IMA Journal of Numerical Analysis (2012) Page 1 of 15
doi:10.1093/imanum/drnxxx

Numerical solution of the modified Bessel equation

Michael Carley†

[Received on 30 April 2008; revised on 29 August 2008]

A Green's function based solver for the modified Bessel equation has been developed with the primary motivation of solving the Poisson and biharmonic equations in cylindrical geometries. The method is implemented using a Discrete Hankel Transform and a Green's function based on the modified Bessel functions of the first and second kind. The computation of these Bessel functions has been implemented to avoid scaling problems due to their exponential and singular behaviour, allowing the method to be used for large order problems, as would arise in solving the Poisson equation with a dense azimuthal grid. The method has been tested on monotonically decaying and oscillatory inputs, checking for errors due to interpolation and/or aliasing. The error has been found to reach machine precision and to have computational time linearly proportional to the number of nodes.

Keywords: modified Bessel function, Poisson equation, biharmonic equation

1. Introduction

This paper is motivated by the requirement for a Poisson solver for cylindrical domains. Such a solver is a basic element in solving a range of physical problems and accurate methods have been developed for Cartesian (Genovese *et al.*, 2007) and cylindrical (Chen *et al.*, 2000) domains. An issue which arises in solving the problem in a cylindrical coordinate system is the singularity which arises at the axis due to the form of the differential operator. A recent paper by Pataki & Greengard (2011) introduces a Green's function solver for the Poisson equation, which avoids problems with this singularity, and automatically imposes a radiation boundary condition, by using an integral formulation for the solution.

The method which Pataki & Greengard (2011) develop is based on Fourier transforms in the axial and azimuthal coordinate, followed by the solution of a modified Bessel equation in the radial coordinate. They test the accuracy of their method on an axisymmetric problem with monotonic decay in radius, and show its application to an asymmetric problem. A problem which arises in their algorithm is that the integration technique used does not work well for large azimuthal orders, i.e. meshes dense in angle, or for large axial wavenumbers, i.e. meshes dense in the axial coordinate, due to the poor scaling of the modified Bessel functions which appear in the Green's function for the problem.

This motivated an attempt to find a robust method for solving the modified Bessel equation, which would work for large wavenumbers and azimuthal orders. The main application is intended to be the modelling of flows in axisymmetric domains which requires high resolution in the azimuthal direction, e.g. 160 nodes for a turbulent jet (Freund, 2001), 512 nodes for turbulent thermal convection (Shishkina *et al.*, 2009). Other applications may require yet finer resolution: a two-dimensional problem in astrophysics has used 3200 points in azimuth (Li *et al.*, 2009).

The method developed, which is described in the rest of this paper, circumvents numerical difficulties in the high order solver through the use of the Discrete Hankel Transform (DHT), tabulated

†Department of Mechanical Engineering, University of Bath, Bath BA2 7AY, United Kingdom (m.j.carley@bath.ac.uk)

integrals, and recursions for the modified Bessel functions.

2. Problem formulation

The Poisson equation in cylindrical coordinates is:

$$u_{rr}(r,\theta,z)+\frac{1}{r}u_r(r,\theta,z)+\frac{1}{r}u_{\theta\theta}(r,\theta,z)+u_{zz}(r,\theta,z)=f(r,\theta,z), \tag{2.1}$$

where (r,θ,z) are cylindrical coordinates, u is the solution and f is some forcing term. With the problem defined on nodes regularly spaced in θ and z, this equation can be solved (Pataki & Greengard, 2011) by using the FFT to Fourier transform u and f in θ and z to yield a set of modified Bessel equations:

$$u^{(n)}_{rr}(r,\kappa)+\frac{1}{r}u^{(n)}_r(r,\kappa)-\left(\frac{n^2}{r^2}+\kappa^2\right)u^{(n)}(r,\kappa)=f^{(n)}(r,\kappa), \tag{2.2}$$

where n is the azimuthal order and κ the axial wavenumber, with $u^{(n)}$ and $f^{(n)}$ the Fourier transformed solution and forcing term respectively. After solving Equation 2.2 for each value of n and κ, $u^{(n)}(r,\kappa)$ can be inverse Fourier transformed to yield the solution $u(r,\theta,z)$.

Pataki & Greengard (2011) give a method for solving this modified Bessel equation, subject to a radiation boundary condition at some outer radius $r=R$. This is done using the Green's function for the modified Bessel equation with the solution written:

$$u^{(n)}(r,\kappa)=\int_0^R G_n(\kappa,r,s)f^{(n)}(s,\kappa)\,\mathrm{d}s. \tag{2.3}$$

The Green's function G_n is:

$$G_n(\kappa,r,s)=\begin{cases} -sI_n(\kappa r)K_n(\kappa s), & r\leqslant s;\\ -sK_n(\kappa r)I_n(\kappa s), & r\geqslant s,\end{cases} \tag{2.4}$$

where $I_n(x)$ and $K_n(x)$ are the modified Bessel functions of the first and second kind respectively. The first of these, I_n, grows exponentially, while K_n decays exponentially, but has $x^{-n}\log x$ behaviour at the origin, leading to some numerical problems caused by the scaling of the Green's function. In practice, the solution is computed as:

$$\begin{aligned}u^{(n)}(r,\kappa)=&-\frac{K_n(\kappa r)}{K_n(\kappa R_m)}\int_0^r I_n(\kappa s)K_n(\kappa R_m)f^{(n)}(s)s\,\mathrm{d}s\\ &-I_n(\kappa r)K_n(\kappa R_m)\int_r^R\frac{K_n(\kappa s)}{K_n(\kappa R_m)}f^{(n)}(s)s\,\mathrm{d}s,\end{aligned} \tag{2.5}$$

where the radial domain is divided at radii R_m which are used to set reference values of the modified Bessel functions. The product $I_n(\kappa s)K_n(\kappa R_m)$, and the ratio $K_n(\kappa s)/K_n(\kappa R_m)$, are thus well-scaled avoiding problems in computation, as long as $|\kappa(s-R_m)|$ is not too large. Pataki (2011) reports that the integration is performed using a dyadic grid, and that by scaling the modified Bessel functions as they are computed, the method works well for $n\lesssim 40$, corresponding to 80 points in the azimuthal mesh.

