Evaluating infinite integrals involving Bessel functions of arbitrary order

S.K. Lucas and H.A. Stone
Division of Applied Sciences
Harvard University
Cambridge MA 02138 U.S.A.

Submitted April 1994, Revised August 1994.

Abstract

The evaluation of integrals of the form $I_n = \int_0^\infty f(x)J_n(x)\,dx$ is considered. In the past, the method of dividing an oscillatory integral at its zeros, forming a sequence of partial sums, and using extrapolation to accelerate convergence has been found to be the most efficient technique available where the oscillation is due to a trigonometric function or a Bessel function of order $n = 0, 1$. Here, we compare various extrapolation techniques as well as choices of endpoints in dividing the integral, and establish the most efficient method for evaluating infinite integrals involving Bessel functions of any order n, not just zero or one. We also outline a simple but very effective technique for calculating Bessel function zeros.

1 Introduction

Calculating integrals on $[0, \infty)$ with oscillatory integrands is a more difficult problem than that for the case of an eventually monotonic integrand. Various techniques have been proposed for calculating integrals of the form

$$I = \int_a^\infty f(x)w(x)\,dx, \tag{1}$$

where $w(x)$ is an oscillatory function such as $\sin \omega x$, $\cos \omega x$, or $J_0(\omega x)$, and $f(x)$ is eventually monotonic. Blakemore *et al.* [1] reviewed several numerical methods, and came to the conclusion that an "integration then summation" procedure was the most efficient method. Assuming the zeros of $w(x)$ are known, the integral (1) is divided at these zeros, and an alternating sequence is summed to compute the integral. The sequence $\{I^j\}_{j=0}^\infty$ is formed, where

$$I^j = \sum_{i=0}^{j} u_i = \sum_{i=0}^{j} \int_{x_i}^{x_{i+1}} f(x)w(x)\,dx, \tag{2}$$

$x_0 = a$ and $x_1, x_2, \ldots$ are the zeros of $w(x)$ greater than a. Although $I^j \to I$ as $j \to \infty$, the sequence typically converges slowly, and so an extrapolation technique is used to accelerate convergence. This technique was originally introduced by Longman [9] using extrapolation by the Euler transform, and Blakemore *et al.* [1] used the ϵ-algorithm of Wynn [22] as an accelerator. Piessens and Branders [14] also survey various techniques for evaluating integrals involving Bessel functions.

More recently, popular textbooks on numerical integration (Davis and Rabinowitz [3] and Evans [4]) state that integration then summation with extrapolation, henceforth indicated as ISE, is the most efficient method for evaluating integrals such as (1). Davis and Rabinowitz [3] described various extrapolation techniques, including the Euler transform, the ϵ-algorithm, and the W transform due to Sidi [16], but provide no indication of the relative effectiveness of the different extrapolation procedures. Evans [4] also described the ISE method, and concluded that the ϵ-algorithm is superior to the Euler transform, but described no direct comparison with the W transform. Here, we shall compare the ϵ-algorithm to the modified W, or mW, transform of Sidi [18]. It is interesting to note that as recently as 1985, Lyness [10] outlined the ISE method using the Euler transform.

Integrals of the form (1) with $w(x) = \sin \omega x$ or $\cos \omega x$ are Fourier integrals, and standard numerical packages such as IMSL [23] include routines for their evaluation. The IMSL integration routines are mainly the QUADPACK routines of Piessens *et al.* [13]. However, for the case that the oscillation is due to a Bessel function, there is no standard routine for calculating integrals of the form

$$I_n = \int_a^\infty f(x) J_n(x)\, dx, \tag{3}$$

where n is an integer greater than or equal to zero. The standard IMSL infinite integral routine `dqdagi( )` performs quite poorly on integrals such as (3). However, if the QUADPACK routines are available, Piessens *et al.* [13] include a program which implements the ISE method using the ϵ-algorithm, though it requires an additional subroutine to calculate the zeros of $J_n(x)$. This combination, ϵ-algorithm and zeros as integration endpoints, will be one of several cases considered here.

As an alternative, (3) can be considered as a Hankel transform. Suter [19] reviewed various techniques for evaluating Hankel transforms, and, in addition to the ISE method, included asymptotic approximations, convolution algorithms, and projection methods. A projection method involves transforming the integral (3) into a Fourier integral which can be evaluated using FFT techniques or an ISE method. For example, Linz [8] derived the identity

$$\int_0^\infty f(x) J_0(\rho x)\, dx = \frac{2}{\pi} \int_0^\rho (\rho^2 - s^2)^{-1/2} \int_0^\infty f(x) \cos xs\, dx\, ds. \tag{4}$$

Cree and Bones [2] compared various algorithms to numerically calculate (3), and concluded projection methods were the best for evaluating Hankel transforms. However, they included the restriction that $f(x)$ is only known at sample points, and hence reject ISE methods. Our investigation has shown that, assuming f is known as an analytic function, there is roughly the same amount of effort involved in evaluating (1) regardless of whether w is a trigonometric or Bessel function. Thus, a projection method based on (4) will be substantially slower than an ISE method due to the additional level of integration, and will not be considered further.

The objective of this paper is to evaluate (3) using various ISE methods, where we compare not only extrapolation methods (§2), but also choices for x_i, the endpoints of the integrals in (2) (§3). Further, we shall consider evaluating (3) for arbitrary n (§4). In all the literature of which we are aware, the only numerical examples given are for the Bessel functions J_0 or J_1 only. For larger orders, Bessel function behaviour is more complex, and we shall see that more numerical care is required.

2 Extrapolation Methods

We consider here various extrapolation techniques, which we wish to compare when using ISE methods to evaluate (3).

2.1 The Euler Transform

The Euler transform is the most commonly employed choice for improving the convergence of a (slowly) converging sequence (Davis and Rabinowitz [3]). Given a sequence of terms $\{u_i\}_{i=0}^{\infty}$, their infinite sum may be written as

$$\sum_{i=0}^{\infty} u_i = \frac{1}{2}\left(u_0 + \mathrm{M}u_0 + \mathrm{M}^2 u_0 + \ldots\right), \tag{5}$$

where M is the forward average operator, and $\mathrm{M}u_0 = (u_0 + u_1)/2$, $\mathrm{M}^2 u_0 = \mathrm{M}(\mathrm{M}u_0) = (u_0 + 2u_1 + u_2)/4$ etc. It can be shown that the right hand series of (5) converges to the same value as the left hand series, and often much more rapidly. Often it is desirable to start with a later term, say u_m, so that

$$\sum_{i=0}^{\infty} u_i = \sum_{i=0}^{m-1} u_i + \frac{1}{2}\left[u_m + \mathrm{M}u_m + \mathrm{M}^2 u_m + \ldots\right]. \tag{6}$$

Unfortunately, there is no obvious way of choosing m to optimize the result obtained from (6). Lyness [10] used the Euler transform with the forward average operator. The form quoted in Davis and Rabinowitz [3], which uses the forward difference operator instead of the forward average operator shown here, can only deal with purely oscillatory sequences of terms. A purely oscillatory sequence can be written as $\{(-1)^i u_i\}_{i=0}^{\infty}$, where all the u_i's are of the same sign.

