Evaluating infinite integrals involving products of Bessel functions of arbitrary order

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

Submitted April 1994, Revised August 1994.

Abstract

The difficulties involved with evaluating infinite integrals involving products of Bessel functions are considered, and a method for evaluating these integrals is outlined. The method makes use of extrapolation on a sequence of partial sums, and requires rewriting the product of Bessel functions as the sum of two more well-behaved functions. Numerical results are presented to demonstrate the efficiency of this method, where it is shown to be significantly superior to standard infinite integration routines.

1 Introduction

We are concerned here with the evaluation of integrals of the form

$$I_{a,b,\rho,\tau} = \int_0^\infty f(x) J_a(\rho x) J_b(\tau x)\, dx, \tag{1}$$

where a, b are non-negative integer constants, and ρ, τ are positive real constants. Integrals such as (1) occur in problems involving particle motion in an unbounded rotating fluid (Tanzosh and Stone [10] and Davis [2]), in magnetohydrodynamic flow, crack problems in elasticity (Tezer [11]), and distortions of nearly circular lipid domains (Stone and McConnell [9]). Standard integration techniques, such as the infinite integration routine in the IMSL package [13], which is in fact a routine from the QUADPACK integration library of Piessens *et al.* [7], perform extremely poorly on infinite integrals with an oscillatory integrand, and the form of the oscillation of products of Bessel functions makes the IMSL routine even less useful. Linz and Kropp [3] outlined a method for evaluating $I_{0,0,\rho,\tau}$ which involves a double integral on $[0,\rho] \times [0,\tau]$ whose integrand is itself an infinite integral with a cosine kernel that was evaluated using standard Fourier integral routines. While this method works for general ρ and τ, our investigations have shown it to be very expensive in terms of number of function evaluations, and not much better than simply using the IMSL routine.

Lucas and Stone [4] have considered the simpler problem of evaluating integrals of the form

$$\int_0^\infty f(x)J_n(x)\,dx, \tag{2}$$

for arbitrary n. Such integrals are best evaluated using an integration, summation, then extrapolation method, henceforth indicated as ISE. The range is typically subdivided at the zeros of $J_n(x)$, and a series of results of alternating sign are summed to compute the integral. Since the sequence of partial sums usually converges slowly, an extrapolation technique is used to accelerate convergence. Lucas and Stone [4] showed that the mW transform of Sidi [8] was the most efficient extrapolation technique for the ISE method, where the range was divided at the zeros of $J_n(x)$ rather than at multiples of π as used in [8]. Marginally better results were obtained when the midpoints of successive zeros of $J_n(x)$ were used as endpoints. An automatic integration routine based on these results is easily implemented as described in [4].

Unfortunately, an ISE method as presented in Lucas and Stone [4] cannot simply be applied to integrals such as (1). For example, figure 1 plots $J_{10}(x)$ as well as the product $J_5(x)J_{10}(3x/2)$. An ISE method is most efficient for evaluating integrals such as (2), since the essentially sinusoidal oscillation due to a single Bessel function, e.g. $J_{10}(x)$, is straightforward to integrate, and leads to a sequence of results of alternating sign. As shown in figure 1, the oscillation caused by a product of Bessel functions such as $J_5(x)J_{10}(3x/2)$ is more complicated, resulting from the combination of two oscillating functions, one of higher frequency superimposed upon one of lower frequency. When an infinite integral involving $J_5(x)J_{10}(3x/2)$ is approximated by an ISE method using the zeros of $J_5(x)J_{10}(3x/2)$ as integral endpoints, the series of results is not of alternating sign, and extrapolation gives results which are no better – and often poorer – than the sum of results, which converge slowly to the integral value.

To circumvent the above difficulties, we describe a method whereby we rewrite the product $J_a(\rho x)J_b(\tau x)$ as the sum of two oscillating functions, and an ISE method to evaluate integrals such as (2) is applied twice to evaluate integrals with the form (1). Several complications involved in the method are addressed also. The numerical tests reported here show that extremely good results approaching machine precision can be obtained with under a thousand function evaluations, which compares extremely well to the IMSL infinite integraion routine, which typically returns at most a few digits of accuracy for up to fifteen thousand function evaluations.

2 Simplifying the Integrand

Wong [12] observed that $J_\nu^2(t)$ can be written using Hankel functions $H_\nu^{(1)}(t)$ and $H_\nu^{(2)}(t)$ as

$$\begin{aligned} J_\nu^2(t) &= \frac{1}{4}\left\{[H_\nu^{(1)}(t)]^2 + [H_\nu^{(2)}(t)]^2\right\} + \frac{1}{2}H_\nu^{(1)}(t)H_\nu^{(2)}(t) \\ &= \frac{1}{2}\left[J_\nu^2(t) - Y_\nu^2(t)\right] + \frac{1}{2}\left[J_\nu^2(t) + Y_\nu^2(t)\right], \end{aligned} \tag{3}$$

where $Y_\nu(\mathrm{t})$ is the Bessel function of the second kind. Using asymptotic results for Hankel functions, Wong [12] showed that for large t the first term on the right hand side of (3) is oscillatory, and the second term is monotonically decreasing.

