An algorithm for the rapid evaluation of special function transforms

Michael O'Neil, Franco Woolfe, and Vladimir Rokhlin
Technical Report YALEU/DCS/TR-1399
April 15, 2008

We introduce a fast algorithm for the numerical application to arbitrary vectors of several special function transforms. The algorithm requires $\mathcal{O}(n \log(n))$ operations to apply to an arbitrary vector any $n \times n$ matrix such that the rank of any $p \times q$ contiguous submatrix is bounded by a constant times pq/n. These rank bounds are proven here for the case of the Fourier-Bessel transform. Numerical experiments demonstrate a much wider applicability. The performance of the algorithm is illustrated via several numerical examples.

An algorithm for the rapid evaluation of special function transforms

Michael O'Neil, Franco Woolfe, and Vladimir Rokhlin
Technical Report YALEU/DCS/TR-1399
April 15, 2008

The authors were supported in part by ONR Grant #N00014-07-1-0711, NGS Grant #HM1582-06-1-2039, DARPA Grant #FA9550-07-1-0541, Schlumberger-Doll Contract #02659, AFOSR Grant #FA9550-06-1-0239, DARPA Grant #HR0011-05-1-0002, and AFOSR Grant #F49620-03-C-0031.

Keywords: *fast, transform, algorithm, matrix, Fourier-Bessel, special functions*

1 Introduction

Special function transforms are a widely used and well-understood tool of applied mathematics; they are encountered, inter alia, in weather and climate modeling [19], [20], [21], tomography [14], [9], electromagnetics [13], and acoustics [6], [16]. Examples of such transforms include the Fourier transform, the Fourier-Bessel transform, orthogonal polynomial transforms, etc.

We present an algorithm for the numerical computation of several special function transforms. Suppose that S is a change of basis matrix between the standard basis and a basis of special functions. We begin by compressing each of the submatrices of S shown in Figure 1; we refer to these submatrices as Level 0. Each subsequent level consists of submatrices obtained from the previous level by merging horizontally and splitting vertically. Level 1 is illustrated in Figure 2 and Level 2 is illustrated in Figure 3. In the last level, each submatrix extends across an entire row of S, as illustrated in Figure 4. We compress each submatrix at each level. We then use the compressed submatrices to apply the matrix S to an arbitrary vector; this requires $\mathcal{O}(n\log(n))$ operations to apply any matrix such that the rank of any $p \times q$ contiguous submatrix is bounded by a constant times pq/n. We prove the required rank bounds for the case of the Fourier-Bessel transform in Theorem 4.3. Numerical examples demonstrate a much wider applicability.

In addition to enabling the fast application of certain matrices, the algorithm of the present paper can be used as a tool for matrix compression. The $n \times n$ matrices examined are compressed using approximately $\mathcal{O}(n\log(n))$ memory.

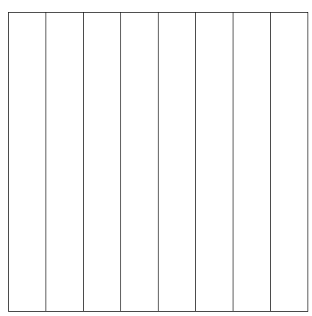

Figure 1: Level 0

It should be pointed out that the algorithm of this paper is very similar to that of [12] and has been motivated by the latter. In particular, the term "butterfly algorithm" is introduced in [12], due to its similarity to the butterfly stage in the Fast Fourier Transform (FFT).

It is not the purpose of this paper to review the extensive literature on the subject of algorithms for special function transforms. For a detailed survey of the literature we refer the reader to [15], [17], [22], and the references therein. In brief, the algorithm described in [17] handles associated Legendre functions, spheroidal wave functions of all orders, associated Laguerre functions, and the Fourier-Bessel transform. The algorithm described in [22] handles associated Legendre functions, Hermite functions, associated Laguerre functions, the Fourier-Bessel transform, and sums of Bessel functions of varying orders. The algorithm described in [22] is much preferable to that of [17] in most circumstances.

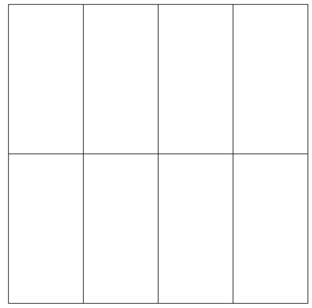

Figure 2: Level 1

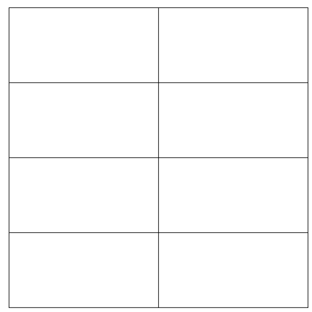

Figure 3: Level 2

The algorithm of the present article is quite easy to implement, and, furthermore, applies directly to Fourier-Bessel series, whereas the algorithms of [17] and [22] do not. The structure of our algorithm is notably different. This paper illustrates our algorithm by treating the example of the Fourier-Bessel transform in full detail. We also illustrate the application of the butterfly algorithm to a wide range of special function transforms numerically, in Section 6 below.

