ALGORITHM 715: SPECFUN — A Portable FORTRAN Package of Special Function Routines and Test Drivers

W. J. CODY
Argonne National Laboratory

SPECFUN is a package containing transportable FORTRAN special function programs for real arguments and accompanying test drivers. Components include Bessel functions, exponential integrals, error functions and related functions, and gamma functions and related functions.

Categories and Subject Descriptors. G.1.0 [**Numerical Analysis**]. General—*numerical algorithms*; G.4 [**Mathematics of Computing**]: Mathematical Software—*certification and testing*

General Terms: Algorithms

Additional Key Words and Phrases: Bessel functions, error function, exponential integrals, gamma function

1. INTRODUCTION

FUNPACK, a small package of special function programs [7, 10] developed in the 1970s, was designed for near-optimal performance on a limited set of computers. Because the programs were highly machine-dependent and not intended to be moved to other machines, the package has become outmoded. We present here SPECFUN, a new and more complete package containing highly transportable FORTRAN programs for 27 special functions. In addition to the functions, the package contains a complete set of transportable self-contained test drivers that may also be used to assess the quality of function programs from other collections. These test drivers may ultimately prove the most useful part of the package; to our knowledge, no other test drivers are available.

In the next section we review the overall design of the package (presented in more detail elsewhere [11]), including portability issues. Section 3

This work was supported by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under contract W-31-109-Eng-38.
Author's address: Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439-4801.

discusses the function programs in the package, and Section 4 briefly summarizes the test programs.

2. PACKAGE DESIGN

SPECFUN differs considerably from the earlier FUNPACK package in several important aspects: portability, the functions available, and the inclusion of programs to measure performance of the functions. The latter two differences are discussed in the following sections; here we concentrate on the portability issue.

The design goal was to achieve portability with minimum degradation of the accuracy and robustness found in FUNPACK. Clearly, portability comes with a price—the loss of features such as precise overflow/underflow threshold tests in this case, features that depend intimately on the machine representation of data.

Most of the FUNPACK programs expressed approximation coefficients and error thresholds in the native representation for the host machine (a major reason the programs are not portable). In contrast, all coefficients and thresholds in SPECFUN are given in decimal in DATA statements. This design introduces two different potentially damaging sources of error: the error necessarily introduced by the conversion of these decimal strings to internal representations, and the imprecision in error thresholds. The effect of conversion error is minimized by selecting approximation forms that are numerically stable, and then evaluating them with attention to details that minimize the buildup of rounding error. Lack of precision in error thresholds cannot be circumvented, however. The best that can be done is to provide conservative values (so that overflow, for example, is not triggered because the test threshold is slightly too large) and to accept the resultant trapping of a few theoretically acceptable arguments as the price for portability.

We use two general approaches to evaluating the functions: minimax approximations for functions of one variable, and recurrence techniques for functions of two variables (the Bessel functions for arbitrary real argument and order where the programs normally return a sequence of function values). Approximation coefficients and certain of the recursion parameters are selected to provide between 18 and 20 significant decimal digits of accuracy. The programs thus provide full single-precision accuracy on all computers currently used for serious scientific computation (assuming, of course, that the elementary function library is decent), and double-precision accuracy on most such machines (the exceptions being machines like the Crays with more than 60-bit double-precision arithmetic). The decision to use only one approximation, independent of the underlying machine precision, means that the programs may be inefficient on short-wordlength machines, evaluating a higher-degree approximation than would otherwise be necessary. We view this as the lesser of two evils, the alternative being to provide different approximations (and possibly different computational paths) for each of several different wordlengths. Of course, parameters for recurrence schemes are adjusted wherever possible to match machine wordlength.

The comments at the beginning of each program provide complete details on calling sequences, error returns, citations for coefficients and algorithms, other programs required, and machine-dependent parameters (error thresholds, etc.). The parameter discussion includes a mathematical definition of each parameter in sufficient detail to permit its computation for a new machine, and a tabulation of suggested values for popular machines. These values must be inserted in the appropriate DATA statements in each program as part of the process of installing the programs (default values in the DATA statements are those for machines based on IEEE arithmetic).

Other than the provision of these machine-dependent constants, program installation involves only some minor global editing changes. All other coefficients, variable declarations (we impose strong typing), function calls, or other constructs that might vary with declared precision are duplicated in statements containing CS in columns 1 and 2 for the single-precision versions and CD in those columns for the double-precision versions. A single-precision version of a program is thus obtained by changing all such occurrences of CS to blanks, and a double-precision version by instead changing the CDs to blanks. Note that one or the other of these global changes must be made.