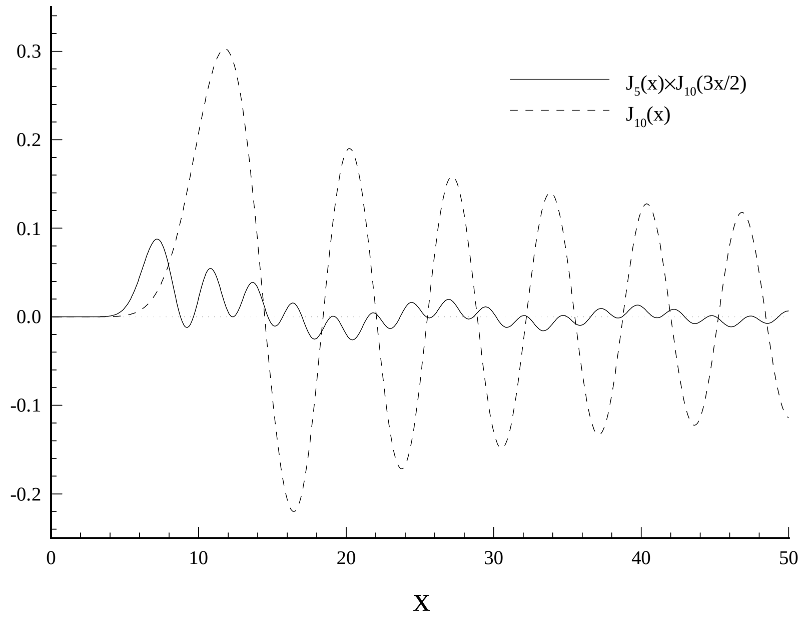


Figure 1: Graph of the functions $J_{10}(x)$ and $J_5(x)J_{10}(3x/2)$.

As a generalization of the above result, we write

$$J_a(\rho x)J_b(\tau x) = h_1(x;a,b,\rho,\tau) + h_2(x;a,b,\rho,\tau), \tag{4}$$

where

$$\begin{aligned} h_1(x;a,b,\rho,\tau) &= \frac{1}{2}\left\{J_a(\rho x)J_b(\tau x) - Y_a(\rho x)Y_b(\tau x)\right\}, \quad \text{and} \\ h_2(x;a,b,\rho,\tau) &= \frac{1}{2}\left\{J_a(\rho x)J_b(\tau x) + Y_a(\rho x)Y_b(\tau x)\right\}. \end{aligned} \tag{5}$$

Using the asymptotic results, valid for large x,

$$\begin{aligned} J_a(\rho x) &\sim \sqrt{\frac{2}{\pi\rho x}}\cos\left(\rho x - \frac{a\pi}{2} - \frac{\pi}{4}\right), \quad \text{and} \\ Y_a(\rho x) &\sim \sqrt{\frac{2}{\pi\rho x}}\sin\left(\rho x - \frac{a\pi}{2} - \frac{\pi}{4}\right), \end{aligned} \tag{6}$$

it can be shown that, provided $x \gg 1$,

$$\begin{aligned} h_1(x;a,b,\rho,\tau) &\sim \frac{1}{\pi\sqrt{\rho\tau}x}\cos\left\{(\rho+\tau)\,x - \frac{(a+b+1)\pi}{2}\right\}, \quad \text{and} \\ h_2(x;a,b,\rho,\tau) &\sim \frac{1}{\pi\sqrt{\rho\tau}x}\cos\left\{(\rho-\tau)\,x + \frac{(a-b)\pi}{2}\right\}. \end{aligned} \tag{7}$$

Thus, for $\rho \neq \tau$, both h_1 and h_2 asymptotically approach cosine functions, in a similar manner to the Bessel function J_a, and an ISE method will be as effective for integrating h_1 or h_2 on an infinite interval as for integrating J_a. If $\rho = \tau$, h_1 still approaches a cosine function, but h_2 approaches a monotonically decreasing function, which can be integrated easily using the IMSL routine for infinite integrals `dqdagi( )`. The high frequency behaviour of $J_a(\rho x)J_b(\tau x)$ is represented by h_1, while the low order frequency behaviour is represented by h_2. This splitting of $J_a(\rho x)J_b(\tau x)$ into the functions h_1 and h_2 is analogous to the trigonometric identity $\cos A \cos B = \{\cos(A - B) + \cos(A + B)\}/2$.

3 Implementing the ISE Method

Once $J_a(\rho x)J_b(\tau x)$ has been rewritten using (4) and (5), an ISE method can be applied to both $\int_0^\infty f(x)h_1(x; a, b, \rho, \tau)\,dx$ and $\int_0^\infty f(x)h_2(x; a, b, \rho, \tau)\,dx$, using the mW transform and the zeros of h_1 and h_2 respectively. However, there are some complicating factors which must be considered. The functions h_1 and h_2 are singular at zero due to the nature of Y_a, and while the zeros of h_1 are easy to find (see §3.2) those of h_2 can be difficult to find for all values of the parameters a, b, ρ and τ.

3.1 The Singular Nature of h_1 and h_2 at $x = 0$

While $J_a(x)$ is well behaved at $x = 0$, $Y_a(x)$ is singular at $x = 0$. The singularity is logarithmic for $a = 0$, and of the form x^{-a} for $a > 0$. For a, $b > 0$, definite integrals with left endpoint $x = 0$ that involve the functions h_1 or h_2 do not exist. The easiest way around this problem is to integrate h_1 and h_2 on intervals with left endpoints greater than zero. While the first zeros of h_1 and h_2 are obvious choices for the left endpoints, our experience has shown that they are not the best ones. If one of the Bessel functions of the second kind is of large magnitude while the other oscillates, the initial oscillations of the product $Y_a(\rho x)Y_b(\tau x)$ have large magnitudes, and dominate the results of integrals involving h_1 or h_2. Since the contributions to h_1 and h_2 from Bessel functions of the second kind are of opposite sign, these large magnitude results will cancel when the integrals involving h_1 and h_2 are summed, and we thus have a case of catastrophic cancellation, where a loss of accuracy occurs by taking the difference of two almost identical numbers. This effect is magnified since there is inherent numerical error in the calculation. A more accurate numerical approximation is obtained if the left endpoints of the integrals involving h_1 and h_2 are zeros of h_1 and h_2 in the region where the term $Y_a(\rho x)Y_b(\tau x)$ no longer dominates. We thus choose the left endpoint to be the largest of the first zeros of $Y_a(\rho x)$ and $Y_b(\tau x)$, which we label $ymax$. Up to $ymax$, we can simply evaluate $\int_0^{ymax} f(x)J_a(\rho x)J_b(\tau x)\,dx$ using an adaptive method such as IMSL's `dqdag( )` [13]. Since this integrand is smooth, a result of high accuracy can be obtained easily. Even if a and b differ significantly, $J_a(\rho x)J_b(\tau x)$ on $[0, ymax]$ will only have a finite number of oscillations, usually just a few, and an adaptive rule will still be adequate. Thus, to avoid the singular nature of h_1 and h_2 at $x = 0$, we calculate the integral (1) as