The present paper has the following structure: Section 2 sets the notation. Section 3 collects various known facts which later sections utilize. Section 4 provides the principal lemmas which Section 5 uses to construct an algorithm. Section 5 describes the algorithm of the present paper, providing details about its computational costs. Section 6 illustrates the performance of the algorithm via several numerical examples. Section 7 draws several conclusions and discusses possible extensions.

2 Notation

We define $\mathbb{R}$ to be the set of all real numbers. We define $\mathbb{C}$ to be the set of all complex numbers. Throughout this paper, we use $i = \sqrt{-1}$.

For a real number x, we define $\lfloor x \rfloor$ to be the largest integer n which satisfies $n \leq x$.

For any positive integer l, we define we define $\mathbf{1}$ to be the real $l \times l$ matrix whose (j, k) entry is 1 if $j = k$, and 0 if $j \neq k$, for integers j, k such that $1 \leq j \leq l$ and $1 \leq k \leq l$. We

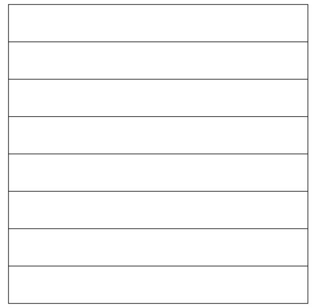

Figure 4: The last level

refer to $\mathbf{1}$ as the identity matrix of size l.

Suppose that a and b are real numbers and that $f(x)$ is a complex valued function, defined for every real number x such that $a \leq x \leq b$. We define the $L^2[a,b]$ norm of f via the formula

$$\|f\|_{[a,b]} = \sqrt{\int_a^b |f(x)|^2 \, dx}. \tag{2.1}$$

We define $L^2[a,b]$ to be the set of all complex valued functions defined on the interval $[a,b]$ such that $\|f\|_{[a,b]} < \infty$.

Suppose that a, b, u, and v are real numbers such that $a < b$ and $u < v$. Suppose further that $A : L^2[a,b] \to L^2[u,v]$ is an integral operator with kernel $k(x,t) : [a,b] \times [u,v] \to \mathbb{C}$ given by the formula

$$(Af)(x) = \int_a^b k(x,t) f(x) \, dx. \tag{2.2}$$

We define the spectral norm of the kernel k or the operator A via the formula

$$\|A\|_2 = \|k\|_2 = \sup_{f \in L^2[a,b]} \frac{\|Af\|_{[u,v]}}{\|f\|_{[a,b]}}. \tag{2.3}$$

If v is an $n \times 1$ vector we define its norm $\|v\|$ via the formula

$$\|v\| = \sum_{j=1}^{n} |v_j|^2, \tag{2.4}$$

where v_j is the j^{th} entry in v, for every integer j such that $1 \leq j \leq n$. If S is a $n \times m$ matrix, we define its norm $\|S\|$ via the formula

$$\|S\| = \max_{0 \neq w \in \mathbb{R}^m} \frac{\|Sw\|}{\|w\|}. \tag{2.5}$$

3 Mathematical preliminaries

3.1 Facts from numerical analysis

3.1.1 Numerical rank

Suppose that m is a postitive integer and that a, b, u, v, and ε are real numbers such that $a < b$ and $u < v$, with $\varepsilon > 0$. Suppose further that $A : L^2[a, b] \to L^2[u, v]$ is the integral operator with kernel $k : [a, b] \times [u, v] \to \mathbb{C}$, given by the formula

$$(Af)(x) = \int_a^b k(x,t) f(x)\, dx. \tag{3.1}$$

The operator A in (3.1) is defined to have rank m to precision ε if m is the least integer such that there exist $2m$ functions $g_1, g_2, \ldots, g_{m-1}, g_m$ and $h_1, h_2, \ldots, h_{m-1}, h_m$ satisfying $\|k(x,t) - \sum_{k=1}^{m} g_k(x) h_k(t)\|_2 = \varepsilon$.

3.1.2 The interpolative decomposition

The following lemma states that, for any $m \times n$ matrix A whose rank is r, there exist an $m \times r$ matrix B whose columns constitute a subset of the columns of A, and a $r \times n$ matrix P, such that

1. some subset of the columns of P makes up the $r \times r$ identity matrix,
2. P is not too large, and
3. $BP = A$.

Moreover, the lemma provides an analogous approximation BP to A when the exact rank of A is not r, but the $(r+1)^{\text{st}}$ singular value of A is nevertheless small. We refer to B as a *column skeleton matrix* and to P as an *interpolation matrix*. We refer to the expression $BP = A$ as an *interpolative decomposition* of A. The lemma can be found, in a slightly different form, in [11], [2], and [8].

Lemma 3.1 *Suppose that m and n are positive integers, and A is a complex $m \times n$ matrix.*

Then, for any positive integer r with $r \le m$ and $r \le n$, there exist a complex $r \times n$ matrix P, and a complex $m \times r$ matrix B whose columns constitute a subset of the columns of A, such that

1. *some subset of the columns of P makes up the $r \times r$ identity matrix,*
2. *no entry of P has an absolute value greater than 2,*
3. $\|P\| \le \sqrt{4r(n-r)+1}$,
4. *the least (that is, r^{th} greatest) singular value of P is at least 1,*
5. $BP = A$ *when $r = m$ or $r = n$, and*

6. $\|BP - A\| \le \sqrt{4r(n-r)+1}\,\sigma_{r+1}$ *when* $r < m$ *and* $r < n$*, where* σ_{r+1} *is the* $(r+1)^{\text{st}}$ *greatest singular value of* A*.*

Remark 3.2 In this paper, we use the numerical scheme for the computation of the matrix P described in [2]. The scheme is stable and requires $\mathcal{O}(rmn)$ floating point operations and $\mathcal{O}(mn)$ floating point words of memory. The reader is referred to [11], [2], and [8] for a more detailed description of matrix skeletonization and related techniques.