We have used different names for some single- and double-precision routines in the error and gamma families to be consistent with the traditional names established years ago in IBM libraries. The single- and double-precision versions of all other function programs have the same name. There are several reasons for this. First, when we used different names in the FUNPACK package many years ago, some users complained. Applications using our functions were difficult to move between IBM machines, for example, where computations were normally done in double precision, and CDC machines where computations were normally done in single precision. Second, system-maintained libraries, even today, normally contain special function routines in only one precision, typically single precision on machines like CRAYs and double precision on machines like those based on IEEE arithmetic.

One final note on package design: programs sharing core computations are packaged together in a packet. Such a packet contains two or more function programs intended to be called by the user, together with a computational subroutine intended to be called only by the function programs, i.e., a subroutine whose usage is intended to be limited to internal package calls. The names of these computational subroutines have been chosen so the user is not likely to duplicate them unintentionally in his own program.

3. THE FUNCTIONS

Table I lists the functions in SPECFUN by family. Unless otherwise noted here, use of a function from this package requires only the declaration of the function and its argument, and the usual invocation for a FUNCTION-type FORTRAN subprogram.

The first family listed is a packet of three exponential integral programs (EI, EONE, and EXPEI) and one computational core program (CALCEI)

Table I. Contents of SPECFUN

Function	Function Program	Auxiliary Routine
Exponential Integrals		
$Ei(x)$	EI	CALCEI
$E_1(x)$	EONE	CALCEI
$e^{-x}E_1(x)$	EXPEI	CALCEI
Gamma Functions		
$\Gamma(x)$	GAMMA, DGAMMA	
$\ln \Gamma(x)$	ALGAMA, DLGAMA	
$\Psi(x)$	PSI	
Error Functions		
$\mathrm{erf}(x)$	ERF, DERF	CALERF
$\mathrm{erfc}(x)$	ERFC, DERFC	CALERF
$e^{x^2}\mathrm{erfc}(x)$	ERFCX, DERFCX	CALERF
Dawson's Integral	DAW	
Normal Distribution	ANORM	
J Bessel Functions		
$J_0(x)$	BESJ0	CALJY0
$J_1(x)$	BESJ1	CALJY1
$J_\nu(x)$	RJBESL	
Y Bessel Functions		
$Y_0(x)$	BESY0	CALJY0
$Y_1(x)$	BESY1	CALJY1
$Y_\nu(x)$	RYBESL	
I Bessel Functions		
$I_0(x)$	BESI0	CALCI0
$e^{-\lvert x\rvert}I_0(x)$	BESEI0	CALCI0
$I_1(x)$	BESI1	CALCI1
$e^{-\lvert x\rvert}I_1(x)$	BESEI1	CALCI1
$I_\nu(x)$, $e^{-x}I_\nu(x)$	RIBESL	
K Bessel Functions		
$K_0(x)$	BESK0	CALCK0
$e^{x}K_0(x)$	BESEK0	CALCK0
$K_1(x)$	BESK1	CALCK1
$e^{x}K_1(x)$	BESEK1	CALCK1
$K_\nu(x)$, $e^{x}K_\nu(x)$	RKBESL	

patterned after a similar packet in FUNPACK. These programs exploit rational minimax approximations [20, 21] that are theoretically accurate to 18 significant decimal digits. EONE avoids an error return for negative arguments by computing with $|x|$. The programs return default values of zero or the largest machine-representable number, as appropriate, for zero arguments or arguments exceeding machine-dependent limits.

Next come functions related to the gamma function. The programs for $\Gamma(x)$ (GAMMA and DGAMMA, where the D designates the double-precision program) are based on an algorithm outlined in [8]. These programs exploit an unpublished minimax approximation for $1 \leq x \leq 2$ and a published approximation [24] for $12 \leq x$. Other arguments are handled with the reflection formula (for negative arguments) and the recurrence relation. These programs return the largest machine-representable number at singularities and

for arguments greater in magnitude than a machine-dependent threshold. The companion programs for $\ln \Gamma(x)$ (ALGAMA and DLGAMA) exploit published minimax approximations [15, 24] in an algorithm detailed in [15]. The argument for this function must be positive and less than a machine-dependent upper limit. These routines return the largest positive machine number for out-of-range arguments. The family also includes a program for $\Psi(x)$ (PSI), similar to the corresponding FUNPACK program, and based on minimax approximations from [19]. This program returns the largest representable floating-point number, with appropriate sign, for arguments in a machine-dependent neighborhood of zero, and negative arguments less than a machine-dependent threshold.

