GEOPHYSICS, VOL. 48, NO. 12 (DECEMBER 1983); P. 1671–1686, 1 FIG., 2 TABLES.

Numerical integration of related Hankel transforms by quadrature and continued fraction expansion

Alan D. Chave*

ABSTRACT

An algorithm is presented for the accurate evaluation of Hankel (or Bessel) transforms of algebraically related kernel functions, defined here as the non-Bessel function portion of the integrand, that is more widely applicable than the standard digital filter methods without enormous increases in computational burden. The algorithm performs the automatic integration of the product of the kernel and Bessel functions between the asymptotic zero crossings of the latter and sums the series of partial integrations using a continued fraction expansion, equivalent to an analytic continuation of the series. The integrands may be saved to allow the rapid computation of related transforms without recalculating the kernel or Bessel functions, and the integration steps use interlacing quadrature formulas so that no function evaluations are wasted when it is necessary to increase the order of the quadrature rule. The continued fraction algorithm allows very slowly divergent or even formally divergent integrals to be computed quite easily. The local error is controlled at each step in the algorithm, and accuracy is limited largely by machine resolution. The algorithm is written in Fortran and is listed in an Appendix along with a driver program that illustrates its features. The driver program and subroutine are available from the SEG Business Office.

INTRODUCTION

Hankel transforms (sometimes called Bessel transforms) written in the standard form

$$\hat{f}(k) = \int_0^\infty d\rho\ \rho f(\rho) J_\nu(k\rho)$$
$$f(\rho) = \int_0^\infty dk\ k \hat{f}(k) J_\nu(k\rho) \qquad (1)$$

are ubiquitous in the mathematical treatment of physical problems involving cylindrical symmetry, as observed in disciplines like optics, electromagnetism, and seismology. The function J_ν in equation (1) is a Bessel function of the first kind of real order ν which exhibits decaying, oscillatory behavior for increasing argument along the real axis, and equation (1) is formally equivalent to a double Fourier transform. The parameter k will be called the wavenumber, while ρ is usually the horizontal range. The pair $\{f(\rho), \hat{f}(k)\}$ is referred to as the kernel functions when it appears under the integral sign; the transforms may also be written with the multiplicative ρ and k terms absorbed into $f(\rho)$ and $\hat{f}(k)$, respectively. For most physical problems the order of the Bessel function is an integer, and for that case only the $\nu = 0$ and 1 cases need be considered since higher order Bessel functions are related to them by standard recursion formulas. Integrals of the form (1) can be evaluated in closed form for only a restricted set of kernel functions (Watson, 1962), and recourse to numerical methods is generally required for the study of real problems.

The standard numerical approach to the computation of Hankel transforms has become the digital filter method since the work of Ghosh (1971). Suitable changes of the independent variables k and ρ in equation (1) are applied to turn the direct integrals into convolution integrals; these are discretized to a finite convolutional sum. The Bessel function of the new variable plays the role of a known digital filter through which the kernel functions are passed as the signal to yield the transformed result as the filter output. Design criteria for the filter are of necessity largely ad hoc and were discussed by Anderson (1979). Reasonable (5 figure) accuracy is typically achieved for monotonic, rapidly decreasing kernel functions at moderate values of ρ. More recent developments, including adaptive and lagged convolution to minimize kernel function evaluations, were covered by Anderson (1982).

For some types of problems the digital filter method is less useful; examples occur at very small values of the range ρ, where the kernel function may be changing rapidly when compared to the Bessel function, and when high numerical precision is required. Direct numerical integration of equation (1) was discussed by Cornille (1972) using the half-cycles of the Bessel function as subintervals combined with Euler's transformation to sum the series of partial integrations. A similar approach is adopted here, with several major improvements, including (1) interlacing quadrature formulas are used so that none of the costly kernel or Bessel function evaluations is

Manuscript received by the Editor February 25, 1983; revised manuscript received May 31, 1983.
*Institute of Geophysics and Planetary Physics, Scripps Institution of Oceanography, University of California at San Diego, La Jolla, CA 92093.

wasted when it is necessary to increase the order of the rule to achieve a desired accuracy, (2) automatic integration is feasible due to the use of interlacing quadrature weights, (3) the integrands may be saved so that algebraically related kernels can be treated without reevaluation of the kernels or Bessel functions, and (4) a continued fraction method is used to sum the series of partial integrations. The latter is a very general means for the summation of series ranging from rapidly convergent to divergent and is equivalent to the analytic continuation of the series. While the algorithm presented here is generally not as fast as the digital filter approach, it is far more efficient than other direct integration schemes and should be regarded as complementing rather than replacing the standard method.

ALGORITHM DESCRIPTION

The algorithm applies numerical quadrature to the evaluation of a sequence of partial integration terms

$$p_n = \int_{z_n}^{z_{n+1}} dk\ \hat{g}(k) J_\nu(k\rho), \tag{2}$$

where $\hat{g}(k) = k\hat{f}(k)$ in equation (1) and z_n is the nth zero of $J_\nu(x)$ normalized by the range ρ. Asymptotic forms for the large zeros of Bessel functions were given in Watson (1962) or standard references; these are quite adequate even for the smallest zeros in this work. The approximation of the linear functional (2) by a finite sum

$$p_n \approx \sum_{j=1}^{N} h_j \hat{g}(a_j) J_\nu(a_j \rho), \tag{3}$$

where a_j is the abscissa and h_j is the weight, is covered in standard texts on numerical analysis (Ralston and Rabinowitz, 1978). Gauss formulas possess the property of minimizing the number of abscissa points (and hence integrand evaluations) required to achieve a specified accuracy when compared to other approaches. Most Gauss rules require recomputation of all of the integrands when the order is changed, so that the proper order must be predetermined for good efficiency. Patterson (1968) developed methods for the optimal addition of points to a quadrature formula so that only the new integrand values need to be calculated when the order is increased, making adaptive integration to a specified error computationally feasible. The weights h_j and a_j in equation (3) used in the algorithm of this paper consist of a three-point Gauss rule with extension to 7, 15, 31, 63, 127, and 255 common-point, interlacing forms; these correspond to integrating polynomials of degree 5, 11, 23, 47, 95, 191, and 383, respectively. In practice, each of the partial integrations (2) is computed by adding new weights to the quadrature rule until a combined relative-absolute error criterion is satisfied. At each step, the kernel and Bessel function values are retained so that none of these costly evaluations need be repeated; these can optionally be placed in common storage so that algebraically related kernels can be treated without recomputing the integrands, in a manner similar to that used by Anderson (1979) with the digital filter method.

The complete integral is formally obtained by summing the partial integrations; direct summation is feasible only for rapidly convergent integrals. The terms in the series are often alternating in sign due to the Bessel function's oscillatory behavior, and the result may be very slowly convergent and will diverge if the kernel increases faster than $k^{1/2}$. Convergence acceleration using various nonlinear transformations of the series were reviewed by Bender and Orszag (1978). The most general of these is the Padé approximant technique, which replaces a slowly converging or diverging series by a rapidly converging expression, each term of which is equivalent to the ratio of two finite sums. Padé summation should be regarded as a method for the analytic continuation of the series; diverging series suggest the presence of singularities which are removed by the transformation. General Padé algorithms require $O(N^3)$ operations, where N is the number of terms in the sum, but the closely related continued fraction approach yields similar results with only $O(N^2)$ operations. Sequential terms of the latter are formally equivalent to a stair-step path in the Padé table (Baker, 1975).

The series of partial integrations

$$S = \sum_{i=0}^{N} p_i \tag{4}$$

may be recast into the continued fraction

$$S = \cfrac{d_0}{1 + \cfrac{d_1}{1 + d_2 \over \vdots \over d_n}}, \tag{5}$$

where there is a direct correspondence between the continued fraction coefficients d_i and the summands p_i. As terms are added to equation (4), only the last coefficient in equation (5) is computed; recursive algorithms were given by Baker (1975) and Hänggi et al (1978, 1980). The latter is used in the algorithm of this paper.

In practice, the continued fraction (5) is reevaluated as additional partial integrations are added to the series (4) and the process terminates when a combined relative-absolute error criterion is satisfied. It is not necessary to compute a portion of the series explicitly and use the continued fraction method only in the asymptotic region of the kernel since the Padé approximants are an analytic continuation method rather than a simple convergence acceleration method. In the author's experience, the continued fraction converges quite rapidly, and more than 30 or so terms are almost never required. The continued fraction algorithm is only slightly effective for rapidly convergent integrals, but for slowly convergent or divergent ones the summation behavior is quite dramatic.

ALGORITHM USAGE

The algorithm contained in Appendix A is written in the ANSI X3.9–1978 (Fortran 77) version of Fortran. For generality, the kernel function is assumed to be complex valued, but complex arithmetic is not used internally, so that double precision calculations may be used when necessary. For best results, the computations should be done with at least 60 significant bits; this requires double precision on the common 32-bit machines (e.g., VAX, Prime). To use the routines on larger machines (e.g., CDC, CRAY-1), only the implicit double precision statements in the subroutines and the type declaration for the functions need be changed.

The usual entry point to the algorithm yields the automatic integration of the Hankel transform to a specified accuracy by increasing the order of the quadrature rule until a convergence criterion is satisfied. The call has the form

CALL BESAUT (BESR, BESI, ORDER, NL, NU, R, FUNCT, RERR, AERR, NPCS, NEW, IERR)

where the variable types follow the usual Fortran conventions. The input arguments are

ORDER = order of the Bessel function.

NL, NU = lower and upper limit for the quadrature order, where NL, NU = 1, ... 7 selects 3, 7, 15, 31, 63, 127, and 255 point rules. These parameters are provided to allow user control of the cost of the integration. For fully automatic integration, set NL = 1 and NU = 7.

R = argument of the Bessel function where $R \geq 0$. This corresponds to the range ρ in (1).

FUNCT = subroutine to return the kernel function. FUNCT must be declared EXTERNAL in the calling program. The access is of the form

CALL FUNCT (X, YR, YI)

where X is the wavenumber [k in equation (1)] supplied by BESAUT and YR and YI are the returned real and imaginary parts of the kernel function. Any additional parameters should be passed to FUNCT in a common block.

RERR, AERR = relative and absolute error parameters used for termination. The algorithm controls the local error, defined as the difference between the result of two sequential quadrature orders, so that for termination

ABS (local error) $\leq$ RERR* ABS (RESULT) + AERR

where RESULT is the answer at the higher quadrature order. The criterion must be satisfied separately by the real and imaginary parts. Setting AERR = 0 results in a pure relative error test which may fail if the answer is very small. For a mixed error test, the criterion corresponds roughly to a relative error test when the solution is much larger than AERR and to an absolute error test when the solution is smaller than AERR. Note that both RERR and AERR must be nonnegative.