2.2 The ϵ-algorithm

A well-known transform, originally due to Shanks [15], became known as the ϵ-algorithm after Wynn [22] introduced an efficient computational algorithm to aid in its evaluation. The ϵ-algorithm is a nonlinear transformation of a sequence of (slowly) converging numbers which identifies oscillatory transients in the sequence and attempts to remove them. Given a sequence of partial sums $\{A_n\}_{n=0}^{\infty}$, we define

$$\begin{gathered} \epsilon_n^{(-1)} = 0, \quad \epsilon_n^{(0)} = A_n, \\ \epsilon_n^{(p)} = \epsilon_{n+1}^{(p-2)} + \left[\epsilon_{n+1}^{(p-1)} - \epsilon_n^{(p-1)}\right]^{-1}. \end{gathered} \tag{7}$$

Then $\epsilon_n^{(2k)}$ is identified as the kth Shanks' transform of the sequence $\{A_n\}$.

2.3 The mW transform

Sidi [16] introduced an extrapolation technique, known as the W transform, for a large class of infinite oscillatory integrals. This technique requires asymptotic information about the integrand as the integration variable tends to infinity. Later, Sidi [18] introduced the modified W, or mW, transform, where the only asymptotic information required is the eventual distance between the zeros of the integrand. The mW transform to evaluate $\int_a^\infty g(x)\,dx$ is described in Sidi [18] as

$$\begin{aligned}
&F(x_s) = \int_a^{x_s} g(x)\,dx, \quad \psi(x_s) = \int_{x_s}^{x_{s+1}} g(x)\,dx, \\
&M_{-1}^{(s)} = F(x_s)/\,\psi(x_s), \quad N_{-1}^{(s)} = 1/\psi(x_s), \\
&M_p^{(s)} = \left(M_{p-1}^{(s)} - M_{p-1}^{(s+1)}\right)\Big/\left(x_s^{-1} - x_{s+p+1}^{-1}\right), \\
&N_p^{(s)} = \left(N_{p-1}^{(s)} - N_{p-1}^{(s+1)}\right)\Big/\left(x_s^{-1} - x_{s+p+1}^{-1}\right), \\
&W_p^{(s)} = M_p^{(s)}\Big/\,N_p^{(s)},
\end{aligned} \tag{8}$$

for $s = 0, 1, \ldots$ and $p = 0, 1, \ldots$. Here, the x_s's are the zeros of g after a. At the pth level of the transform, $W_p^{(0)}$ gives the best approximation to the integral $\int_a^\infty g(x)\,dx$, and involves $p+3$ terms in the expansion of (2).

For the case of the oscillatory integral (3), where $g(x) = f(x)J_n(x)$, Sidi [18] calculated integrals with $a = 0$ and $n = 0$ or 1. For these cases, the integral endpoints were chosen as $x_s = (s+1)\pi$, $s = 0, 1, \ldots$.

3 Choosing the Interval Endpoints

The ISE method typically involves integration between the zeros of the oscillatory function. For the case that the oscillatory function is trigonometric, finding zeros is a trivial exercise. However, finding the zeros of a Bessel function is more difficult. In this section, we first review previously available methods for obtaining zeros of Bessel functions, and then outline a simple but very efficient procedure, based upon Newton's method, for evaluating the zeros of a Bessel function of any order. We also describe a simple modification due to Lyness [10] which avoids the need to evaluate exact zeros for the interval endpoints needed for integration.

3.1 Finding the Zeros of a Bessel Function of Arbitrary Order

Finding the zeros of a Bessel function $J_n(x)$ is an interesting problem, for which no routine is currently available in IMSL [23]. The use of tabulated results is possible (e.g. Olver [12]), but this would involve a large amount of data entry, and would be useless if the particular order n of interest is not tabulated, or if more zeros are required than are available. Several asymptotic results for finding zeros are also given in [12], but these are of limited use for a general numerical procedure. Recently, Ikebe *et al.* [7] outlined a method for finding the zeros of a Bessel function of any order using a formulation that involves forming a tridiagonal symmetric matrix whose eigenvalues are the zeros. An error estimate for the results is also reported. The disadvantage of the method of Ikebe *et al.* [7] is that an eigenvalue problem has to be solved, and a matrix large enough to guarantee the required accuracy for a given number of zeros must be formed. If more zeros are required than are calculated initially, then a larger matrix has to be formed and its eigenvalues determined. Also, time may be wasted finding zeros that are not necessary. A referee has made us aware of Temme [21], which we will comment on further at the end of this section.

As an alternative to the above procedures, we outline here a method for calculating the zeros of $J_n(x)$ for arbitrary n, where further zeros are calculated only as required, with high accuracy, and with minimal computational effort. The only assumption is the availability of a routine to calculate accurately a Bessel function for a particular argument x, e.g. the IMSL routine `dbsjns( )`. We shall use the standard notation that the ith zero of $J_n(x)$ is denoted by $j_{n,i}$.

It is known (e.g. Sneddon [20]) that for large x,

$$J_n(x) \sim \sqrt{\frac{2}{\pi x}} \cos\left(x - \frac{n\pi}{2} - \frac{\pi}{4}\right), \tag{9}$$

which implies that for large x the zeros of $J_n(x)$ are separated by approximately π. Thus, if we know the zero $j_{n,i-1}$, we can say that $j_{n,i} \sim j_{n,i-1} + \pi$. Newton's method can be used to improve this approximation to $j_{n,i}$. Using the identity

$$J_n'(x) = \frac{n}{x} J_n(x) - J_{n+1}(x), \tag{10}$$

the Newton iteration is

$$x_{i+1} = x_i - \frac{J_n(x_i)}{\frac{n}{x} J_n(x_i) - J_{n+1}(x_i)}. \tag{11}$$

It is convenient when using (11) that the routine `dbsjns( )` returns the values of all Bessel functions up to a given order for a particular x. The zeros of $J_n(x)$ can be seen in figure 1, where plots of $J_n(x)$ for $n = 0, 10$ and 50 are shown.