$$\begin{aligned}\int_0^\infty f(x)J_a(\rho x)J_b(\tau x)\,dx = \; & \int_0^{ymax} f(x)J_a(\rho x)J_b(\tau x)\,dx \; + \\ & \int_{ymax}^\infty f(x)\left(h_1(x; a, b, \rho, \tau) + h_2(x; a, b, \rho, \tau)\right)\,dx.\end{aligned} \tag{8}$$

3.2 Evaluating the Zeros of h_1 and h_2

To use the mW transform effectively, we need to locate the zeros of h_1 and h_2, where the ith zero of h_j ($j = 1$ or 2) is denoted as $h_{j,i}$. For a, $b = 0$ or 1, using the zeros of the cosine approximations to h_1 and h_2 in (7) is perfectly acceptable. For larger a or b, the simple but highly efficient Newton method of finding zeros of Bessel functions is applicable [4]; given $h_{j,i-2}$ and $h_{j,i-1}$ for $j = 1$ or 2, we first approximate $h_{j,i}$ by $h_{j,i-1} + (h_{j,i-1} - h_{j,i-2})$, and improve the approximation of the zero to high precision using Newton's method. For $J_\nu(x)$ and $Y_\nu(x)$, asymptotic expansions are available which can be used with Newton's method to find the first two zeros to initialize the method. Unfortunately, such expansions are not available for h_1 or h_2.

However, we are not interested in the first zeros of h_1 or h_2, but are interested in the first zeros after the largest of the first zeros of $Y_a(\rho x)$ and $Y_b(\tau x)$. If we redefine $h_{j,1}$ as the first zero of h_j *after* the Bessel functions of the second kind no longer dominate for $j = 1$ or 2, then they can be found as follows: find an approximation to the first zeros of $Y_a(\rho x)$ and $Y_b(\tau x)$ from the appropriate asymptotic expansion. Olver [6] showed that the first zero of $Y_a(x)$, denoted $y_{a,1}$, is approximately

$$\begin{aligned} y_{a,1} \ \simeq\ & a + 0.9315768a^{1/3} + 0.260351a^{-1/3} + 0.01198a^{-1} - \\ & 0.0060a^{-5/3} - 0.001a^{-7/3} + \ldots . \end{aligned} \tag{9}$$

While this asymptotic approximation only achieves high accuracy for large a, it still gives results of sufficient accuracy for our purposes for $a \simeq 1$. Starting from the largest of the two initial zeros, increase x with sufficiently small increments until two successive values of the function h_j have different sign. Take the midpoint of these two successive values as the initial approximation to $h_{j,1}$, and use Newton's method to obtain a more accurate result. To then find $h_{j,2}$, start with $h_{j,1}$ and again take sufficiently small steps in x until there is a change in sign, and use Newton's method to get a more accurate result. Once $h_{j,1}$ and $h_{j,2}$ have been found, later zeros can be found by the stepping method described above.

To complete the algorithm, we need a definition of 'sufficiently small increments' for x. Using the asymptotic results (7), we know that for large x the zeros of h_1 and h_2 are separated by $\pi/(\rho+\tau)$ and $\pi/|\rho-\tau|$, respectively. For most values of the parameters a, b, ρ, and τ, these asymptotic values are approached quickly, and the initial separation between zeros is reasonably close to the asymptotic value. Thus, a reasonable choice for sufficiently small steps is $\pi/[4(\rho+\tau)]$ for h_1 and $\pi/[4|\rho-\tau|]$ for h_2. However, if $\rho/\tau \simeq 1$ and a and b are widely separated, then there are further complications due to nonregular behaviour of h_2, which we shall discuss next.

3.3 Initial Poor Behaviour of h_2

For cases where a and b are close together (e.g. $|a-b| \le 5$), h_2 is well behaved. However, as the orders of the Bessel functions are increasingly separated, h_2 will not immediately approach simple oscillatory behaviour for $\rho \simeq \tau$. Figure 2 plots $h_2(x)$ when the orders are separated by 20 and 100, and $\rho \simeq \tau$. The plots begin at the first zero of h_2 after the Bessel functions of the second kind no longer dominate. We see that there is an intermediate region before simple oscillation begins, or oscillation begins immediately, but with increasingly small initial gap between zeros as the difference between orders increases. The larger the difference between orders of the Bessel functions, the larger the range of ρ/τ for which h_2 exhibits this intermediate behaviour. For ρ/τ sufficiently different from one, h_2 will settle into simple oscillation immediately. It should be noted that h_1 is perfectly well behaved whatever values of the parameters are used; it is only the low frequency behaviour of $J_a(\rho x)J_b(\tau x)$ which exhibits poor behaviour.