The third family is a collection of error-like functions. Programs for the error function (ERF and DERF), the complementary error function (ERFC and DERFC), and the exponentially scaled complementary error function (ERFCX and DERFCX) are packaged together with one core computational program (CALERF). The approximations used here [6] are theoretically accurate to at least 18 significant decimal digits. ERFC and ERFCX return zero for arguments greater than machine-dependent thresholds, and ERFCX returns the largest floating-point number for negative arguments less than a machine-dependent threshold. Next is Dawson's integral (DAW). This program is similar to the FUNPACK program, exploiting minimax approximations and an algorithm outlined in [16]. There is a machine-dependent argument beyond which the program returns a zero result. The final member of this family is the normal distribution function (ANORM). The approximations used in this program are derived from those for the error function.

The remainder of the functions in SPECFUN are Bessel functions grouped into four families, the J, Y, I, and K functions. Each family consists of function programs for orders 0 and 1 and a subroutine-type program for sequences of functions starting at an arbitrary fractional order. In addition, the I and K families include routines for exponentially scaled functions.

For efficiency, the programs for $J_0(x)$ (BESJ0) and $Y_0(x)$ (BESY0) are packaged into one packet with a core computational program (CALJY0). Similarly, the programs for $J_1(x)$ (BESJ1) and $Y_1(x)$ (BESY1) are packaged into one packet with a core program (CALJY1). In both packets, the main computation uses unpublished minimax approximations for $|x| \leq 8$ and approximations from [24] for larger arguments. The first few zeros of all of these functions are built into the approximations so that relative accuracy is maintained for $|x| \leq 8$. Only absolute accuracy is maintained for $x > 8$, however. These programs return zero for arguments greater than machine-dependent thresholds, and BESY0 and BESY1 return the negative of the largest floating-point number for zero or negative arguments. The programs BESJ0 and BESJ1 duplicate capabilities in FUNPACK.

The subroutine RJBESL is a heavily modified version of a program written by Sookne [25] that computes both I and J Bessel functions of real argument and integer order. Modifications for SPECFUN include the restriction of the computation to the J Bessel functions for nonnegative arguments, the generalization to arbitrary nonnegative fractional order, and elimination of

underflow problems. The new program returns an array of Bessel functions $J_\alpha(x), J_{\alpha+1}(x), \ldots, J_{\alpha+n-1}(x)$ for $0 \le x <$ XLARGE and $0 \le \alpha < 1$, where XLARGE is arbitrarily set to 10^4. The calling sequence for this routine is

```
CALL RJBESL(X,ALPHA,NB,B,NCALC)
```

where X and ALPHA are input floating-point values for x and α, respectively, NB is an input INTEGER variable with the value n, B is a floating-point output array of at least length NB to hold the computed function values, and NCALC is an INTEGER output variable containing error information in the form of the number of correct function values returned, or a coded indicator for an improper argument (such as a negative value for X or NB). In case of an argument error, B(1) is set to zero, but the remaining contents of the array B are not altered by the program. RJBESL does require that a GAMMA function be available.

RYBESL is the counterpart of RJBESL for the Y Bessel functions. The program draws heavily on Temme's Algol program for $Y(a,x)$ and $Y(a+1,x)$ [26], and on Campbell's program for $Y_\nu(x)$ [3]. Temme's scheme is used for arguments in the nonasymptotic region, and Campbell's scheme is used in the asymptotic region. With the original authors' permissions, segments of code from both sources have been translated into FORTRAN 77, merged, and heavily modified. Modifications include parameterization of machine dependencies, use of a new internal approximation for $\ln \Gamma(x)$, and built-in protection against overflow and destructive underflow. RYBESL returns an array of Bessel functions $Y_\alpha(x), Y_{\alpha+1}(x), \ldots, Y_{\alpha+n-1}(x)$ for $0 < x <$ XLARGE and $0 \le \alpha < 1$, where XLARGE is a machine-dependent threshold. The calling sequence is

```
CALL RYBESL(X,ALPHA,NB,BY,NCALC)
```

where the arguments and error indications are similar to those for RJBESL. In case of an argument error, BY(1) is set to zero, but the remaining elements of the array are unaltered.