While adding π is acceptable for an approximation of the next zero for large x, the estimate is unacceptable for initial zeros, especially for large n, as is clear from figure 1. For a particular n, the distance between zeros is largest between initial zeros, and this separation converges to π as $x \to \infty$. This behavior suggests an efficient scheme for finding the next zero: given two previous zeros $j_{n,i-2}$ and $j_{n,i-1}$, approximate $j_{n,i}$ by

$$j_{n,i} \simeq j_{n,i-1} + (j_{n,i-1} - j_{n,i-2}) \quad \text{for } i \geq 3, \tag{12}$$

and use the Newton iteration (11) to improve the approximation. Since (12) cannot be used for $i = 1$ or 2, we may instead utilize the asymptotic results of Olver [11], that

$$\begin{aligned} j_{n,1} &\sim n + 1.8557571 n^{1/3} + 1.033150 n^{-1/3} - 0.00397 n^{-1} - \\ &\quad 0.0908 n^{-5/3} + 0.043 n^{-7/3} + \ldots, \quad \text{and} \\ j_{n,2} &\sim n + 3.2446076 n^{1/3} + 3.158244 n^{-1/3} - 0.08331 n^{-1} - \\ &\quad 0.8437 n^{-5/3} + 0.864 n^{-7/3} + \ldots, \quad \text{for} \quad n \geq 1. \end{aligned} \tag{13}$$

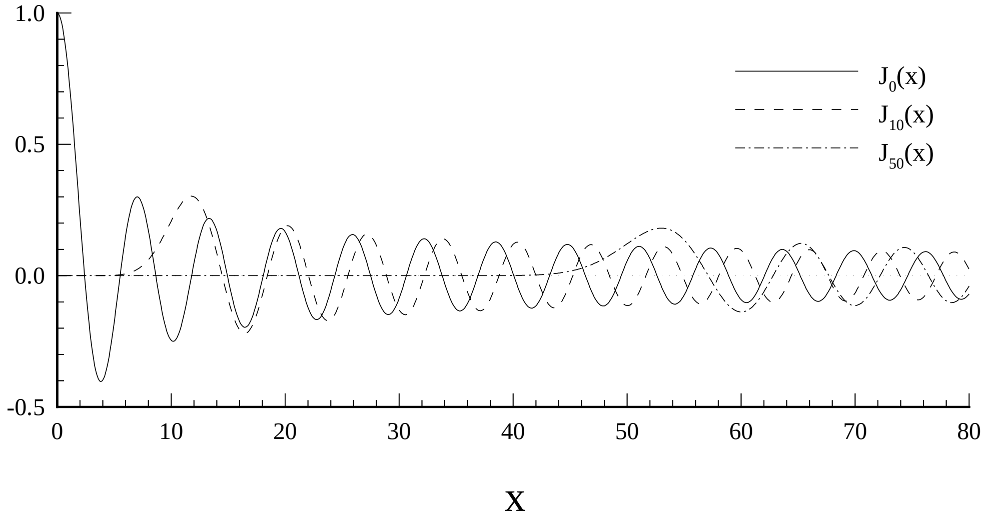

Figure 1: Graphs of the Bessel functions $J_0(x)$, $J_{10}(x)$, and $J_{50}(x)$. For large orders, the difference between initial zeros is much larger than π.

The results in (13) are asymptotic expansions for large n, but even for $n = 1$ the results are acceptable as initial approximations for Newton's method. For $n = 0$, the known zeros $j_{0,1} \simeq 2.404826$ and $j_{0,2} \simeq 5.520078$ can be used.

We thus propose the following method for finding the zeros $j_{n,i}$ of $J_n(x)$: for $i = 1$ and 2, use (13) as an initial approximation, and use the Newton iteration (11) to improve the accuracy. For $i > 2$, use (12) as an initial approximation for (11). We have found that typically 3-5 iterations of (11) yield machine accuracy for the zeros in double precision FORTRAN, and the approach has the further advantages that zeros can be calculated as needed, and only as many as are necessary are obtained. The method and implementation is fast and general. For example, when demanding machine accuracy, which is about sixteen figures in double precision FORTRAN on a Sun Sparc 10, calculating the first one hundred zeros of $J_0(x)$ took less than 0.1 seconds of CPU, the first one hundred zeros of $J_{100}(x)$ took approximately 0.1 seconds, and the first one hundred zeros of $J_{995}(x)$ took approximately 0.5 seconds.

Temme [21] used a very similar procedure to that described here. A higher order version of Newton's method and a far more complicated method for approximating the zeros initially are described. While Temme [21] reports typically 1-3 iterations are required (compared to 3-5 for our implementation), each iteration is computationally more expensive, as is initially approximating the successive zeros. We feel that the simplicity of the method described here is worth consideration.

3.2 Lyness' Modification

Lyness [10] suggested a simple modification to the ISE method, based on the observation that calculation of the zeros $j_{n,i}$ was more difficult and time consuming than the actual evaluation of integrals and subsequent extrapolation: due to the asymptotic result (9), choose interval endpoints π apart. For the case $a = 0$ in (1), the two possibilities considered were

$$x_0 = 0, \quad x_j = \pi \left(j + k + \frac{3}{4} + \frac{n}{2} \right), \tag{14}$$

and

$$x_0 = 0, \quad x_j = \pi \left(j + k + \frac{1}{4} + \frac{n}{2} \right), \tag{15}$$

where (14) corresponds asymptotically to the zeros of $J_n(x)$, and (15) to the extrema. k is an integer chosen so that x_1 is beyond the first zero or extrema, respectively, of $J_n(x)$. The use of (15) means each integral involves half the integrand positive and half negative, so the magnitude of the integrals is likely to be smaller than the corresponding integrals with integrands that are either always positive or negative, and hence the oscillation in the partial sums will be smaller, and convergence should be improved. Thus, Lyness proposed using (14) or (15) instead of the actual zeros as a way to improve the efficiency of the integration method.

The method recommended by Sidi [18] for the mW transform in fact uses this technique of assuming all integration intervals are π apart, but uses multiples of π as endpoints as opposed to the endpoint values determined by (14) or (15). It should be noted that Sidi [18] independently arrived at this choice of endpoints as a particular case for Bessel functions – the mW transform is applicable to a wide range of oscillatory infinite integrals.