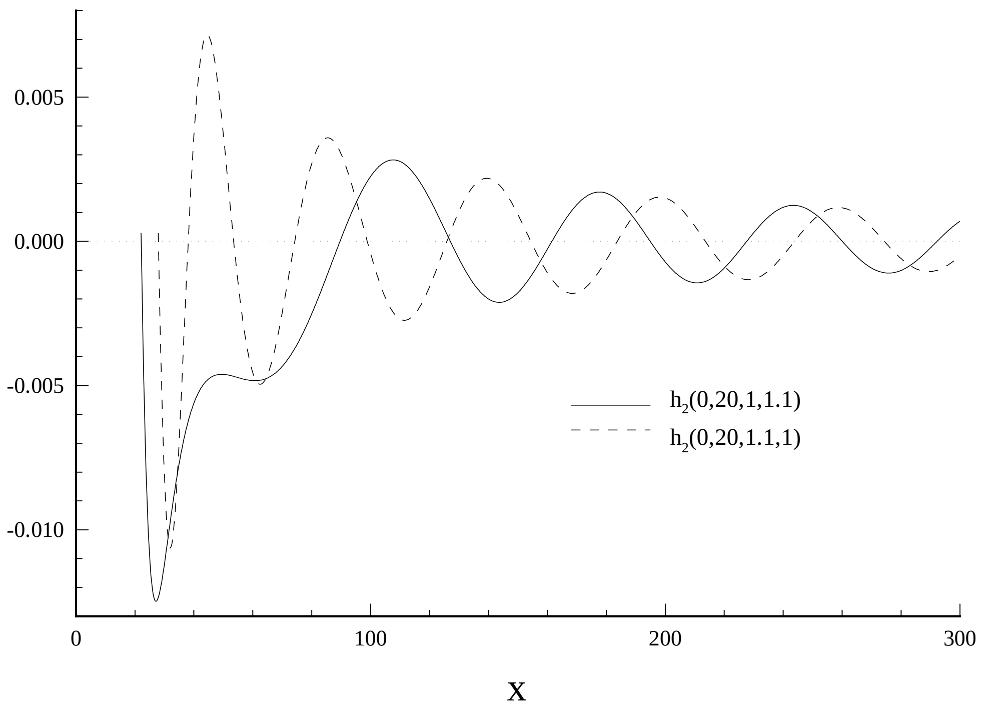
Finding the zeros of h_2 for examples such as those in figure 2 will not be successful by the method described in section §3.2. Indeed, it is only where h_2 settles down to simple oscillation that we should apply extrapolation to improve convergence. Unfortunately, there is no simple analytic method for deciding where simple oscillation will begin, based on knowing a, b, ρ and τ. However, there is a way around this problem.

Lyness [5] observed that it is not really necessary to know the exact zeros of a Bessel function to use extrapolation as a method for evaluating integrals such as (2). Lyness suggested using the zeros of the asymptotic cosine approximation (6a) to the Bessel function as endpoints for integration, and using extrapolation on these partial sums. In Lucas and Stone [4], it was shown that this method, using the ϵ-algorithm for extrapolation, could be used even for large order Bessel functions, although results were somewhat poorer than for the case where the zeros of the Bessel function were used. This suggests that instead of using the exact zeros of h_2, which are difficult to find numerically when $\rho \simeq \tau$ and a and b are well separated, we can use the zeros of h_2's asymptotic approximation (7b). While the initial terms in the sequence of partial sums will not be converging until the oscillatory behaviour of h_2 begins, the ϵ-algorithm as implemented in QUADPACK's routine `dqext( )` is such that early poor results are ignored, and when convergence begins, it is detected and appropriate extrapolated results returned.

Since there is no simple way of deciding whether the parameters for a particular integral are such that h_2 will not initially be well behaved beyond actually plotting h_2 and observing the form of the function, if there is any doubt then after the Bessel functions of the second kind no longer dominate we recommend using the zeros of (7b) as the endpoints of integrals, which leads to a sequence of partial sums for the integral involving h_2. The ϵ-algorithm is used for extrapolation in this case. Again, we emphasize that h_1 is always well behaved, its exact zeros can be found quickly and easily in all cases, and the mW transform will be the most efficient extrapolation technique for the integral involving h_1.

3.4 Algorithm For the Method

Now that the complications due to the singular nature of h_1 and h_2 at $x = 0$ and the possible poor behaviour of h_2 have been recognized, we outline an algorithm for the solution of infinite integrals involving the product of Bessel functions (1). The routines `dqdag( )` and `dqdagi( )` are IMSL [13] routines for adaptive integration over finite and infinite integrals respectively, `dqext( )` is the QUADPACK [7] routine which implements the ϵ-algorithm, and `mWtrans( )` is a routine to implement the mW transform of Sidi [8]. The algorithm we propose here is outlined in figure 3. For a given function $f(x)$, parameters a, b, ρ and τ, and a requested error bound, we return the integral value as well as an estimate of its error, and the number of function evaluations required.