3.2 Special functions

3.2.1 Fourier series

The following information concerning discrete Fourier transforms on equispaced nodes can be found, for example, in [3]. We also describe discrete Fourier transforms for nonequispaced nodes, referring the reader to [4] and [5] for more information.

The functions e^{inx}, with integer n, are orthogonal on the interval $[-\pi, \pi]$, that is, for any integers m and n such that $m \neq n$,

$$\int_{-\pi}^{\pi} e^{imx} e^{inx}\, dx = 0. \tag{3.2}$$

The functions e^{inx}, with integer n, form a basis for $L^2[-\pi, \pi]$ so that for any square integrable function $f : [-\pi, \pi] \to \mathbb{C}$, there exist real numbers $c_0, c_1, c_2, \ldots$ such that

$$f(x) = \sum_{j=-\infty}^{\infty} c_j e^{ijx}. \tag{3.3}$$

In numerical applications, the series in (3.3) is truncated after n terms, for some appropriately chosen integer n. The function f is thus approximated by the trigonometric polynomial $p : [-\pi, \pi] \to \infty$, given by the formula

$$f(x) \approx p(x) = \sum_{j=-n}^{n} c_j e^{ijx}. \tag{3.4}$$

Formula (9.2.4) in [3] states that the complex number c_j in equation (3.4) is given by the formula

$$c_j = \frac{1}{2n+1} \sum_{k=0}^{2n} p(x_k) e^{-ijx_k}, \tag{3.5}$$

where the real number x_k is defined via the formula

$$x_k = \frac{2\pi k}{2n+1}, \tag{3.6}$$

for integers j, k such that $-n \le j \le n$ and $0 \le k \le 2n$.

The discrete Fourier transform maps the $2n+1$ values $p(x_k)$ of the function p at the nodes x_k to the $2n+1$ coefficients c_j in (3.4), for $-n \le j \le n$ and $0 \le k \le 2n$. The inverse discrete Fourier transform maps the $2n+1$ coefficients c_j in (3.4) to the $2n+1$ values $p(x_k)$ of the function p at the nodes x_k, for $0 \le k \le 2n$ and $-n \le j \le n$. The fast Fourier transform (FFT) algorithm for computing the forward and inverse discrete Fourier transforms is widely known (see, for example, [3]). Note that the FFT and equation (3.5) rely on the facts that $x_0, x_1, \ldots, x_{2n-1}, x_{2n}$ are equispaced and j is an integer between $-n$ and n.

The algorithm described in Section 5 can be used to evaluate sums of the form

$$\alpha_j = \sum_{k=0}^{2n} p(x_k)\, e^{-i\omega_j x_k}, \tag{3.7}$$

where $x_0, x_1, \ldots, x_{2n-1}, x_{2n}$ are arbitrary real numbers between 0 and 2π and ω_0, ω_1,..., ω_{2n-1}, ω_{2n} are arbitrary real numbers between $-n$ and n. If $x_0, x_1, \ldots, x_{2n-1}, x_{2n}$ are nonequispaced, we refer to the sum in equation (3.7) as a discrete Fourier transform for nonequispaced nodes. Calculation of a Fourier transform for nonequispaced nodes requires that we apply to the vector $(p(x_0), p(x_1), \ldots, p(x_{2n-1}), (x_{2n}))^\top$ the matrix T_F defined by the formula

$$T_F = \begin{pmatrix} e^{-i\omega_0 x_0} & \cdots & e^{-i\omega_0 x_{2n}} \\ \vdots & \ddots & \vdots \\ e^{-i\omega_{2n} x_0} & \cdots & e^{-i\omega_{2n} x_{2n}} \end{pmatrix}. \tag{3.8}$$

Remark 3.3 If the nodes $x_0, x_1, \ldots, x_{2n-1}, x_{2n}$ are equispaced the use of the FFT to calculate the forward and inverse Fourier transforms is faster than the algorithm of the present paper. If the nodes $x_0, x_1, \ldots, x_{2n-1}, x_{2n}$ are not equispaced, the forward and inverse Fourier transforms can be calculated via the application of the matrix T_F defined in equation (3.8); this is accelerated by the algorithm described in Section 5. However, the algorithms of [4] and [5] are faster than the algorithm of this paper for computing nonequispaced Fourier transforms. Unlike the methods of [4] and [5] the method of this paper works for many transforms besides the Fourier transform.

3.2.2 Bessel functions

Following the standard practice, we will be denoting by J_m the Bessel function of the first kind of order m and by $H_m^{(1)}$ the Hankel function of the first kind of order m. Whenever m is an integer, J_m is analytic in the whole complex plane, and H_m has a singularity at 0 and a branch cut along the negative real axis. The properties of Bessel functions are extremely well-known, and the reader is referred (for example) to [1] and [25].