NPCS = number of pieces into which the partial integration is divided at each step. This parameter is usually set to 1. For very small values of R, when the kernel changes much more rapidly than the Bessel functions, it may be necessary to subdivide each interval by increasing NPCS from 1. This should be done with caution, as it could result in excessive usage of computer time. Note that NPCS $\geq$ 1.

NEW = parameter to indicate the storage mode for the integral evaluations. NEW = 1 indicates the first call to BESAUT with a particular kernel function so that the product of the kernel and Bessel functions is calculated and saved for each abscissa value and partial integration. NEW = 2 indicates that a previously saved result is to be used, and only additional abscissa values and partial integrations are computed explicitly. This allows algebraically related kernels to be treated without reinitializing. The storage arrangement is described below.

The output arguments are

BESR, BESI = real and imaginary parts of the Hankel transform.

IERR = error flag, where IERR = 0 indicates a normal return and IERR = 1 indicates a failure to converge at the highest quadrature order. The solution obtained at the highest order is contained in BESR and BESI when IERR = 1.

An additional entry point, BESTRN, is described in the code contained in Appendix A. This subroutine returns the Hankel transform at a fixed Gauss order and has the additional capability of explicit calculation of the transform over an initial interval, allowing the user to defeat the continued fraction convergence method when desired.

The saved kernel abscissas and values are stored in COMMON/BESINT/with the form

```
DIMENSION NW(NTERM), KARG(255, NTERM),
  KERN(510, NTERM)
COMMON/BESINT/NK,
              NP, NPS, KARG, KERN
```

where NTERM is a parameter set in the code to keep storage requirements reasonable; it may be altered at compile time. The arrays KARG and KERN must be declared REAL or DOUBLE PRECISION. The abscissa values are stored in KARG (I, J), I = 1, ... 2** (NK (J) + 1) − 1, J = 1, ... NPS and the corresponding real and imaginary parts of the integrands are stored in KERN (I, J), KERN (I + 1, J), I = 1, ... 2* (2** (NK (J) + 2) − 1), J = 1, ... NPS, where the index I refers to the abscissa points for the quadrature rule arranged in the order used and J refers to the number of the partial integrands as in equation (2). The code automatically stops saving the result when storage is full, avoiding memory access problems. The treatment of related kernels is illustrated in the driver program of Appendix B.

A second common block, COMMON/TEST/, may be used to access run statistics. There are three integer parameters in the block, containing the highest quadrature order used, the total number of calls to the kernel function routine, and the number of partial integrations required to achieve convergence.

In addition to the two subroutines listed above, there are six other functions or subroutines in the algorithm. These include three subroutines not normally accessed directly by the user

BESQUD computes the integral (2) over a specified interval at a specified quadrature order,

PADECF sums a complex series using the continued fraction algorithm of Hänggi et al (1978),

CF computes a continued fraction starting at the bottom in a numerically stable fashion,

and three functions that may be user supplied

JBESS (X, ORDER) returns the Bessel function of the first kind of order ORDER at argument X. This should be interfaced to standard library routines, e.g. IMSL or SLATEC. Note that JBESS is a REAL or DOUBLE PRECISION function,

ZEROJ (N, ORDER) returns the Nth zero of the Bessel function of the first kind of order ORDER. The ORDER = 0 and 1 case, computed using an asymptotic expansion, is contained in the routines in Appendix A,

DOT (N, X1, INCL, X2, INC2) returns the dot product of two vectors with variable incrementing allowed. The BLAS routine SDOT (Lawson et al, 1979) should be used on machines where single precision arithmetic of sufficient accuracy is available. A plug-in double precision replacement is contained in Appendix A.

EXAMPLES AND APPLICATIONS

A sequence of eight Hankel transforms of elementary functions was selected to demonstrate the features and capabilities of the algorithm; a driver program incorporating them is contained in Appendix B. The integrals are separated into four types, starting with the rapidly convergent forms

$$\int_0^\infty dk\, ke^{-\alpha k^2} J_0(k\rho) = \frac{e^{-\rho^2/4\alpha}}{2\alpha} \tag{6}$$

$$\int_0^\infty dk\, e^{-k} J_1(k\rho) = \frac{\sqrt{\rho^2+1}-1}{\rho\sqrt{\rho^2+1}} \tag{7}$$

and including the slowly convergent integrals

$$\int_0^\infty dk\, J_0(k\rho) = \frac{1}{\rho} \tag{8}$$

$$\int_0^\infty dk\, \frac{k}{\sqrt{k^2+\alpha^2}} J_0(k\rho) = \frac{e^{-\alpha\rho}}{\rho}. \tag{9}$$

The algebraically divergent types

$$\int_0^\infty dk\, kJ_0(k\rho) = 0 \tag{10}$$

$$\int_0^\infty dk\, k\sqrt{k^2+\alpha^2} J_0(k\rho) = \frac{e^{-\alpha\rho}}{\rho^3}(\alpha\rho+1) \tag{11}$$

and the oscillatory kernel forms

$$\int_0^\infty dk \cos kJ_1(k\rho) = \begin{cases} \dfrac{\sqrt{1-\rho^2}-1}{\rho\sqrt{1-\rho^2}} & (\rho \geq 1) \\ \dfrac{1}{\rho} & \end{cases} \tag{12}$$

$$\int_0^\infty dk\, \frac{\cos k}{k} J_1(k\rho) = \begin{cases} 0 & \\ \dfrac{\sqrt{\rho^2-1}}{\rho} & (\rho > 1), \end{cases} \tag{13}$$

$$(\rho < 1) \qquad (\rho \leq 1)$$

where $\alpha = (1+i)/\sqrt{2}$. Only the first pair of integrals (6)–(7) can be handled by the digital filter method, and the divergent pair (10)–(11) can only be numerically integrated using an analytic continuation algorithm such as the continued fraction expansion. The last three pairs of integrals are algebraically related, and the appropriate program feature is used in the driver of Appendix B.

Results for the eight test integrals at short, moderate, and long range for two values of the relative error parameter are shown in Tables 1 and 2. The columns show the integral number corresponding to integrals (6)–(13), the error flag IERR returned by BESAUT, the Bessel function argument R, the NEW parameters passed to BESAUT, the relative error RERR, the real and imaginary parts of the numerical result, the real and imaginary parts of the analytic result, the number of the quadrature rule used at the last iteration, the total number of calls to FUNCT, and the number of partial integrations used in the continued fraction algorithm at the last iteration. The absolute error AERR is always taken to be 1000 times smaller than the relative error.

Table 1 contains the results for RERR = 10^{-5}, which are comparable in accuracy to ZHANKS, where that algorithm is applicable (Anderson, 1979). Convergence is easily achieved for all of the cases, with the two rapidly convergent integrals requiring the fewest number of partial integrations and the oscillatory kernel cases needing the most. In general, a higher order quadrature rule is required for small values of the range R; this is accomplished automatically by the code. Note that the divergent integrals are easily handled, requiring a comparable amount of computer time to the slowly convergent or oscillatory integrals. Integrals (6) and (7) were included in the test program for ZHANKS in Anderson (1979), and typically required 150 kernel function evaluations for similar accuracy to the results in Table 1. At moderate values of the parameter R, the direct method requires only slightly more computer time.

Table 2 shows similar computations with RERR = 10^{-10}, a far more stringent requirement with a concomitant increase in computational overhead. Convergence was achieved for all cases save the rapidly divergent form (11) at the shortest range. This is caused by a lack of sufficient numerical precision in the

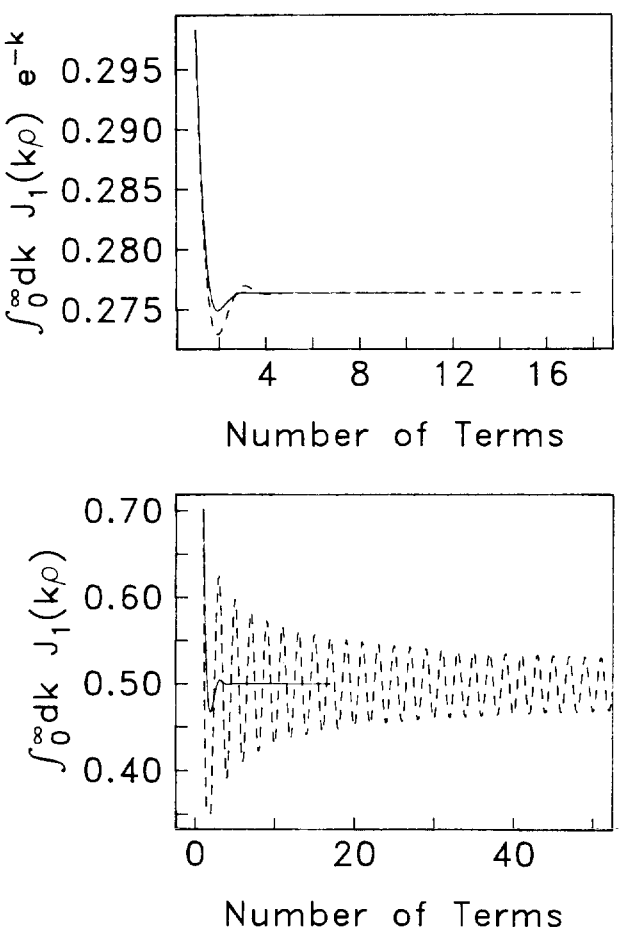

FIG. 1. The top plot shows the convergence behavior of the continued fraction algorithm (solid) and direct summation of the partial integrations (dashed) for the rapidly convergent integral (7) using a relative error requirement of 10^{-12}. The continued fraction expansion results in more rapid convergence by a factor of over 50 percent. The bottom plot shows similar results for the slowly convergent integral (14). The continued fraction expansion has converged after 18 terms in the series of partial integrations, while the direct sum has not converged after 100 terms.

Table 1.