For many applications (Freund, 2001; Shishkina *et al.*, 2009; Li *et al.*, 2009, for example) it is desirable to use a denser mesh than this and so a different approach was sought for the solution of

Equation 2.2. The natural transform technique for problems in polar coordinates is the Discrete Hankel Transform (DHT), which expands a function as a series of ordinary Bessel functions $J_n(x)$. For a function $f(r)$, $0 \leqslant r \leqslant R$:

$$f(r) \approx \sum_{m=1}^{M} \hat{f}_m J_n(\alpha_m r) \tag{2.6}$$

where $J_n(\alpha_m R) \equiv 0$ and $\hat{f}_m$ denotes the mth coefficient of the expansion of f. If the DHT of $f^{(n)}(r,\kappa)$ is available, the solution of Equation 2.2 can be immediately written:

$$u^{(n)}(r,\kappa) = \sum_{m=1}^{M} \hat{f}_m \int_0^R J_n(\alpha_m s) G_n(\kappa, r, s)\,\mathrm{d}s. \tag{2.7}$$

The integrals required in Equation 2.7 are given in standard tables (Gradshteyn & Ryzhik, 1980; Prudnikov *et al.*, 2003):

$$\int_0^r I_n(\kappa s) J_n(\alpha s) s\,\mathrm{d}s = \left[\alpha J_{n+1}(\alpha r) I_n(\kappa r) + \kappa I_{n+1}(\kappa r) J_n(\alpha r)\right] \frac{r}{\alpha^2+\kappa^2}, \tag{2.8a}$$

$$\int_0^r K_n(\kappa s) J_n(\alpha s) s\,\mathrm{d}s = \left[\left(\frac{\alpha}{\kappa}\right)^n + \alpha r J_{n+1}(\alpha r) K_n(\kappa r) - \kappa r K_{n+1}(\kappa r) J_n(\alpha r)\right] \frac{1}{\alpha^2+\kappa^2} \tag{2.8b}$$

which gives a solution for the problem in terms of the DHT coefficients $\widehat{f^{(n)}}_m$ and κ. As written, this solution is correct, but not numerically useful, due to the poor scaling of the modified Bessel functions, especially for large values of κ and/or n.

The reason for the failure of Equations 2.8 for large κ and/or n is seen most clearly by examining the product $I_n(\kappa r)K_n(\kappa R)$ which is required when Equation 2.4 is substituted into Equation 2.7. In practice, this cannot be computed directly for large order or wavenumber, since I_n will be prone to overflow and K_n to underflow. A practical way of computing the product would be to compute:

$$I_n(\kappa r)K_n(\kappa R) = \left[I_n(\kappa r)\mathrm{e}^{-\kappa r}\right]\left[K_n(\kappa R)\mathrm{e}^{\kappa R}\right]\mathrm{e}^{\kappa(r-R)}, \tag{2.9}$$

where the terms in square brackets are the scaled versions of the modified Bessel functions which can be computed directly by numerical libraries (Galassi *et al.*, 2005). This leaves the need to compute the exponential $\exp[\kappa(r-R)]$. If this is to be computed accurately, the machine imposes a limit on $\kappa(R-r)$ (note that $R \geqslant r$). If r is small, this is effectively a limit on κR. If we assume the size of the domain is fixed, this limits the discretization which can be applied in z. In a practical application, the mesh will be refined in all three axes simultaneously, so that as κ increases, so does n, leading to faster growth of I_n, faster decay of K_n and larger values of κR, resulting in overflow and underflow. In the next section, the integral formula is rewritten to avoid the problems introduced by mesh refinement and higher azimuthal order.

3. Numerical implementation

In order to avoid numerical difficulties caused by poor scaling of the modified Bessel function, Equations 2.8 are rewritten, noting that (Gradshteyn & Ryzhik, 1980, 8.477.1)

$$I_n(\kappa r)K_{n+1}(\kappa r) + K_n(\kappa r)I_{n+1}(\kappa r) = \frac{1}{\kappa r},$$

and used to give the convolution of the Green's function with the ordinary Bessel function as:

$$\int_0^R G_n(\kappa,r,s)J_n(\alpha s)\,\mathrm{d}s =$$

$$-R\frac{I_n(\kappa r)K_n(\kappa R)}{\alpha^2+\kappa^2}\left[\alpha J_{n+1}(\alpha R)-\kappa J_n(\alpha R)\frac{K_{n+1}(\kappa R)}{K_n(\kappa R)}\right]-\frac{J_n(\alpha r)}{\alpha^2+\kappa^2},\quad r\neq 0, \tag{3.1a}$$

$$=\frac{1}{\alpha^2+\kappa^2},\quad r=0,n=0, \tag{3.1b}$$

$$=0,\quad r=0,n\neq 0. \tag{3.1c}$$

Written in this form, the modified Bessel functions appear only as the products $I_n(x)K_n(x)$ and $I_n(\kappa r)K_n(\kappa R)$, which can be computed using ratios of modified Bessel functions, $I_{n+1}(x)/I_n(x)$ and $K_{n+1}(x)/K_n(x)$. These ratios can be computed directly using standard functional relations, while the products are calculated using the same ratios combined with modified Bessel functions of order zero, which can be computed accurately and stably. Implementation of the solution technique thus requires two elements, a method for the calculation of ratios of modified Bessel functions, and a method for computing the DHT. In practice, the input will not be defined on the nodes of the DHT, so an interpolation scheme will also be required. The method has been coded making use of GSL, the GNU Scientific Library (Galassi *et al.*, 2005), which provides functions for the computation of scaled versions of the modified Bessel functions, directly returning $I_n(x)\exp[-x]$ and $K_n(x)\exp[x]$. The algorithm has been designed to use these scaled functions, to avoid problems of underflow and overflow.

