An Algorithm for the fast Hankel transform

Sharad Kapur and Vladimir Rokhlin

YALEU/DCS/TR-1045

August 1995

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

An algorithm for the fast Hankel transform[*]

Sharad Kapur[†]
Vladimir Rokhlin[‡]

August 2, 1995

Abstract

In this paper, we construct an algorithm for the rapid evaluation of the Hankel transform. The algorithm decomposes the Hankel transform into a product of two integral operators, the first of which is evaluated rapidly via the fast cosine transform. The second operator has a singular kernel. It is evaluated rapidly by a combination of a version of the fast multipole method (FMM) with a quadrature formula designed to account for the presence of the singularity. The asymptotic time complexity of the algorithm is $O(N \log N)$. In practice, (for a function tabulated at the Nyquist sampling rate) the cost of the algorithm is roughly three times that of the Fast Fourier transform of length N, provided that the calculations are performed to single precison accuracy. In double precision, the ratio is approximately 5. Numerical experiments are presented demonstrating the speed and accuracy of the scheme.

1 Introduction

Hankel transforms are frequently encountered in applied mathematics, engineering and computational physics. Their applications include vibrations of a circular membrane, flow of heat in a circular cylinder, wave propagation in a three-dimensional medium and many others. However, attempts to use Hankel transforms as a numerical tool (as opposed to analytical apparatus) tend to meet with a serious difficulty: given a function

[*]This research, performed at Yale University, was supported in part by the Office of Naval research under grant N00014-89-J-1527

[†]Department of Computer Science, Yale University, Box 20825, New Haven, Connecticut 06520-8285 (kapur-sharad@cs.yale.edu, http://www.cs.yale.edu/HTML/YALE/CS/HyPlans/kapur-sharad.html).

[‡]Departments of Mathematics and Computer Science, Yale University, Box 20825, New Haven, Connecticut 06520-8285 (rokhlin@cs.yale.edu).

$f : [0, \infty] \to \mathbf{R}$, tabulated at N nodes, it takes $O(N^2)$ operations to obtain the numerical Hankel transform

$$\int_0^\infty x f(x) J_\nu(a \cdot x) dx, \tag{1}$$

for N values of a.

Remark 1.1 In recent years, a variety of numerical schemes have been developed for the rapid evaluation of the Hankel transform based on the observation that the transformation

$$x = e^{-u}, \tag{2}$$

and

$$a = e^{v}, \tag{3}$$

converts (1) into a convolution (see, [1], [2], [12], [17]). There are two disadvantages to this approach: this procedure requires samples on an exponential grid and produces output on a similar grid. A more serious problem is the fact that the input function needs to be tabulated at least 7 times that of the Nyquist sampling rate in order to obtain single precision accuracy (also see, [2] for more precise estimates on the degree of oversamping required).

In this paper, we develop a procedure for the rapid evaluation of integrals of the form (1), to any degree of precision, requiring CPU time proportional to $N \log N$. More specifically, suppose that $h = \frac{A}{N-1}$, $x_i = ih$, $a_j = \frac{\pi j}{(N-1)h}$, and $f : [0, A] \to \mathbf{R}$ is a function tabulated at N equispaced nodes $x_0, x_1, ..., x_{N-1}$. Then the integrals

$$g(a_j) = \int_0^A x f(x) J_0(a_j x) dx, \tag{4}$$

are computed for all $j = 0, 1, 2..., N-1$, in $O(N \log N)$ operations.

Our algorithm for the Hankel transform is based on several well known facts from classical analysis. The algorithm decomposes the Hankel transform into a product of two integral operators, the first of which is evaluated rapidly by a combination of the fast cosine transform with quadrature formula of the type developed in [10]. The second operator is evaluated rapidly by a combination of a version of the fast multipole method with yet another quadrature formula derived in [10]. All calculations are performed to full double precision accuracy.

In the following section, we summarize several facts from approximation theory and numerical analysis to be used in the subsequent sections. Section 3 provides the numerical apparatus for the algorithm, and is divided into three subsections. In §3.1 a procedure for the accurate evaluation of a cosine transform of a non-periodic function is described. In §3.2 we design a very high-order end-point corrected trapezoidal rule for integrals of the form (4). In §3.3 we present an algorithm based on a version of the fast multipole method (FMM) which computes a trapezoidal approximation to the integrals (4) in $O(N)$

operations. In §4 we combine the results of Sections 3.1, 3.2, and 3.3 to construct a fast algorithm for the Hankel transform. Finally, §5 contains the results of several numerical experiments, and in §6 we discuss several straightforward generalizations and applications of the algorithm of this paper.

2 Mathematical and Numerical Preliminaries

In this section, we summarize several well known results from classical analysis and approximation theory to be used in this paper.

2.1 An expression for the Bessel function J_0

As is well known, for any $z \in \mathbf{C}$,

$$J_0(z) = \frac{1}{\pi}\int_0^{\pi} \cos(z\cos\theta)d\theta, \tag{5}$$

where J_0 denotes the Bessel function of order 0 (see, for example [3]). It follows immediately from (5) that, for any function $f \in c^2[0, A] \to \mathbf{R}$, and any real number a,

$$\int_0^A f(x)J_0(ax)dx = \frac{1}{\pi}\int_{-a}^{a} \frac{1}{\sqrt{a^2-u^2}}\int_0^A f(x)cos(ux)dxdu. \tag{6}$$

2.2 Chebyshev Polynomials

In this section, we summarize several well known facts about Chebyshev polynomials. The following three classical definitions can be found, for example, in [9].

Definition 2.1 *The n-th degree Chebyshev polynomial $T_n(x)$ is defined by the following equivalent formulae:*

$$T_n(x) = \cos(n \arccos x), \tag{7}$$

$$T_n(x) = \frac{1}{2}\cdot\left((x+\sqrt{x^2-1})^n + (x-\sqrt{x^2-1})^n\right). \tag{8}$$

The proof of the following lemma can be found in, for example, in[3].

Lemma 2.1 *Suppose that $n \geq 0$ is an integer. Then*

$$\int_{-1}^{1} \frac{T_0(u)du}{\sqrt{1-u^2}} = \pi, \tag{9}$$

and,

$$\int_{-1}^{1} \frac{T_n(u)du}{\sqrt{1-u^2}} = 0, \tag{10}$$

for all $n \geq 1$.

Definition 2.2 *The roots $\{t_1,\ldots,t_n\}$ of the n-th degree Chebyshev polynomial T_n lie in the interval $[-1,1]$ and*

$$t_k = -\cos\left(\frac{2k-1}{n}\cdot\frac{\pi}{2}\right), \tag{11}$$

for $k=1,\ldots,n$. They are referred to as Chebyshev nodes of order n.

Definition 2.3 *Given an integer $n \geq 1$, we will denote by $u_1,\ldots,u_n$ the set of polynomials of order $n-1$ defined by the formulae*

$$u_j(t) = \prod_{k\neq j}^{n} \frac{t-t_k}{t_j-t_k}, \tag{12}$$

for $j=1,\ldots,n$, where t_k are defined by (11).

The proofs of the following two lemmas are well known from classical theory of Chebyshev approximation, and can be found, for example, in [6], [5].

Lemma 2.2 *Suppose that $p \geq 2$, $b > 0$, and x_0 are real numbers such that $|x_0| < b$. Suppose further that $\{t_1,\ldots,t_p\}$ are Chebyshev nodes on the interval $[x_0-b, x_0+b]$. Then,*

$$\left| \frac{1}{\sqrt{y^2-x_0^2}} - \sum_{m=1}^{p} \frac{1}{\sqrt{y^2-t_m^2}}\cdot u_m(x_0) \right| < O\left(\frac{1}{5^p}\right) \tag{13}$$

for all $|y| > 3b$.