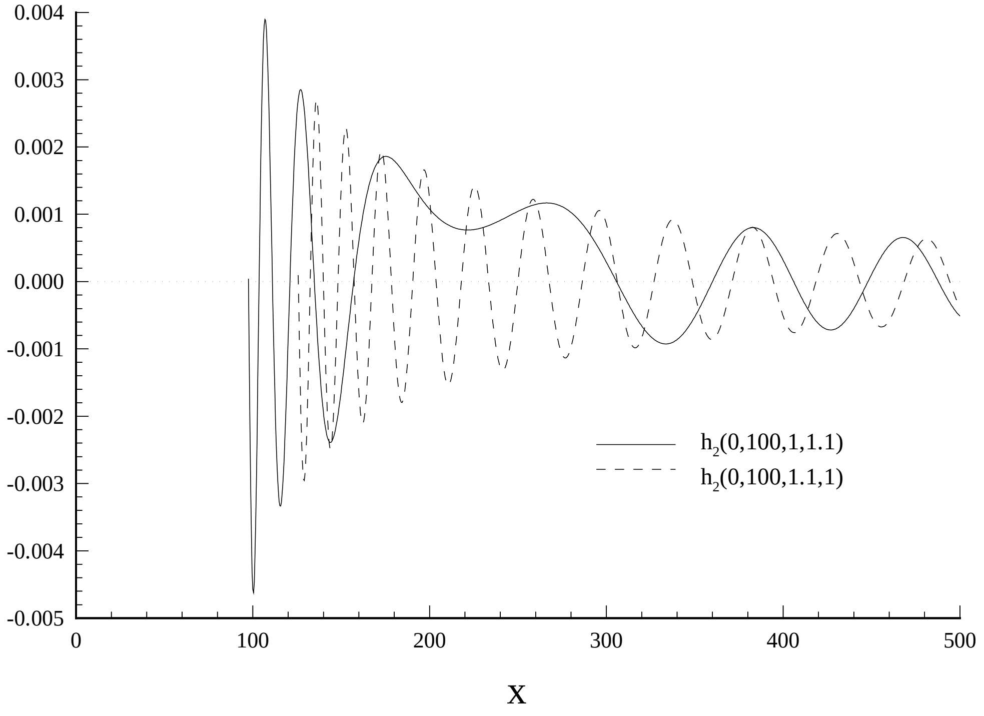


Figure 2: Graphs of (a) $h_2(0, 20, 1, 1.1)$, $h_2(0, 20, 1.1, 1)$, and (b) $h_2(0, 100, 1, 1.1)$ and $h_2(0, 100, 1.1, 1)$. Note the more complicated behaviour of h_2 when $\rho \simeq \tau$ and a and b are widely separated.

```
Input $a$, $b$, $\rho$, $\tau$, and a requested error bound
**If** $\rho = \tau$, **Then**
    Calculate $h_{1,1}$, and for the following integrals, let the required error bound be one
      third the input requested error bound
    Evaluate $I_1 = \int_0^{h_{1,1}} f(x) J_a(\rho x) J_b(\tau x)\, dx$ using dqdag( )
    Evaluate $I_2 = \int_{h_{1,1}}^{\infty} f(x) h_2(x; a, b, \rho, \tau)\, dx$ using dqdagi( )
    Evaluate $I_3 = \int_{h_{1,1}}^{\infty} f(x) h_1(x; a, b, \rho, \tau)\, dx$ using the ISE method based on mWtrans( )
    Return $I_{a,b,\rho,\tau} = I_1 + I_2 + I_3$, as well as the estimate of error and total number of
      function evaluations
**Else** ($\rho \neq \tau$)
    Calculate $h_{1,1}$, $h_{2,1}$, and for the following integrals, let the required error bound be
      one quarter the input requested error bound
    Evaluate $I_2 = \int_{h_{1,1}}^{\infty} f(x) h_1(x; a, b, \rho, \tau)\, dx$ using the ISE method based on mWtrans( )
    Evaluate $I_3 = \int_{h_{2,1}}^{\infty} f(x) h_2(x; a, b, \rho, \tau)\, dx$ using the ISE method based on dqext( )
    **If** $h_{1,1} < h_{2,1}$ **Then**
        Evaluate $I_1 = \int_0^{h_{1,1}} f(x) J_a(\rho x) J_b(\tau x)\, dx$ using dqdag( )
        Evaluate $I_4 = \int_{h_{1,1}}^{h_{2,1}} f(x) h_2(x; a, b, \rho, \tau)\, dx$ using dqdag( )
    **Else** ($h_{1,1} > h_{2,1}$)
        Evaluate $I_1 = \int_0^{h_{2,1}} f(x) J_a(\rho x) J_b(\tau x)\, dx$ using dqdag( )
        Evaluate $I_4 = \int_{h_{2,1}}^{h_{1,1}} f(x) h_1(x; a, b, \rho, \tau)\, dx$ using dqdag( )
    **Endif**
    Return $I_{a,b,\rho,\tau} = I_1 + I_2 + I_3 + I_4$, as well as the estimate of error and total number
      of function evaluations
**Endif**
```

Figure 3: The algorithm for a method of evaluating integrals such as (1).

4 Results

Based on the discussion in §3, we have implemented for comparison purposes the following three versions of the method for evaluating integrals such as (1):

approx – use the zeros of the cosine approximations (7) as the integral endpoints $h_{j,i}$ and the mW transform for both h_1 and h_2 infinite integrals,

exact – calculate and use the actual zeros of h_1 and h_2 as integral endpoints, and use the mW transform for both h_1 and h_2 infinite integrals,

general – calculate the zeros of h_1 as integral endpoints and use them with the mW transform for the infinite integral involving h_1, and for the infinite integral involving h_2 use zeros of the cosine approximation (7b) and the ϵ-algorithm.

The *approx* method is for use when a and b are zero or one only. The *exact* method is for a and b any integers, as long as the ratio ρ/τ is far enough from unity for h_2 to be well behaved. How far is 'far enough' depends on the difference between a and b as well as their magnitudes, and no easy way is known to calculate a bound upon which the *exact* method will work. The *general* method is designed to be successful for any combination of a, b, ρ and τ.

In the results reported here, we shall compare the numerical error to the number of function evaluations. The function evaluation count includes the number of times h_1 or h_2 are evaluated in calculating their zeros. This count is included because evaluating h_1 or h_2 is typically the most time consuming part of evaluating the integrands, and so is a better representation of the computational effort required in evaluating integrals such as (1). The error curves are obtained by running programs based on the algorithm in figure 3 with requested absolute errors of 10^{-n}, $n = 0, 1, 2, \ldots$, and obtaining a set of results whereby we can compare the actual error to the number of function evaluations. We consider the three test integrals:

$$\begin{aligned} &\int_0^\infty J_0(x)J_1(3x/2)\,dx = 2/3, \\ &\int_0^\infty x^{-4}J_0(x)J_5(2x)\,dx = 27/4096, \quad \text{and} \\ &\int_0^\infty \frac{x}{1+x^2}J_0(x)J_{20}(1.1x)\,dx \simeq -6.05074\,79030\,49 \times 10^{-3}. \end{aligned} \tag{10}$$

Equations (10a) and (10b) are specific cases of the Abramowitz and Stegun [1] equations 11.4.41 and 11.4.42, respectively. The result for (10c) is calculated by application of the *general* method with increasingly strict error bounds until convergence to machine precision is obtained.