3.1 *Ratios of modified Bessel functions*

The ratios of modified Bessel functions can be computed using standard functional relations (Gradshteyn & Ryzhik, 1980, 8.486):

$$I_{n-1}(x)=\frac{2n}{x}I_n(x)+I_{n+1}(x), \tag{3.2a}$$

$$K_{n+1}(x)=\frac{2n}{x}K_n(x)+K_{n-1}(x), \tag{3.2b}$$

using an approach similar to that of Amos (1974), who writes the ratios of successive functions as:

$$\frac{I_n(x)}{I_{n-1}(x)}=\frac{x}{2n+xI_{n+1}(x)/I_n(x)}, \tag{3.3a}$$

$$\frac{K_{n+1}(x)}{K_n(x)}=\frac{2n}{x}+\frac{K_{n-1}(x)}{K_n(x)}. \tag{3.3b}$$

The recursion for $I_n(x)/I_{n-1}(x)$ is stable for descending n while that for $K_{n+1}(x)/K_n(x)$ is stable for increasing n. The recursion for $K_{n+1}(x)/K_n(x)$ is seeded with $K_1(x)/K_0(x)$, computed using the scaled form of $K_0(x)$ and $K_1(x)$. The recursion for $I_n(x)/I_{n-1}(x)$ is seeded using Olver's asymptotic formula (National Institute of Standards and Technology, 2010, 10.41.10) for modified Bessel functions of large order, as recommended by Amos (1974), starting at order equal to the larger of $n+8$ and 32.

The asymptotic expansion is given by:

$$I_n(x) \sim \frac{\mathrm{e}^{n\eta}}{(2\pi n)^{1/2}(1+z^2)^{1/4}} \sum_{q=0}^{\infty} \frac{u_q(t)}{n^q}, \tag{3.4}$$

$$z = x/n,\ \eta = (1+z^2)^{1/2} + \log \frac{z}{1+(1+z^2)^{1/2}},\ t = 1/(1+z^2)^{-1/2},$$

$$u_0(t) = 1,$$

$$u_1(t) = (3t - 5t^3)/24,$$

$$u_2(t) = (81t^2 - 462t^4 + 385t^6)/1152,$$

$$u_3(t) = (30375t^3 - 369603t^5 + 765765t^7 - 425425t^9)/414720,$$

while for small arguments, $(x/2)^2 < n+1$, the series expansion of $I_n(x)$ is used (Gradshteyn & Ryzhik, 1980, 8.445).

Given a sequence of ratios of modified Bessel functions, the products in Equation 3.1 can be computed as:

$$I_n(\kappa r)K_n(\kappa r) = \left[I_0(\kappa r)\mathrm{e}^{-\kappa r}\right]\left[K_0(\kappa r)\mathrm{e}^{\kappa r}\right] \prod_{i=0}^{n-1} \left[\frac{I_{i+1}(\kappa r)}{I_i(\kappa r)}\right]\left[\frac{K_{i+1}(\kappa r)}{K_i(\kappa r)}\right], \tag{3.5}$$

and

$$I_n(\kappa r)K_n(\kappa R) = \left[I_0(\kappa r)\mathrm{e}^{-\kappa r}\right]\left[K_0(\kappa R)\mathrm{e}^{\kappa R}\right] \prod_{i=0}^{n-1} A \left[\frac{I_{i+1}(\kappa r)}{I_i(\kappa r)}\right]\left[\frac{K_{i+1}(\kappa R)}{K_i(\kappa R)}\right], \tag{3.6}$$

$$A = \mathrm{e}^{\kappa(r-R)/n},$$

where terms of the form $I_n(x)\exp[-x]$ and $K_n(x)\exp[x]$ are computed directly using the scaled form of the modified Bessel functions. The problems inherent in computing $\exp[-\kappa R]$ when κR is large are avoided by scaling on n to keep A in a reasonable range of values. The ratios of successive modified Bessel functions are well scaled and multiplying them in pairs as in the products of Equations 3.5 and 3.6 avoids underflow and overflow problems.

3.2 *Discrete Hankel Transform*

The coefficients of the DHT are computed using the method of Lemoine (1994). This is essentially a quadrature rule based on the zeros of the ordinary Bessel function of order n, $J_n(x)$. The function to be transformed is specified at these zeros x_m, $0 \leqslant m < M$, $J_n(x_m) = 0$, and the DHT is given by a matrix multiplication of the vector of input data $f(x_m)$ with the matrix entries given by:

$$B^{(n)}_{mj} = \frac{2}{x_M} \frac{J_n(x_m x_j / x_M)}{|J_{n+1}(x_m)J_{n+1}(x_j)|}. \tag{3.7}$$

In the calculations presented here, the GSL implementation (Galassi *et al.*, 2005) of Lemoine's method was used, but with a modification to compute the zeros of $J_n(x)$ using the $O(M)$ algorithm of Glaser *et al.* (2007).

In order to compute the DHT, the input must be specified at the zeros of the Bessel function. If only one order n is of interest, this presents no difficulties, but if the solution is to be found for multiple values

of n, as in solving a Poisson equation, for example, an interpolation scheme is required to transfer the input from the problem mesh onto the DHT nodes, in particular because the zeros are not the same for different orders of Bessel function.

3.3 *Interpolation*

Given that an interpolation scheme will almost always be needed, the approach used by Pataki & Greengard (2011) has been adopted. The domain $0 \leqslant r \leqslant R$ is divided into N blocks, $R_n \leqslant r \leqslant R_{n+1}$, $n = 0, \ldots, N-1$. Each block is discretized with P points, given by the Chebyshev nodes of the second kind:

$$r_{nP+p} = \frac{R_{n+1} + R_n}{2} + \frac{R_{n+1} - R_n}{2} \cos \frac{p\pi}{P}, \quad p = 0, 1, \ldots, P. \tag{3.8}$$

Evaluation of $f^{(n)}(r, \kappa)$ within each block is performed using barycentric Lagrangian interpolation (Berrut & Trefethen, 2004). Since Equation 3.1 can be computed directly at arbitrary values of r, the solution is generated on the input nodes, with no requirement for interpolation from the DHT nodes.