The next family is the I Bessel family. In addition to programs analogous to those in the previous two families, this family includes programs for exponentially scaled functions of order 0 and 1, and an additional parameter to the general-order subroutine to provide exponentially scaled functions there. The programs for $I_0(x)$ and $e^{-|x|}I_0(x)$ (BESI0 and BESEI0, respectively) are packaged with the core program CALCI0. The corresponding programs for $I_1(x)$ and $e^{-|x|}I_1(x)$ (BESI1 and BESEI1) form a packet with the core program CALCI1. The main computations in these programs evaluate slightly modified minimax approximations from [2]. BESI0 and BESI1 return the largest floating-point number for arguments greater in magnitude than machine-dependent thresholds. These two packets duplicate facilities provided in FUNPACK.

RIBESL is the counterpart to RJBESL in this family. It too is based on Sookne's program [25]. Modifications include the restriction of the computation to the I Bessel functions for nonnegative arguments, the extension to arbitrary nonnegative fractional order, the inclusion of optional exponential

scaling, and elimination of underflow problems. An earlier version of this program was published in [9]. RIBESL returns an array of Bessel functions $I_\alpha(x), I_{\alpha+1}(x), \ldots, I_{\alpha+n-1}(x)$ or of exponentially scaled Bessel functions $e^{-|x|}I_\alpha(x), e^{-|x|}I_{\alpha+1}(x), \ldots, e^{-|x|}I_{\alpha+n-1}(x)$ for $0 \le x$ and $0 \le \alpha < 1$. In addition, x is bounded above by machine-dependent thresholds which differ for the scaled and unscaled computations. The calling sequence is

```
CALL RIBESL(X,ALPHA,NB,IZE,B,NCALC)
```

where the INTEGER input parameter IZE is set to 1 if unscaled functions are to be computed, and to 2 if exponentially scaled functions are to be computed. All other parameters are similar to those for RJBESL or RYBESL. In case of an argument error, none of the elements of the array B is altered. RIBESL also requires a GAMMA function.

The final family of K Bessel functions contains programs entirely analogous to those in the I Bessel family. The programs for $K_0(x)$ and $e^x K_0(x)$ (BESK0 and BESEK0) are packaged with the core computation CALCK0, while those for $K_1(x)$ and $e^x K_1(x)$ (BESK1 and BESEK1) are packaged with the core computation CALCK1. Arguments to these functions must be strictly positive and less than machine-dependent thresholds. All of the main computations use slightly modified minimax approximations from [1]. BESK0 and BESK1 return the largest floating-point number for nonpositive arguments, and return zero for arguments larger than machine-dependent thresholds. FUNPACK contained similar programs.

RKBESL returns an array of Bessel functions $K_\alpha(x), K_{\alpha+1}(x), \ldots, K_{\alpha+n-1}(x)$ or of exponentially scaled Bessel functions $e^x K_\alpha(x), e^x K_{\alpha+1}(x), \ldots e^x K_{\alpha+n-1}(x)$ for $0 < x$ and $0 \le \alpha < 1$. In addition, x is bounded above by a machine-dependent threshold for the unscaled functions. The calling sequence is

```
CALL RKBESL(X,ALPHA,NB,IZE,BK,NCALC)
```

where the arguments are defined similarly to those for RIBESL. The program is based on one written by Campbell [4, 5]. Modifications include the addition of nonscaled functions; parameterization of machine dependencies; and the use of more accurate, locally generated internal approximations for $\sinh(x)$ and $\sin(x)$. In case of an argument error, none of the elements of the array BK is altered.

4. THE TEST PROGRAMS

Table II lists the 18 test programs and an auxiliary program in SPECFUN. The tests are all self-contained. They rely on carefully selected and implemented identities or analytic expansions to check accuracy and provide some checks of robustness (error returns, etc.).