Lemma 2.3 *Suppose that $p \geq 2$, $b > 0$, and y_0 are real numbers such that $|y_0| > 3b$. Suppose further that $\{t_1,\ldots,t_p\}$ are Chebyshev nodes on the interval $[y_0-b, y_0+b]$. Then,*

$$\left| \frac{1}{\sqrt{y_0^2-x^2}} - \sum_{m=1}^{p} \frac{1}{\sqrt{t_m^2-x^2}}\cdot u_m(y_0) \right| < O\left(\frac{1}{5^p}\right) \tag{14}$$

for all $|x| < b$.

2.3 High-order corrected trapezoidal rules for non-singular functions

In this section we summarize several results obtained by the authors in [10].

Definition 2.4 *Suppose that a, b are a pair of real numbers such that $a < b$, and that $n \geq 2$ is an integer. For a function $f : [a, b] \to \mathbf{R}$, the n-point trapezoidal rule $T_R^n(f)$ is defined by the formula,*

$$T_R^n(f) = h\left(\sum_{i=0}^{n-1} f(a+ih) - \left(\frac{f(a)+f(b)}{2}\right)\right), \tag{15}$$

with

$$h = (b-a)/(n-1). \tag{16}$$

Definition 2.5 *Suppose that m, k, i are integers, with $m \geq 3$ and odd. Then we define the real coefficients $D_{i,k}^m$ by the formula*

$$D_{i,k}^m = \frac{(-1)^{\frac{m-1}{2}+k}}{(\frac{m-1}{2}+k)!(\frac{m-1}{2}-k)!} a_{k,2i-1}^m (2i-1)!, \tag{17}$$

for any k, i such that $1 \leq k \leq \frac{m-1}{2}$, and $1 \leq i \leq \frac{m-1}{2}$, with the coefficients $a_{k,l}^m$ given by the recurrence relation

$$\begin{aligned}
a_{1,1}^3 &= 1, \\
a_{1,2}^3 &= 1, \\
a_{k,l}^{2k+1} &= (k-k^2)a_{k-1,l}^{2k-1} + a_{k-1,l-1}^{2k-1} + a_{k-1,l-2}^{2k-1}, \\
a_{k,l}^{m+2} &= a_{k,l-2}^m - \left(\frac{m+1}{2}\right)^2 a_{k,l}^m,
\end{aligned}$$

with $a_{k,l}^m = 0$, for all $k \leq 0$, or $l \leq 0$, or $m \leq 1$.

Definition 2.6 *Suppose that n, m, are a pair of integers with $m \geq 3$ and odd, and $n \geq 2$. Suppose further that a, b are a pair of real numbers such that $a < b$, $h = (b-a)/(n-1)$, and $f : [a-mh, b+mh] \to \mathbf{R}$ is an integrable function, and T_R^n is defined in (15). We define the corrected trapezoidal rule $T_{\beta^m}^n$ for non-singular functions by the formula*

$$\begin{aligned}
T_{\beta^m}^n(f) &= T_R^n(f) \\
&+ h\sum_{k=1}^{\frac{m-1}{2}} (-f(b+kh) + f(b-kh) + f(a+kh) - f(a-kh))\beta_k^m.
\end{aligned} \tag{18}$$

The real coefficients β_k^m are given by the formula

$$\beta_k^m = \sum_{l=1}^{\frac{m-1}{2}} \frac{D_{l,k}^m B_{2l}}{(2l)!}, \tag{19}$$

where B_{2l} are the Bernoulli numbers, and $D_{l,k}^m$ is defined in (17).

The proof of the following two theorems can be found in [10].

Theorem 2.4 *If $m \geq 3$ is an odd integer then for any k such that $-\frac{m-1}{2} \leq k \leq \frac{m-1}{2}$,*

$$| \beta_k^m | < 1, \tag{20}$$

where the coefficients β_k^m are defined in (19).

Theorem 2.5 *Suppose that m, n are a pair of integers with $m \geq 3$ and odd, and $n \geq 2$. Suppose further that a, b are a pair of real numbers such that $a < b$. Then, the end-point corrected trapezoidal rule $T_{\beta m}^n$ is of order m, i.e., for any $f \in c^m[a - mh, b + mh]$ there exists a real number $c > 0$ such that*

$$| T_{\beta m}^n(f) - \int_a^b f(x)dx | < \frac{c}{n^m}. \tag{21}$$

The following lemma provides an error estimate for the approximation to the integral given by the trapezoidal rule. It can be found, for example in [3].

Lemma 2.6 *(Euler-Maclaurin formula) Suppose that a, b are a pair of real numbers such that $a < b$, and that $m \geq 1$ is an integer. Further, let B_k denote the Bernoulli numbers*

$$B_2 = \frac{1}{6}, B_4 = \frac{-1}{30}, B_6 = \frac{1}{42}, ...,. \tag{22}$$

If $f \in c^{2m+2}[a, b]$ (i.e., f has $2m + 2$ continuous derivatives on $[a, b]$), then there exists a real number ξ, with $a < \xi < b$, such that

$$\int_a^b f(x)dx = T_R^n(f) + \sum_{l=1}^{m} \frac{h^{2l} B_{2l}}{(2l)!}(f^{(2l-1)}(b) - f^{(2l-1)}(a)) - \frac{h^{2m+2} B_{2m+2}}{(2m+2)!} f^{2m+2}(\xi). \tag{23}$$

2.4 The fast cosine transform

The following definition of the discrete cosine transform, can be found for example, in [19].

Definition 2.7 *For a real sequence $\{f_0, \ldots, f_{N-1}\}$, the discrete cosine transform $\{F_j^T\}$ is defined by the formula*

$$F_j^T = \frac{\pi}{N-1}\left(\sum_{i=1}^{N-2} f_i \cos\left(\frac{\pi \cdot j \cdot i}{N-1}\right) + \frac{f_0 + (-1)^j f_{N-1}}{2}\right) \tag{24}$$

for all $j = 0, 1, ..., N-1$.

Remark 2.1 The fast cosine transform is an algorithm, based on the FFT (see, for example [19]), to evaluate the discrete cosine transform (DCT) in $O(N \log N)$ operations.

Remark 2.2 The DCT is a trapezoidal approximation (see Definition 2.4) to the exact cosine transform. More specifically, suppose that $f \in c^2[0,\pi]$ and $\{F_j^T\}$ is defined in (24). Then

$$F_j^T \approx \int_0^\pi f(x)\cos(j \cdot x)dx, \tag{25}$$

for all $j = 0, 1, ..., N-1$.

The following well known theorem provides an error estimate for the approximation to the exact cosine transform given by the discrete cosine transform. The proof follows immediately from the combination of (24) and Lemma 2.6.

Theorem 2.7 *Suppose that $f \in c^2[0,\pi]$. Suppose further that $h = \pi/(N-1)$, $x_i = ih$, and $f_i = f(x_i)$. Then the discrete cosine transform is second order convergent, i.e., there exists some real $c > 0$ such that*

$$\mid F_j^T - \int_0^\pi f(x)\cos(j \cdot x) \mid < \frac{c}{N^2} \tag{26}$$

for all $j = 0, 1, ..., N-1$.

The following theorem is less widely known. The proof also follows immediately from the combination of (24) and Lemma 2.6; it can also be found in [5].

Theorem 2.8 *Suppose that $f \in c^m[0,\pi]$ is an even function (i.e., f(x) = f(-x)). Suppose further that $h = \pi/(N-1)$, $x_i = ih$, and $f_i = f(x_i)$. Finally, suppose that $f(\pi) = f'(\pi) = f''(\pi) = ... = f^m(\pi) = 0$. Then the discrete cosine transform is a rule of order m, i.e., there exists some real $c > 0$ such that*

$$\mid F_j^T - \int_0^\pi f(x)\cos(j \cdot x) \mid < \frac{c}{N^m} \tag{27}$$

for all $j = 0, 1, ..., N-1$.

