, x)$ for large x, which is quite sound. The second is based on a formula that expresses $U(a, b, x)$ as a difference of two M-type confluent hypergeometric functions, which is unsound because of the potential for massive numerical cancellation. In consequence, although the documentation claims that the effective ranges of the variables are given by

$$-50 \le a \le 50, \quad -50 \le b \le 50, \quad -100 \le x \le 100,$$

with the exclusion of integer values of b and nonpositive integer values of a, extensive regions are inadmissible. These include, for example,

$$\begin{cases} a = 0.5, \ b = 0.5, \ 7.4 \le x \le 19.9; \ a = 2.7, \ b = 5.4, \ 10.1 \le x \le 437.4; \\ a = 20, \ b = 2.5, \ 0.34 \le x \le 20,000. \end{cases}$$

For HPRGMT the documentation states that a and b can have any real values between -10^{10} and 10^{10}, c can have any real value between -10^{20} and 10^{20} (other than a negative integer) and x can have any real value between -10^{10} and 1. But, again, these claims are misleading. For example, when x is

in the interval [0, 1) HPRGMT sums the hypergeometric series

$$F(a, b; c; x) = \sum_{n=0}^{\infty} \frac{a(a+1)\cdots(a+n-1)b(b+1)\cdots(b+n-1)}{c(c+1)\cdots(c+n-1)} \frac{x^n}{n!}.$$

Since the radius of convergence is 1 the algorithm must fail for values of x sufficiently close to 1. Sample intervals of failure were found to be:

$$\begin{aligned} &a = b = 100, && c = 1, && 0.95 \le x < 1; \\ &a = b = 10000, && c = 1, && 0.0013 \le x < 1; \\ &a = b = 10^8, && c = 1, && 10^{-10} \le x < 1. \end{aligned}$$

Even when failure does not occur, execution can be extremely slow. Thus for $a = b = c = 1$ and $x = 0.99999$, 20 minutes elapsed on an IBM PS2 computer before the answer was produced.

The weaknesses in the algorithms used for $J_\nu(x)$, $Y_\nu(x)$ and $U(a, b, x)$ were avoidable, if only because robust software for generating these functions is already available; see, for example, [3], [8]. We also observe that instead of normalizing the trial values of $J_\nu(x)$ obtained in the Miller algorithm via a single value of $J_\nu(x)$, the identity

$$(\tfrac{1}{2}x)^\nu = \sum_{k=0}^{\infty} \frac{(\nu + 2k)\Gamma(\nu + k)}{k!} J_{\nu+2k}(x), \qquad \nu \neq 0, -1, -2, \ldots,$$

[6, equation (9.1.87)], could have been used instead. This is the appropriate generalization of the identity

$$1 = J_0(x) + 2J_2(x) + 2J_4(x) + \cdots$$

that the authors use in a routine BESJ (§8.7) for generating functions of integer order. Again, a stable way of generating $U(a, b, x)$ is backward integration of the confluent hypergeometric equation, with initial values derived from the asymptotic expansions of $U(a, b, x)$ and $\partial U(a, b, x)/\partial x$ for large x.

In addition to occasional poor choices of algorithm, we noticed instances of poor choices of the actual functions being generated. Thus, functions that exhibit exponential growth or decay when the argument x is large would have been better replaced by their logarithms. This would greatly increase the threshold at which overflow or underflow occurs. Such functions include the Gamma function, $\Gamma(x)$, the exponential integral, $\mathrm{Ei}(x)$, the complementary error function, $\mathrm{erfc}\, x$, the Airy functions, $\mathrm{Ai}(x)$ and $\mathrm{Bi}(x)$, the incomplete Gamma function $\Gamma(a, x)$, the modified Bessel functions $I_\nu(x)$ and $K_\nu(x)$, and the confluent hypergeometric function $M(a, b, x)$. In the case of $\Gamma(x)$, a separate routine *is* given for $\ln|\Gamma(x)|$ in §13.3 but since this is constructed simply by taking logarithms of the values obtained by the library routine for $\Gamma(x)$ there is no increase in the overflow threshold. It is the logarithm that should have been generated first!

Troublesome singularities can sometimes be avoided by introduction of appropriate factors. Thus each of the functions $M(a, c, x)$ and $F(a, b; c; x)$ has poles at $c = 0, -1, -2, \cdots$, but both $M(a, c, x)/\Gamma(c)$ and $F(a, b; c; x)/\Gamma(c)$ are entire functions of c; this was pointed out many years ago in [7]. On the other hand, there is a nonsensical statement in §9.2 that $U(a, b, x)$ is not well defined when a and b are negative integers and $|b| \geq |a|$. In fact, as noted in [7, p. 258], $U(a, b, x)$ is entire in a and b. It is the poor algorithm used to compute $U(a, b, x)$ that may fail when a or b is an integer or close to an integer.