NN	IERR	R	NEW	RERR	NUMERR	NUMERI	EXACTR	EXACTI	NG	NF	NI
1	0	0.05	1	0.1E-05	0.3535533215	-0.3532409596	0.3535533276	-0.3532409657	7	510	2
2	0	0.05	1	0.1E-05	0.2495322244E-01	0.0000000000E+00	0.2495322244E-01	0.0000000000E+00	5	126	2
3	0	0.05	1	0.1E-05	20.00000018	0.0000000000E+00	20.00000000	0.0000000000E+00	3	165	11
4	0	0.05	2	0.1E-05	19.29318262	-0.6824013551	19.29318266	-0.6824013839	6	1365	12
5	0	0.05	1	0.1E-05	-0.3157985989E-09	0.0000000000E+00	0.0000000000E+00	0.0000000000E+00	3	255	17
6	0	0.05	2	0.1E-05	-7999.770418	9.764356001	-7999.770489	9.764356133	6	1456	13
7	0	0.05	1	0.1E-05	-0.2504697217E-01	0.0000000000E+00	-0.2504697287E-01	0.0000000000E+00	6	1426	11
8	0	0.05	2	0.1E-05	-0.2117880160E-09	0.0000000000E+00	0.0000000000E+00	0.0000000000E+00	6	0	8
1	0	2.00	1	0.1E-05	0.2457791604	-0.1928180254E-01	0.2457791619	-0.1828179962E-01	4	155	5
2	0	2.00	1	0.1E-05	0.2763932025	0.0000000000E+00	0.2763932023	0.0000000000E+00	3	105	7
3	0	2.00	1	0.1E-05	0.5000000045	0.0000000000E+00	0.5000000000	0.0000000000E+00	3	165	11
4	0	2.00	2	0.1E-05	0.1895626011E-01	-0.1200712142	0.1895625755E-01	-0.1200712131	4	207	12
5	0	2.00	1	0.1E-05	0.1165897232E-08	0.0000000000E+00	0.0000000000E+00	0.0000000000E+00	3	180	12
6	0	2.00	2	0.1E-05	-0.5389270131E-01	0.6576733900E-01	-0.5389269888E-01	0.6576733929E-01	3	30	14
7	0	2.00	1	0.1E-05	0.4999999952	0.00000000000+00	0.5000000000	0.0000000000E+00	3	315	21
8	0	2.00	2	0.1E-05	0.8660254053	0.0000000000E+00	0.8660254038	0.0000000000E+00	3	15	22
1	0	100.00	1	0.1E-05	0.6951423145E-09	0.1121111965E-10	0.0000000000E+00	0.0000000000E+00	2	56	8
2	0	100.00	1	0.1E-05	0.9900004657E-02	0.0000000000E+00	0.9900005000E-02	0.0000000000E+00	3	150	10
3	0	100.00	1	0.1E-05	0.1000000303E-01	0.0000000000E+00	0.1000000000E-01	0.0000000000E+00	3	135	9
4	0	100.00	2	0.1E-05	0.4756845487E-09	-0.5298624909E-09	-0.4852101624E-34	-0.1952576719E-32	2	0	8
5	0	100.00	1	0.1E-05	0.5976479299E-09	0.0000000000E+00	0.0000000000E+00	0.0000000000E+00	2	56	8
6	0	100.00	2	0.1E-05	0.3208175355E-09	0.4330536062E-09	-0.1345885512E-34	0.1434515570E-34	2	0	8
7	0	100.00	1	0.1E-05	0.9999999879E-02	0.0000000000E+00	0.1000000000E-01	0.0000000000E+00	3	150	10
8	0	100.00	2	0.1E-05	0.9999499985	0.0000000000E+00	0.9999499987	0.0000000000E+00	3	30	12

Table 2.

NN	IERR	R	NEW	RERR	NUMERR	NUMERI	EXACTR	EXACTI	NG	NF	NI
1	0	0.05	1	0.1E-09	0.3535533215	-0.3532409596	0.3535533276	-0.3532409657	7	510	2
2	0	0.05	1	0.1E-09	0.2495322244E-01	0.0000000000E+00	0.2495322244E-01	0.0000000000E+00	5	126	2
3	0	0.05	1	0.1E-09	20.00000000	0.0000000000E+00	20.00000000	0.0000000000E+00	3	240	16
4	0	0.05	2	0.1E-09	19.29318268	-0.6824013726	19.29318266	-0.6824013839	7	3840	16
5	0	0.05	1	0.1E-09	-0.3874014568E-13	0.0000000000E+00	0.0000000000E+00	0.0000000000E+00	4	682	22
6	1	0.05	2	0.1E-09	-7999.770490	9.764380040	-7999.770489	9.764356133	7	24818	100
7	0	0.05	1	0.1E-09	-0.2504697287E-01	0.0000000000E+00	-0.2504697287E-01	0.0000000000E+00	6	2166	17
8	0	0.05	2	0.1E-09	0.1311979782E-13	0.0000000000E+00	0.0000000000E+00	0.0000000000E+00	6	0	14
1	0	2.00	1	0.1E-09	0.2457791604	-0.1928180254E-01	0.2457791619	-0.1928179962E-01	4	186	6
2	0	2.00	1	0.1E-09	0.2763932022	0.0000000000E+00	0.2763932023	0.0000000000E+00	4	310	10
3	0	2.00	1	0.1E-09	0.5000000000	0.0000000000E+00	0.5000000000	0.0000000000E+00	3	240	16
4	0	2.00	2	0.1E-09	0.1895626091E-01	-0.1200712156	0.1895625755E-01	-0.1200712131	4	287	17
5	0	2.00	1	0.1E-09	0.2580024523E-13	0.0000000000E+00	0.0000000000E+00	0.0000000000E+00	4	558	18
6	0	2.00	2	0.1E-09	-0.5389270093E-01	0.6576733896E-01	-0.5389269888E-01	0.6576733929E-01	4	31	19
7	0	2.00	1	0.1E-09	0.5000000000	0.0000000000E+00	0.5000000000	0.0000000000E+00	4	961	31
8	0	2.00	2	0.1E-09	0.8660254038	0.0000000000E+00	0.8660254038	0.0000000000E+00	4	31	32
1	0	100.00	1	0.1E-09	-0.1624977841E-12	0.9647859905E-13	0.0000000000E-00	0.0000000000E+00	3	195	13
2	0	100.00	1	0.1E-09	0.9900005000E-02	0.0000000000E+00	0.9900005000E-02	0.0000000000E+00	3	225	15
3	0	100.00	1	0.1E-09	0.1000000000E-01	0.0000000000E+00	0.1000000000E-01	0.0000000000E+00	3	225	15
4	0	100.00	2	0.1E-09	0.2766694312E-12	0.1347755968E-13	-0.4852101624E-34	-0.1952576719E-32	3	0	14
5	0	100.00	1	0.1E-09	-0.9360270729E-13	0.0000000000E+00	0.0000000000E+00	0.0000000000E+00	3	195	13
6	0	100.00	2	0.1E-09	0.1643696453E-13	0.3603324315E-13	-0.1345885512E-34	0.1434515570E-34	3	15	14
7	0	100.00	1	0.1E-09	0.1000000000E-01	0.0000000000E+00	0.1000000000E-01	0.0000000000E+00	3	225	15
8	0	100.00	2	0.1E-09	0.9999499987	0.0000000000E+00	0.9999499987	0.0000000000E+00	3	45	18

continued fraction algorithm as larger and larger terms are added to the series, and better behavior would be obtained if this portion of the algorithm (subroutines PADECF and CF) used a longer word length (e.g., REAL * 16 on the VAX or DOUBLE PRECISION on a mainframe computer).

Figure 1 illustrates the power of the continued fraction expansion. The top plot shows the integral (7) with $R = 2$ and RERR = 10^{-12}, AERR = 10^{-15}. The solid line shows the continued fraction result, which converged after 11 terms, and the dashed line shows the direct sum of the partial integrands, which required 18 terms for identical convergence. The bottom part of the figure shows the slowly convergent integral

$$\int_0^x dk\, J_1(k\rho) = \frac{1}{\rho}, \tag{14}$$

where the continued fraction result converges after 18 terms, but the direct sum is alternating and approaches the correct result very slowly. It is likely that numerical round-off will prevent the latter from ever reaching convergence.

The application of this algorithm to a geophysical problem was discussed in Chave and Cox (1982) for a seafloor-based horizontal electric dipole (HED). The two fundamental electromagnetic modes for this type of source are nearly out of phase, necessitating high accuracy in the calculation of the electromagnetic fields. Correction for the finite length of real sources requires integration with a combination of Hankel transforms as integrands, another application with fairly stringent precision requirements. The direct integration algorithm of this paper has worked well in this application, and yields good user control of the computer time and numerical accuracy.

ACKNOWLEDGMENTS

This work was supported by the Office of Naval Research and by the National Science Foundation under grant OCE 81-10399.

REFERENCES

Anderson, W. L., 1979, Numerical integration of related Hankel transforms of orders 0 and 1 by adaptive digital filtering: Geophysics, v. 44, p. 1287–1305.

——— 1982, Fast Hankel transforms using related and lagged convolutions: ACM Trans. on Math. Software, v. 8, p. 344–368.

Baker, G. A., Jr., 1975, Essentials of Padé approximants: New York, Academic Press.

Bender, C. M., and Orszag, 1978, Advanced mathematical methods for scientists and engineers: New York, McGraw-Hill.

Chave, A. D., and Cox, C. S., 1982, Controlled electromagnetic sources for measuring electrical conductivity beneath the oceans, 1. forward problem and model study: J. Geophys. Res., v. 87, p. 5327–5338.

Cornille, P., 1972, Computation of Hankel transforms: SIAM Rev., v. 14, p. 278–285.

Ghosh, P. P., 1971, The application of linear filter theory to the direct interpretation of geoelectrical resistivity sounding measurements: Geophys. Prosp., v. 19, p. 192–217.

Hänggi, P., Roesel, F., and Trautmann, P., 1978, Continued fraction expansions in scattering theory and statistical non-equilibrium mechanics: Z. Naturforsch., v. 33a, p. 402–417.

——— 1980, Evaluation of infinite series by use of continued fraction expansions: a numerical study: J. Comp. Phys., v. 37, p. 252–258.

Lawson, C. L., Hanson, E. J., Kincaid, D. R., and Krogh, F. T., 1979, Basic linear algebra subprograms for Fortran usage: ACM Trans. on Math. Software, v. 6, p. 308–323.

Patterson, T. N. L., 1968, The optimum addition of points to quadrature formulae: Math. Comp., v. 22, p. 847–856.

Ralston, A., and Rabinowitz, P., 1978, A first course in numerical analysis, 2nd ed.: New York, McGraw-Hill.

Watson, G. N., 1962, A treatise on the theory of Bessel functions, 2nd ed.: Cambridge, Cambridge Univ. Press.