3.4 *Summary of algorithm*

Given the elements described above, the solution algorithm can be summarized as follows. For a given order n, wavenumber κ and input $f^{(n)}(r, \kappa)$, $0 \leqslant r \leqslant R$:

1. generate, if necessary, the DHT matrix and corresponding nodes r_m;
2. if necessary, interpolate $f^{(n)}(r, \kappa)$ onto the nodes r_m;
3. perform the DHT to yield $\widehat{f^{(n)}}_m$;
4. evaluate Equation 2.7 at the input nodes using Equations 3.1.

4. Numerical tests

The solution method is tested using a function which can be varied to examine the performance of the algorithm with regard to potential sources of error. The main sources of error in the algorithm arise from the interpolation schemes and aliasing. These errors arise in both the direct method, where the input is specified on the DHT nodes, and when the input must be interpolated from another mesh onto these points.

Interpolation errors arise when the interpolation scheme is unable to accurately resolve the function which is being interpolated. This can occur because the interpolation method proper does not have the required properties to give a well-converged estimate of the underlying function, or because the interpolation nodes are not dense enough to take advantage of an otherwise good interpolation method. In this sense, 'interpolation scheme' refers both to the explicitly stated polynomial interpolation method used to transfer data from the input mesh to the DHT nodes, and to the interpolation which is performed implicitly in the quadrature scheme of the DHT.

The second source of error is aliasing, when the point distribution is not dense enough to capture the spatial frequencies present in the input. This can happen in the DHT, if the analytically defined input has wavenumbers α_m with $m > M$, so that the expansion of Equation 2.6 does not contain the full set of radial wavenumbers α_m present in the input. Clearly, it can also happen in the Chebyshev interpolation

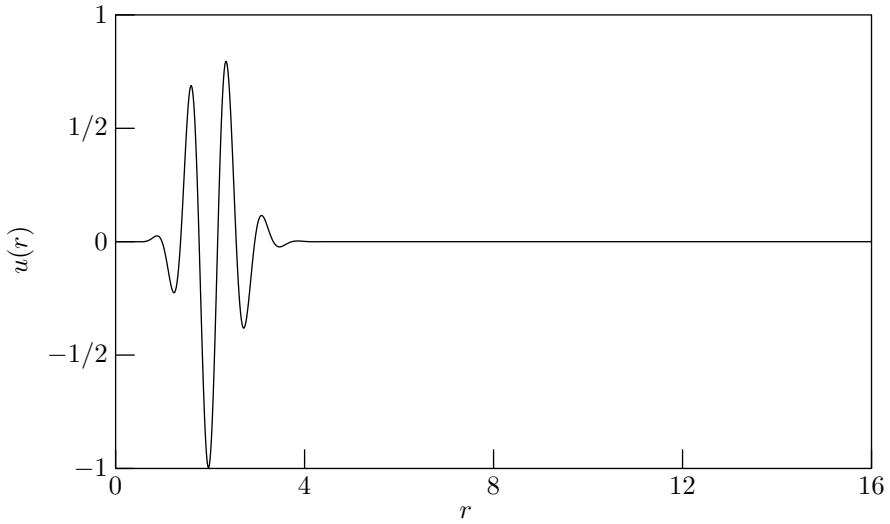

FIG. 1. Test function with $\alpha = 1$, $\beta = 8$, $m = 8$

scheme if the node density is not high enough, even if the DHT would otherwise contain enough nodes to capture the full behaviour of the input.

In order to assess the performance of the algorithm against these criteria, the test function:

$$u^{(n)}(r) = (r/r_{\max})^m \mathrm{e}^{-(r^2 - r_{\max}^2)/\alpha^2} \cos \beta r,\ m \geqslant n, \tag{4.1}$$
$$r_{\max} = \alpha(m/2)^{1/2},$$

has been adopted. By varying the parameters α and β, the function can be varied from monotonically decaying, as in Pataki and Greengard's test (2011), to oscillatory, Figure 1. The r^m term is required for validity of the solution and introducing the terms in $r_{\max}$ scales the amplitude of the cosine on its maximum, so that the maximum amplitude of $u^{(n)}$ is one, reducing errors caused by very large values of r^m.

In testing the algorithm, $\alpha = 1$, $P = 16$, and $R = 16$. The parameters varied were N the number of blocks in r, M the number of DHT nodes, β the frequency of $u^{(n)}$ and κ. The order n was tested up to 128, with $m = n$ in the evaluation of $u^{(n)}(r)$. The error measure is the L_∞ norm:

$$\varepsilon = \frac{\max |u_c^{(n)}(r, \kappa) - u^{(n)}(r)|}{\max |u^{(n)}(r)|}, \tag{4.2}$$

where $u_c^{(n)}(r, \kappa)$ is the computed solution.

4.1 *Solution accuracy and computation time*

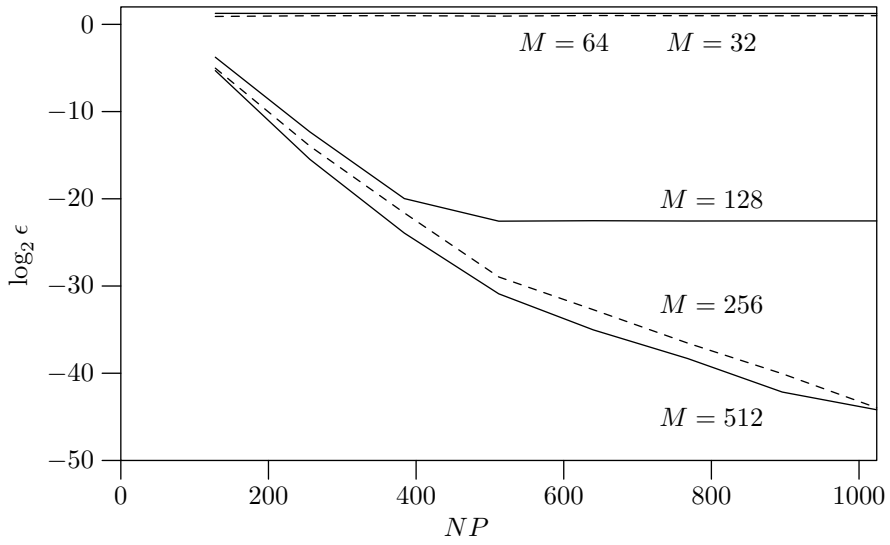

FIG. 2. Solution error for Hankel transform method: $n = 32$, $\kappa = 16$, $\beta = 16$; $M = 32, 64, 128, 256, 512$