4 Comparison of Methods

There are several different possible combinations of extrapolation and endpoint choices available for an ISE method to evaluate infinite integrals involving Bessel functions. We consider:

euler – using the Euler transform with the Bessel function zeros $j_{n,i}$ as endpoints,

mW – using the mW transform with multiples of π as endpoints, as used by Sidi [16, 18],

mW offset – using the mW transform with multiples of π as endpoints, but choosing the first end point such that it is greater than the first zero of $J_n(x)$,

mW zeros – using the mW transform with Bessel function zeros as endpoints,

mW extrema – using the mW transform, modified with the recommendation of Lyness [10] to use approximate extrema as endpoints defined as

$$x_i = \frac{1}{2} \left(j_{n,i} + j_{n,i+1} \right), \tag{16}$$

eps zeros – using the ϵ-algorithm with Bessel function zeros as endpoints,

eps extrema – using the ϵ-algorithm with endpoints as defined in (16),

eps app zeros – using the ϵ-algorithm with endpoints as defined in (14), where the first endpoint is greater than the first zero of $J_n(x)$.

We evaluate all integrals on $[x_i, x_{i+1}]$ using the routine `dqag( )` from Piessens *et al.* [13], with a requested relative error bound of 10^{-12}. Due to the smooth nature of the integrands, most integrals for our test problems require only one application of the fifteen point Gauss-Kronrod rule, and numerical results are accurate to machine precision.

As a first test, we apply the *euler*, *mW* and *eps zeros* methods to the set of integrals

$$\begin{aligned}
&(a) \quad \int_0^\infty x^2 J_0(x)\,dx = -1, \\
&(b) \quad \int_0^\infty \frac{1}{2}\ln(1+x^2) J_1(x)\,dx = K_0(1) \simeq 0.42102\,44382\,40708\,333, \\
&(c) \quad \int_0^\infty \frac{x}{1+x^2} J_0(x)\,dx = K_0(1), \quad \text{and} \\
&(d) \quad \int_0^\infty \frac{1-e^{-x}}{x\ln(1+\sqrt{2})} J_0(x) = 1.
\end{aligned} \tag{17}$$

These integrals are based on examples from Hasegawa and Sidi [6], Sidi [18] and Piessens *et al.* [13]. The reader may note that integral (17a) is formally divergent, but converges in the sense of Abel summability (see Sidi [17]). The particular methods chosen to numerically evaluate these integrals are the standard ones used by Blakemore *et al.* [1] and Sidi [18]. Figure 2 compares the relative error to the number of intervals that have to be integrated in equation (2). As expected, the Euler transform is inferior to the ϵ-algorithm. We have observed this result for a wide variety of sequences, not just those produced in infinite oscillatory integral evaluations. The Euler transform will not be considered further. Also, in the cases (a), (c) and (d), we observe the superior performance of the mW transform compared to the ϵ-algorithm. The mW transform seems much more efficient than the ϵ-algorithm, which contradicts the conclusion of Evans [4] for infinite oscillatory integrals where the claim is made that the mW transform has a similar effectiveness to the ϵ-algorithm.

As mentioned earlier, to the best of our knowledge, all previous tests of numerical methods for infinite integrals involving Bessel functions were only for J_0 and J_1. We now compare the various methods for Bessel functions of higher order. As a test integral, we will evaluate

$$\int_0^\infty \frac{x}{1+x^2} J_n(x)\,dx = \begin{cases} 0.42102\,44382\,4071 & \text{for} \quad n=0, \quad (a), \\ 9.89705\,45308\,402 \times 10^{-2} & \text{for} \quad n=10, \quad (b), \\ 9.99899\,97000\,302 \times 10^{-3} & \text{for} \quad n=100, \quad (c). \end{cases} \tag{18}$$

The numerical results were obtained by successively using more terms in the sequences until the returned results were the same to machine precision. The same results shown here were obtained regardless of which of the converging extrapolation methods was used. Figures 3, 4 and 5 show respectively the relative error versus number of intervals versus for the three cases in equation (18), where we compare both the extrapolation method and the manner of choosing integration endpoints.

Figure 3 shows the results for the integral (18a), which is the same integral as (17c). Error curves for the *mW offset* and *eps app zeros* methods are not included. The *mW offset* method is identical to the *mW* method here because the first zero of $J_0(x)$ is less than π. The *eps app zeros* method error curve is virtually identical to that for the *eps zeros method* since for small n (9) is quite a good approximation, and the approximate zeros of (14) are very close to the