Appendix

```
      SUBROUTINE BESAUT(BESR,BESI,ORDER,NL,NU,R,FUNCT,RERR,AERR,
     $                  NPCS,NEW,IERR)
C
C     PERFORMS AUTOMATIC CALCULATION OF BESSEL TRANSFORM TO SPECIFIED
C     RELATIVE AND ABSOLUTE ERROR
C
C     ARGUMENT LIST:
C
C     BESR,BESI-REAL AND IMAGINARY PARTS RETURNED BY BESAUT
C     ORDER-ORDER OF THE BESSEL FUNCTION
C     NL-LOWER LIMIT FOR GAUSS ORDER TO START COMPUTATION
C     NU-UPPER LIMIT FOR GAUSS ORDER
C        NU,NL=1,...7 SELECTS 3,7,15,31,63,127,AND 255 POINT GAUSS
C        QUADRATURE BETWEEN THE ZERO CROSSINGS OF THE BESSEL FUNCTION
C     R-ARGUMENT OF THE BESSEL FUNCTION
C     FUNCT-EXTERNAL SUBROUTINE TO EVALUATE KERNEL FUNCTION AT
C           A GIVEN WAVENUMBER.
C           THE CALL IS OF THE FORM
C              CALL FUNCT(X,YR,YI)
C           WHERE X IS SUPPLIED BY BESQUD AND YR AND YI ARE THE RETURNED
C           REAL AND IMAGINARY PARTS. ALL OTHER ARGUMENTS TO FUNCT MUST BE
C           PASSED IN COMMON.
C     RERR,AERR-RELATIVE AND ABSOLUTE ERROR FOR TERMINATION
C           BESAUT TERMINATES WHEN INCREASING THE GAUSS ORDER DOES NOT
C           CHANGE THE RESULT BY MORE THAN RERR OR WHEN THE ABSOLUTE ERROR
C           IS LESS THAN AERR OR WHEN A GAUSS ORDER OF NU IS REACHED.
C     NPCS-NUMBER OF PIECES INTO WHICH EACH PARTIAL INTEGRAND IS DIVIDED.
C           ORDINARILY SET TO ONE. FOR VERY SMALL VALUES OF R WHERE
C           THE KERNEL FUNCTION IS APPRECIABLE ONLY OVER THE FIRST FEW
C           LOOPS OF THE BESSEL FUNCTION, NPCS MAY BE INCREASED TO ACHIEVE
C           REASONABLE ACCURACY.
C     NEW   IF NEW=1, THE INTEGRANDS ARE COMPUTED AND SAVED AT EACH GAUSS
C           ORDER. IF NEW=2, PREVIOUSLY COMPUTED INTEGRANDS ARE USED. NOTE
C           THAT ORDER,R, AND NPCS MUST NOT BE CHANGED WHEN SETTING NEW=2.
C     IERR-ERROR PARAMETER
C           IERR=0--NORMAL RETURN
C           IERR=1--RESULT NOT ACCURATE TO RERR DUE TO TOO LOW A GAUSS
C           ORDER OR CONVERGENCE NOT ACHIEVED IN BESTRN
C
C     SUBROUTINES REQUIRED:
C       BESTRN,BESQUD,JBESS,PADECF,CF,ZEROJ,DOT
C
C     A.CHAVE IGPP/UCSD
C
      IMPLICIT DOUBLE PRECISION (A-H,C-Z)
      DIMENSION XSUM(1)
      PARAMETER (NSUM=0)
C     COMMON /TEST/ CONTAINS RUN STATISTICS
C     NG=HIGHEST GAUSS ORDER USED
C     NF=TOTAL FUNCTION EVALUATIONS  FOR ALL CALLS TO BESTRN
C     NI=TOTAL NUMBER OF PARTIAL INTEGRANDS ON LAST CALL
C     TO BESTRN ONLY
      COMMON /TEST/NG,NF,NI
      EXTERNAL FUNCT
      NF=0
      IF(NL.GT.NU)THEN
        BESR=0.
        BESI=0.
        IERR=1
        RETURN
      ENDIF
      NW=MAX(NEW,1)
      CALL BESTRN(BESR,BESI,ORDER,NL,R,FUNCT,.1*RERR,.1*AERR,
     $            NPCS,XSUM,NSUM,NW,IERR)
      IF((IERR.NE.0).AND.(NL.EQ.7))THEN
        NG=NL
        RETURN
      ELSE
        OLDR=BESR
        OLDI=BESI
        DO 10 N=NL+1,NU
          CALL BESTRN(BESR,BESI,ORDER,N,R,FUNCT,.1*RERR,.1*AERR,
     $                NPCS,XSUM,NSUM,2,IERR)
          IF((IERR.NE.0).AND.(N.EQ.7))THEN
            BESR=OLDR
            BESI=OLDI
            NG=N
            RETURN
          ELSEIF((ABS(BESR-OLDR).LE.RERR*ABS(BESR)+AERR).AND.(ABS(BESI-
     $           OLDI).LE.RERR*ABS(BESI)+AERR))THEN
            NG=N
            RETURN
          ELSE
            OLDR=BESR
            OLDI=BESI
          ENDIF
   10   CONTINUE
      ENDIF
      NG=7
      IERR=1
      RETURN
      END
      SUBROUTINE BESTRN(BESR,BESI,ORDER,NG,R,FUNCT,RERR,AERR,NPCS,
     $                  XSUM,NSUM,NEW,IERR)
C
C     COMPUTES BESSEL TRANSFORM OF SPECIFIED ORDER DEFINED AS
C     INTEGRAL(FUNCT(X)*J-SUB-ORDER(X*R)*DX) FROM X=0 TO INFINITY
C     COMPUTATION IS ACHIEVED BY INTEGRATION BETWEEN THE ASYMPTOTIC
C     ZERO CROSSINGS OF THE BESSEL FUNCTION USING GAUSS QUALRATURE.
C     THE RESULTING SERIES OF PARTIAL INTEGRANDS IS SUMMED BY CALCULATING
C     THE PADE APPROXIMANTS TO SPEED UP CONVERGENCE.
C
C     ARGUMENT LIST:
C
C     BESR,BESI  REAL AND IMAGINARY PARTS RETURNED BY BESTRN
C     ORDER  ORDER OF THE BESSEL FUNCTION
C     NG  NUMBER OF GAUSS POINTS TO USE IN THE QUADRATURE ROUTINE.
C         NG=1 THROUGH 7 SELECTS 3,7,15,31,63,126,AND 255 TERMS.
C     R     ARGUMENT OF THE BESSEL FUNCTION
C     FUNCT   EXTERNAL ROUTINE TO EVALUATE KERNEL FUNCTION AT ARGUMENT
C             X. THE CALL IS OF THE FORM
C               CALL FUNCT(X,YR,YI)
C             WHERE YR AND YI ARE THE RETURNED REAL AND IMAGINARY PARTS.
C     RERR,AERR  SPECIFIED RELATIVE AND ABSOLUTE ERROR FOR THE
C             CALCULATION.  THE INTEGRATION
C             TERMINATES WHEN AN ADDITIONAL TERM DOES NOT CHANGE THE
C             RESULT BY MORE THAN RERR*RESULT+AERR
C     NPCS  NUMBER OF PIECES INTO WHICH EACH PARTIAL INTEGRAND IS DIVIDED,
```

```
C       ORIGINARILY SET TO ONE. FOR VERY SMALL VALUES OF RANGE WHERE
C       THE KERNEL FUNCTION IS APPRECIABLE ONLY OVER THE FIRST FEW
C       LOOPS OF THE BESSEL FUNCTION, NPCS MAY BE INCREASED TO ACHIEVE
C       REASONABLE ACCURACY. NOTE THAT NPCS AFFECTS ONLY THE PADE
C       SUM PORTION OF THE INTEGRATION, OVER X(NSUM) TO INFINITY.
C XSUM  VECTOR OF VALUES OF THE KERNEL ARGUMENT OF FUNCT FOR WHICH
C       EXPLICIT CALCULATION OF THE INTEGRAL IS DESIRED, SO THAT THE
C       INTEGRAL OVER 0 TO XSUM(NSUM) IS ADDED TO THE INTEGRAL OVER
C       XSUM(NSUM) TO INFINITY WITH THE PADE METHOD INVOKED ONLY FOR
C       THE LATTER. THIS ALLOWS THE PADE SUMMATION METHOD TO BE
C       OVERRIDDEN AND SOME TYPES OF SINGULARITIES TO BE HANDLED.
C NSUM  NUMBER OF VALUES IN XSUM, MAY BE ZERO.
C NEW   DETERMINES METHOD OF KERNEL CALCULATION
C       NEW=0 MEANS  CALCULATE BUT DO NOT SAVE INTEGRANDS
C       NEW=1 MEANS CALCULATE KERNEL BY CALLING FUNCT-SAVE KERNEL
C             TIMES BESSEL FUNCTION
C       NEW=2 MEANS USE SAVED KERNELS TIMES BESSEL FUNCTIONS IN
C             COMMON /BESINT/. NOTE THAT ORDER,R,NPCS,XSUM, AND
C             NSUM MAY NOT BE CHANGED WHEN SETTING NEW=2.
C IERR  ERROR PARAMETER
C       0 NORMAL RETURN-INTEGRAL CONVERGED
C       1 MEANS NO CONVERGENCE AFTER NSTOP TERMS IN THE PADE SUM
C
C SUBROUTINES REQUIRED:
C BESQUD,PADECF,CF,ZEROJ,DOT,JBESS
C
C A.CHAVE IGPP/UCSD
C
C NTERM IS MAXIMUM NUMBER OF BESSEL FUNCTION LOOPS STORED IF NEW.NE.0
C NSTOP IS MAXIMUM NUMBER OF PADE TERMS
C
      PARAMETER (NTERM=120, NSTOP=100)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION KARG,KERN,LASTR,LASTI
      DIMENSION KARG(255,NTERM),KERN(510,NTERM),SR(NSTOP),SI(NSTOP),
     $          NK(NTERM),XSUM(1)
C COMMON /TEST/ CONTAINS RUN STATISTICS
C NGAUSS=HIGHEST GAUSS ORDER USED
C NF=TOTAL FUNCTION EVALUATIONS FOR ALL CALLS TO BESTRN
C NI=TOTAL NUMBER OF PARTIAL INTEGRANDS USED IN PADECF ON LAST CALL
C TO BESTRN
      COMMON /TEST/NGAUSS,NF,NI
C IF NEW.GT.0, COMMON /BESINT/ CONTAINS THE KERNEL FUNCTION ARGUMENTS
C AND VALUES KARG(I,N),KERN(I,N) FOR N=1 TO NPS AND I=1 TO
C 2**(NK(N)+1)-1.
C KERN(I,N), KERN(I+1,N) CONTAIN THE REAL AND IMAGINARY PARTS OF THE
C KERNEL FUNCTION FOR THE CORRESPONDING ARGUMENT VALUE KARG(I,N).
C EACH VALUE OF N CORRESPONDS TO ONE ENTRY IN XSUM OR TO ONE INTERVAL
C BETWEEN ZERO CROSSINGS OF THE BESSEL FUNCTION DIVIDED BY NPCS.
C NOTE THAT UP TO NTERM VALUES ARE STORED THE INTEGRATION PROCEEDS
C FOR N.GT.NTERM AS IF NEW=0. NP IS AN INDEX PASSED INTERNALLY BY
C BESTRN.
      COMMON /BESINT/NK,NP,NPS,KARG,KERN
      EXTERNAL FUNCT
      IF(NEW.EQ.2)THEN
        NPO=NPS
      ELSE
        DO 5 I=1,NTERM
          NK(I)=0
    5   CONTINUE
        NPS=0
        NPO=NTERM
      ENDIF
C  CHECK FOR TRIVIAL CASE
      IF((ORDER.NE.0.0).AND.(R.EQ.0.))THEN
        BESR=0.
        BESI=0.
        IERR=0
        RETURN
      ENDIF
      NI=0
      NW=NEW
      NP=1
      NPB=1
      L=1
      B=0.0
      SUMR=0.0
      SUMI=0.0
      XSUMR=0.0
      XSUMI=0.0
      IF(NSUM.GT.0)THEN
C  COMPUTE BESSEL TRANSFORM EXPLICITLY ON (0,XSUM(NSUM))
        LASTR=0.0
        LASTI=0.0
        DO 10 N=1,NSUM
          IF((NW.EQ.2).AND.(NP.GT.NPO))NW=1
          IF(NP.GT.NTERM)NW=0
          A=B
          B=XSUM(N)
          CALL BESQUD(A,B,TERMR,TERMI,NG,NW,ORDER,R,FUNCT)
          XSUMR=XSUMR+TERMR
          XSUMI=XSUMI+TERMI
          IF((ABS(XSUMR-LASTR).LE.RERR*ABS(XSUMR)+AERR).AND.
     $       (ABS(XSUMI-LASTI).LE.RERR*ABS(XSUMI)+AERR))THEN
            BESR=XSUMR
            BESI=XSUMI
            IERR=0
            NPS=MAX(NP,NPS)
            RETURN
          ELSE
            NP=NP+1
            LASTR=XSUMR
            LASTI=XSUMI
          ENDIF
   10   CONTINUE
C  FIND FIRST ZERO CROSSING OF BESSEL FUNCTION BEYOND XSUM(NSUM)
   15   CONTINUE
        IF(ZEROJ(NPB,ORDER).GT.XSUM(NSUM)*R)GOTO 20
        NPB=NPB+1
        GOTO 15
      ENDIF
C  ENTRY POINT FOR PADE SUMMATION OF PARTIAL INTEGRANDS
   20 CONTINUE
      LASTR=0.0
      LASTI=0.0
      A=B
      B=ZEROJ(NPB,ORDER)/R
C  CALCULATE TERMS AND SUM WITH PADECF, QUITTING WHEN CONVERGENCE IS
```

```
C  OBTAINED
      DO 30 N=1,NSTOP
        IF(NPCS.EQ.1)THEN
          IF((NW.EC.2).AND.(NP.GT.NPO))NW=1
          IF(NP.GT.NTERM)NW=2
          CALL BESQUD(A,B,TERMR,TERMI,NG,NW,ORDER,R,FUNCT)
          NP=NP+1
        ELSE
          TERMR=0.
          TERMI=0.
          XINC=(B-A)/NPCS
          AA=A
          BB=A+XINC
          DO 25 I=1,NPCS
            IF((NW.EC.2).AND.(NP.GT.NPO))NW=1
            IF(NP.GT.NTERM)NW=0
            CALL BESQUD(AA,BB,TR,TI,NG,NW,ORDER,R,FUNCT)
            TERMR=TERMR+TR
            TERMI=TERMI+TI
            AA=BB
            BB=BB+XINC
            NP=NP+1
   25     CONTINUE
        ENDIF
        NI=NI+1
        SR(L)=TERMR
        SI(L)=TERMI
        CALL FALECF(SUMR,SUMI,SR,SI,L)
        IF((ABS(SUMR-LASTR).LE.RERR*ABS(SUMR)+AERR).AND.
     $     (ABS(SUMI-LASTI).LE.RERR*ABS(SUMI)+AERR))THEN
          BESR=XSUMR+SUMR
          BESI=XSUMI+SUMI
          IERR=0
          NPS=MAX(NP-1,NPS)
          RETURN
        ELSE
          LASTR=SUMR
          LASTI=SUMI
          NPB=NPB+1
          A=B
          B=ZEROJ(NPB,ORDER)/R
          L=L+1
        ENDIF
   30 CONTINUE
      BESR=XSUMR+SUMR
      BESI=XSUMI+SUMI
      IERR=1
      NPS=MAX(NP,NPS)
      RETURN
      END
      SUBROUTINE BESQUD(A,B,BESR,BESI,NG,NEW,ORDER,R,F)
C
C  THIS SUBROUTINE CALCULATES THE INTEGRAL OF F(X)*J-SUB-N(X*R)
C  OVER THE INTERVAL A TO B AT A SPECIFIED GAUSS ORDER
C  THE RESULT IS OBTAINED USING A SEQUENCE OF 1,3,7,15,31,63,
C  127, AND 255 POINT INTERLACING GAUSS FORMULAE SO THAT NO INTEGRAND
C  EVALUATIONS ARE WASTED. THE KERNEL FUNCTIONS MAY BE SAVED SO THAT
C  BESSEL TRANSFORMS OF SIMILAR KERNELS ARE COMPUTED WITHOUT NEW
C  EVALUATION OF THE KERNEL.
C
C  DETAILS ON THE FORMULAE ARE GIVEN IN 'THE OPTIMUM ADDITION OF POINTS