Figure 4 shows the results for the three methods *approx*, *exact*, and *general* applied to integral (10a), a case where a and b are 0 and 1. The curves for the *approx* and *exact* methods have similar form, with the advantage to the *approx* method being almost entirely due to not having to calculate the zeros of h_1 or h_2. The *general* method is the least efficient of the three methods, which is expected due to the poorer convergence properties of the ϵ-algorithm compared to those of the mW transform, as discussed in Lucas and Stone [4]. When this integral was evaluated by the IMSL infinite integration routine `dqdagi( )`, the best it could do was an error of 2.6×10^{-2} with 14985 function evaluations, and a returned error code indicating slow convergence.

Figure 5 shows the results for the integral (10b) using the *exact* and *general* methods. Both the actual error and the estimated error returned from the routines are displayed. In this case the *general* method performs nearly as well as the *exact* method. In fact, if the estimate of error is used as the performance criterion, the two methods are nearly identical. This behaviour can be explained when we consider the 2nd to 5th actual errors for the *exact* method. In this region, the mW transform gives much better results than the ϵ-algorithm – so much better, in fact, that the error is due to error in the actual integrals between zeros. It is only from the 6th result for the *exact* method shown here that the integration error is small enough, and further terms are added to the sequence for the mW transform. Due to the slower convergence of the ϵ-algorithm, more terms are used, and so a better error estimate is produced. However, this may be considered of minor importance, since both methods provide excellent results for an otherwise intractable integral.

As a further guide to the performance of the algorithm in figure 3, figure 6 shows both (a) the oscillatory part of the integrand in (10b), and (b) the four pieces it is divided into according to the algorithm. The I_n correspond to which of the integrals from the algorithm in figure 3 is being calculated. The curve I_2 is $h_1(x; 0, 5, 1, 2)$, and the curve I_3 is $h_2(x; 0, 5, 1, 2)$. The much simpler oscillatory nature of these functions is why this technique is so efficient.

Finally, table 1 shows the results of evaluating the integral (10c) by the *general* method. This is a case where h_2 is badly behaved, and the *exact* method is not appropriate as there is no simple way of finding the zeros of h_2. The results again indicate that this method is well suited to evaluating integrals such as (1). Successive rows where number of function evaluations increase while the actual error stays the same is indicates that more integration points are required for a stricter error bound on some interval, but the increased accuracy in the terms of the sequence does not improve the extrapolated result on the sequence. Also included in figure 7 is a plot of the product $J_0(x)J_{20}(1.1x)$ as an indication of the difficulty such integrals represent.

5 Conclusion

We have developed a method for efficiently and accurately evaluating integrals of the form (1) by rewriting the product of Bessel functions as the sum of two functions whose oscillatory behaviour is such that an established ISE method performs as well as for the infinite integrals involving a single Bessel function. By using the adaptive IMSL routines `dqdag( )` and `dqdagi( )` for integration, the mW transform and ϵ-algorithm for extrapolation, and methods for finding or approximating the zeros of h_1 and h_2, we have developed an automatic routine that can successfully evaluate (1) for general required error bound and parameters a, b, ρ and τ. While the methods of finding *ymax* (see §3.1) and approximating the zeros of h_2 may not be ideal, the techniques described here have been shown to be successful methods.

Several straightforward refinements can be applied to this method as required. There is nothing in the theory behind this method that requires a and b to be integers. We have also

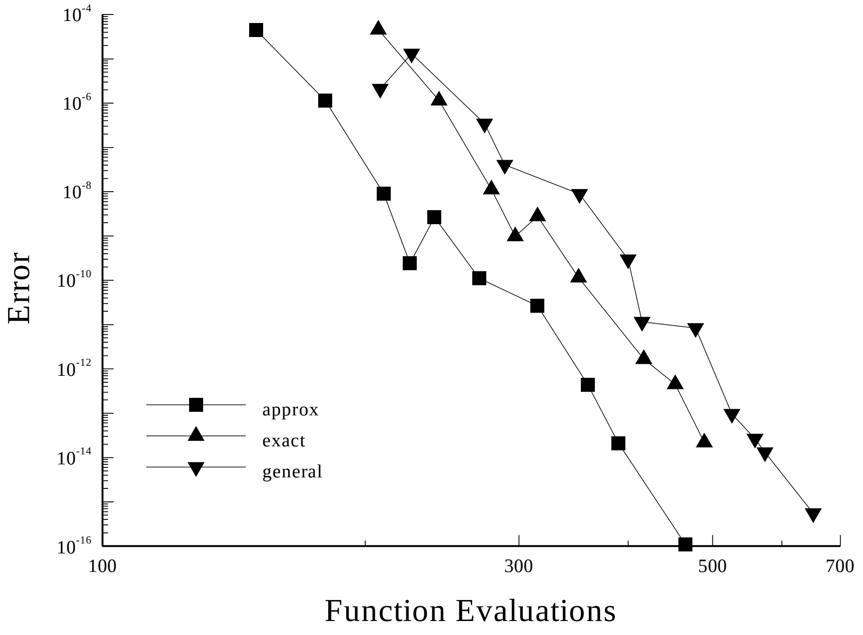


Figure 4: Comparing error to number of function evaluations for the integral (10a) by various methods.