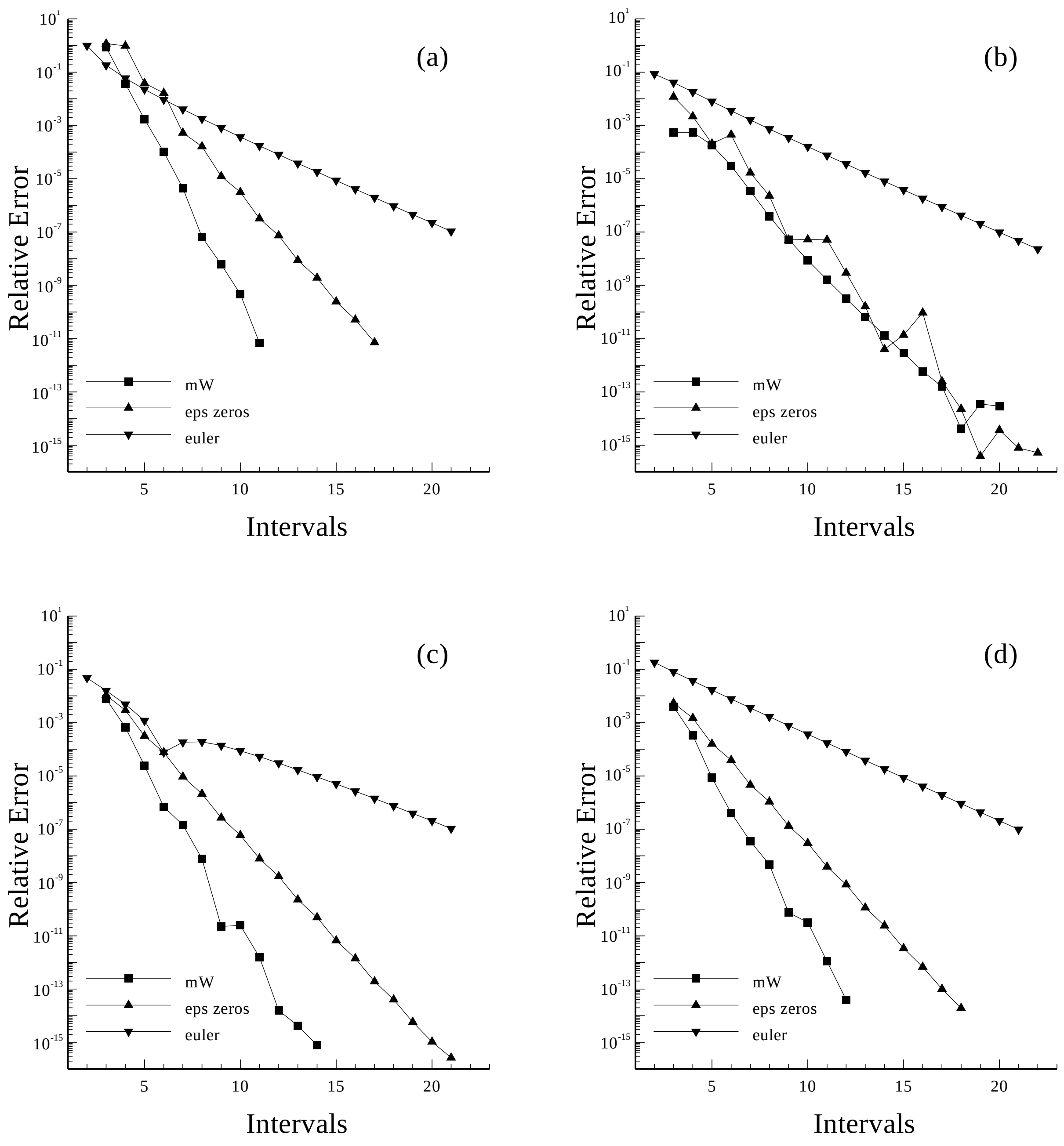

Figure 2: Comparing relative error to number of intervals used to numerically approximate the infinite integrals in (17). The ISE methods *euler*, *mW* and *eps zeros* are used.

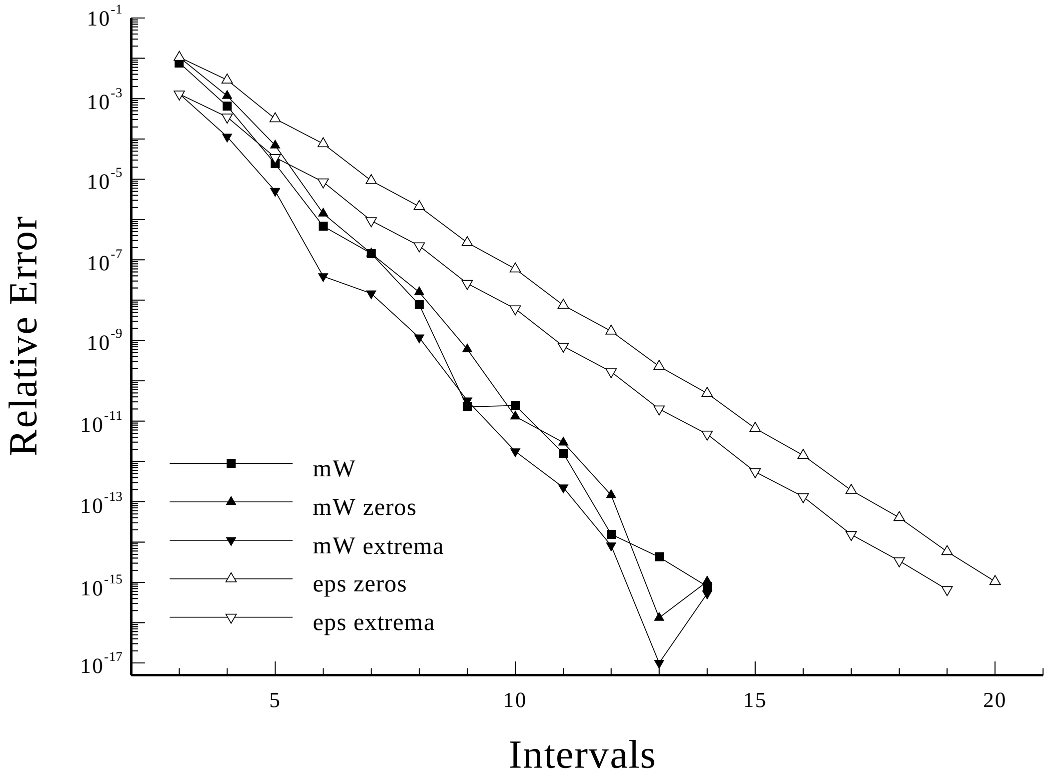

Figure 3: Comparing relative error to number of intervals for the integral in (18) for $n = 0$ by various methods.

actual zeros. Thus, the sequences produced by the *eps zeros* and *eps app zeros* methods are almost identical, and the error curves on applying extrapolation are indistinguishable. In fact, if for $n = 0$ any method that uses zeros or extrema instead uses the approximations to these values as given by (14) or (15), then the error curves are again the same as those found using zeros or extrema. Figure 3 illustrates that using extrema as endpoints gives sequences with less error, as predicted by Lyness [10]. The choice of endpoints π apart other than at zeros or extrema gives an error curve in between those for zeros and extrema. This result explains why the *mW* method gives an error curve between those for *mW zeros* and *mW extrema.* The *mW extrema* method gives the best result for integral (18a), but there is little to choose between any of the three methods that are based upon the mW transform.

Figure 4 shows the results for integral (18b), where we are now dealing with the $J_{10}(x)$, a Bessel function of moderate order. The error curves for both the *mW* and *mW offset* methods are poor. The number of intervals has to be extended beyond forty before these curves regain a downward trend. The *eps app zeros* method, which is similar to the *mW* and *mW offset* methods in that endpoints are chosen exactly π apart, does not perform worse than the *eps zeros* or *eps extrema* methods. We do see, however, that the methods involving exact zeros or extrema continue to perform as well as for (18a) with the *mW extrema* method again giving the best error curve.