Accuracy tests all use randomly generated arguments, usually 2000 of them, from appropriate intervals. These arguments are first "purified" (perturbed in a test-dependent way) to ensure that certain auxiliary arguments to be determined from them are exact machine numbers. The purified

Table II. Test Drivers in SPECFUN

Driver	Functions Tested	Auxiliary Routine
EITEST	EI	DSUBN
GAMTST	GAMMA, DGAMMA	
ALGTST	ALGAMA, DLGAMA	
PSITST	PSI	
ERFTST	ERF, ERFC ERFCX	
DAWTST	DAW	
ANRTST	ANORM	
J0TEST	BESJ0	
J1TEST	BESJ1	
RJTEST	RJBESL	
Y0TEST	BESY0	
Y1TEST	BESY1	
RYTEST	RYBESL	
I0TEST	BESI0, BESEI0	
I1TEST	BESI1, BESEI1	
RITEST	RIBESL	
K0TEST	BESK0, BESEK0	
K1TEST	BESK1, BESEK1	
RKTEST	RKBESL	

arguments are then used in calls to the function under test. The auxiliary arguments are used in an independent computation of the function value based on an identity or analytic expansion. The results of these two computations are compared, and the loss of significant digits (in the native radix for the machine) is estimated as follows.

Let β denote the machine floating-point radix, p the number of base-β digits in the floating-point significand, and E the relative difference between the two computations for a particular argument. Then the reported statistics are

$$MRE = p + \ln(\max|E|)/\ln(\beta),$$

and

$$RMS = \max\left[0.0, p + \ln\left(\sum E^2/N\right)/(2\ln(\beta))\right],$$

where the sum in *RMS* is taken over all arguments in a given interval. *MRE* estimates the maximum relative error in the interval and *RMS* is an estimate of the root-mean-square error for the interval, both measured in terms of lost base-β digits. Note that the computation of *RMS* has been adjusted so *RMS* never reports a negative loss of significant digits.

Values for p, β, and other machine parameters are dynamically determined with the auxiliary routine MACHAR [12], which is included in the package for completeness. A random number generator, REN, is also included. (Warning: REN is good enough for our simple needs, but is not intended to be used for a general-purpose generator.) All of the test programs in SPECFUN call MACHAR and REN. Therefore, the programs will malfunction on any machine on which MACHAR malfunctions (see [12]), although they can be customized for a particular machine by supplying the necessary constants in DATA statements and removing the call to MACHAR.

The accuracy test in the first test driver, EITEST, is based on a local Taylor series expansion [18]. Derivatives are generated as needed by using the auxiliary routine DSUBN, a slightly modified version of Gautschi and Klein's algorithm for the derivatives of e^x/x [22, 23]. See [18] for details of the test procedure, an error analysis for the testing process, and typical statistics obtained for EI and routines from several other sources on some typical machines. Although this program specifically tests the accuracy only of EI, the tests are easily altered to look at the other functions in the family. We have not done that because our programs share a core computational element whose accuracy is adequately tested in checking EI. Robustness checks are applied to all three of our exponential integral routines, however.

The next three test drivers, GAMTST, ALGTST, and PSITST, look at the gamma family of functions. The identities used in these programs, the error analysis for the testing procedures, and sample test results are given in [13].

The next three test drivers, ERFTST, DAWTST, and ANRTST, check the error function family. ERFTST checks ERF, ERFC, and ERFCX using methods discussed in detail in [14]. The accuracy tests in DAWTST are based on local Taylor series expansions similar in concept to those discussed in [18]. ANRTST tests ANORM using methods similar to those in ERFTST.

Our accuracy tests for Bessel programs involve two different approaches, the use of multiplication formulas and local Taylor series expansions. See [17] for a detailed discussion of the first approach, including an error analysis and typical results for RKTEST. Accuracy tests in J0TEST and J1TEST are based on local Taylor series, while those in Y0TEST, Y1TEST, and all of the tests for K Bessel functions are based on multiplication theorems. RJTEST, RYTEST, I0TEST, I1TEST, and RITEST all use multiplication formulas for testing with small arguments, and local Taylor series otherwise.

With the exception of the Taylor series-based tests for the integer-order Bessel functions, which require functions of orders 0 and 1 to initialize the Taylor series, all of the accuracy tests in our programs involve only the program under test and possibly one or two elementary functions.

ACKNOWLEDGMENTS

This paper and the algorithms it describes owe much to contributions from others. In addition to those whose work has been cited earlier, L. Stoltz and G. Zazi participated in the design and preparation of several of the elements of the package. G. Pieper, as usual, patiently read this manuscript and

offered suggestions for improving the text. The author is grateful to all who participated in this effort.