The algorithm of Section 5 accelerates the evaluation of the Fourier-Bessel transform as described in Sections 3.2.3 and 3.2.4. In addition, the algorithm described in Section 5 accelerates the evaluation of sums of Bessel and Hankel functions over varying orders. These sums are analogous to expansions in orthogonal polynomials (see, for example, [24] and [23]). For any non-negative real number x and complex numbers α_0, α_1,...,α_{m-2}, α_{m-1}, we consider the sums

$$g(x) = \sum_{k=0}^{m-1} \alpha_k\, J_k(x), \tag{3.9}$$

and

$$w(x) = \sum_{k=0}^{m-1} \alpha_k \, H_k^{(1)}(x), \tag{3.10}$$

To evaluate the function g, defined in (3.9), at the real points $x_1, x_2, \ldots, x_{m-1}, x_m$ we must apply to the vector $(\alpha_0, \alpha_1, \ldots, \alpha_{m-2}, \alpha_{m-1})^\top$ the matrix E_J defined via the formula

$$E_J = \begin{pmatrix} J_0(x_1) & \cdots & J_{m-1}(x_1) \\ \vdots & \ddots & \vdots \\ J_0(x_m) & \cdots & J_{m-1}(x_m) \end{pmatrix}. \tag{3.11}$$

To evaluate the function w, defined in (3.10), at the real points $x_1, x_2, \ldots, x_{m-1}, x_m$ we must apply to the vector $(\alpha_0, \alpha_1, \ldots, \alpha_{m-2}, \alpha_{m-1})^\top$ the matrix $E_H^{(1)}$ defined via the formula

$$E_H^{(1)} = \begin{pmatrix} H_0^{(1)}(x_1) & \cdots & H_{m-1}^{(1)}(x_1) \\ \vdots & \ddots & \vdots \\ H_0^{(1)}(x_m) & \cdots & H_{m-1}^{(1)}(x_m) \end{pmatrix}. \tag{3.12}$$

The algorithm described in Section 5 accelerates the application of the matrices E_J and $E_H^{(1)}$ defined in (3.11) and (3.12) respectively.

In this paper, we will need two identities connecting Bessel functions with Chebyshev polynomials. Equation (3.13) is a reformulation of Formula 7.355.1 in [7] and equation (3.14) is a reformulation of Formula 7.355.2 in [7].

Lemma 3.4 *For any non-negative integer k and positive real number y,*

$$(-1)^k \frac{\pi}{2} J_{2k+1}(y) = \int_0^1 \frac{T_{2k+1}(s)\sin(ys)}{\sqrt{1-s^2}} \, ds \tag{3.13}$$

and

$$(-1)^k \frac{\pi}{2} J_{2k}(y) = \int_0^1 \frac{T_{2k}(s)\cos(ys)}{\sqrt{1-s^2}} \, ds. \tag{3.14}$$

3.2.3 The Fourier-Bessel transform

We consider the two dimensional Fourier transform $\hat{g}$ of the function g, defined via the formula

$$\hat{g}(\xi, \eta) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) \, e^{-ix\xi - iy\eta} \, dx \, dy. \tag{3.15}$$

We next express $\hat{g}$ in polar coordinates, obtaining the function γ defined via the formula

$$\gamma(\rho, \tau) = \hat{g}(\xi, \eta), \tag{3.16}$$

where

$$\xi = \rho \cos(\tau) \tag{3.17}$$

and

$$\eta = \rho \sin(\tau), \tag{3.18}$$

for any non-negative real number ρ and any real number τ such that $-\pi \leq \tau \leq \pi$. For each fixed non-negative real number ρ, we consider the Fourier series of the function $\gamma(\rho, \tau)$,

$$\gamma(\rho, \tau) = \sum_{m=-\infty}^{\infty} \gamma_m(\rho) e^{imt}, \tag{3.19}$$

where the Fourier coefficient $\gamma_m(\rho)$ is given by the formula

$$\gamma_m(\rho) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \gamma(\rho, \tau) e^{-im\tau} \, d\tau, \tag{3.20}$$

for every integer m. We next express g in polar coordinates, obtaining the function f defined via the formula

$$f(r, t) = g(x, y), \tag{3.21}$$

where

$$x = r\cos(t) \tag{3.22}$$

and

$$y = r\sin(t), \tag{3.23}$$

for any non-negative real number r and any real number t such that $-\pi \leq t \leq \pi$. It is well known (see, for example, page 137 in [18]) that the Fourier coefficient $\gamma_m(\rho)$ defined in (3.20) satisfies

$$\gamma_m(\rho) = (-i)^m \int_0^{\infty} r J_m(r\rho) f_m(r) \, dr, \tag{3.24}$$

where J_m is the Bessel function of the first kind of order m and $f_m(r)$ is the m^{th} Fourier coefficient of the function $f(r, t)$ given by the formula

$$f_m(r) = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(r, t) e^{-imt} \, dt, \tag{3.25}$$

for every integer m. The function $\gamma_m(\rho)$ defined in (3.24) is known as the Fourier-Bessel transform of $f_m(r)$.