3 Numerical apparatus

3.1 The corrected fast cosine transform

Remark 3.1 When a function is even, the discrete cosine transform provides a remarkably good approximation to the exact cosine transform (see Theorem 2.8). For functions that are not even, we use end-point corrections to accelerate the convergence of the DCT.

Definition 3.1 *For a finite real sequence $\{f_{-(n-1)}, \ldots, f_0, \ldots, f_{n-1}\}$ we define the corrected discrete cosine transform $\{F_j^C\}$ by the formula*

$$F_j^C = F_j^T + \frac{\pi}{N-1}\sum_{i=1}^{N-2}(-f_{-i} + f_i)\cos\left(\frac{\pi \cdot i \cdot j}{N-1}\right)\beta_i^{2N-3}, \tag{28}$$

for all $j = 0, 1, ..., N-1$, where $\{F_j^T\}$ is defined in (24), and $\{\beta_i^{2N-3}\}$ are the correction coefficients defined in (19).

The following corollary provides an error estimate for the approximation to the exact cosine transform given by the corrected discrete cosine transform (CDCT). It is an immediate consequence of Theorem 2.5.

Corollary 3.1 *Suppose that $f \in c^m[-\pi, \pi]$. Suppose further that $h = \pi/(N-1)$, $x_i = ih$, and $f_i = f(x_i)$ for all $i = 0, \pm 1, \ldots, \pm N-1$. Finally, suppose that $f(\pi) = f'(\pi) = f''(\pi) = ... = f^m(\pi) = 0$. Then there exists some real $c > 0$ such that*

$$\mid F_j^C - \int_0^\pi f(x)\cos(j \cdot x) \mid < \frac{c}{N^m} \tag{29}$$

for all $j = 0, 1, ..., N-1$.

Remark 3.2 The CDCT requires that the function be tabulated outside the interval of integration $[0 : \pi]$. However, if a function is odd (or even) this requirement is obviated, i.e., the function needs to be tabulated only within the interval of integration.

Observation 3.3 *Suppose that $f \in c^m[0, \pi]$ is an even function (i.e., $f(x) = f(-x)$) satisfying the conditions of Corrollary 3.1. Then it follows immediately from (28) that*

$$F_j^C = F_j^T, \tag{30}$$

for all $j = 0, 1, ..., N-1$.

Observation 3.4 *Suppose that $f \in c^m[0, \pi]$ is an odd function (i.e., $f(x) = -f(-x)$) satisfying the conditions of Corrollary 3.1. Then it follows immediately from (28) that*

$$F_j^C = F_j^T - \frac{\pi}{N-1}\sum_{i=1}^{N-2}(2 \cdot f_i \cdot \beta_i^{2N-3})\cos\left(\frac{\pi \cdot i \cdot j}{N-1}\right) \tag{31}$$

for all $j = 0, 1, ..., N-1$.

3.1.1 Rapid evaluation of the corrected discrete cosine transform (CDCT)

Suppose that $\{f_{-(n-1)}, \ldots, f_0, \ldots, f_{n-1}\}$ is a finite real sequence. Suppose further that we define the real sequence $\{\hat{f}_0, \ldots, \hat{f}_{n-1}\}$ by the formulae

$$\hat{f}_0 = f_0, \quad \hat{f}_{N-1} = f_{N-1}, \tag{32}$$

and,

$$\hat{f}_i = f_i + \beta_i^{2N-3}(-f_{-i} + f_i), \tag{33}$$

for all $i = 1, 2, ..., N-2$, where the real coefficients β_k^{2N-3} are defined in (19). Then it follows immediately from the combination of (28) and (32) that

$$F_j^C = \frac{\pi}{N-1}\left(\sum_{i=1}^{N-2} \hat{f}_i \cos\left(\frac{\pi \cdot i \cdot j}{N-1}\right) + \frac{\hat{f}_0 + (-1)^j \hat{f}_{N-1}}{2}\right) \tag{34}$$

for all $j = 0, 1, 2, ..., N-1$.

Remark 3.5 Given a real sequence $\{f_{-(n-1)}, \ldots, f_0, \ldots, f_{n-1}\}$, the sequence $\{\hat{f}_0, \ldots, \hat{f}_{n-1}\}$ can be computed (using (32), (33)) in $O(N)$ operations. Subsequently sums of the form (34) can be computed in $O(N \log N)$ operations using the FCT (see Remark 2.1). Thus the corrected discrete cosine transform $\{F_j^C\}$ can be computed in $O(N \log N) + O(N)$ operations.

3.2 End-point corrected trapezoidal quadrature rules for singular functions of the form $\frac{F(u)}{(a^2-u^2)^{1/2}}$

In this section we develop an end-point corrected quadrature formula to approximate the definite integral

$$\int_{-a}^{a} \frac{F(u)}{\sqrt{a^2-u^2}} du, \tag{35}$$

with $a > 0$, and $F \in c^k[-a, a]$, an even function (i.e., $F(-u) = F(u)$).

We define the corrected trapezoidal rule $T_{\nu^N}^N$ by the formula

$$T_{\nu^N}^N\left(\frac{F(u)}{\sqrt{a^2-u^2}}\right) = h \sum_{l=-(N-2)}^{N-2} \frac{F(x_l)}{\sqrt{a^2-x_l^2}} + h \sum_{i=1}^{k} \nu_i^N \frac{F(y_i)}{\sqrt{|a^2-(y_i)^2|}}, \tag{36}$$

where $h = a/(N-1)$, $x_l = lh$, $y_i = a - hi$ for all $1 \le i \le k/2$, and $y_i = a + h(i - k/2)$ for all $k/2 + 1 \le i \le k$. We will use the expression $T_{\nu^N}^N$ with appropriately chosen ν_i^N as a quadrature formulae to approximate integrals of the form (35), and the following construction provides a tool for finding ν_i^N, so that the rule is of order $2k-2$, i.e., there exists a real $c > 0$ such that

$$\left| T_{\nu^N}^N\left(\frac{F(u)}{\sqrt{a^2-u^2}}\right) - \int_{-a}^{a} \frac{F(u)}{\sqrt{a^2-u^2}} du \right| < \frac{c}{N^{2k-2}}. \tag{37}$$

Remark 3.6 The correction nodes $\{y_1, \ldots, y_k\}$ are equispaced nodes on both sides of a, the location of of the singularity (i.e., half the correction nodes lie outside the interval of integration). The approach we use in this paper is similar to that of [10]; where a class of quadrature formulae is presented applicable to functions with end-point singularities, taking advantage of functional information outside the interval of integration.

3.2.1 Construction of the quadrature weights ν^N

For any pair of positive integers k, N $(k < N)$, we will consider the following system of linear algebraic equations with respect to the unknowns $\{\nu_1^N, \ldots, \nu_k^N\}$:

$$\sum_{p=1}^{k} \frac{T_{2i-2}(y_p)}{\sqrt{|a^2 - (y_p)^2|}} \nu_p^N = \frac{1}{h}\Big(\int_{-a}^{a} \frac{T_{2i-2}(u)}{\sqrt{a^2 - u^2}} du - h \sum_{l=-(N-2)}^{N-2} \frac{T_{2i-2}(x_l)}{\sqrt{a^2 - x_l^2}}\Big), \tag{38}$$

for all $i = 1, 2, \ldots, k$, where h, x_l, and y_i are defined in (36). In (38) T_{2i-2} is the Chebyshev polynomial defined in (7).

The following observation is used in the development of our algorithm for the fast Hankel transform.