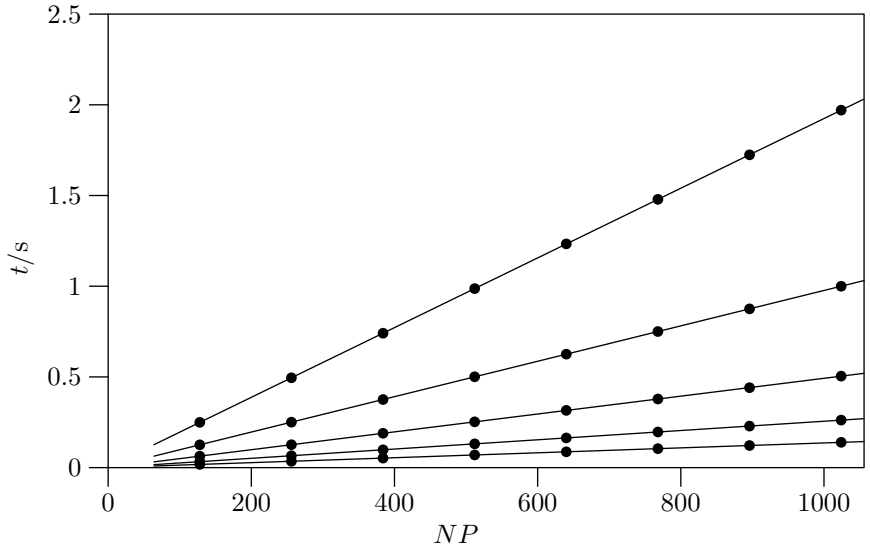

FIG. 3. Solution time for Hankel transform method: $n = 32$, $\kappa = 16$, $\beta = 16$; $M = 32, 64, 128, 256, 512$; symbols show computation time, solid line shows linear fit

Details of the accuracy of the method are presented in the next section, where they are compared to results computed using dyadic quadrature. Here, we show the trends to be expected from the Hankel transform technique, applied to a reference case. Figure 2 shows the error in the computed solution for a range of parameters where there is an aliasing problem for lower numbers of Hankel transform coefficients, but with the error reducing as the number of coefficients is increased, $M \geqslant 128$. Figure 3

TABLE 1. *Error in Hankel transform and dyadic quadrature solutions,* $\beta = 0$

κ	16		64		256	
n	ε_H	ε_D	ε_H	ε_D	ε_H	ε_D
16	2.1×10^{-14}	2.2×10^{-15}	2.1×10^{-14}	5.7×10^{-15}	2.1×10^{-14}	3.5×10^{-14}
32	1.0×10^{-14}	2.7×10^{-15}	1.5×10^{-14}	1.2×10^{-14}	1.5×10^{-14}	7.6×10^{-14}
64	4.3×10^{-14}	—	5.9×10^{-14}	—	5.5×10^{-14}	8.9×10^{-13}
128	1.8×10^{-13}	—	2.0×10^{-13}	—	2.0×10^{-13}	—

TABLE 2. *Error in Hankel transform and dyadic quadrature solutions,* $\beta = 16$

κ	16		64		256	
n	ε_H	ε_D	ε_H	ε_D	ε_H	ε_D
16	4.9×10^{-14}	3.3×10^{-15}	5.2×10^{-14}	7.0×10^{-15}	5.7×10^{-14}	5.6×10^{-14}
32	5.0×10^{-14}	5.6×10^{-15}	5.5×10^{-14}	1.0×10^{-14}	7.7×10^{-14}	9.5×10^{-14}
64	4.6×10^{-14}	—	4.9×10^{-14}	—	5.8×10^{-14}	7.7×10^{-13}
128	2.6×10^{-13}	—	2.5×10^{-13}	—	2.5×10^{-13}	—

TABLE 3. *Error in Hankel transform and dyadic quadrature solutions,* $\beta = 32$

κ	16		64		256	
n	ε_H	ε_D	ε_H	ε_D	ε_H	ε_D
16	1.6×10^{-09}	2.1×10^{-11}	1.2×10^{-09}	4.6×10^{-11}	1.5×10^{-09}	8.3×10^{-11}
32	1.8×10^{-09}	2.2×10^{-11}	1.1×10^{-09}	4.8×10^{-11}	1.5×10^{-09}	8.7×10^{-11}
64	1.8×10^{-09}	—	1.0×10^{-09}	—	1.4×10^{-09}	7.8×10^{-11}
128	1.1×10^{-09}	—	9.1×10^{-10}	—	1.3×10^{-09}	—

shows the computation time required for the solution. As might be expected, the effort scales linearly with the number of grid points. For a given NP, it also scales linearly with M the number of Hankel transform points, since the computational effort is dominated by the evaluation of Equation 3.1 while the application of the DHT takes a negligibly small time by comparison. The computational effort is thus $O(MNP)$, with accuracy controlled by both the Hankel transform and by the grid density.

4.2 *Comparison with dyadic quadrature*

A point of some interest, and the motivation for the development of the present method, is the failure of the dyadic quadrature approach for $n \gtrapprox 40$ (Pataki, 2011). Tables 1–4 give the solution error for β =0, 16, 32, 64 and n =16, 32, 64, 128 for the dyadic quadrature method, ε_D, and for the Hankel transform approach, ε_H. The dyadic quadrature was tested with 2,4,8, and 16 intervals using 8 and 16 point quadrature rules. The Hankel transform method used 32, 64, 128, 256, and 512 point transforms. The same mesh was used for both approaches, with a maximum of 1024 points. The reported error is the minimum found for each choice of κ, n and β. Failure of the dyadic quadrature method, due to floating point exceptions or 'not-a-number' (NaN) results, is marked in the tables by a dash. In the cases marked as failures, the dyadic quadrature approach failed for all numbers of intervals used.