We could continue in this vein and we could also provide a substantial list of typographical errors in the manual and operational defects in the software.[3] Instead, let us turn now to the topic of validation. Several articles have been written on the difficulty of checking software for the generation of mathematical functions; see, for example, [3]. How were the UL Library routines validated by the authors? The only clue supplied in the manual appears to be the statement on p. 36 that the least accurate results in the output of an algorithm always occur at the interface with another algorithm. Presumably this means that there has been substantial cross-checking at these interfaces. This is indeed a powerful type of check, but one that does not guard against all types of algorithmic and programming errors, as we have already observed with the routines BESJR and BESYR.

In fairness to the authors there are two subsections (§§6.2, 6.3) in which they encourage users to check output by using identities satisfied by the higher transcendental functions. For example, the error and Bessel functions are both special cases of confluent hypergeometric functions. However, these kinds of identities are rather specialized, and there is always the danger that they may have already been used in constructing the library routine. For example, there are identities that relate the Airy functions $\mathrm{Ai}(x)$, $\mathrm{Bi}(x)$ and their derivatives to Bessel functions and modified Bessel functions of orders $\pm 1/3$, $\pm 2/3$. But if these identities are used as cross-checks, then the inaccuracies we noted above for the routine, AIRYA and AIRYAD may not show up because the same kind of inaccuracy is present in a (parent) routine BESKR (§8.14) for the modified Bessel function $K_\nu(x)$. In contrast, a powerful form of cross-check that is *not* mentioned in the manual, but which is widely applicable, is to employ identities of Wronskian or Casoratian type, for example,

$$\mathrm{Ai}(x)\mathrm{Bi}'(x) - \mathrm{Ai}'(x)\mathrm{Bi}(x) = 1/\pi,$$
$$J_{\nu+1}(x)Y_\nu(x) - J_\nu(x)Y_{\nu+1}(x) = 2/(\pi x).$$

[3]For example, there is a statement on p. 24 that the computer will halt when a library routine is called and a co-processor is not present. We found that is not always true; furthermore, instead of identifying the absence of a co-processor as the problem, the error messages misleadingly report errors in the library routine.

These were the checks that we used ourselves to detect errors in the UL Library routines as well as to ascertain whether or not the UL Library output is correct in cases of discrepancy with output from other libraries.

Let us summarize our findings. The UL Library provides PC users with a new set of routines for generating an extensive collection of higher transcendental functions, indeed most of the functions tabulated in the N.B.S. Handbook. The library is relatively easy to install and its price is reasonable. It will provide a useful tool for mathematical physicists and other scientists, especially those engaged in calculations of exploratory type. However, in our comparatively small sample we encountered failures of various kinds, including some extremely serious ones. Because of this, we conclude that the authors may have lacked some experience and expertise, or perhaps just the necessary time, for the mammoth task of constructing a robust library of the kind intended. In consequence, users will need to exercise great care with any output from the library, applying independent checks wherever possible. Furthermore, users must also be prepared for disappointments: the viable ranges of a routine may turn out to be a good deal less than is claimed in the documentation, especially in the case of functions that have not been treated by earlier software workers.

For heavy systematic computations many users will find the more robust IMSL and NAG libraries [4], [5] to be preferable. The variety of functions covered in these libraries is not as large, but the viable ranges of the variables are considerably more extensive and the precision is often higher. Moreover, both IMSL and NAG provide many desirable features, such as linear algebra packages, in addition to routines for generating higher transcendental functions.

In preparing this review I am heavily indebted to Dr. Daniel Lozier of the National Institute of Standards and Technology who helped devise and carry out the numerical testing, and also to Sang Chin, a student at Duke University, who assisted.

F.W.J.O.

1. American National Standard Programming Language FORTRAN, ANSI X3., 9–1978, American National Standards Institute, 1430 Broadway, New York, NY 10018.
2. D. E. Amos, S. L. Daniel, and M. K. Weston, CDC 6600 *subroutines* IBESS *and* JBESS *for Bessel Functions* $I_\nu(x)$ *and* $J_\nu(x)$, $x \geq 0$, $\nu \geq 0$, ACM Trans. Math. Software **3** (1977), 76–92.
3. W. J. Cody, *Software for special functions*, Rend. Sem. Mat. Univ. Politec. Torino, Special issue: *Special functions: theory and computation* (1985), 91–116.
4. IMSL *Library Reference Manual*, IMSL, 7500 Bellaire Boulevard, Houston, TX 77036-5085, Edition 9, June 1982.
5. *NAG Fortran Mini Manual—Mark* 12, Numerical Algorithms Group, Inc., 1101 31st St., Suite 100, Downers Grove, IL 60515–1263. First edition, March 1987.
6. National Bureau of Standards, *Handbook of mathematical functions* (M. Abramowitz and I. A. Stegun, eds.), Applied Mathematics Series No. 55, U.S. Government Printing Office, Washington, D.C., 1964.
7. F. W. J. Olver, *Asymptotics and special functions*, Academic Press, New York, 1974.
8. N. M. Temme, *The numerical computation of the confluent hypergeometric function* $U(a, b, z)$, Numer. Math. **41** (1983), 63–82.