Observation 3.7 *The linear system (38) is independent of the length of the interval a i.e., the quadrature weights $\nu_1^N, \nu_2^N, \ldots, \nu_k^N$ are only dependent on the number of points N used in the trapezoidal approximation to the integral (35).*

Thus, by substituting $a = 1$ in (36), the unknowns $\{\nu_1^N, \nu_2^N, \ldots, \nu_k^N\}$ alternatively can be determined by solving the system of equations:

$$\sum_{p=1}^{k} \frac{T_{2i-2}(y_p)}{\sqrt{|1 - y_p^2|}} \nu_p^N = \frac{1}{h}\Big(\int_{-1}^{1} \frac{T_{2i-2}(u)}{\sqrt{1 - u^2}} du - h \sum_{l=-(N-2)}^{N-2} \frac{T_{2i-2}(x_l)}{\sqrt{1 - x_l^2}}\Big), \tag{39}$$

for all $i = 1, 2, \ldots, k$, where $h = \frac{1}{N-1}$, $x_l = lh$, $y_p = 1 - hp$ for all $1 \le p \le k/2$, and $y_p = 1 + h(p - k/2)$ for all $k/2 + 1 \le p \le k$.

Remark 3.8 Any polynomial basis can be chosen to construct the linear system (39) above. Our choice of the Chebyshev polynomials T_i as the polynomial basis is simply for the reason that the right-hand side of of the linear system (39) can be simplified due to Lemma 2.1.

Hence, we may alternatively solve the following system of equations to obtain the quadrature weights $\nu_1^N, \nu_2^N, \ldots, \nu_k^N$:

$$\sum_{p=1}^{k} \frac{T_{2i-2}(y_p)}{\sqrt{|1 - y_p^2|}} \nu_p^N = \frac{1}{h}\Big(\pi - h \sum_{l=-(N-2)}^{N-2} \frac{T_{2i-2}(x_l)}{\sqrt{1 - x_l^2}}\Big), \tag{40}$$

for $i = 1$, and

$$\sum_{p=1}^{k} \frac{T_{2i-2}(y_p)}{\sqrt{|1-y_p^2|}} \nu_p^N = \frac{1}{h}(-h \sum_{l=-(N-2)}^{N-2} \frac{T_{2i-2}(u_l)}{\sqrt{1-x_l^2}}), \tag{41}$$

for $i = 2, 3, ..., k$.

3.2.2 Convergence of the rule $T_{\nu^N}^N$

The use of expressions $T_{\nu^N}^N$ as quadrature formulae to approximate integrals of the form (35) is based on the following theorem. The proof of the theorem can be found in [10].

Theorem 3.2 *Suppose that $k, N > 2$ are a pair of positive integers. Further, suppose that h, a are positive real numbers with $h = a/(N-1)$. Also, suppose that the systems (40), (41) have solutions $(\nu_1^N, \nu_2^N, ..., \nu_k^N)$ for all N. Finally, suppose that $F \in c^k[-a-kh, a+kh]$ is an even function. Then the rule $T_{\nu^N}^N$ is of order $2k-2$, i.e., there exists some real number c such that*

$$| T_{\nu^N}^N(\frac{F(u)}{\sqrt{a^2-u^2}}) - \int_{-a}^{a} \frac{F(u)}{\sqrt{a^2-u^2}} du |< \frac{c}{N^{2k-2}} \tag{42}$$

The following theorem easily follows from the combination of (28), Corollary 3.2, and Corollary 3.1.

Theorem 3.3 *Suppose that $f \in c^{2k}[-\pi, \pi]$ is an odd or even function. Suppose further f_j and $\hat{f}_j$ are defined in (32) for all $j = 0, 1, \ldots, N-1$. Finally, suppose that*

$$F_i^C = \frac{\pi}{N-1}(\sum_{j=0}^{N-1} \hat{f}_j \cos(\frac{\pi \cdot i \cdot j}{N-1}), \tag{43}$$

Then there exist real numbers c_j such that

$$| T_{\nu^j}^j(\frac{F^C(u)}{\sqrt{j^2-u^2}}) - \int_0^{\pi} f(x) J_0(j \cdot x) dx |< \frac{c_j}{j^{2k-2}}, \tag{44}$$

for all $j = 0, \pm 1, \pm 2, \pm 3, ..., \pm N-1$.

3.2.3 Approximation of the Hankel tranform using the rule $T_{\nu_j}^j$

Suppose that we define the trapezoidal approximation g^T to the Hankel transform by the formula

$$g_j^T = h \sum_{l=-(j-1)}^{j-1} \frac{F_l^C}{\sqrt{j^2-l^2}}, \tag{45}$$

and the correction g^C by the formula

$$g_j^C = h \sum_{i=1}^{k} \nu_i^j \frac{F_{j-i}^C}{\sqrt{j^2 - (j-i)^2}}, \tag{46}$$

for all $j = \pm 1, \pm 2, ..., \pm N-1$, for $1 \le i \le k/2$, and $y_i^j = j + (i - k/2)$ for $k/2+1 \le i \le k$. It follows immediately from the combination of (45), (46), and (36) that

$$|\, (g_j^C + g_j^T) - \int_0^\pi f(x) J_0(j \cdot x) dx \,| < \frac{c_j}{j^{2k-2}}, \tag{47}$$

for all $j = 0, \pm 1, \pm 2, \pm 3, ..., \pm N-1$.

Remark 3.9 It is obvious that g_j^T can be computed for all $j = 0, \pm 1 \pm 2, ..., \pm N-1$,in $O(N^2)$ operations, and g_j^C can be computed in $O(kN)$ operations. In the following section we discuss how to compute sums of the form (45) in $O(N)$ operations.

Remark 3.10 The numerical stability of the scheme developed above, for the approximation of integrals of the form (35), is dependent on the assumption that the size of the quadrature weights $(\nu_1^j, \nu_2^j, ..., \nu_k^j)$ is small. We have observed in [10] that the size of the quadrature weights can be suppressed (for both singular and non-singular functions) by using functional information outside the interval of integration. It is observed empirically that the quadrature weights $(\nu_1^j, \nu_2^j, ..., \nu_k^j)$ are always of $O(1)$.

Remark 3.11 The authors have been unable to construct a quadrature rule which is independent of the number j of points used in the uncorrected trapezoidal rule. However, this is a minor deficiency since the weights in such cases can be precomputed and stored.

3.3 A fast multipole method in one-dimension for sums of the form $\sum_{k=1}^{j-1} \frac{\alpha_k}{(y_j^2 - x_k^2)^{1/2}}$

In this section we consider the problem of computing the sums

$$f_j = \sum_{k=1}^{j-1} \frac{\alpha_k}{\sqrt{y_j^2 - x_k^2}} \tag{48}$$

for $j = 1, \ldots, N$, where $\{x_1, \ldots, x_N\}$ and $\{y_1, \ldots, y_N\}$, $\{\alpha_1, \ldots, \alpha_N\}$, and $\{f_1, \ldots, f_N\}$ are sets of real numbers.

Remark 3.12 For the remainder of this section, we shall assume without loss of generality that $x_i, y_i \in [-1, 1]$ for $i = 1, \ldots, N$.

Remark 3.13 Sums of the form (48) are a simple reformulation of the trapezoidal approximation to the Hankel transform, defined in (45).

The fast multipole algorithm of [9] computes sums of a slightly different form than (48) in $O(N)$ arithmetic operations, described described by the formulae

$$f_j = \sum_{k=1}^{N} \frac{\alpha_k}{w_j - z_k} \tag{49}$$

for $j = 1, \ldots, N$, where $\{z_1, \ldots, z_N\}$ and $\{w_1, \ldots, w_N\}$ are sets of complex numbers. From a physical viewpoint, this corresponds to the evaluation of the electrostatic field due to N charges which lie in the plane. The two and three dimensional scenarios for the N-body problem have been discussed in some depth (see, for example, [9]). In recent years the the analysis and applications of one dimensional problems have been invesigated in [6], [7], [4]. In [6] the sums

$$f_j = \sum_{k=1}^{N} \frac{\alpha_k}{y_j - x_k} \tag{50}$$

for $j = 1, 2, \ldots, N$, are computed in $O(N)$ using a combination of chebyshev approximation techniques, singular value decompositon (SVD) based compression. In [4], an algorithm is constructed to evalauate the function f

$$f(t) = \sum_{j=0}^{n-1} \alpha_j \cdot P_j(t), \tag{51}$$

at the nodes $t_0, t_1, \ldots, t_{n-1}$, in expending CPU time proportional to $O(N)$ operations. In fact, an early implementation of the FBT used a version of the algorithm developed in [4].