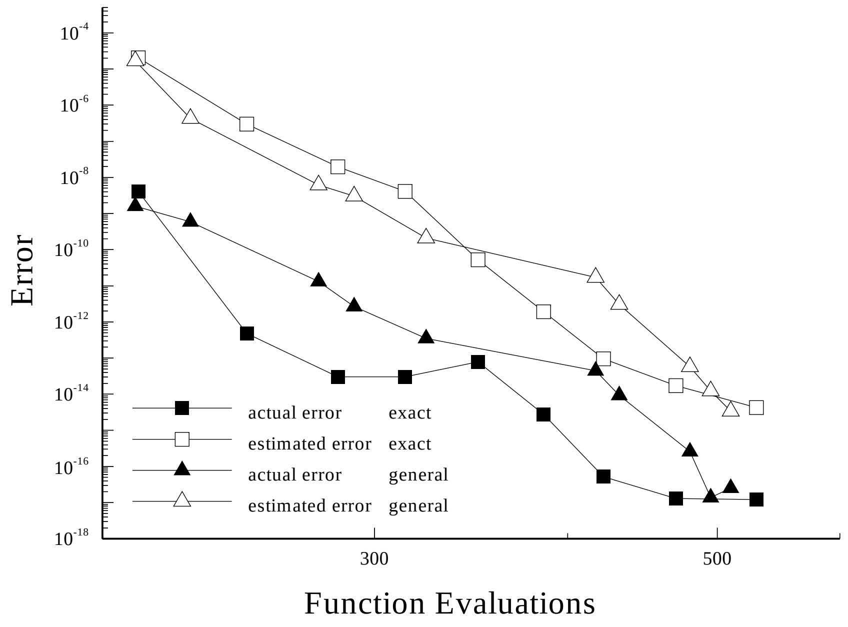
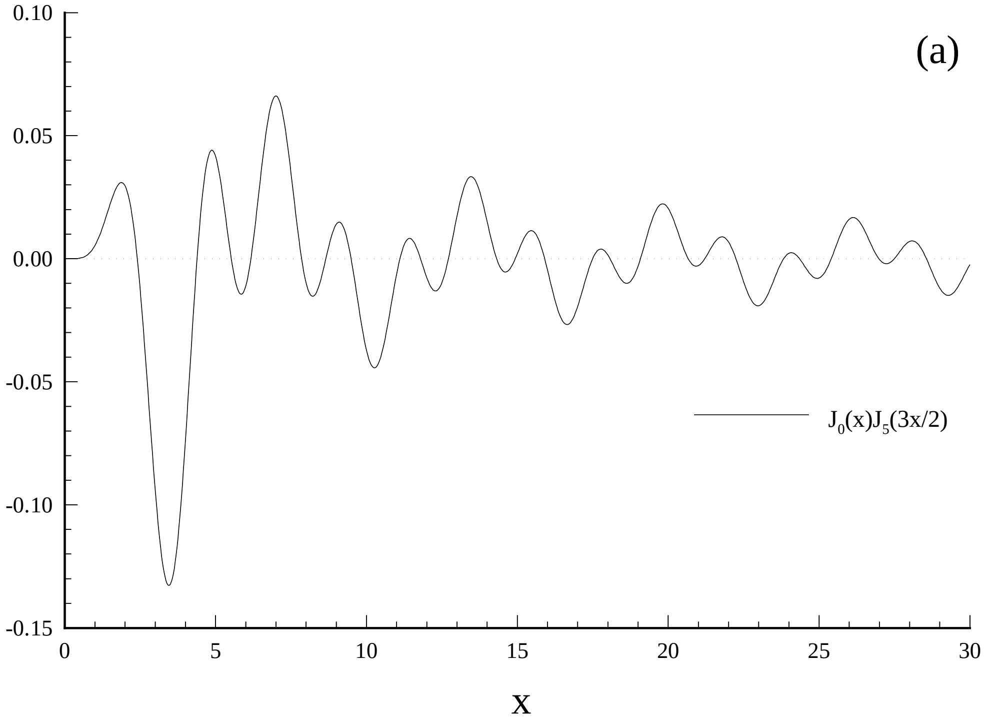


Figure 5: Comparing actual and estimated error to number of function evaluations for the integral (10b) by two methods.



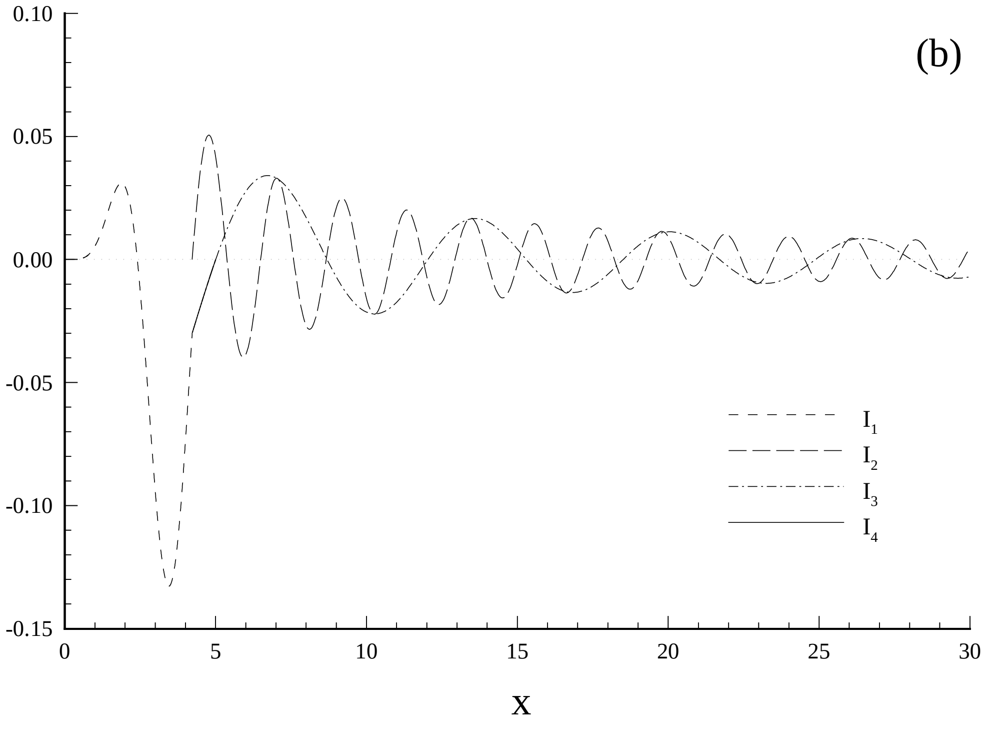


Figure 6: (a) The function $J_0(x)J_5(3x/2)$, and (b) the four pieces it is split into when integrated on $[0, \infty)$ by this method.