3.2.4 Fourier-Bessel series

In this section, we continue to use the notation of Section 3.2.3. We now suppose that m is a non-negative integer and that the functions f and g are zero outside of the disk of radius R centered at the origin, where R is a positive real number.

We define the real function $\tilde{J}_m : [0, \infty) \to \mathbb{R}$ via the formula

$$\tilde{J}_m(\rho) = J_m(2\pi\rho R). \tag{3.26}$$

We define the positive real numbers $0 < \rho_1 < \rho_2 < \rho_3 \dots$ to be the zeros of the function $\tilde{J}_m$ defined in (3.26), that is

$$\tilde{J}_m(\rho_k) = 0. \tag{3.27}$$

We define the functions $\bar{J}_{m,1}, \bar{J}_{m,2}, \bar{J}_{m,3}, \ldots$ on the interval $[0, R]$ via the formula

$$\bar{J}_{m,k}(r) = \frac{\sqrt{2r}\; J_m(2\pi\rho_k r)}{R\, J_{m+1}(2\pi\rho_k R)}, \tag{3.28}$$

for any positive integer k and non-negative integer m. The functions $\bar{J}_{m,1}, \bar{J}_{m,2}, \bar{J}_{m,3}, \ldots$ are orthonormal, that is for any positive integers j and k such that $j \neq k$,

$$\int_0^R \bar{J}_{m,j}(r)\, \bar{J}_{m,k}(r)\, dr = 0 \tag{3.29}$$

and

$$\int_0^R \left(\bar{J}_{m,j}(r)\right)^2 dr = 1. \tag{3.30}$$

The functions $\bar{J}_{m,1}, \bar{J}_{m,2}, \bar{J}_{m,3}, \ldots$ are dense in $L^2[0, R]$; consequently there exist real numbers $\beta_m^1, \beta_m^2, \beta_m^3, \ldots$ such that

$$\sqrt{r}\, f_m(r) = \sum_{k=1}^{\infty} \beta_m^k \bar{J}_{m,k}(r), \tag{3.31}$$

for any real number r such that $0 \leq r \leq R$, where $f_m(r)$ is defined in (3.25). The sum on the right-hand side of equation (3.31) is known as a Fourier-Bessel series. It follows from (3.29) and (3.30) that the coefficient β_m^k in equation (3.31) is the inner product of $\bar{J}_{m,k}(r)$ and $\sqrt{r}\, f_m(r)$, that is,

$$\beta_m^k = \int_0^R \sqrt{r}\, \bar{J}_{m,k}(r) f_m(r)\, dr, \tag{3.32}$$

for any non-negative integer m and positive integer k. The numbers $\beta_m^1, \beta_m^2, \beta_m^3, \ldots$ defined in (3.32) are known as Fourier-Bessel coefficients. It follows from (3.32) and (3.28) that

$$\beta_m^k = \frac{\sqrt{2}}{R\, J_{m+1}(2\pi\rho_k R)} \int_0^R r\, J_m(2\pi\rho_k r) f_m(r)\, dr. \tag{3.33}$$

It follows from (3.33) and the fact that f is zero outside the disk of radius R centered at the origin that the Fourier-Bessel coefficients $\beta_m^1, \beta_m^2, \beta_m^3, \ldots$ are a discretized version of the Fourier-Bessel transform $\gamma_m(\rho)$ defined in equation (3.24), that is,

$$\beta_m^k = \frac{\sqrt{2}}{R\, J_{m+1}(2\pi\rho_k R)} i^m \gamma_m(2\pi\rho_k). \tag{3.34}$$

Remark 3.5 The integrand $r\, J_m(2\pi\rho_k r) f_m(r)$ in equation (3.33) has infinitely many continuous derivatives, provided that the function $f_m(r)$ has infinitely many continuous derivatives. We use the Gaussian quadrature based on the nodes of Legendre polynomials to approximate the integral in (3.33) accurately. Specifically, we use the nodes y_j on the interval $[0, R]$ and corresponding weights w_j, defined in Formula 25.4.30 in [1].

Remark 3.6 To ensure that we sample at the Nyquist rate or higher, we discretize at slightly greater than two nodes per wavelength, using n nodes in the calculation of the Fourier-Bessel coefficients β_m^1, β_m^2, ..., $\beta_m^{n/2-m-C-1}$, $\beta_m^{n/2-m-C}$, where m is the order of the Fourier-Bessel transform. $C = 10$ is sufficient when the precision ε of the computations is 10^{-10} and $n \geq 20$. In fact, C depends weakly on ε; it is easily shown that C must be of the order $\log(1/\varepsilon)$.