In this section we briefly describe an $O(N)$ algorithm for the computation of (48) which is based on the one-dimensional FMM of [6], and [4]. We assume that the reader is familiar with [9].

3.3.1 General strategy

We will illustrate by means of a simple example how Chebyshev expansions can be used to evaluate expressions of the form (48) more efficiently. We will also give an informal description of how the method of this simple example is used in the construction of a fast algorithm for the general case.

First we introduce a definition which formalizes the notion of well-separated intervals on the real line. This is simply the one-dimensional analog of the definition of well-separatedness in [9].

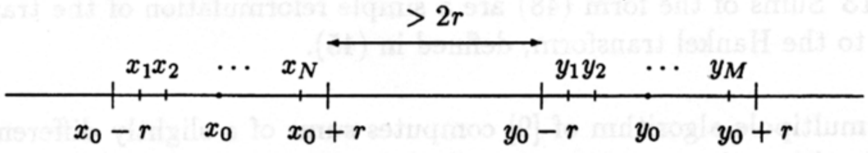

Figure 1: Well-separated intervals on the line.

Definition 3.2 *Let $\{x_1, \ldots, x_N\}$ and $\{y_1, \ldots, y_M\}$ be two sets of points in $\mathbf{R}$. We say that the sets $\{x_i\}$ and $\{y_i\}$ are* well-separated *if there exist points $x_0, y_0 \in \mathbf{R}$ and a real $r > 0$ such that*

$$\begin{aligned} |x_i - x_0| &< r \quad \forall \;\; i = 1, \ldots, N, \\ |y_i - y_0| &< r \quad \forall \;\; i = 1, \ldots, M, \;\; \textit{and} \\ |x_0 - y_0| &> 4r. \end{aligned} \tag{52}$$

Suppose now that $\{x_1, \ldots, x_N\}$ and $\{y_1, \ldots, y_M\}$ are well-separated sets of points in $\mathbf{R}$ (see Figure 1), that $\{\alpha_1, \ldots, \alpha_N\}$ is a set of complex numbers, and that we wish to compute the numbers $f(y_1), \ldots, f(y_M)$ where the function $f : \mathbf{R} \to \mathbf{C}$ is defined by the formula

$$f(x) = \sum_{k=1}^{N} \frac{\alpha_k}{\sqrt{x^2 - x_k^2}}. \tag{53}$$

A direct evaluation of (53) at the points $\{y_1, \ldots, y_M\}$ requires $O(NM)$ arithmetic operations. We will describe two different ways of speeding up this calculation based on the following two observations. The observations follow from a combination of Lemma 2.2, Lemma 2.3, and the triangle inequality.

Observation 3.14 *Suppose that $p > 2$ is an integer. Further, suppose that we define the real coefficients Φ_m (representing the far field) by the formula*

$$\Phi_m = \sum_{k=1}^{N} \alpha_k \cdot u_m(x_k), \tag{54}$$

for all $m = 1, 2, ..., p$, where u_m is defined in (12). Finally, suppose that

$$\tilde{f}_1(y_j) = \sum_{m=1}^{p} \frac{1}{\sqrt{y_j^2 - t_m^2}} \Phi_m \tag{55}$$

for all $j = 1, 2, ..., M$, where $t_1, t_2, ..., t_p$ are the Chebyshev coefficients defined in (11). Then,

$$| f(y_j) - \tilde{f}_1(y_j) | < O(\frac{1}{5^p}). \tag{56}$$

for all $j = 1, 2, ..., M$.

Computation of the coefficients Φ_j requires $O(Np)$ operations, and a subsequent evaluation of $\tilde{f}_1(y_1), \ldots, \tilde{f}_1(y_M)$ is an $O(Mp)$ procedure. The total computational cost of approximating (53) to a relative precision $1/5^p$ is then $O(Np + Mp)$ operations.

Observation 3.15 *Suppose that $p > 2$ is an integer. Further, suppose that we define the real coefficients Ψ_m (representing the local expansion) by the formula*

$$\Psi_j = \sum_{k=1}^{N} \frac{\alpha_k}{\sqrt{t_j^2 - x_k^2}} \tag{57}$$

for all $m = 1, 2, ..., p$, where $t_1, t_2, ..., t_p$ are the Chebyshev coefficients defined in (11). Finally, suppose that

$$\tilde{f}_2(y_j) = \sum_{m=1}^{p} \Psi_m u_m(j) \tag{58}$$

for all $j = 1, 2, ..., N$, where u_m is defined in (12). Then,

$$\mid f(y_j) - \tilde{f}_2(y_j) \mid < O(\frac{1}{5^p}). \tag{59}$$

for all $j = 1, 2, ..., M$.

Computation of the coefficients Ψ_j requires $O(Np)$ operations, and a subsequent evaluation of $\tilde{f}_2(y_1), \ldots, \tilde{f}_2(y_M)$ is an $O(Mp)$ procedure. Again the total computational cost of approximating (53) to a relative precision $1/5^p$ is $O(Np + Mp)$ operations.

Consider now the general case, where the points $\{x_1, \ldots, x_N\}$ and $\{y_1, \ldots, y_M\}$ are arbitrarily distributed on the interval $[-1, 1]$ (see Remark 3.12). We use a hierarchy of grids to subdivide the computational domain $[-1, 1]$ into progressively smaller subintervals, and to subdivide the sets $\{x_i\}$ and $\{y_i\}$ according to subinterval (see Figure 2). A tree structure is imposed on this hierarchy, so that the two subintervals resulting from the bisection of a larger (parent) interval are referred to as its children. Two Chebyshev expansions are associated with each subinterval: a far-field expansion for the points within the subinterval, and a local expansion for the points which are well-separated from the subinterval. Interactions between pairs of well-separated subintervals can be computed via these Chebyshev expansions in the manner described above, and all other interactions at the finest level can be computed directly. Once the precision has been fixed, the computational cost of the entire procedure is $O(N)$ operations. We refer the reader to the algortithm of [6], or [4] for a detailed description.

3.3.2 A more efficient algorithm

Chebyshev expansions are not the most efficient means of representing interactions between well-separated intervals. All the matrix operators of the algorithm are numerically rank-deficient, and can be further compressed by a suitable change of basis. The orthogonal matrices required for this basis change are obtained via singular value decompositions (SVDs) of appropriate matrices.

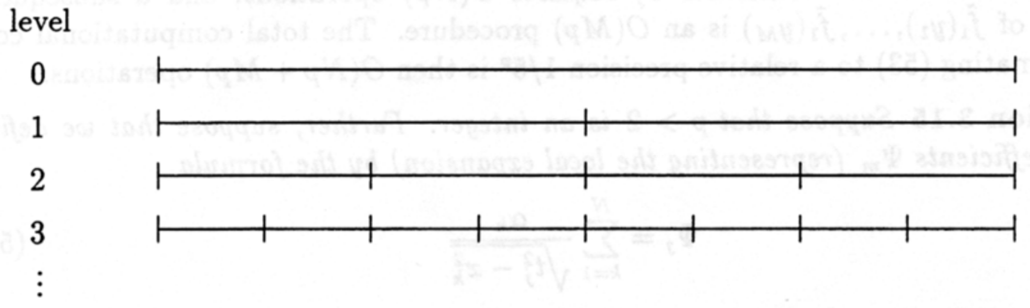

Figure 2: Hierarchy of subintervals.