TABLE 4. *Error in Hankel transform and dyadic quadrature solutions,* $\beta = 64$

κ	16		64		256	
n	ε_H	ε_D	ε_H	ε_D	ε_H	ε_D
16	3.0×10^{-04}	3.6×10^{-06}	1.0×10^{-04}	3.8×10^{-06}	7.8×10^{-05}	4.1×10^{-06}
32	1.8×10^{-04}	2.3×10^{-06}	6.2×10^{-05}	2.5×10^{-06}	5.0×10^{-05}	3.0×10^{-06}
64	3.1×10^{-04}	—	9.2×10^{-05}	—	7.6×10^{-05}	4.0×10^{-06}
128	1.6×10^{-04}	—	7.7×10^{-05}	—	7.7×10^{-05}	—

TABLE 5. *Computation time for Hankel transform and dyadic quadrature solution,* $\beta = 0$

κ	16		64		256	
n	t_H	t_D	t_H	t_D	t_H	t_D
16	0.20	0.99	0.26	1.46	0.25	2.59
32	0.51	0.44	0.19	1.39	0.25	2.50
64	1.06	—	0.89	—	1.05	1.80
128	0.78	—	0.60	—	0.59	—

TABLE 6. *Computation time for Hankel transform and dyadic quadrature solution,* $\beta = 16$

κ	16		64		256	
n	t_H	t_D	t_H	t_D	t_H	t_D
16	1.55	0.99	0.78	3.06	0.78	4.61
32	1.97	0.96	1.98	2.92	1.00	4.46
64	2.78	—	2.79	—	2.78	2.44
128	1.25	—	1.19	—	1.18	—

TABLE 7. *Computation time for Hankel transform and dyadic quadrature solution,* $\beta = 32$

κ	16		64		256	
n	t_H	t_D	t_H	t_D	t_H	t_D
16	1.55	0.99	1.56	3.06	1.55	2.67
32	1.97	1.90	1.98	2.92	1.00	5.13
64	2.78	—	2.79	—	2.78	2.44
128	4.66	—	4.51	—	4.47	—

In both cases, the error where a solution has been computed is small, and the errors from both methods are comparable to machine precision. Table 4 shows quite large errors, which appear to be due to aliasing when $\beta = 64$. This could arise in any method where the mesh cannot resolve the input and affects both the dyadic quadrature and Hankel transform results.

In all of the results presented, however, the dyadic quadrature method fails for $n = 128$ and for

TABLE 8. *Computation time for Hankel transform and dyadic quadrature solution,* $\beta = 64$

κ	16		64		256	
n	t_H	t_D	t_H	t_D	t_H	t_D
16	1.56	0.99	1.56	3.06	1.56	5.31
32	1.97	0.96	1.98	1.47	1.97	5.13
64	2.78	—	2.79	—	2.78	2.44
128	4.66	—	4.51	—	4.47	—

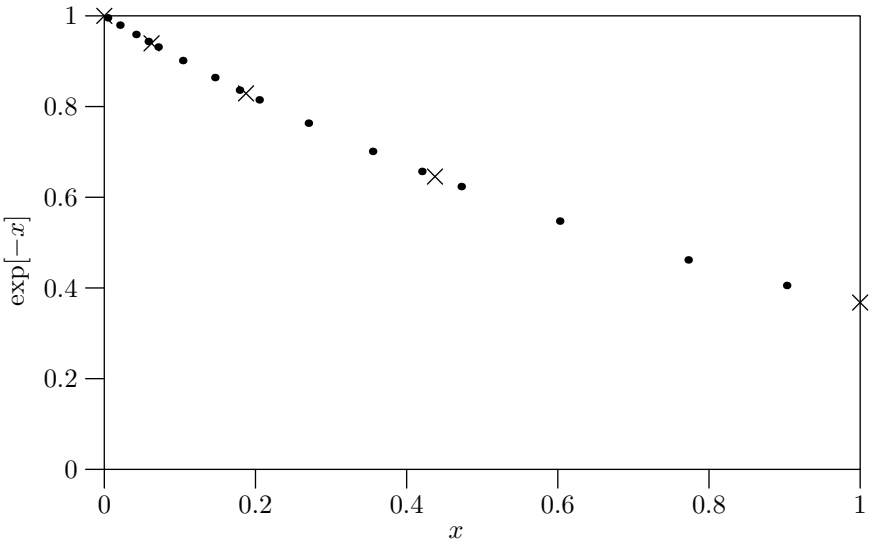

FIG. 4. Nodes for dyadic quadrature of a decaying exponential: dots, quadrature nodes; crosses, endpoints of sub-intervals.

$n = 64$, $\kappa = 16, 64$. In order to examine the breakdown of the method, and the reasons for it, we present a sample calculation using a dyadic grid with varying numbers of sub-intervals.

For an exponentially decaying integrand, dyadic quadrature works by dividing the interval of integration into a series of sub-intervals, each of whose lengths is twice that of its predecessor, in order to match the discretization to the behaviour of the function being integrated. (For an exponentially increasing integrand, successive sub-intervals have their lengths halved.) A quadrature rule is then applied to compute the integral on each sub-interval. Figure 4 shows the quadrature nodes (dots) and sub-interval limits (crosses) for integration of a decaying exponential from 0 to 1, using a four point Gauss–Legendre rule on four dyadic intervals.