To calculate the Fourier-Bessel coefficients, it is necessary to evaluate the integral in (3.33). As described in Remark 3.5, we evaluate the integral in (3.33) by using the Gaussian quadrature with nodes y_j and weights w_j. We truncate the series in equation (3.31) after $n/2-m-C$ terms, where n is the number of nodes y_j used to discretize the integral in equation (3.33), m is the order of the Fourier-Bessel transform, and C is as described in Remark 3.6. The function $\sqrt{r} f_m(r)$ is thus approximated by the function $p : [0, R] \to \mathbb{R}$ given by the formula

$$\sqrt{r} f_m(r) \approx p(r) = \sum_{k=1}^{n/2-m-C} \beta_m^k \, \bar{J}_{m,k}(r). \tag{3.35}$$

We define the Fourier-Bessel series transform $U_m^n : \mathbb{R}^n \to \mathbb{R}^{n/2-m-C}$ of order m and size n via the formula

$$U_m^n \big(f_m(y_1), f_m(y_2), \ldots, f_m(y_{n-1}), f_m(y_n)\big)^\top = (\beta_m^1, \beta_m^2, \ldots, \beta_m^{n/2-m-C-1}, \beta_m^{n/2-m-C})^\top. \tag{3.36}$$

For every integer k such that $1 \leq k \leq n/2 - m - C$, we define the real number γ_m^k to be the approximation of the real number β_m^k given in (3.33) obtained by using the Gaussian quadrature with nodes y_j on the interval $[0, R]$ and corresponding weights w_j. That is,

$$\gamma_m^k = \frac{\sqrt{2}}{R\, J_{m+1}(2\pi\rho_k R)} \sum_{j=1}^{n} w_j y_j J_m(2\pi\rho_k y_j) f_m(y_j) \tag{3.37}$$

or

$$\beta \approx \gamma = U_m^n f_m, \tag{3.38}$$

where

$$U_m^n = \frac{\sqrt{2}}{R} S_m^n T_m^n W_m^n, \tag{3.39}$$

$$\beta = \left(\beta_m^1, \beta_m^2, \ldots, \beta_m^{n/2-m-C-1}, \beta_m^{n/2-m-C}\right)^\top, \tag{3.40}$$

$$\gamma = (\gamma_m^1, \gamma_m^2, \ldots, \gamma_m^{n/2-m-C-1}, \gamma_m^{n/2-m-C})^\top, \tag{3.41}$$

$$f_m = (f_m(y_1), f_m(y_2), \ldots, f_m(y_{n-1}), f_m(y_n))^\top, \tag{3.42}$$

$$S_m^n = \begin{pmatrix} \frac{1}{J_{m+1}(2\pi\rho_1 R)} & & & 0 \\ & \frac{1}{J_{m+1}(2\pi\rho_2 R)} & & \\ & & \ddots & \\ 0 & & & \frac{1}{J_{m+1}(2\pi\rho_{n/2-m-C} R)} \end{pmatrix}, \tag{3.43}$$

$$T_m^n = \begin{pmatrix} J_m(2\pi\rho_1 y_1) & \cdots & J_m(2\pi\rho_1 y_n) \\ \vdots & \ddots & \vdots \\ J_m(2\pi\rho_{n/2-m-C}\, y_1) & \cdots & J_m(2\pi\rho_{n/2-m-C}\, y_n) \end{pmatrix}, \tag{3.44}$$

and

$$W_n = \begin{pmatrix} w_1 y_1 & & & 0 \\ & w_2 y_2 & & \\ & & \ddots & \\ 0 & & & w_n y_n \end{pmatrix}. \tag{3.45}$$

The algorithm described in Section 5 accelerates the application of the matrix T_n^m in (3.44) and thus accelerates the evaluation of the Fourier-Bessel series transform.

We define the inverse Fourier-Bessel series transform $Q_m^n : \mathbb{R}^{n/2-m-C} \to \mathbb{R}^n$ of order m and size n via the formula

$$Q_m^n(\beta_m^1, \beta_m^2, \dots, \beta_m^{n/2-m-C-1}, \beta_m^{n/2-m-C})^\top = \left(\frac{p(y_1)}{\sqrt{y_1}}, \frac{p(y_2)}{\sqrt{y_2}}, \dots, \frac{p(y_{n-1})}{\sqrt{y_{n-1}}}, \frac{p(y_n)}{\sqrt{y_n}} \right)^\top, \tag{3.46}$$

where $\beta_m^1, \beta_m^2, \dots, \beta_m^{n/2-m-C-1}, \beta_m^{n/2-m-C}$ and p satisfy equation (3.35), and $y_1, y_2, \dots, y_{n-1}, y_n$ are real numbers. It follows from (3.35) and (3.46) that

$$f_m \approx Q_m^n \beta, \tag{3.47}$$

where f_m is the vector defined in (3.42), the vector β is defined in equation (3.40), and Q_m^n is the inverse Fourier-Bessel series transform defined in (3.46). It follows from (3.28) and (3.35) that, in matrix notation, equation (3.46) becomes

$$Q_m^n \beta = \frac{\sqrt{2}}{R} T_m^{n\,\top} S_m^n \beta, \tag{3.48}$$

where β is the vector defined in (3.40), the matrix S_m^n is defined in (3.43), and T_m^n is the matrix defined in (3.44).