C  TO QUADRATURE FORMULAE' BY T.N.L. PATTERSON, MATHS.COMP.
C  22,847-856 (1968). GAUSS WEIGHTS TAKEN FROM
C  COMM. A.C.M. 16,694-699 (1973)
C
C ARGUMENT LIST:
C
C A       LOWER LIMIT OF INTEGRATION
C B       UPPER LIMIT OF INTEGRATION
C BESR,BESI RETURNED INTEGRAL VALUE REAL AND IMAGINARY PARTS
C NG NUMBER OF POINTS IN THE GAUSS FORMULA.  NG=1,...,7
C       SELECTS 3,7,15,31,63,127,AND 255 POINT QUADRATURE.
C NEW SELECTS METHOD OF KERNEL EVALUATION
C     NEW=0 CALCULATES KERNELS BY CALLING F - NOTHING SAVED
C     NEW=1 CALCULATES KERNELS BY CALLING F AND SAVES KERNEL TIMES
C           BESSEL FUNCTION IN COMMON /BESINT/
C     NEW=2 USES SAVED KERNEL TIMES BESSEL FUNCTIONS IN
C           COMMON /BESINT/
C ORDER ORDER OF THE BESSEL FUNCTION
C R     ARGUMENT OF THE BESSEL FUNCTION
C F     F(X) IS THE EXTERNAL INTEGRAND SUBROUTINE
C
C  A.CHAVE IGPP/UCSD
C
C  MAXIMUM NUMBER OF BESSEL FUNCTION LOOPS THAT CAN BE SAVED
      PARAMETER (NTERM=100)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION KARG,KERN,JPESS
      DIMENSION WT(254),WA(127),NWA(7),NWT(7),KARG(255,NTERM),
     $          KERN(510,NTERM),FUNCT(254),FR1(64),FI1(64),FR2(64),
     $          FI2(64),BES1(64),BES2(64),NK(NTERM)
C  COMMON /TEST/ CONTAINS RUN STATISTICS
C  NGAUSS=HIGHEST GAUSS ORDER USED
C  NF=TOTAL FUNCTION EVALUATIONS  FOR ALL CALLS TO BESTRN
C  NI=TOTAL NUMBER OF PARTIAL INTEGRANDS ON LAST CALL
C  TO BESTRN ONLY
      COMMON /TEST/NGAUSS,NF,NI
      COMMON /BESINT/NK,NP,NPS,KARG,KERN
C  STARTING ADDRESSES IN ARRAYS WT AND WA AT EACH ORDER
      DATA NWT/1,3,7,15,31,63,127/,NWA/1,2,4,8,16,32,64/
C  ARRAY WT CONTAINS GAUSS QUADRATURE WEIGHTS STORED AS NEEDED AT
C  EACH ORDER.  ARRAY WA CONTAINS THE CORRESPONDING ARGUMENT WEIGHTS.
      DATA (WT(I),I=1,20)/
     $0.555555555555555556D+00,0.888888888888888888889D+00,
     $0.26848808986833344073D+00,0.104656226026467265519D+00,
     $0.401397414775962222291D+00,0.450916538659547114235D+00,
     $0.134415255243784220036D+00,0.516032829970797396971E-01,
     $0.20062852937698902103D+00,0.170017196299402603390D-01,
     $0.929271953151245376786D-01,0.171511909136391380779D+00,
     $0.219156858401587496401D+00,0.225510499798206687739D+00,
     $0.672077542959907035404D-01,0.259075980961766535654E-01,
     $0.100314278611795578770D+00,0.843456573932110624630D-02,
     $0.464628932617579865414E-01,0.857559204499903511540-01/
      DATA (WT(I),I=21,40)/
     $0.109578421055924638240+00,0.254478079156187441540-02,
     $0.164460498543878109340E-01,0.359571033071293220970-01,
     $0.569795094941233574120-01,0.768796204990035310430E-01,
     $0.936271099812644736170-01,0.105669893580234809740+00,
     $0.111956873020953456880E+00,0.112755256720768691610+00,
```

```
$0.336038771482077305420-01,0.129038460100351265626D-01,
$0.501571393058995374140-01,0.421763044155885483911-02,
$0.232314466399102669443D-01,0.428779620250077344930-01,
$0.547892105279628650321-01,0.126515655623006801140-02,
$0.822300795723592966930-02,0.179785515681282703330-01/
 DATA (WT(I),I=41,60)/
$0.284897547458335486130-01,0.364398102494555320390-01,
$0.468135549906280124030-01,0.528349467901165198621-01,
$0.559784361047631940800-01,0.363221481845530659690-03,
$0.257904979468566827241-02,0.611550682211724633970-02,
$0.104982469096213218980-01,0.154067504665594978020-01,
$0.205942339159127111490-01,0.258696793227214746911D-01,
$0.310735511116879648800-01,0.360644327807825726400-01,
$0.407155101169443189341-01,0.449145316536321974140-01,
$0.485643304066731987160-01,0.515832539520484587771-01,
$0.539054993352660639270-01,0.554814043566559639880-01/
 DATA (WT(I),I=61,80)/
$0.562776998312543012730-01,0.563776283603847173880-01,
$0.168019385741038652710-01,0.645190005017573692280-02,
$0.250785696529497687070-01,0.210881524572663287930-02,
$0.116157233199551347270-01,0.214389800125036672461-01,
$0.273946052639814325160-01,0.632607319362633544220-03,
$0.411150397865469304720-02,0.898927578406413572330-02,
$0.142448773729167743060-01,0.192199051247277660190-01,
$0.234067774953140062010-01,0.264174733950582599310-01,
$0.279892182552381597040-01,0.180739564445388357820-03,
$0.128952408261041739210-02,0.305775341017553113610-02/
 DATA (WT(I),I=81,100)/
$0.524912345480885912510-02,0.770337523323797418480-02,
$0.102971169579563555240-01,0.129348396636607373455D-01,
$0.155367755558439824400-01,0.180322163903912863200-01,
$0.203577550584721594670-01,0.224572658268168987070-01,
$0.242821652033365993580-01,0.257916269760242293880-01,
$0.269527496676330319630-01,0.277407021782796811994D-01,
$0.281388499156271506360-01,0.505360952078625176250-04,
$0.377746646326984660270-03,0.938369848542381500790-03,
$0.168114286542146990630-02,0.256676494379402037310-02,
$0.357286278351729964940-02,0.467105037211432174740-02/
 DATA (WT(I),I=101,120)/
$0.584344967585356395076D-02,0.707248999543355546800-02,
$0.834283875396815770560-02,0.964117729702536695300-02,
$0.109557333878379016480-01,0.122758305600827700870-01,
$0.135915710097655467900-01,0.148936416648151820350-01,
$0.161732187295777199420-01,0.174219301594641737470-01,
$0.186318462561387901860-01,0.197954950480974994880-01,
$0.209058514458120238520-01,0.219563663053178249390-01,
$0.229406642293877457610-01,0.238540521060385400800-01,
$0.246905247444876769090-01,0.254457699654647656130-01,
$0.261156733767060976800-01,0.266966229274503599060-01/
 DATA (WT(I),I=121,140)/
$0.271855132296247918190-01,0.275797495664818730350-01,
$0.278772514766137016090-01,0.280764557938172466070-01,
$0.281763190330166021310-01,0.281868141801923586940-01,
$0.840096928705193263540-02,0.322595002508786846140-02,
$0.125392848264748843530-01,0.105440762286331677220-02,
$0.580786165997756736350-02,0.107194900062519336230-01,
$0.136973026319907162580-01,0.316303660822264476890-03,
$0.205575198932734652360-02,0.449463789203206786160-02,
$0.712243868645838715320-02,0.960995256236388300970-02,
$0.117033888747657003101D-21,0.132087366975291299660-01/
 DATA (WT(I),I=141,160)/
$0.139946091276190798520-01,0.903727346587511492610-04,
$0.644762041305724779330-03,0.152887670508776556840-02,
$0.262456172740442956260-02,0.385166761663987092410-02,
$0.514855847897817776180-02,0.646741983182368672740-02,
$0.776838777792199122000-02,0.901610819519564316000-02,
$0.101788775292360797330-01,0.112286329134080493540-01,
$0.121410826016682996790-01,0.128958134880121146940-01,
$0.134763748338161515982D-01,0.138703510891398409970-01,
$0.140694249578135753180-01,0.251578703842806610890-04,
$0.188873264506504913660-03,0.469184924247850409751-03/
 DATA (WT(I),I=161,180)/
$0.640571432710722463651-03,0.128438247189701017680-02,
$0.178644639175864982470-02,0.233552518605716087370-02,
$0.292172493791781975380-02,0.353624499771677773400-02,
$0.417141937698407885280-02,0.482058886485126834760-02,
$0.547786669391895082400-02,0.613791529804138504350-02,
$0.679578550488277339480-02,0.744662283240759101740-02,
$0.808660936478885997100-02,0.871096507973208687360-02,
$0.931592412806939509320-02,0.989774752404867497440D-02,
$0.104529257229060119260-01,0.109781831526589124700-01,
$0.114704821146938743800-01,0.119270260530192700400-01/
 DATA (WT(I),I=181,200)/
$0.123452623722438384550-01,0.127228849827323829060-01,
$0.130578366883530488400-01,0.133483114637251799530-01,
$0.135927566148123959100-01,0.137898747832409366170-01,
$0.139386257383068508040-01,0.140382278969086233030-01,
$0.140881595916508301065D-01,0.693793643241082671700-05,
$0.532752936697806131250-04,0.135754910949228719730-03,
$0.249212400482997294020-03,0.389745284473282293220-03,
$0.554295314930374714920-03,0.740262804244503330460-03,
$0.945361516858525382460-03,0.116748411742995940770-02,
$0.140492799565514464270-02,0.165611272815445260520-02/
 DATA (WT(I),I=201,220)/
$0.191971297101387241250-02,0.219440692536383883880-02,
$0.247695822665756793070-02,0.277219576459345099400-02,
$0.307301843470257832340-02,0.338039799108692038230-02,
$0.369337791702565081830-02,0.401106872407502339890-02,
$0.433264096809298285450-02,0.465731729975685477730-02,
$0.498436456476553860120-02,0.531308660518705656630-02,
$0.564281810138444415850-02,0.597291956550816580490-02,
$0.630277344908575871720-02,0.663178124290187894100-02,
$0.695936140939042293940-02,0.728494798055380706390-02,
$0.760798966571905658320-02,0.792794933422948491103D-02/
 DATA (WT(I),I=221,240)/
$0.824430376303286880306D-02,0.855654356130768961920-02,
$0.886417320948249426410-02,0.916671116356078840670-02,
$0.946368999383006529430-02,0.975465653631741146110-02,
$0.100391722044055840798D-01,0.103168123309476216820-01,
$0.105871679048851979310-01,0.108498440893373140990-01,
$0.111044611340069265370-01,0.113506543159805956020-01,
$0.115880740330439525680-01,0.118163858908302235760-01,
$0.120352707852795626300-01,0.122444249816119858990-01,
$0.124435601907140352630-01,0.126324036435422078650-01,
$0.128106981638773619670-01,0.129782022395373992860-01/
 DATA (WT(I),I=241,254)/
$0.131346900919601528360-01,0.132799517439305306500-01,
$0.134137930851100985130-01,0.135360359349562136140-01,
```

```
     $0.136465181625712914286D-01,0.137459034430018966632D-01,
     $0.138316319095064286761-01,0.139060196013254612641-01,
     $0.139681588065169385161-01,0.140179680394566088810D-01,
     $0.140553822726499642772D-01,0.140803519625536613250-01,
     $0.140928452769162406355I-01,0.140944077090096179347D-01/
     DATA (WA(I),I=1,40)/
     $0.774596669241483377241+00,0.960491268780802028342D+00,
     $0.434243719346802558001D+00,0.993831963212755022211+00,
     $0.888459232872256998890+00,0.621102946737226402941D+00,
     $0.223366866128966881631D+00,0.999098124967667597661D+00,
     $0.381531145537401066671+00,0.929554857429740056671D+00,
     $0.836722938616866873554D+00,0.702496206491527077611D+00,
     $0.531319743644375622797D+00,0.333113539325797683309D+00,
     $0.112486694313316662575D+00,0.999872888120357611941D+00,
     $0.897206259377221959081+00,0.966684757547429479941D+00,
     $0.972182874748581796560+00,0.946342858373402905151D+00/
     DATA (WA(I),I=21,40)/
     $0.912371156957004292501D+00,0.866390793819369047715D+00,
     $0.826942531952217611801D+00,0.739756044352694758681D+00,
     $0.652929666224780595461D+00,0.577195710052045814841D+00,
     $0.483618226945641027561+00,0.383359324198730346921D+00,
     $0.277749822021824315071+00,0.168235251552207464981D+00,
     $0.563443130465927899721D-01,0.999098243035489159858D+00,
     $0.999596795671910661325I+00,0.998316635318407392531D+00,
     $0.995724124698407188511+00,0.991495721178106132401+00,
     $0.985371499585203711111+00,0.977141514639705714161D+00,