The test integral to be evaluated is:

$$I = \int_0^r \frac{K_n(\kappa s)}{K_n(\kappa R_m)} f^{(n)}(s) s \, \mathrm{d}s, \tag{4.3}$$

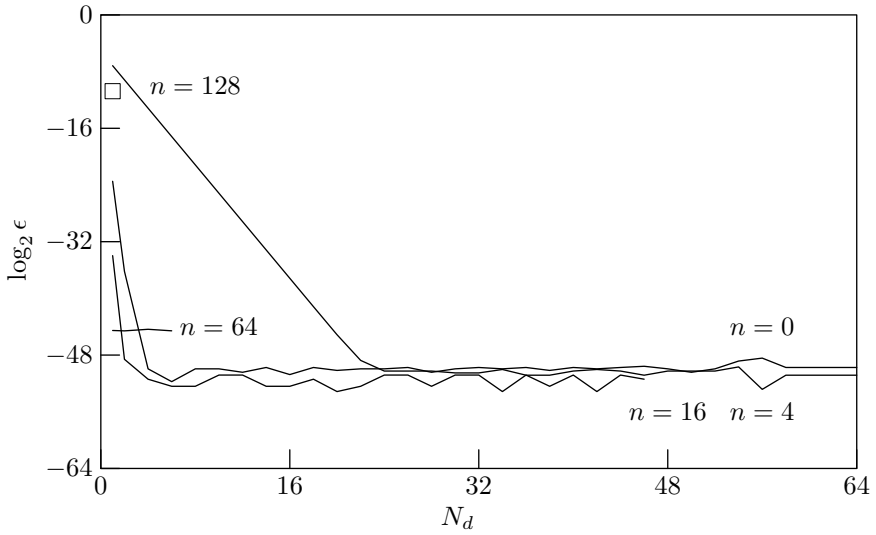

FIG. 5. Error in dyadic quadrature evaluation of Equation 4.3 for varying n

with

$$f^{(n)}(s) = J_n(\alpha s),$$

so that the function $f^{(n)}(s)$ has the required regularity properties at $s = 0$ and I is given by Equation 2.8. For the example presented here, $\kappa = \alpha = 128$, $r = 1/2$ and $R_m = 1$, with $0 \leqslant n \leqslant 128$. A 16 point Gauss–Legendre quadrature rule was used on N_d dyadic intervals, with $2 \leqslant N_d \leqslant 64$. The relative error was computed as:

$$\varepsilon = \frac{|I_c - I_r|}{|I_r|}, \tag{4.4}$$

where I_c is the computed value and I_r the exact, reference value.

Figure 5 shows the error as a function of N_d and of n. For the lower order problem, $n = 0, 4$, the error behaves as expected, with a reasonably large initial value, for a small number of dyadic sub-intervals, which then decreases to near machine precision as N_d increases. For $n = 16$, however, the method fails at $N_d = 46$, and for $n = 64$, it fails at $N_d = 6$. For $n = 128$, the method fails for $N_d > 1$. The reason for the failure is readily apparent from Figure 4: as the number of dyadic sub-intervals increases, there is a concentration of quadrature nodes near the origin. These nodes eventually have locations comparable to machine precision (or to the nth root of machine precision), leading to overflow problems in computing the singular function $K_n(\kappa s)$. Reducing the number of intervals, or using a lower order quadrature rule, can alleviate the problem, but at the expense of reduced accuracy. Likewise, using a smaller value of R_m to keep the ratio $K_n(\kappa s)/K_n(\kappa R_m)$ within reasonable bounds can improve the situation but only up to a point. The concentration of nodes near the origin is an unavoidable outcome of using dyadic quadrature, so that any measures to maintain accuracy are only delaying the inevitable breakdown of the method at some value of n where no reasonable quadrature rule escapes having a node too close to the origin.

5. Solution of the biharmonic equation

For completeness, we extend the solution method to the biharmonic equation $\nabla^4 u = f$, which is handled by Pataki & Greengard (2011) as part of the calculation of the collision operator. Applying Fourier transforms in θ and z as before gives a differential equation for the coefficients $u^{(n)}(r,\kappa)$:

$$u^{(n)}_{rrrr} + \frac{2}{r}u^{(n)}_{rrr} - \left(\frac{1+2n^2}{r^2} + 2\kappa^2\right)u^{(n)}_{rr} + \left(\frac{1+2n^2}{r^3} - \frac{2\kappa^2}{r}\right)u^{(n)}_{r} + \left(\frac{n^4-4n^2}{r^4} + \frac{2\kappa^2 n^2}{r^2} + \kappa^4\right)u^{(n)} = f^{(n)}. \tag{5.1}$$

As before, Equation 5.1 can be solved using a Green's function $G^{(b)}_n(\kappa,r,s)$, which is given by:

$$G^{(b)}_n(\kappa,r,s) = \begin{cases} -\left[I_n(\kappa r)sK'_n(\kappa s) + rI'_n(\kappa r)K_n(\kappa s)\right]s/2\kappa, & r \leqslant s; \\ -\left[K_n(\kappa r)sI'_n(\kappa s) + rK'_n(\kappa r)I_n(\kappa s)\right]s/2\kappa, & r \geqslant s, \end{cases} \tag{5.2}$$

where the prime denotes differentiation with respect to the argument. This Green's function is immediately recognizable as a derivative of the Green's function for the Poisson equation:

$$G^{(b)}_n(\kappa,r,s) = \frac{1}{2\kappa}\frac{\partial}{\partial\kappa}G_n(\kappa,r,s), \tag{5.3}$$

so that:

$$\begin{aligned}
&\int_0^R G^{(b)}_n(\kappa,r,s)J_n(\alpha s)\,\mathrm{d}s = \\
&-\frac{1}{2\kappa}\frac{R}{\alpha^2+\kappa^2}\left[\alpha R J_{n+1}(\alpha R)\left(\frac{n}{\kappa R} - \frac{K_{n+1}(\kappa R)}{K_n(\kappa R)}\right) + nJ_n(\alpha R)\left(\frac{\kappa R}{n} + \frac{K_{n+1}(\kappa R)}{K_n(\kappa R)}\right)\right]I_n(\kappa r)K_n(\kappa R) \\
&+\frac{1}{2\kappa}\left[\frac{2\kappa}{(\alpha^2+\kappa^2)^2} - \frac{r}{\alpha^2+\kappa^2}\left(\frac{n}{\kappa r} + \frac{I_{n+1}(\kappa r)}{I_n(\kappa r)}\right)\right]\left[\alpha R J_{n+1}(\alpha R) - \kappa R J_n(\alpha R)\frac{K_{n+1}(\kappa R)}{K_n(\kappa R)}\right]I_n(\kappa r)K_n(\kappa R) \\
&+\frac{1}{(\alpha^2+\kappa^2)^2}J_n(\alpha r),
\end{aligned} \tag{5.4}$$