Remark 3.7 The matrix Q_m^n defined in (3.46) is the right inverse of U_m^n defined in (3.39), that is

$$U_m^n Q_m^n = \mathbf{1}, \tag{3.49}$$

where $\mathbf{1}$ is the $(n/2-m-C) \times (n/2-m-C)$ identity matrix. Indeed, suppose that β_m^1, β_m^2, $\dots$, $\beta_m^{n/2-m-C-1}$, $\beta_m^{n/2-m-C}$ are real numbers and that f_m is the real function defined via the formula

$$\sqrt{y}\, f_m(y) = \sum_{k=1}^{n/2-m-C} \beta_m^k \bar{J}_{m,k}(y). \tag{3.50}$$

It follows from (3.50) and the definition of the function p in (3.35) that

$$\sqrt{y}\, f_m(y) = p(y). \tag{3.51}$$

It follows from (3.46) and (3.51) that

$$Q_m^n(\beta_m^1, \beta_m^2, \dots, \beta_m^{n-1}, \beta_m^n)^\top = (f_m(y_1), f_m(y_2), \dots, f_m(y_{n-1}), f_m(y_n))^\top. \tag{3.52}$$

Finally, (3.49) follows from (3.36) and (3.52).

4 Analytical apparatus

The algorithm of this paper relies on the observation that for certain $n \times n$ matrices the rank to precision ε of any $p \times q$ contiguous submatrix is proportional to pq/n. The present section contains a proof of this fact for the the Fourier-Bessel transform (see Theorem 4.3, below). The principal tool used in the proof of Theorem 4.3 is Lemma 4.1.

The following lemma states that the rank of the normalized Fourier transform with kernel $e^{i\gamma\xi\tau/4}$ is bounded by a constant times γ, at any fixed precision ε. This lemma can be found (in a slightly different form) in [10].

Lemma 4.1 *Suppose that δ, ε, and γ are positive real numbers such that*

$$0 < \varepsilon < 1. \tag{4.1}$$

Suppose further that the operator $F : L^2[-1,1] \to L^2[-1,1]$ is given by the formula

$$(Fh)(\tau) = \int_{-1}^{1} e^{i\gamma\xi\tau/4} h(\xi)\, d\xi. \tag{4.2}$$

Then, F has rank to precision ε at most

$$N = (1+\delta)\left(\frac{\gamma}{2\pi} + \frac{E}{\delta}\right) + 3, \tag{4.3}$$

where

$$E = 2\sqrt{2\ln\left(\frac{4}{\varepsilon}\right)\ln\left(\frac{6\sqrt{\frac{1}{\sqrt{\delta}}+\sqrt{\delta}}}{\varepsilon}\right)}. \tag{4.4}$$

Remark 4.2 If the Fourier transform with kernel e^{ixt} is restricted to a rectangle in the (x,t) plane then its rank is bounded by a constant times the area of the rectangle.

Indeed, suppose that the operator $A : L^2[a,b] \to L^2[u,v]$ is given by the formula

$$(Ag)(t) = \int_a^b e^{ixt} g(x)\, dx. \tag{4.5}$$

Defining

$$\xi = \frac{2(x-a)}{b-a} - 1, \tag{4.6}$$

$$\tau = \frac{2(t-u)}{v-u} - 1, \tag{4.7}$$

and

$$h(\xi) = g(x) \tag{4.8}$$

yields that the operator A has the same rank as the operator $F : L^2[-1,1] \to L^2[-1,1]$ defined in (4.2), with kernel $e^{i\gamma\xi\tau/4}$, where $\gamma = (b-a)(v-u)$ is the area of the rectangle $[a,b] \times [u,v]$ in the (x,t) plane. It then follows from Lemma 4.1 that the operator A defined in (4.5) has rank at most N, where N is defined in (4.3).

The following theorem states that if the Fourier-Bessel transform with kernel $x\,J_m(xt)$ (see equation (3.24)) is restricted to a rectangle in the (x,t) plane, its rank at any fixed precision ε is bounded by a constant times the area of the rectangle.

Theorem 4.3 *Suppose that a, b, R, u, v, δ, and ε are real numbers such that R, δ, $\varepsilon > 0$ and $u < v$, with $0 < a < b < R$. Suppose further that m is a non-negative integer and $U_m : L^2[a,b] \to L^2[u,v]$ is the integral operator given by the formula*

$$(U_m f)(t) = (-i)^m \int_a^b x\, J_m(xt)\, f(x)\, dx. \tag{4.9}$$

Then, the rank of U_m to precision $R^2\varepsilon/\pi$ is at most

$$M = 2(1+\delta)\left(\frac{(b-a)(v-u)}{2\pi} + \frac{E}{\delta}\right) + 6, \tag{4.10}$$

where E is the real number given by the formula

$$E = 2\sqrt{2\ln\left(\frac{4}{\varepsilon}\right)\ln\left(\frac{6\sqrt{\frac{1}{\sqrt{\delta}}+\sqrt{\delta}}}{\varepsilon}\right)}. \tag{4.11}$$

Proof.

We prove the theorem in the case where

$$m = 2k, \tag{4.12}$$

for some non-negative integer k. The proof when $m = 2k+1$, for some non-negative integer k is similar.