     $0.966637051558416567091D+00,0.953730004642576113641D+00/
     DATA (WA(I),I=41,60)/
     $0.938320397779592883651D+00,0.920340025470012420731D+00,
     $0.899744699776940036641+00,0.876513414484705269741D+00,
     $0.850644459476835027976I+00,0.822156254364982407371D+00,
     $0.791084933799848361431D+00,0.757483966380513637931D+00,
     $0.721423265370038915461D+00,0.682957431091079228091D+00,
     $0.642276642509759513771+00,0.599403930242242892971D+00,
     $0.554495132631932548871D+00,0.507667775753371660215D+00,
     $0.459132011989832332871D+00,0.408979821222688672411D+00,
     $0.357403837831532152361D+00,0.304576441556714043341D+00,
     $0.250676730303483176611+00,0.195897502711100153921D+00/
     DATA (WA(I),I=61,80)/
     $0.140424233152560174591+00,0.844540400837108837101D-01,
     $0.281646489497426943391D-01,0.999997596377474646721D+00,
     $0.999943996207054375761+00,0.999760490924432047331D+00,
     $0.999360338025023561931D+00,0.998745614468095114701D+00,
     $0.997805354495957274561+00,0.996514145914890273851D+00,
     $0.995483150260062100051D+00,0.992721344282788615331D+00,
     $0.990151370420771591981+00,0.987022527954034467191D+00,
     $0.983511657578632728761+00,0.979406261670882683811D+00,
     $0.974734459752402667761+00,0.969486465950245923177I+00,
     $0.963644621569812132521D+00,0.957188216109860962741D+00/
     DATA (WA(I),I=81,100)/
     $0.950115297521294876561+00,0.942411565191063059811+00,
     $0.934066436157725788001D+00,0.925078932907075652361D+00,
     $0.915437587155765040641+00,0.905140358813261595191D+00,
     $0.894186455633555902286I+00,0.882568840247341906841+00,
     $0.870293055548113905851D+00,0.857358310886232156531+00,
     $0.843766882672708601041D+00,0.829522194637401400181D+00,
     $0.814622787655137413411+00,0.799092290906084140180D+00,
     $0.782919394118283016391D+00,0.766117819303760990721D+00,
     $0.748696293616936602821D+00,0.730664521242181261331D+00,
     $0.712033155362252034591D+00,0.692813769779114702891D+00/
     DATA (WA(I),I=101,120)/
     $0.673018830230418479201D+00,0.652661665410017496101D+00,
     $0.631756437711942304111+00,0.610318113715186400161D+00,
     $0.588362434447662541431D+00,0.565905885423654422621+00,
     $0.542965666498311490491D+00,0.519559661537457021991D+00,
     $0.495706407918761460171D+00,0.471425065871658876931D+00,
     $0.446735387662028473741D+00,0.421657686626163300061D+00,
     $0.396212806057615939181D+00,0.370422065795007823014D+00,
     $0.344307341599438022781D+00,0.317890812068476683181D+00,
     $0.291195148518246681961D+00,0.264243372410926761941D+00,
     $0.237058845589829727211D+00,0.209665238243181194771D+00/
     DATA (WA(I),I=121,127)/
     $0.182086499675925219825I+00,0.154346811481378108691D+00,
     $0.126470584372301966851D+00,0.984823965981192020901I-01,
     $0.704069760428551790631D-01,0.422691647653636032121D-01,
     $0.140938864107824626141D-01/
C    CHECK FOR TRIVIAL CASE
      IF(A.GE.B)THEN
        BESR=0.
        BESI=0.
        RTURN
      ENDIF
      IF(NEW.EQ.2)GOTO 200
C    SCALE FACTORS
      SUM=(B+A)/2.
      DIFF=(B-A)/2.
C    ONE POINT GAUSS
      CALL F(SUM,FZEROR,FZEROI)
      NF=NF+1
      BESR=JBESS(SUM*R,ORDER)
      FZEROR=BESR*FZEROR
      FZEROI=BESR*FZEROI
      IF(NEW.EQ.1)THEN
        KARG(1,NP)=SUM
        KERN(1,NP)=FZEROR
        KERN(2,NP)=FZEROI
        LA=2
        LK=3
      ENDIF
      N=1
      KN=1
    5 CONTINUE
C    STEP THROUGH GAUSS ORDERS
      DO 40 K=KN,NG
C    COMPUTE NEW FUNCTION VALUES
        NA=NWA(K)
        DO 10 J=1,NWA(K)
          X=WA(NA)*DIFF
          NA=NA+1
          SUMP=SUM+X
          SUMM=SUM-X
          CALL F(SUMP,FR1(J),FI1(J))
          CALL F(SUMM,FR2(J),FI2(J))
          NF=NF+2
          BES1(J)=JBESS(SUMP*R,ORDER)
          BES2(J)=JBESS(SUMM*R,ORDER)
          IF(NEW.GE.1)THEN
            KARG(LA,NP)=SUMP
```

```
            KARG(LA+1,NP)=SUMM
            LA=LA+2
          ENDIF
   10   CONTINUE
C  COMPUTE PRODUCTS OF KERNELS AND BESSEL FUNCTIONS
        DO 20 J=1,NWA(K)
          FR1(J)=BES1(J)*FR1(J)
          FI1(J)=BES1(J)*FI1(J)
          FR2(J)=BES2(J)*FR2(J)
          FI2(J)=BES2(J)*FI2(J)
          FUNCT(N)=FR1(J)+FR2(J)
          FUNCT(N+1)=FI1(J)+FI2(J)
          N=N+2
   20   CONTINUE
        IF(NEW.GE.1)THEN
          DO 30 J=1,NWA(K)
            KERN(LK,NP)=FR1(J)
            KERN(LK+1,NP)=FI1(J)
            KERN(LK+2,NP)=FR2(J)
            KERN(LK+3,NP)=FI2(J)
            LK=LK+4
   30     CONTINUE
        ENDIF
   40 CONTINUE
C  COMPUTE DOT PRODUCT OF WEIGHTS WITH INTEGRAND VALUES
      NW=NWT(NG)
C  DOT SHOULD BE REPLACED WITH SDOT FROM SCILIB ON CRAY-1 IF D.P. IS
C  NOT USED
      ACUMR=DOT(NW,WT(NW),1,FUNCT(1),2)
      ACUMI=DOT(NW,WT(NW),1,FUNCT(2),2)
      BESR=(ACUMR+WT(2*NW)*FZEROR)*DIFF
      BESI=(ACUMI+WT(2*NW)*FZEROI)*DIFF
      IF(NP.LE.NTERM)NK(NP)=NG
      RETURN
  200 CONTINUE
C  CONSTRUCT FUNCT FROM SAVED KERNELS
      DIFF=(B-A)/2.
      FZEROR=KERN(1,NP)
      FZEROI=KERN(2,NP)
      K=MIN(NK(NP),NG)
      NW=NWT(K)
      LK=3
      DO 210 N=1,2*NW,2
        FUNCT(N)=KERN(LK,NP)+KERN(LK+2,NP)
        FUNCT(N+1)=KERN(LK+1,NP)+KERN(LK+3,NP)
        LK=LK+4
  210 CONTINUE
      IF(NK(NP).GE.NG)THEN
C  NO ADDITIONAL ORDERS REQUIRED-COMPUTE DOT PRODUCT OF WEIGHTS WITH
C  INTEGRAND VALUES AND RETURN
        ACUMR=DOT(NW,WT(NW),1,FUNCT(1),2)
        ACUMI=DOT(NW,WT(NW),1,FUNCT(2),2)
        BESR=(ACUMR+WT(2*NW)*FZEROR)*DIFF
        BESI=(ACUMI+WT(2*NW)*FZEROI)*DIFF
        RETURN
      ELSE
C  COMPUTE ADDITIONAL ORDERS BEFORE TAKING DOT PRODUCT
        SUM=(B+A)/2.
        KN=K+1
        LA=LK/2+1
        GOTO 5
      ENDIF
      END
      SUBROUTINE PADECF(SUMR,SUMI,SR,SI,N)
C
C  COMPUTES SUM(S(I)) I=1,...N BY COMPUTATION OF PADE APPROXIMANT
C  USING CONTINUED FRACTION EXPANSION.  FUNCTION IS DESIGNED TO BE
C  CALLED SEQUENTIALLY AS N IS INCREMENTED FROM 1 TO ITS FINAL VALUE.
C  THE NTH CONTINUED FRACTION COEFFICIENT IS CALCULATED AND
C  STORED AND THE NTH CONVERGENT RETURNED.  IT IS UP TO THE USER TO
C  STOP THE CALCULATION WHEN THE DESIRED ACCURACY IS ACHIEVED.
C  ALGORITHM FROM HANGGI ET AL., Z.NATURFORSCH. 33A,402-417 (1977)
C  IN THEIR NOTATION, VECTORS CFCOR,CFCOI ARE LOWER CASE D, VECTORS DR,
C  DI ARE UPPER CASE D, VECTORS XR,XI ARE X, AND VECTORS SR,SI ARE S
C
C  A.CHAVE IGPP/UCSD
C
      PARAMETER (NC=100)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION SR(1),SI(1),XR(NC),XI(NC),DR(0:NC),DI(0:NC),CFCOR(NC),
     $          CFCOI(NC)
      DATA DR(0)/-1./,DI(0)/0./
      SAVE XR,XI,DR,DI,CFCOR,CFCOI
      IF(N.LT.3)THEN
C  INITIALIZE FOR RECURSIVE CALCULATIONS
        IF(N.EQ.1)THEN
          DO 10 I=1,NC
            XR(I)=0.
            XI(I)=0.
   10     CONTINUE
        ENDIF
        DR(N)=SR(N)
        DI(N)=SI(N)
        DENOM=DR(N-1)**2+DI(N-1)**2
        CFCOR(N)=-(DR(N-1)*DR(N)+DI(N-1)*DI(N))/DENOM
        CFCOI(N)=-(DR(N-1)*DI(N)-DR(N)*DI(N-1))/DENOM
        CALL CF(SUMR,SUMI,CFCOR,CFCOI,N)
        RETURN
      ELSE
        L=2*INT((N-1)/2)
C  UPDATE X VECTORS FOR RECURSIVE CALCULATION OF CF COEFFICIENTS
        DO 20 K=L,4,-2
          XR(K)=XR(K-1)+CFCOR(N-1)*XR(K-2)-CFCOI(N-1)*XI(K-2)
          XI(K)=XI(K-1)+CFCOR(N-1)*XI(K-2)+CFCOI(N-1)*XR(K-2)
   20   CONTINUE
        XR(2)=XR(1)+CFCOR(N-1)
        XI(2)=XI(1)+CFCOI(N-1)
C  INTERCHANGE OLD AND EVEN PARTS
        DO 30 K=1,MAX(1,L-1),2
          T1=XR(K)
          T2=XI(K)
          XR(K)=XR(K+1)
          XI(K)=XI(K+1)
          XR(K+1)=T1
          XI(K+1)=T2
   30   CONTINUE
C  COMPUTE FIRST COEFFICIENTS
        DR(N)=SR(N)
```

```
        II(N)=SI(N)
        DO 40 K=1,MAX(1,L/2)
          DR(N)=DR(N)+SR(N-K)*XR(2*K-1)-SI(N-K)*XI(2*K-1)
          DI(N)=DI(N)+SI(N-K)*XR(2*K-1)+SR(N-K)*XI(2*K-1)
   40   CONTINUE
C  COMPUTE NEW CF COEFFICIENT
        DENOM=DR(N-1)**2+DI(N-1)**2
        CFCOR(N)=-(DR(N)*DR(N-1)+DI(N)*DI(N-1))/DENOM
        CFCOI(N)=-(DR(N-1)*DI(N)-DR(N)*DI(N-1))/DENOM
C  EVALUATE CONTINUED FRACTION
        CALL CF(SUMR,SUMI,CFCOR,CFCOI,N)
        RETURN
      ENDIF
      END
      SUBROUTINE CF(RESR,RESI,CFCOR,CFCOI,N)
C
C  EVALUATES A COMPLEX CONTINUED FRACTION BY RECURSIVE DIVISION
C  STARTING AT THE BOTTOM. AS USED BY PADECF
C  RESR,RESI ARE REAL AND IMAGINARY PARTS RETURNED
C  CFCOR,CFCOI ARE REAL AND IMAGINARY VECTORS OF CONTINUED FRACTION
C  COEFFICIENTS
C
      PARAMETER (NC=100,NCM=NC-1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION CFCOR(1),CFCOI(1),ONE(NC)
      DATA ONE/0.,NCM*1./
      RESR=ONE(N)+CFCOR(N)
      RESI=CFCOI(N)
      DO 10 K=N-1,1,-1
        DENOM=RESR**2+RESI**2
        RESRO=RESR
        RESR=ONE(K)+(RESR*CFCOR(K)+RESI*CFCOI(K))/DENOM
        RESI=(RESRO*CFCOI(K)-RESI*CFCOR(K))/DENOM
   10 CONTINUE
      RETURN
      END
      DOUBLE PRECISION FUNCTION ZEROJ(NZERO,ORDER)
C
C  COMPUTES ZERO OF BESSEL FUNCTION OF THE FIRST KIND FROM
C  MCMAHON'S ASYMPTOTIC EXPANSION
C  NZERO-NUMBER OF THE ZERO
C  ORDER-ORDER OF THE BESSEL FUNCTION (0 OR 1)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (ZT1=-1.D0/8.D0,ZT2=124.D0/1536.D0,ZT3=-120928.D0/
     $           491520.D0,ZT4=40174316E.D0/220200960.D0)
      PARAMETER (OT1=3.D0/8.D0,OT2=-36.D0/1536.D0,OT3=113184.D0/
     $           491520.D0,OT4=-1951209.D0/220200960.D0)
      PARAMETER (PI=3.14159265358979300)
      IF(ORDER.EQ.0.0)THEN
        BETA=(NZERO-.25)*PI
        ZEROJ=BETA-ZT1/BETA-ZT2/BETA**3-ZT3/BETA**5-ZT4/BETA**7
      ELSEIF(ORDER.EQ.1.0)THEN
        BETA=(NZERO+.25)*PI
        ZEROJ=BETA-OT1/BETA-OT2/BETA**3-OT3/BETA**5-OT4/BETA**7
      ENDIF
      RETURN
      END
      DOUBLE PRECISION FUNCTION DOT(N,X1,INC1,X2,INC2)
C
C  COMPUTES DOT PRODUCT OF TWO D.P. VECTORS WITH NONUNIT
C  INCREMENTING ALLOWED. REPLACEMENT FOR BLAS SUBROUTINE SDOT.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X1(1),X2(1)
      IF(INC2.GT.0)THEN
        K=1
      ELSE
        K=N*ABS(INC2)
      ENDIF
      DOT=0.0
      IF(INC1.GT.0)THEN
        DO 10 I=1,N,INC1
          DOT=DOT+X1(I)*X2(K)
          K=K+INC2
   10   CONTINUE
      ELSE
        DO 20 I=N,1,INC1
          DOT=DOT+X1(I)*X2(K)
          K=K+INC2
   20   CONTINUE
      ENDIF
      RETURN
      END
      DOUBLE PRECISION FUNCTION JBESS(X,ORDER)
C
C  COMPUTES BESSEL FUNCTION OF ORDER ORDER AND ARGUMENT X BY CALLING
C  SLATEC ROUTINES DBESJ0 AND DBESJ1
C  MAY BE REPLACED WITH IMSL ROUTINES MMBSJ0,MMBSJ1
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      IF(ORDER.EQ.0.0)THEN
        JBESS=DBESJ0(X)
      ELSEIF(ORDER.EQ.1.0)THEN
        JBESS=DBESJ1(X)
      ENDIF
      RETURN
      END
```