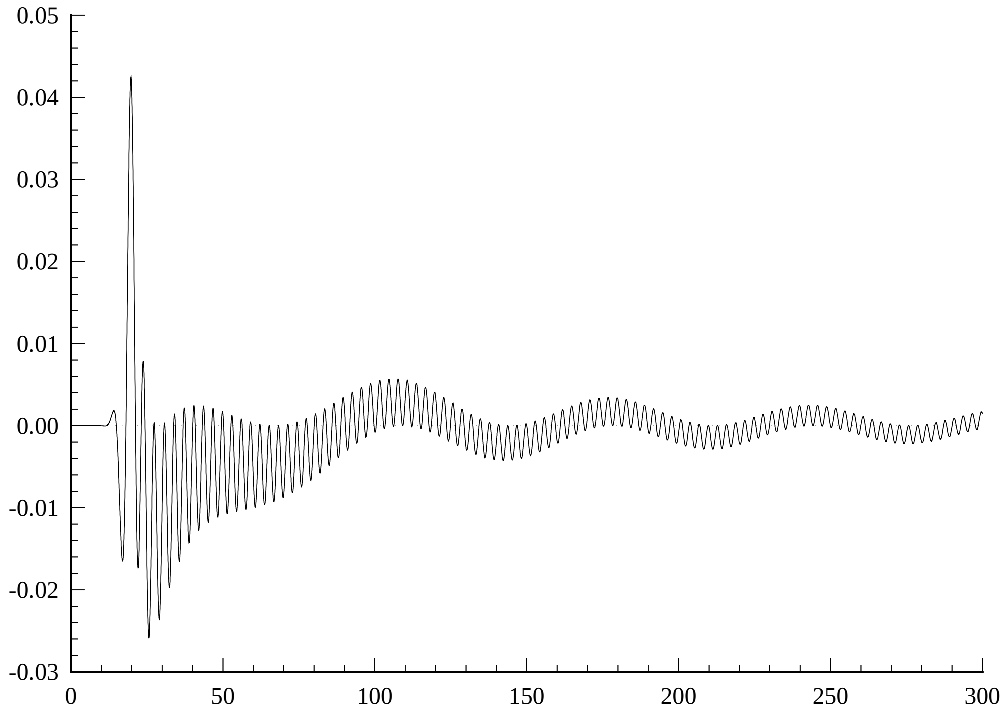


Figure 7: A plot of $J_0(x)J_{20}(1.1x)$.

formed a routine that can deal with Bessel functions of fractional order ν and ξ. Also, we are not limited to integrands involving $J_\nu(\rho x)J_\xi(\tau x)$. A product of any two Bessel functions of the first or second kinds can be written as a sum of two functions whose oscillatory behaviour is regular enough to be dealt with by an ISE method. In fact, since our technique is justified by the trigonometric behaviour of Bessel functions for large x, we have also been able to use this method to deal with products of sine or cosine functions and Bessel functions. Finally, although slightly more complicated in derivation, the method can also be applied to infinite integrals involving products of more than two Bessel functions of general order and/or sine or cosine functions, if such an integral is ever required.

Acknowledgements

We gratefully acknowledge NSF-PYI Award CT5-8957043 (to H.A. Stone) and the MERCK Foundation for funding for this study. Thanks also go to Dr. John Tanzosh for motivating this study.

References

[1] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions* (Dover, New York, 1972).

[2] A.M.J. Davis, Drag modifications for a sphere in a rotational motion at small non-zero Reynolds

req. err.	est. err.	N	act. err.
10^{0}	5.90×10^{-4}	239	6.54×10^{-7}
10^{-3}	1.61×10^{-4}	258	3.77×10^{-7}
10^{-4}	2.11×10^{-5}	318	3.77×10^{-7}
10^{-5}	1.28×10^{-6}	367	2.32×10^{-9}
10^{-6}	2.30×10^{-7}	415	6.07×10^{-9}
10^{-7}	1.18×10^{-8}	478	6.85×10^{-11}
10^{-8}	7.07×10^{-10}	541	5.14×10^{-13}
10^{-9}	4.30×10^{-10}	601	5.14×10^{-13}
10^{-10}	1.86×10^{-11}	664	3.00×10^{-13}
10^{-11}	1.32×10^{-12}	712	1.23×10^{-13}
10^{-12}	3.74×10^{-13}	805	5.34×10^{-15}
10^{-13}	2.05×10^{-15}	871	4.55×10^{-15}

Table 1: Results for integrating (10c) by the *general* method. Table headings are abbreviations for requested error, estimated error, number of function evaluations, and actual error.

---

and Taylor numbers: wake interference and possible Coriolis effects, *J. Fluid Mech.* **237** (1992) 13-22.

[3] P. Linz and T.E. Kropp, A note on the computation of integrals involving products of trigonometric and Bessel functions, *Math. Comp.* **27** (1973) 871-872.

[4] S.K. Lucas and H.A. Stone, Evaluating infinite integrals involving Bessel functions of arbitrary order, submitted to *J. Comput. Appl. Math.* (1994).

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

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

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

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

[9] H.A. Stone and H.M. McConnell, Hydrodynamics of quantized shape transitions of lipid domains, *Proc. Roy. Soc. London* (to appear) (1994).

[10] J. Tanzosh and H.A. Stone, Motion of a rigid particle in a rotating viscous flow: An integral equation approach, *J. Fluid. Mech.* **275** (1994) 225-256.

[11] M. Tezer, On the numerical evaluation of an oscillating infinite series, *J. Comput. Appl. Math.* **28** (1989) 383-390.

[12] R. Wong, Asymptotic expansion of $\int_0^{\pi/2} J_\nu^2(\lambda\cos\theta)\,d\theta$, *Math. Comp.* **50** (1988) 229-234.

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