4 The fast Hankel transform

4.1 Informal description of the algorithm

In this section we outline the procedure we use to rapidly evaluate integrals of the form

$$g(a) = \int_0^A f(x) J_0(ax)dx. \tag{60}$$

for all N values of A. More specifically, suppose that $h = \pi/(N-1)$, $x_i = ih$ and $f : [0,\pi] \to \mathbf{R}$ is a function tabulated at N $x_0, x_1, ..., x_{N-1}$. Then

$$g_j = \int_0^\pi f(x) J_0(jx)dx \tag{61}$$

is computed for all $j = 0, 1, 2..., N-1$, in $O(N \log N)$ operations.

Suppose that we define

$$F(a) = \int_0^\pi f(x)\cos(ax)dx, \tag{62}$$

for all $a > 0$. Then it follows immediately from the expression for the Bessel function J_0 defined in (6) that

$$g(a) = \int_{-a}^{a} \frac{F(u)}{\sqrt{a^2 - u^2}} du. \tag{63}$$

Now, given a function $f : [0,\pi]$ tabulated at N equispaced nodes, we evaluate the integrals (62) using a fast cosine transform in $O(N \log N)$, for N equispaced frequencies a. Subsequently, using a high order corrected trapezoidal rule (developed in [10]), we evaluate the integrals (63) in two stages. First, a trapezoidal approximation to the integral is made using the formula

$$g_T(a_j) = h \sum_{l=-(j-1)}^{j-1} \frac{F(u_l)}{\sqrt{a_j^2 - u_l^2}}. \tag{64}$$

The trapezoidal approximation g_T is subsequently corrected with appropriately chosen quadrature weights $\nu_1^j, \nu_2^j, ..., \nu_k^j$ using the formula

$$g_C(a_j) = h \sum_{i=1}^{k} \nu_i^j \frac{F(y_i)}{\sqrt{| a_j^2 - y_i^2 |}}, \tag{65}$$

for all $j = 0, 1, 2, ..., N-1$, where $y_i = a_j - hi$ for $1 \leq i \leq k/2$, and $y_i = a_j + h(i - k/2)$ for $k/2 + 1 \leq i \leq k$. In Section 3.2 we prove that the corrected rule $g_T + g_C$ is of order $2k - 2$, i.e., for all j there exists some real $c > 0$ such that

$$| (g_T(a_j) + g_C(a_j)) - g(a_j) | \leq \frac{c}{j^{2k-2}}. \tag{66}$$

The trapezoidal approximation g_T is rapidly computed in $O(N)$ operations, using a generalized version of the 1-D FMM for the smoothly varying kernel $\frac{1}{\sqrt{a^2-u^2}}$. The correction to the trapezoidal approximation g_C is computed in $O(kN)$ operations using (65). Thus, the total asymptotic time complexity of the algorithm is $O(N \log N) + O(N) + O(pN)$.

4.2 Detailed description of the algorithm

This section contains the detailed description and the complexity analysis of our algorithm for the fast Hankel transform.

Remark 4.1 Without loss of generality, we assume that the function f is tabulated on the interval $[0, \pi]$.

Suppose that $f : [0, \pi] \rightarrow \mathbf{R}$. Further, suppose that $h = \pi/(N-1)$, $x_i = ih$, and the function f is tabulated at the N equispaced nodes points $x_0, x_1, ..., x_{N-1}$. Then

$$g_j = \int_0^{\pi} f(x) J_0(j \cdot x) dx \tag{67}$$

is evaluated for all $j = 0, 1, 2..., N-1$, in $O(N \log N)$ operations. The function is assumed to be tabulated at the Nyquist sampling rate.

Algorithm 4.1

Step	**Complexity**	**Description**
1		**Comment** [Preprocessing: Input Problem size N. Compute the correction coefficients β_k^m using the formula (19). Compute the correction weights ν_p^j by solving the system of equations (40), (41). These correction weights are obtained once and are stored.]

2 $O(N \log N)$ **Comment** [The fast cosine transform of an even or odd function, i.e., for a function f tabulated at n points, $f(x_0), f(x_1), ..., f(x_{N-1})$, compute the integral $F_i^C \approx \int_0^\pi f(x)\cos(i \cdot x)dx$ for all $i = 0, 1, ..., N-1$.]

If the function f is even,

$$F_i^C = h(\sum_{j=0}^{N-1} f_j \cos(\frac{\pi i j}{N-1})),$$

is computed using a fast cosine transform.

If the function f is odd, $\hat{f}$ is calculated using the formula (33) in $O(N)$ operations. Subsequently,

$$F_i^C = h(\sum_{j=0}^{N-1} \hat{f}(x_j)\cos(\frac{\pi i j}{N-1}))$$

is computed using a fast cosine transform.

3 $O(N)$ **Comment** [The trapezoidal approximation to the to the integral (67) is computed in $O(N)$ operations using a version of the one-dimensiona fast multipole method.]

$$g_j^T = h \sum_{l=-(j-1)}^{j-1} \frac{F_l^C}{\sqrt{j^2 - l^2}}.$$

is computed for all $j = 0, 1, ..., N-1$, using the algorithm for the 1-D FMM (see, section 3.3).

4 $O(N)$ **Comment** [The corrections are computed using the stored correction coefficients]

$$g_j^C = h \sum_{i=1}^{k} \nu_i^j \frac{F_{j-i}^C}{\sqrt{j^2 - (j-i)^2}},$$

for all $j = \pm 1, \pm 2, ..., \pm \frac{N-1}{2}$,

5 $O(N)$ **Comment** [The corrections are added to the trapezoidal approximation, resulting in an approximation to the Hankel tranfsorm $\{g_j\}$]

$$g_j = g_j^C + g_j^T, \tag{68}$$

for all $j = 0, 1, \ldots, N-1$.

5 Numerical Results

In this section we present numerical experiments testing the algorithms of this paper.

Example 1: Corrections for the cosine transform

We considered the following integral (of an odd function),

$$\int_0^{2\pi} sin(bx) \cdot cos(bx) \cdot e^{-x^2}. \tag{69}$$

to illustrate the effectiveness of using the correction coefficients β^m (see, 19) for correcting the cosine transform of an odd function (see, (19)). In Table 1 we present convergence results of the standard trapezoidal rule, while in Table 2 we present convergence results using the correction coefficients. In both tables the first column contains the number of nodes discretizing the interval $[0 : 2\pi]$. In Table 1, columns 2-6 contain the relative errors of the standard trapezoidal rule used to evaluate the integral (69) for various values of b. In Table 2, columns 2-6 contain the relative errors of the corrected trapezoidal rule to also evaluate the integral (69) for various values of b. We observe empirically that, an odd function sampled at 4 points per wavelength, can be integrated to double precision accuracy using the correted trapezoidal rule for non-singular functions. Hence, it is possible to accurately evaluate the cosine transform of an odd function if the function is sampled at twice the Nyquist sampling rate.

Example 2: Corrections for the singular kernel

The quadrature weights for the rule $T^j_{\nu^j}$ (see, (36)) are obtained as solutions of linear systems (40), (41). The linear systems used for determining these weights are very ill-conditioned. In order to combat the high condition number, all systems were solved using the mathematical package Mathematica using 100 significant digits. We considered the following integral

$$\int_{-\pi}^{\pi} \frac{\cos(b \cdot u)}{\sqrt{\pi^2 - u^2}} du, \tag{70}$$

to experimentally demostrate the convergence rate of the quadrature rule $T^j_{\nu^j}$. In Table 3 convergence results are presented for the rule rule $T^j_{\nu^j}$ using 20 correction weights $(\nu^j_1, \nu^j_2, ..., \nu^j_{20})$. The first column contains the number of nodes discretizing the interval $[0 : 2\pi]$. Columns 2-4 contain the relative errors of rule to evaluate the integral (70) for various values of b. It can be observed in Table 3 that the rule $T^j_{\nu^j}$ provides single precision accuracy for a function tabulated at twice the Nyquist sampling rate, and double precision accuracy for function tabulated at four times the Nyquist sampling rate.