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER (PI=3.14159265359)
      DOUBLE PRECISION KARG,KERN
      DIMENSION KARG(255,100),KERN(510,100),RERR(9),RNG(9),NK(100)
      COMMON /TEST/NG,NF,NI
      COMMON /BESINT/NK,NF,NPS,KARG,KERN
      EXTERNAL F1,F2,F3,F4,F5,F6,F7,F8
      DATA (RERR(I),I=1,9)/1.E-6,1.E-8,1.E-8,3*1.E-10/,
     $     (RNG(I),I=1,9)/.05,2.,100.,.05,2.,100.,.05,2.,100./
C  ALPHA=(1.0,1.0)/SQRT(2) SO THAT ALPHA**2=(0.,1.)
      SQR2=SQRT(2.)
      DO 100 NN=1,9
        IF((NN.EQ.1).OR.(NN.EQ.4).OR.(NN.EQ.7))THEN
          WRITE(6,899)
          WRITE(6,898)
        ENDIF
        AERR=1.E-1*RERR(NN)
        R=RNG(NN)
C  COMPUTE INTEGRAL(J0(K*R)*K*EXP(-ALPHA*K**2)*DK)
        NEW=1
        CALL BESAUT(BESR,BESI,0.D0,1.7,R,F1,RERR(NN),AERR,1,NEW,IERR)
        ANSR=EXP(-R**2/(4.*SQR2))*(COS(R**2/(4.*SQR2))+
     $       SIN(R**2/(4.*SQR2)))/(2.*SQR2)
        ANSI=EXP(-R**2/(4.*SQR2))*(SIN(R**2/(4.*SQR2))-
     $       COS(R**2/(4.*SQR2)))/(2.*SQR2)
        WRITE(6,900)1,IERR,SNGL(R),NEW,SNGL(RERR(NN)),BESR,BESI,
     $              ANSR,ANSI,NG,NF,NI
C  COMPUTE INTEGRAL(J1(K*R)*EXP(-K)*DK)
        NEW=1
        CALL BESAUT(BESR,BESI,1.D0,1.7,R,F2,RERR(NN),AERR,1,NEW,IERR)
        ANSR=(SQRT(R**2+1.0)-1.0)/(R*SQRT(R**2+1.0))
        ANSI=0.0
        WRITE(6,900)2,IERR,SNGL(R),NEW,SNGL(RERR(NN)),BESR,BESI,
     $              ANSR,ANSI,NG,NF,NI
C  COMPUTE INTEGRAL(J0(K*R)*DK)
        NEW=1
        CALL BESAUT(BESR,BESI,0.D0,1.7,R,F3,RERR(NN),AERR,1,NEW,IERR)
        ANSR=1./R
        ANSI=0.0
        WRITE(6,900)3,IERR,SNGL(R),NEW,SNGL(RERR(NN)),BESR,BESI,
     $              ANSR,ANSI,NG,NF,NI
C  COMPUTE INTEGRAL(J0(K*R)*K/SQRT(K**2+ALPHA**2)*DK)
C  FIRST MOLIFY SAVED KERNELS FOR NEXT CALL TO BESTRN
        DO 10 I=1,MIN(NPS,100)
          LK=1
          DO 10 J=1,2**(NK(I)+1)-1
            X=KARG(J,I)
            XR=SQRT((SQRT(X**4+1.0)+X*X)/2.0)
            XI=SQRT((SQRT(X**4+1.0)-X*X)/2.0)
            KERN(LK+1,I)=-X*XI/(XR**2+XI**2)*KERN(LK,I)
            KERN(LK,I)=X*XR/(XR**2+XI**2)*KERN(LK,I)
            LK=LK+2
   10   CONTINUE
C  SET NEW=2 TO USE SAVED KERNELS
        NEW=2
        CALL BESAUT(BESR,BESI,0.D0,1.7,R,F4,RERR(NN),AERR,1,NEW,IERR)
        ANSR=EXP(-R/SQR2)*COS(R/SQR2)/R
        ANSI=-EXP(-R/SQR2)*SIN(R/SQR2)/R
        WRITE(6,900)4,IERR,SNGL(R),NEW,SNGL(RERR(NN)),BESR,BESI,
     $              ANSR,ANSI,NG,NF,NI
C  COMPUTE INTEGRAL(J0(K*R)*K*LK)
        NEW=1
        CALL BESAUT(BESR,BESI,0.D0,1,7,R,F5,RERR(NN),AERR,1,NEW,IERR)
        ANSR=0.0
        ANSI=0.0
        WRITE(6,900)5,IERR,SNGL(R),NEW,SNGL(RERR(NN)),BESR,BESI,
     $              ANSR,ANSI,NG,NF,NI
C  COMPUTE INTEGRAL(J0(K*R)*K*SQRT(K**2+ALPHA**2)*DK)
C  FIRST MOLIFY SAVED KERNELS FOR NEXT CALL TO BESTRN
        DO 20 I=1,MIN(NPS,100)
          LK=1
          DO 20 J=1,2**(NK(I)+1)-1
            X=KARG(J,I)
            XR=SQRT((SQRT(X**4+1.0)+X*X)/2.0)
            XI=SQRT((SQRT(X**4+1.0)-X*X)/2.0)
            KERN(LK+1,I)=XI*KERN(LK,I)
            KERN(LK,I)=XR*KERN(LK,I)
            LK=LK+2
   20   CONTINUE
C  SET NEW=2 TO USE SAVED KERNELS
        NEW=2
        CALL BESAUT(BESR,BESI,0.D0,1,7,R,F6,RERR(NN),AERR,1,NEW,IERR)
        ANSR=-EXP(-R/SQR2)/R**3*((R/SQR2+1.)*COS(R/SQR2)+
     $       R/SQR2*SIN(R/SQR2))
        ANSI=-EXP(-R/SQR2)/R**3*(R/SQR2*COS(R/SQR2)-
     $       (R/SQR2+1.)*SIN(R/SQR2))
        WRITE(6,900)6,IERR,SNGL(R),NEW,SNGL(RERR(NN)),BESR,BESI,
     $              ANSR,ANSI,NG,NF,NI
C  COMPUTE INTEGRAL(J1(K*R)*COS(K)*DK)
        NEW=1
        CALL BESAUT(BESR,BESI,1.D0,1,7,R,F7,RERR(NN),AERR,1,NEW,IERR)
        IF(R.LT.1.0)THEN
          ANSR=(SQRT(1.0-R*R)-1.0)/(R*SQRT(1.0-R*R))
        ELSE
          ANSR=1./R
        ENDIF
        ANSI=0.0
        WRITE(6,900)7,IERR,SNGL(R),NEW,SNGL(RERR(NN)),BESR,BESI,
     $              ANSR,ANSI,NG,NF,NI
C  COMPUTE INTEGRAL(J1(K*R)*COS(K)/K*DK)
C  FIRST MOLIFY SAVED INTEGRANDS
        DO 30 I=1,MIN(NPS,100)
          LK=1
          DO 30 J=1,2**(NK(I)+1)-1
            KERN(LK,I)=KERN(LK,I)/KARG(J,I)
            LK=LK+2
   30   CONTINUE
C  SET NEW=2 TO USE SAVED KERNELS
        NEW=2
        CALL BESAUT(BESR,BESI,1.D0,1,7,R,F8,RERR(N),AERR,1,NEW,IERR)
        IF(R.GT.1.)THEN
          ANSR=SQRT(R**2-1.)/R
        ELSE
          ANSR=0.0
        ENDIF
        ANSI=0.0
        WRITE(6,900)8,IERR,SNGL(R),NEW,SNGL(RERR(NN)),BESR,BESI,
     $              ANSR,ANSI,NG,NF,NI
```