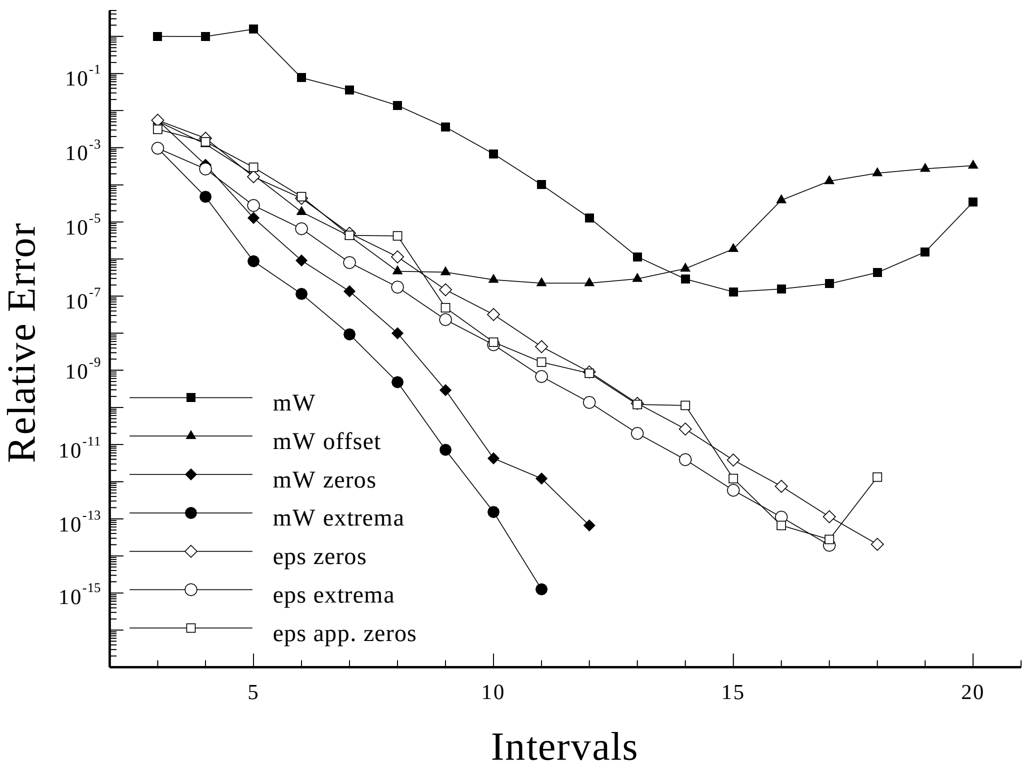

Figure 4: Comparing relative error to number of intervals for the integral in (18) for $n = 10$ by various methods.

Figure 5 shows the results for integral (18c), where we now have the Bessel function of high order $J_{100}(x)$. Since $J_{100}(x) < 10^{-20}$ for $x < 50$, the *mW* method is not shown because it does not give meaningful results since it implies convergence to zero before the oscillatory nature of $J_{100}(x)$ occurs. The *mW offset* method is not much better. Its error curve stays roughly constant and high beyond fifty intervals. The *eps app zeros* method is converging, but it is doing so much slower than the *eps zeros* method. As for the example in figure 4 for $J_{10}(x)$, the methods involving exact zeros or extrema perform the best. In this case, the convergence rate of the mW transform is no better than that for the ϵ-algorithm, but does have an offset downwards. We again conclude that the *mW extrema* method is the best method for this integral.

Another possible choice of interval endpoints, suggested by a referee, are the McMahon's expansions for Bessel function zeros or extrema. These approximations to the zeros $j_{n,i}$, given as $\beta - (4n^2 - 1)/8\beta$ where $\beta = (i + n/2 - 1/4)\pi$, are much better than those in (14). When used in conjunction with the ϵ-algorithm, the McMahon zeros give error curves for the integrals in (18) between those for *eps zeros* and *eps app zeros*. When used with the mW transform, we find that the error curve for (18b) has the same slope as *mW zeros* but is shifted upwards, while the result for (18c) is no better than that for *mW offset*. This shows that a more accurate approximation of Bessel function zeros improves the eficiency of the extrapolation methods, and in the case of the mW transform increases the order of the Bessel function where the *mW*

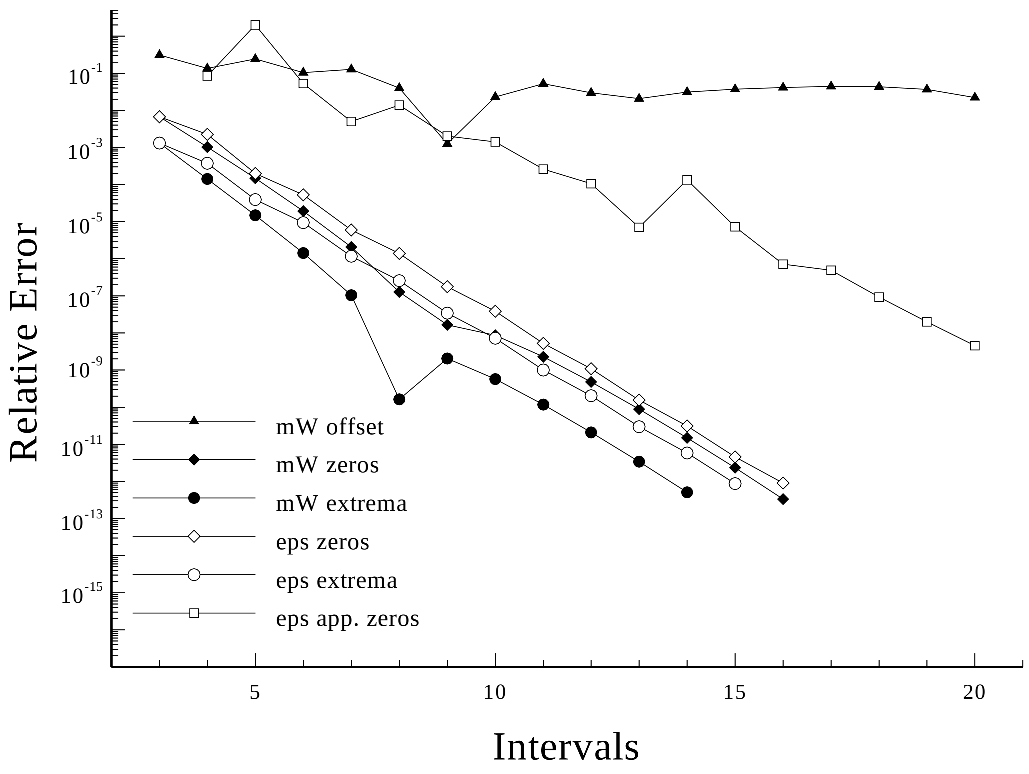

Figure 5: Comparing relative error to number of intervals for the integral in (18) for $n = 100$ by various methods.

---

method fails. However, knowledge of the actual zeros still gives the best results.