using ratios and products of modified Bessel functions already derived in §3.1. The solution procedure is identical to that of §3.4 except that Equation 3.1 is replaced with Equation 5.4. The method was tested using the function of Equation 4.1: computational time was similar to that for the Poisson equation, as might be expected; errors are reported in Tables 9–12. As for the Poisson equation, the error is comparable to machine precision, except as β increases and aliasing becomes apparent, Table 12.

6. Conclusions

A method for the solution of a modified Bessel equation which arises in the solution of the Poisson and biharmonic equations in cylindrical geometries has been presented, based on the Hankel transform. Numerical testing has shown the method to be accurate over a wide range of wave numbers and orders. We conclude that for a proper choice of mesh densities and Hankel transform order, the method can achieve machine precision accuracy, for oscillatory and for decaying solutions. A sample implementation of the algorithm, written in C, is available from the author.

Acknowledgements

I am grateful to Andras Pataki, Courant Institute, New York University, for helpful discussions on the implementation of his Green's function Poisson solver.

TABLE 9. *Error in Hankel transform solution of biharmonic equation,* $\beta = 0$

κ / n	16	64	256
16	2.1×10^{-14}	2.1×10^{-14}	2.1×10^{-14}
32	8.3×10^{-15}	1.5×10^{-14}	1.4×10^{-14}
64	3.8×10^{-14}	5.5×10^{-14}	5.5×10^{-14}
128	1.7×10^{-13}	1.9×10^{-13}	2.0×10^{-13}

TABLE 10. *Error in Hankel transform solution of biharmonic equation,* $\beta = 16$

κ / n	16	64	256
16	5.6×10^{-14}	5.0×10^{-14}	5.6×10^{-14}
32	5.7×10^{-14}	5.0×10^{-14}	7.5×10^{-14}
64	5.6×10^{-14}	4.1×10^{-14}	5.7×10^{-14}
128	3.0×10^{-13}	2.5×10^{-13}	2.5×10^{-13}

TABLE 11. *Error in Hankel transform solution of biharmonic equation,* $\beta = 32$

κ / n	16	64	256
16	5.2×10^{-09}	1.1×10^{-09}	1.5×10^{-09}
32	5.6×10^{-09}	1.0×10^{-09}	1.6×10^{-09}
64	5.1×10^{-09}	1.1×10^{-09}	1.3×10^{-09}
128	2.1×10^{-09}	8.2×10^{-10}	1.2×10^{-09}

TABLE 12. *Error in Hankel transform solution of biharmonic equation,* $\beta = 64$

κ / n	16	64	256
16	2.9×10^{-03}	1.4×10^{-04}	8.0×10^{-05}
32	1.7×10^{-03}	9.3×10^{-05}	5.0×10^{-05}
64	2.6×10^{-03}	1.6×10^{-04}	7.4×10^{-05}
128	8.0×10^{-04}	1.1×10^{-04}	7.5×10^{-05}

References

AMOS, D. E. (1974) Computation of modified Bessel functions and their ratios. *Mathematics of Computation*, **28**, 239–251.

BERRUT, J.-P. & TREFETHEN, L. N. (2004) Barycentric Lagrange interpolation. *SIAM Review*, **46**, 501–517.

CHEN, H., SU, Y. & SHIZGAL, B. D. (2000) A direct spectral collocation Poisson solver in polar and cylindrical coordinates. *Journal of Computational Physics*, **160**, 453–469.

FREUND, J. B. (2001) Noise sources in a low-Reynolds-number turbulent jet at Mach 0.9. *Journal of Fluid Mechanics*, **438**, 277–305.

GALASSI, M., DAVIES, J., THEILER, J., GOUGH, B., JUNGMAN, G., BOOTH, M. & ROSSI, F. (2005) *GNU Scientific Library Reference Manual*. Bristol, United Kingdom: Network Theory Ltd.

GENOVESE, L., DEUTSCH, T. & GOEDECKER, S. (2007) Efficient and accurate three-dimensional Poisson solver for surface problems. *Journal of Chemical Physics*, **127**, 054704.

GLASER, A., LIU, X. & ROKHLIN, V. (2007) A fast algorithm for the calculation of the roots of special functions. *SIAM Journal on Scientific Computing*, **29**, 1420–1438.

GRADSHTEYN, I. & RYZHIK, I. M. (1980) *Table of integrals, series and products*, 5th edn. London: Academic.

LEMOINE, D. (1994) The discrete Bessel transform. *Journal of Chemical Physics*, **101**, 3936–3944.

LI, S., BUONI, M. J. & LI, H. (2009) A fast potential and self-gravity solver for nonaxisymmetric disks. *The Astrophysical Journal Supplement Series*, **181**, 244.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (2010) Digital library of mathematical functions. `http://dlmf.nist.gov/`.

PATAKI, A. (2011). personal communication.

PATAKI, A. & GREENGARD, L. (2011) Fast elliptic solvers in cylindrical coordinates and the Coulomb collision operator. *Journal of Computational Physics*, **230**, 7840–7852.

PRUDNIKOV, A. P., BRYCHKOV, Y. A. & MARICHEV, O. I. (2003) *Integraly i ryady: Tom 2 Spetsial'nye funktsii (Integrals and series: Volume 2 Special functions)*, 2nd edn. Moscow: Fizmatlit.

SHISHKINA, O., SHISHKIN, A. & WAGNER, C. (2009) Simulation of turbulent thermal convection in complicated domains. *Journal of Computational and Applied Mathematics*, **226**, 336–344.