```
100 CONTINUE
898 FORMAT(' NN',2X,'IERR',5X,'P',1X,'NFE',4X,'RERR',10X,'NUMERR',
     $       12X,'NUMERI',12X,'EXACTR',12X,'EXACTI',8X,'NG',5X,
     $       'NF',4X,'NI')
899 FORMAT(1H1)
900 FORMAT(2I4,I1,4X,I1,4X,F6.2,3X,I1,3X,F8.1,2X,4G18.10,2X,I1,
     $       3X,I5,2X,I3)
    END
    SUBROUTINE F1(X,Y1,Y2)
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    PARAMETER (SQR2=1.414213562)
    Y1=X*EXP(-X**2/SQR2)*COS(X**2/SQR2)
    Y2=-X*EXP(-X**2/SQR2)*SIN(X**2/SQR2)
    RETURN
    END
    SUBROUTINE F2(X,Y1,Y2)
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    Y1=EXP(-X)
    Y2=0.0
    RETURN
    END
    SUBROUTINE F3(X,Y1,Y2)
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    Y1=1.0
    Y2=0.0
    RETURN
    END
    SUBROUTINE F4(X,Y1,Y2)
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    XR=SQRT((SQRT(X**4+1.0)+X*X)/2.)
    XI=SQRT((SQRT(X**4+1.0)-X*X)/2.)
    Y1=X*XR/(XR**2+XI**2)
    Y2=-X*XI/(XR**2+XI**2)
    RETURN
    END
    SUBROUTINE F5(X,Y1,Y2)
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    Y1=X
    Y2=0.0
    RETURN
    END
    SUBROUTINE F6(X,Y1,Y2)
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    XR=SQRT((SQRT(X**4+1.0)+X*X)/2.)
    XI=SQRT((SQRT(X**4+1.0)-X*X)/2.)
    Y1=X*XR
    Y2=X*XI
    RETURN
    END
    SUBROUTINE F7(X,Y1,Y2)
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    Y1=COS(X)
    Y2=0.0
    RETURN
    END
    SUBROUTINE F8(X,Y1,Y2)
    IMPLICIT DOUBLE PRECISION (A-H,O-Z)
    Y1=COS(X)/X
    Y2=0.0
    RETURN
    END
```