Based on the examples shown in figures 3-5, our conclusion is that the *mW extrema* method is the best method for evaluating infinite integrals involving Bessel functions of arbitrary order. As the results of figures 4 and 5 illustrate, knowledge of the zeros of the Bessel function is more important as n increases beyond zero or one. An improved understanding of this fact can be gained by comparing the sequences of partial sums for the cases where approximate and exact zeros are used as endpoints of the intervals for (18c) – see figure 6. The number of terms shown are those required to give a relative error of at most 5×10^{-7} using the ϵ-algorithm – the methods *eps zeros* and *eps app zeros*. We see that the sequence for exact zeros is purely oscillatory, while that for approximate zeros is oscillatory, but not term by term, and the error at successive terms is not necessarily decreasing, although the overall trend is convergent.

The results of figures 3–5 show that the mW transform is better able to accelerate convergence for purely oscillatory sequences than the ϵ-algorithm, and the results of figures 4 and 5 show that the mW transform only seems to work well for purely oscillatory sequences, but performs better than the ϵ-algorithm in these cases. The *mW offset* method could be adjusted by choosing the first endpoint as a multiple of π large enough that the terms $\psi(x_s)$ in (8) are purely oscillatory, but the initial integral would then be over a potentially large number of oscillations and require substantial effort to evaluate, thus making the *mW offset* method still

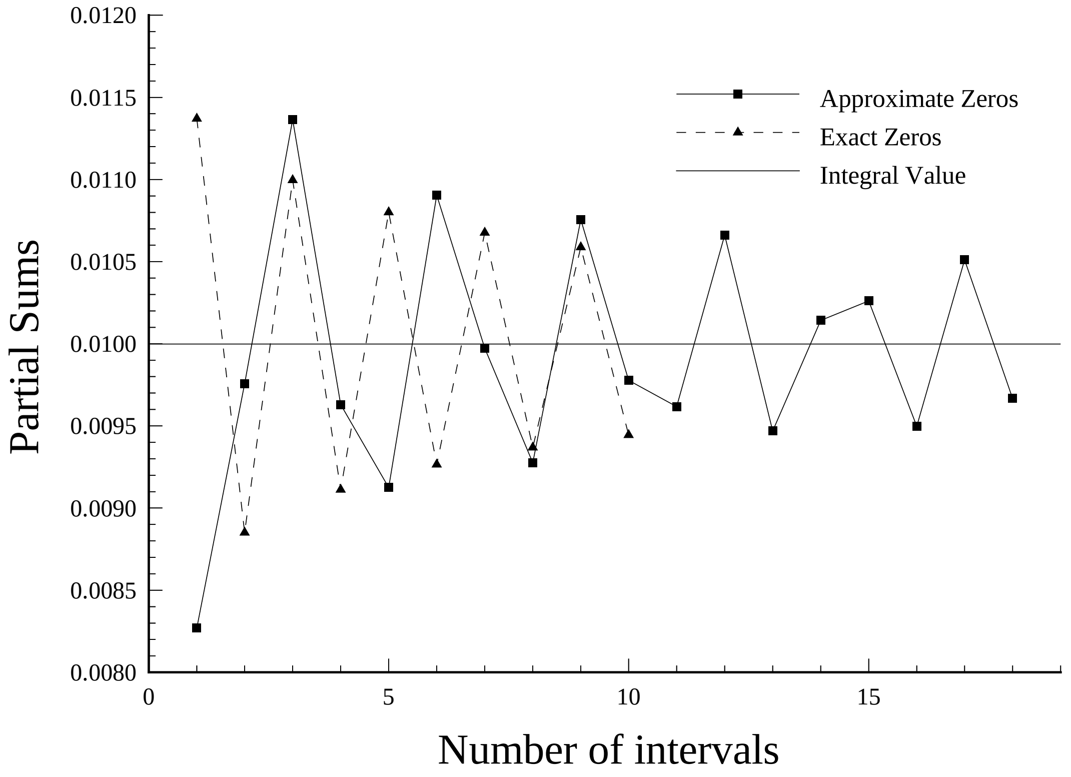

Figure 6: The sequences of partial sums in evaluating (18c) using an interval of π (Approximate Zeros), and knowing the zeros of $J_{100}(x)$ (Exact Zeros). The terms shown are those required for a relative error of at most 5×10^{-7} using the ϵ-algorithm. The value of the integral, which the sequences are converging to, is shown as the horizontal line.

much less efficient than the *mW zeros* or *mW extrema* methods. The mW transform works perfectly well on $J_0(x)$ or $J_1(x)$ with endpoints as multiples of π because the distance between zeros of these Bessel functions is already almost exactly π, and so a purely oscillatory sequence is obtained. Therefore, in terms of an overall comparison, we can conclude that the ϵ-algorithm is a more robust sequence accelerator than the mW transform for infinite oscillatory integral problems, but if the endpoints can be chosen to guarantee a purely oscillatory sequence, then the mW transform shows its superiority.

5 Automatic Implementation of ISE Methods

When implementing an ISE method such that a solution can be obtained with a requested error bound, consideration has to be given to a stopping criterion for the extrapolation and an error bound for the integration on each interval. Piessens *et al.* [13] provided a program to implement the *eps zeros* method, although a routine to find the zeros of the Bessel function is required by the reader. This program uses the same error bound for each interval integral as for the final solution, and uses the routine `dqags( )` to evaluate the integrals. The routine `dqags( )`

uses a twenty-one point Gauss-Kronrod rule as its base rule, which is typically only applied once on each interval. The routine `dqext( )` implements the ϵ-algorithm as well as returning an error estimate which is used as a stopping criterion. The error estimate is calculated as the maximum difference between the current and last three extrapolated results, and so is often pessimistic. Recently, Hasegawa and Sidi [6] have implemented an automatic routine using the mW transform, and use the difference between the current and previous extrapolated results as an error estimate. Unfortunately, this could cause problems when successive values obtained from the transform are close together. As an example, this stopping criterion could fail for integral (17c), where we can see from figure 2c that the ninth and tenth interval results are close together, and could suggest a lower error estimate than the actual error. As a compromise, we suggest the intermediate error estimate of the maximum difference between the current and the previous two extrapolated results.