We have written a computer program in ANSI FORTRAN for the implementation of the algorithm of this paper. This program was tested on a Sun SPARCstation 10 for a variety of input data. Four experiments are described below, and their results are summarized in Tables 4-7. These tables contain error estimates and CPU time requirements for the algorithms, with all computations performed in double precision arithmetic.

The table entries are described below

- The first column in each table contains the problem size N, which was chosen to be a power of 2, ranging from 32 to 1024.
- The second column contains the time required for the fast cosine transform. In Examples 3, and 4 this includes the time to interpolate to a finer grid.
- The third column contains the time required for the one-dimensional FMM.
- The fourth column contains the time required for the correction to the trapezoidal approximation to the Hankel transform.
- The fifth column contains the time required for the the direct implementation of the Hankel transform.
- The sixth column contains the time required for an FFT of the same size.
- The seventh column contains the relative 2-norm E_2 for each result.

Two technical details of our implementations appear worth mentioning here:

- The implementation consists of two main subroutines: the first is an initialization stage in which the elements of the various matrices employed by the algorithms are stored on disk, and the second is the evaluation stage in which these matrices are applied. Successive applications of the linear transformations to multiple vectors requires the initialization to be performed once.
- The parameters for the algorithm were chosen to retain maximum precision while minimizing the CPU time requirements. We found that by using 20 quadrature weights for the correction of the trapezoidal approximation (see, 45) we minimized the CPU time of the algorithm without sacrificing accuracy.

Following are the descriptions of experiments, and tables of numerical results.

Example 1. The Hankel transform for an even function The purpose of this example is to demonstrate the performance of the fast Hankel transform for evaluating the expression

$$g(a_j) = \int_0^{2\pi} f(x) J_0(a_j x) dx \tag{71}$$

where

$$f(x) = (\cos(b \cdot x) + \cos(\frac{b \cdot x}{2}) + \cos(\frac{b \cdot x}{3})) \cdot e^{-x^2}, \tag{72}$$

and $b = \frac{N}{4}$. The function f is tabulated at the equispaced nodes $x_0, x_1, ..., x_{N-1}$, where $h = \frac{2\pi}{N-1}$, and, $x_i = ih$. The Hankel transform (71) is computed for all $a_j = \frac{\pi j}{Nh}$ for all $j = 0, 1, ..., N-1$. The single precision results are provided in Table 4. The algorithm is easily modified to provide double precision results, merely by zero padding the input vector by a vector of the same size (see, Table 3). The double precision results are are provided in Table 5.

Example 2. The Hankel transform for an odd function The purpose of this example is to demonstrate the performance of the fast Hankel transform for evaluating the expression

$$g(a_j) = \int_0^{2\pi} f(x) J_0(a_j x) dx \tag{73}$$

where

$$f(x) = x \cdot ((\cos(b \cdot x) + \cos(\frac{b \cdot x}{2}) + \cos(\frac{b \cdot x}{3})) \cdot e^{-x^2}), \tag{74}$$

and $b = \frac{N}{4}$. The function f is tabulated at the equispaced nodes $x_0, x_1, ..., x_{N-1}$, where $h = \frac{2\pi}{N-1}$, and, $x_i = ih$. The Hankel transform (71) is computed for all $a_j = \frac{\pi j}{Nh}$ for all $j = 0, 1, ..., N-1$. The single precision results are provided in Table 4. The algorithm is easily modified to provide double precision results, merely by zero padding the input vector by a vector of the same size (see, Table 3). The double precision results are are provided in Table 5. We observe that if a function g is defined as follows

$$\begin{aligned} g(u,v) &= (\cos(b \cdot \sqrt{(u^2+v^2)}) + \cos(\frac{b \cdot \sqrt{(u^2+v^2)}}{2}) \\ &+ \cos(\frac{b \cdot \sqrt{(u^2+v^2)}}{3})) \cdot e^{-(u^2+v^2)}, \end{aligned} \tag{75}$$

(i.e., the function is rotationally symmetric) then the Hankel transform may be used to compute the two-dimensional FFT (see,)

$$g(a_k, a_j) = \int_{-2\pi}^{2\pi} \int_{-2\pi}^{2\pi} g(u,v) \cdot e^{ia_k} \cdot e^{ia_j} \tag{76}$$

for all $k = 0, \pm 1, \pm 2, ..., \pm \frac{N-1}{2}$, and all $j = 0, \pm 1, \pm 2, ..., \pm \frac{N-1}{2}$. Column 10 contains the time required by a 2-D FFT to evaluate the integrals (76) Column 11 contains the ratio of the time taken by the 2-D FFT to the time taken by Algorithm 4.1.

Table 1: Accuracy of the uncorrected trapezoidal rule for evaluating integrals $\int_0^{2\pi} sin(bx) \cdot cos(bx) \cdot e^{-x^2}$

N	b=4	b=8	b=16	b=32	b=64
32	0.223E+00	0.108E+01	0.124E+02	0.330E+02	0.358E+02
64	0.519E-01	0.220E+00	0.104E+01	0.261E+02	0.677E+02
128	0.127E-01	0.524E-01	0.218E+00	0.102E+01	0.533E+02
256	0.313E-02	0.129E-01	0.523E-01	0.216E+00	0.101E+01
512	0.779E-03	0.320E-02	0.129E-01	0.521E-01	0.215E+00

Table 2: Accuracy of the corrected trapezoidal rule for evaluating the integrals $\int_0^{2\pi} sin(bx) \cdot cos(bx) \cdot e^{-x^2}$

N	b=4	b=8	b=16	b=32	b=64
32	0.359E-04	0.125E+01	0.126E+02	0.334E+02	0.376E+02
64	0.730E-14	0.346E-11	0.119E+01	0.262E+02	0.679E+02
128	0.708E-14	0.705E-14	0.621E-14	0.115E+01	0.533E+02
256	0.665E-14	0.705E-14	0.554E-14	0.355E-14	0.111E+01
512	0.644E-14	0.749E-14	0.598E-14	0.599E-14	0.133E-13

Table 3: Accuracy of the quadrature rule $T^j_{v_j}$ for evaluating the integrals $\int_{-\pi}^{\pi} \cos(b \cdot u)/(\pi^2 - u^2)^{\frac{1}{2}} du$

N	b=N/2	b=N/4	b=N/8
32	0.165E-01	0.829E-08	0.897E-15
64	0.109E-02	0.334E-07	0.868E-14
128	0.671E-02	0.668E-08	0.298E-13
256	0.784E-02	0.175E-07	0.107E-13
512	0.798E-02	0.242E-07	0.361E-13
1024	0.795E-02	0.257E-07	0.597E-14

Table 4: Numerical results for Example 1 (single precision)

N	t_{cos}	t_{fmm}	t_{corr}	t_{alg}	t_{dir}	t_{fft}	t_{alg}/t_{fft}	E_2 error
32	0.00011	0.00016	0.00016	0.00043	0.00050	0.00011	3.839	0.33351E-07
64	0.00018	0.00048	0.00034	0.00100	0.00142	0.00019	5.319	0.33688E-06
128	0.00040	0.00116	0.00080	0.00236	0.00464	0.00040	5.900	0.22484E-06
256	0.00070	0.00280	0.00170	0.00520	0.01660	0.00082	6.341	0.19274E-06
512	0.00200	0.00600	0.00350	0.01150	0.05900	0.00260	4.423	0.32146E-06
1024	0.00400	0.01200	0.00700	0.02300	0.22900	0.00760	3.026	0.62221E-06

Table 5: Numerical results for Example 1 (double precision)

N	t_{cos}	t_{fmm}	t_{corr}	t_{alg}	t_{dir}	t_{fft}	t_{alg}/t_{fft}	E_2 error
64	0.00020	0.00065	0.00030	0.00115	0.00140	0.00020	5.750	0.27881E-13
128	0.00030	0.00190	0.00080	0.00300	0.00470	0.00038	7.895	0.12463E-12
256	0.00060	0.00500	0.00160	0.00720	0.01660	0.00084	8.571	0.13597E-12
512	0.00150	0.01350	0.00350	0.01850	0.06300	0.00280	6.607	0.19605E-12
1024	0.00417	0.03333	0.00667	0.04417	0.23917	0.00833	5.300	0.26481E-12