REFERENCES

1. RUSSON, A. E., AND BLAIR, J. M. Rational function minimax approximations for the modified Bessel functions $K_0(x)$ and $K_1(x)$. Atomic Energy of Canada Limited Rep. AECL-3461, Chalk River Nuclear Lab., Chalk River, Ontario, Oct. 1969.
2. BLAIR, J. M., AND EDWARDS, C. A. Stable rational minimax approximations to the modified Bessel functions $I_0(x)$ and $I_1(x)$. Atomic Energy of Canada Limited Rep. AECL-4928, Chalk River Nuclear Lab., Chalk River, Ontario, Oct. 1974.
3. CAMPBELL, J. B. Bessel functions $J_\nu(x)$ and $Y_\nu(x)$ of real order and real argument. *Comput. Phys. Commun. 18* (1979), 133–142.
4. CAMPBELL, J. B. A FORTRAN IV subroutine for the modified Bessel functions of the third kind of real order and real argument. Rep. NRC/ERB-925, National Research Council, Canada, 1980.
5. CAMPBELL, J. B. On Temme's algorithm for the modified Bessel functions of the third kind. *ACM Trans. Math. Softw. 6*, 4 (1980), 581–586.
6. CODY, W. J. Rational Chebyshev approximations for the error function. *Math. Comput. 23* (1969), 631–637.
7. CODY, W. J. The FUNPACK package of special function routines. *ACM Trans. Math. Softw. 1*, 1 (1975), 13–25.
8. CODY, W. J. An overview of software development for special functions. In *Lecture Notes in Mathematics, 506, Numerical Analysis*. G. A. Watson, Ed., Springer-Verlag, Berlin, 1976, 38–48.
9. CODY, W. J. Algorithm 597: Sequence of modified Bessel functions of the first kind. *ACM Trans. Math. Softw. 9*, 2 (1983), 242–245.
10. CODY, W. J. FUNPACK—A package of special function routines. In *Sources and Development of Mathematical Software*, W. R. Cowell, Ed., Prentice-Hall, Englewood Cliffs, N.J., 1984, 49–67.
11. CODY, W. J. SPECFUN—A portable special function package. In *New Computing Environments: Microcomputers in Large-Scale Scientific Computing*, A. Wouk, Ed., SIAM, Philadelphia, 1987, 1–12.
12. CODY, W. J. Algorithm 665: MACHAR: A subroutine to dynamically determine machine parameters. *ACM Trans. Math. Softw. 14*, 4 (1988), 303–311.
13. CODY, W. J. Performance evaluation of programs related to the real gamma function. To appear in *ACM Trans. Math. Softw.*
14. CODY, W. J. Performance evaluation of programs for the error and complementary error functions. *ACM Trans. Math. Softw. 16*, 1 (1990), 29–37.
15. CODY, W. J., AND HILLSTROM, K. E. Chebyshev approximations for the natural logarithm of the gamma function. *Math. Comput. 21* (1967), 198–203.
16. CODY, W. J., PACIOREK, K. A., AND THACHER, H. C., JR. Chebyshev approximations for Dawson's integral. *Math. Comput. 24*, 1 (1970), 171–178.
17. CODY, W. J., AND STOLTZ, L. Performance evaluation of programs for certain Bessel functions. *ACM Trans. Math. Softw. 15*, 1 (1989), 41–48.
18. CODY, W. J., AND STOLTZ, L. The use of Taylor series to test accuracy of function programs. To appear in *ACM Trans. Math. Softw.*
19. CODY, W. J., STRECOK, A. J., AND THACHER, H. C., JR. Chebyshev approximations for the psi function. *Math. Comput. 27*, 1 (1973), 123–127.
20. CODY, W. J., AND THACHER, H. C., JR. Rational Chebyshev approximations for the exponential integral $E_1(x)$. *Math. Comput. 22* (1968), 641–649.
21. CODY, W. J., AND THACHER, H. C., JR. Chebyshev approximations for the exponential integral Ei (x). *Math. Comput. 23*, 2 (1969), 289–303.

22. GAUTSCHI, W., AND KLEIN, B J. Recursive computation of certain derivatives—A study of error propagation. *Commun. ACM 13*, 1 (1970), 7–9.
23. GAUTSCHI, W., AND KLEIN, B. J. Remark on Algorithm 282. *Commun. ACM 13*, 1 (1970), 53–54.
24. HART, J. F., ET AL. *Computer Approximations*. Wiley, New York, 1979.
25. SOOKNE, D. J. Bessel functions of real argument and integer order. *NBS J. Res. B 77B* (1973), 125–132.
26. TEMME, N. M. On the numerical evaluation of the ordinary Bessel function of the second kind. *J. Comput. Phys. 21*, 3 (1976), 343–350.

Received October 1990; revised March 1992; accepted March 1992