Our experience has shown that Gauss-Kronrod rules are very effective as integration rules which also provide an error estimates. Gauss-Kronrod rules are $2N+1$ points rules, where $N+1$ points are optimally added to an N point Gauss-Legendre rule, and the error estimate is the difference between the $2N+1$ point rule and the embedded N point rule. The QUADPACK package [13] uses Gauss-Kronrod rules extensively, and the smallest such rule available is a fifteen point rule, which is used as part of an adaptive method in the routine `dqag( )` to provide an excellent general purpose integration tool. The integrals in an ISE method are typically over a half cycle of an oscillation and have smooth integrands. Gauss style rules converge quickly on smooth integrals as the order of the rule increases, and so the error estimate from a Kronrod rule will typically be quite pessimistic here. Hasegawa and Torii [5] have developed a rule based on Chebychev series expansion, where successive rules for comparison have smaller increments than doubling, and sophisticated error estimates are provided that ensure, for smooth integrands, an almost minimum number of function evaluations are required for a particular error bound. They extend this method to oscillatory integrands involving trigonometric functions, and provide results that have extremely low numbers of function evaluations. The Hasegawa and Torii [5] technique, along with the mW transform, is used by Hasegawa and Sidi [6], and applied to some infinite integrals involving Bessel functions of the zeroth and first order. The number of function evaluations reported are typically a factor of two to three lower than simply using `dqag( )` over each interval with the mW transform, mainly due to pessimistic error estimates from the Gauss-Kronrod rule. For low order Bessel functions, the method of Hasegawa and Sidi [6] may indeed be the most efficient routine to evaluate integrals such as (3).

When using an automatic version of an ISE method to evaluate infinite oscillatory integrals, it is error versus number of function evaluation curves that give a guideline to efficiency. When a comparison of the above methods is done with respect to number of function evaluations, the same results already described are found. However, for the $n = 10$ and 100 cases, the error curves for extrema methods are only just superior to those using zeros, sometimes even crossing. For $n = 0$, the difference between extrema and zero methods is as marked as in figure 3. The reason for this is that the first interval for extrema methods is longer than that for zero methods. More function evaluations are thus required for this first interval for extrema methods than zero methods to satisfy the same error bound. This problem is exacerbated as n increases due to the increasingly complicated form of $J_n(x)$ as n increases. However, we still find the mW extrema method to be superior by this criterion.

6 Conclusion

We have considered here various extrapolation methods and ways of choosing interval endpoints for the evaluation of infinite integrals involving Bessel functions of arbitrary order. We conclude that the mW transform is the best accelerator available, with the proviso that the method described by Sidi [18] is extended by choosing the interval endpoints as the average of successive zeros of the Bessel function in the integrand – the *mW extrema* method described above. In reaching this conclusion, we have outlined a simple but nevertheless efficient method for calculating successive Bessel function zeros based on Newton's method.

It should also be noted that the above derivation does not require that the Bessel function is of integer order. By using the IMSL routine `dbsjs( )`, which evaluates Bessel functions of real order, we have evaluated by the same methods $\int_0^\infty f(x) J_\nu(x)\, dx$ for ν real and greater than or equal to zero.

Acknowledgements

We gratefully acknowledge NSF-PYI Award CT5-8957043 (to HAS) and the MERCK Foundation for funding for this study. We thank Dr. John Tanzosh for motivating this study, and also thank Prof. A. Sidi for constructive comments during the research described here and providing us with various papers, including a preprint of Hasegawa and Sidi [6].

References

[1] M. Blakemore, G.A. Evans and J. Hyslop, Comparison of some methods for evaluating infinite range oscillatory integrals, *J. Comp. Physics* **22** (1976) 352-376.

[2] M.J. Cree and P.J. Bones, Algorithms to numerically evaluate Hankel transforms, *Computers Math. Applic.* **26** (1993) 1-12.

[3] P.J. Davis and P. Rabinowitz, *Methods of Numerical Integration,* (Academic Press, Orlando, 2nd ed. 1988).

[4] G. Evans, *Practical Numerical Integration*, (John Wiley & Sons, Chichester, 1993).

[5] T. Hasegawa and T. Torii, Indefinite integration of oscillatory functions by the Chebychev series expansion, *J. Comp. Appl. Math.* **17** (1987) 21-29.

[6] T. Hasegawa and A. Sidi, An automatic integration procedure for infinite range integrals involving oscillatory kernels, submitted to ACM TOMS.

[7] Y. Ikebe, Y. Kikuchi, and I. Fujishiro, Computing zeros and orders of Bessel functions, *J. Comput. Appl. Math.* **38** (1991) 169-184.

[8] P. Linz, A method for computing Bessel function integrals, *Math. Comp.* **26** (1972) 509-513.

[9] I.M. Longman, Note on a method for computing infinite integrals of oscillatory functions, *Proc. Cambridge Philos. Soc.* **52** (1956) 764-768.

[10] J.N. Lyness, Integrating some infinite oscillating tails, *J. Comput. Appl. Math.* **12&13** (1985) 109-117.

[11] F.W.J. Olver, A further method for the evaluation of zeros of Bessel functions, and some new asymptotic expansions for zeros of functions of large order, *Proc. Cambridge Philos. Soc.* **47** (1951) 699-712.

[12] F.W.J. Olver (ed.), *Royal Society Mathematical Tables Vol. 7. Bessel Functions Part III, Zeros and Associated Values* (Cambridge University Press, 1960).

[13] R. Piessens, E. De Doncker-Kapenga, C.W. Uberhuber and D.K. Kahaner, *QUADPACK, a subroutine package for automatic integration* (Springer-Verlag, Berlin, 1983).

[14] R. Piessens and M. Branders, A survey of numerical methods for the computation of Bessel function integrals, *Rend. Sem. Mat. Univers. Politecn. Torino, Fascicolo speciale* (1985) 249-265.

[15] D. Shanks, Non-linear transformations of divergent and slowly convergent sequences, *J. Math. and Phys.* **34** (1955) 1-42.

[16] A. Sidi, The numerical evaluation of very oscillatory infinite integrals by extrapolation, *Math. Comp.* **38** (1982) 517-529.

[17] A. Sidi, Extrapolation methods for divergent oscillatory infinite integrals that are defined in the sense of summability, *J. Comp. Appl. Math.* **17** (1987) 105-114.

[18] A. Sidi, A user-friendly extrapolation method for oscillatory infinite integrals, *Math. Comp.* **51** (1988) 249-266.

[19] B.W. Suter, Foundations of Hankel transform algorithms, *Quart. App. Math.* **49** (1991) 267-279.

[20] I.H. Sneddon, *The Use of Integral Transforms* (McGraw-Hill Book Company, New York, 1972).

[21] T.M. Temme, An algorithm with ALGOL 60 program for the computation of the zeros of ordinary Bessel functions and those of their derivatives, *J. Comp. Physics* **32** (1979) 270-279.

[22] P. Wynn, On a device for computing the $e_m(S_n)$ transformation, *MTAC* **10** (1956) 91-96.

[23] IMSL MATH/LIBRARY Special Functions Version 2.0 (FORTRAN subroutines for mathematical applications) (IMSL, Houston, 1991).