Table 6: Numerical results for Example 2 (single precision)

N	t_{cos}	t_{fmm}	t_{corr}	t_{alg}	t_{dir}	t_{fft}	t_{alg}/t_{fft}	E_2 error	t_{2Dfft}	t_{2Dfft}/t_a
32	0.00046	0.00016	0.00016	0.00078	0.00086	0.00011	7.091	0.76613E-07	0.00335	4.295
64	0.00082	0.00050	0.00038	0.00170	0.00214	0.00021	8.173	0.33767E-06	0.01332	7.835
128	0.00160	0.00116	0.00080	0.00356	0.00588	0.00040	8.900	0.17319E-05	0.05268	14.798
256	0.00350	0.00270	0.00170	0.00790	0.01890	0.00086	9.186	0.18528E-06	0.22000	27.848
512	0.00850	0.00550	0.00350	0.01750	0.06800	0.00260	6.731	0.45411E-06	1.42000	81.143
1024	0.02000	0.01300	0.00700	0.04000	0.25200	0.00740	5.405	0.6342E-06	8.58200	214.550

Table 7: Numerical results for Example 2 (double precision)

N	t_{cos}	t_{fmm}	t_{corr}	t_{alg}	t_{dir}	t_{fft}	t_{alg}/t_{fft}	E_2 error	t_{2Dfft}	t_{2Dfft}/t_a
64	0.00080	0.00065	0.00035	0.00180	0.00205	0.00019	9.474	0.10525E-13	0.01265	7.028
128	0.00160	0.00190	0.00080	0.00430	0.00590	0.00038	11.316	0.85742E-13	0.05190	12.070
256	0.00340	0.00460	0.00180	0.00980	0.01880	0.00084	11.667	0.10078E-12	0.21880	22.327
512	0.00750	0.01400	0.00350	0.02500	0.06700	0.00280	8.929	0.90009E-12	1.55300	62.120
1024	0.02250	0.03167	0.00750	0.06167	0.25583	0.00783	7.872	0.54210E-12	8.72167	141.432

6 Generalizations and Conclusions

It is is well known (see, for example [13]) that a function $f \in c^k(0,a)$ can be expanded in a series

$$f(x) = \sum_{r=1}^{\infty} A_r J_n(\lambda_r x), \tag{77}$$

where $n > -1$, J_n are the Bessel functions of order n, and $\lambda_1, \lambda_2, ...,$ are the positive zeros of $J_n(\lambda a)$. The coefficients A_r are given by the formula

$$\int_0^a x f(x) J_n(\lambda_r x) dx = \frac{a^2}{2} A_r (J_n'(\lambda_r a))^2. \tag{78}$$

Expansions of the form (77) are known as Fourier-Bessel expansions. Fourier-Bessel expansions are encountered in many areas of computational physics. Among the problems leading to them are the vibrations of a circular membrane, flow of heat in a circular cylinder, wave propagation in a three-dimensional layered medium and many others.

The Fourier Transform of a cylindrically symmetric function can be computed with a single integral instead of a double integral as we show below. We introduce polar coordinates in both the spatial and frequency domains,

$$x = r\cos(\phi) \quad \text{and} \quad y = r\sin(\phi),$$
$$u = \rho\cos(\alpha) \quad \text{and} \quad v = \rho\sin(\alpha),$$

so that $ux + vy = r\rho\cos(\phi - \alpha)$. If, $f(x,y) = g(r)$ then the transform

$$F(u,v) \;=\; \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) e^{-i(ux+vy)} dx dy,$$

is just

$$G(\rho) \;=\; \int_{-\pi}^{\pi}\int_0^{\infty} f(r) e^{-ir\rho\cos(\phi-\alpha)} dr d\phi.$$

If we change the order of integration,

$$\int_{-\pi}^{\pi} e^{-ir\rho\cos(\phi-\alpha)} d\phi \;=\; 2\pi J_0(r\rho).$$

Thus, if $F(u,v) = G(\rho)$ then,

$$G(\rho) = 2\pi \int_0^{\infty} r f(r) J_0(r\rho) dr. \tag{79}$$

In fact, most problems involving Fourier Transforms of cylindrically symmetric functions can be reformulated in terms of Fourier-Bessel integrals. This leads to a significant interest in the numerical evaluation Fourier-Bessel integrals. Unfortunately, the direct evaluation of these integrals, given a function tabulated at N points, requires $O(N^2)$ operations. This makes the procedure numerically unattractive for large N, so that in practice Fourier-Bessel Transforms are avoided in favour of Fourier Tranforms whenever possible.

References

[1] ANDERSON, W. L., *Numerical integration of related Hankel transforms of orders 0 and 1 by adaptive digital filtering*, Geophysics, 44, (1979).

[2] ANDERSON, W. L., *Fast Hankel transforms using related and lagged convolutions* ACM Trans. Math. Software. (1982).

[3] M. ABRAMOVITZ AND I. STEGUN, *Handbook of Mathematical Functions.* National Bureau of Standards, Washington, DC., 1964.

[4] BRADLEY K. ALPERT, VLADIMIR ROKHLIN, *A Fast Algorithm for the Evaluation of Legendre Expansions*, Siam J, Sci. Stat Comput, Vol. 12, No. 1., (1991)

[5] G. DAHLQUIST, A. BJORK, *Numerical Methods*, Prentice Hall, NJ (1974)

[6] ALOK DUTT, MING GU, VLADIMIR ROKHLIN, *Fast Algorithms for Polynomial Interpolation, Integration and Differention* Technical Report 977, Yale Computer Science Department, New Haven, CT, (1994)

[7] ALOK DUTT, *Fast Fourier Transforms for Non-Equispaced Data* Technical Report 981, Yale Computer Science Department, New Haven, CT, (1993)

[8] I. S. GRADSHTEYN AND I. M. RYZHIK, *Table of Integrals, Series and Products*, Academic Press Inc., 1980.

[9] L. GREENGARD AND V. ROKHLIN, *A Fast Algorithm for Particle Simulations*, J. Comp. Phys., 73 (1987), pp. 325–348.

[10] SHARAD KAPUR, VLADIMIR ROKHLIN, *High-Order Corrected Trapezoidal Rules for Singular Functions*, Technical Report 1042, Yale Computer Science Department, New Haven, CT, (1994)

[11] SHARAD KAPUR, VLADIMIR ROKHLIN, *The Fast Fourier Transform for Functions with Singularities.* Technical Report (1050), Yale Computer Science Department, New Haven, CT, (1994)

[12] QING-HUO LIU, WENG CHO CHEW, *Applications of the conjugate gradient fast Fourier Hankel transform method with an improved Hankel transform algorithm.* Radio Science. July-August (1994).

[13] T. M. MACROBERT, *An Elementary Treatise on Harmonic Functions with Applications.* Pergammon Press, Oxford, London, 1967.

[14] GREGORY MATVIEYNKO, *On the Evaluation of Bessel functions* Technical Report 903, Yale Computer Science Department, CT, (1992)

[15] ALLAN V. OPPENHEIM, ALAN S. WILLSKY, *Signals and Systems.* Prentice-Hall, Inc. Englewood Cliffs, NJ, (1983)

[16] V. ROKHLIN, *End-point corrected trapezoidal quadrature rules for singular functions*, Computers and Mathematics with Applications, 1990.

[17] SIEGMAN, A. E. *Quasi fast Hankel transform*, Optical letters. (1977).

[18] J. STOER AND R. BULIRSCH, *Introduction to Numerical Analysis*, Springer Verlag, New York, 1980.

[19] J. WALKER, *Fast Fourier Transforms*, CRC Press, Ann Arbor, 1991.