We start by combining (4.9) and (3.14) to obtain

$$(U_{2k} f)(t) = \frac{2}{\pi} \int_a^b x f(x) \left(\int_0^1 \frac{T_{2k}(s)\cos(xts)}{\sqrt{1-s^2}}\, ds, \right) dx \tag{4.13}$$

or

$$(U_{2k} f)(t) = \frac{1}{\pi} \int_a^b x\, f(x) \left(\int_0^1 \frac{T_{2k}(s)(e^{ixts} + e^{-ixts})}{\sqrt{1-s^2}}\, ds \right) dx. \tag{4.14}$$

Introducing the notation

$$x(\xi) = \frac{\alpha}{2}(\xi+1) + a, \tag{4.15}$$

with

$$\alpha = b - a, \tag{4.16}$$

we rewrite (4.14) in the form

$$(U_{2k} f)(t) = \frac{\alpha}{2\pi} \int_{-1}^1 x(\xi)\, h(\xi) \left(\int_0^1 \frac{T_{2k}(s)\, c(\xi,t,s)}{\sqrt{1-s^2}}\, ds \right) d\xi, \tag{4.17}$$

where

$$c(\xi,t,s) = e^{itsa\xi/2+its\alpha/2+itsa} + e^{-its\alpha\xi/2-its\alpha/2-itsa} \tag{4.18}$$

and

$$h(\xi) = f(x). \tag{4.19}$$

Now, introducing the notation

$$\beta = v - u, \tag{4.20}$$

$$\gamma = \alpha\beta, \tag{4.21}$$

$$\tau = \frac{2(t-u)}{\beta} - 1, \tag{4.22}$$

$$\eta(\tau,s,\xi) = e^{ia\beta s\tau/2} e^{i\gamma s\tau/4} e^{iasu} e^{i\alpha su/2} e^{i\gamma s/4} e^{i\beta as/2} e^{i\alpha su\xi/2} e^{i\gamma s\xi/4}, \tag{4.23}$$

we observe that the function c defined in (4.18) assumes the form

$$c(\xi,t,s) = e^{i\gamma\tau s\xi/4}\,\eta(\tau,s,\xi)\; + e^{-i\gamma\tau s\xi/4}\,\eta(\tau,s,\xi)^{-1} \tag{4.24}$$

and that

$$(V_{2k}h)(\tau) = (U_{2k}f)(t), \tag{4.25}$$

where

$$(V_{2k}h)(\tau) = \tag{4.26}$$
$$\frac{\alpha}{2\pi}\int_{-1}^{1} x(\xi)h(\xi)\left(\int_0^1 \frac{T_{2k}(s)\big(\,e^{i\gamma\tau s\xi/4}\eta(\tau,s,\xi) + e^{-i\gamma\tau s\xi/4}\eta(\tau,s,\xi)^{-1}\big)}{\sqrt{1-s^2}}\,ds\right)d\xi.$$

Changing the order of integration in (4.26), we rewrite it in the form

$$(V_{2k}h)(\tau) = (X_{2k}h)(\tau) + (Y_{2k}h)(\tau), \tag{4.27}$$

with

$$(X_{2k}h)(\tau) = \frac{\alpha}{2\pi}\int_0^1 \frac{T_{2k}(s)}{\sqrt{1-s^2}}\int_{-1}^1 e^{i\gamma\tau s\xi/4}h(\xi)x(\xi)\eta(\tau,s,\xi)d\xi\,ds \tag{4.28}$$

and

$$(Y_{2k}h)(\tau) = \frac{\alpha}{2\pi}\int_0^1 \frac{T_{2k}(s)}{\sqrt{1-s^2}}\int_{-1}^1 e^{-i\gamma\tau s\xi/4}h(\xi)x(\xi)\eta(\tau,s,\xi)^{-1}\,d\xi\,ds. \tag{4.29}$$

Finally, defining the operators A_{2k}, B_{2k}, and C_{2k} via the formulas

$$(A_{2k}g)(\tau) = \int_0^1 \frac{T_{2k}(s)}{\sqrt{1-s^2}}g(s,\tau)\,ds, \tag{4.30}$$

$$(B_{2k}h)(s,\tau) = \frac{\alpha}{2\pi}\int_{-1}^1 e^{i\gamma\tau s\xi/4}h(\xi)x(\xi)\eta(\tau,s,\xi)\,d\xi, \tag{4.31}$$

and

$$(C_{2k}h)(s,\tau) = \frac{\alpha}{2\pi}\int_{-1}^1 e^{-i\gamma\tau s\xi/4}h(\xi)x(\xi)\eta(\tau,s,\xi)^{-1}\,d\xi, \tag{4.32}$$

we observe that X_{2k} is the composition of A_{2k} with B_{2k}, that is,

$$X_{2k} = A_{2k} \circ B_{2k} \tag{4.33}$$

and that Y_{2k} is composition of A_{2k} with C_{2k}, that is,

$$Y_{2k} = A_{2k} \circ C_{2k}. \tag{4.34}$$

Due to Lemma 4.1 and the facts that $0 \le x(\xi) \le R$, $|\eta(\tau, s, \xi)| = 1$, and $0 \le \alpha \le R$ the ranks of the operators B_{2k} and C_{2k} are bounded by N to precision $R^2\varepsilon/(2\pi)$, where N is defined in equation (4.3). Therefore (4.27), (4.33), (4.34), and (4.30) yield that the rank of V_{2k} is bounded by $2N$ to precision $R^2\varepsilon/\pi$.

□

5 The butterfly algorithm

This section contains a description of an algorithm for the application of $n \times n$ matrices which have the property that any $p \times q$ contiguous submatrix has rank to precision ε at most a constant times pq/n.

5.1 Informal description of the algorithm

We now illustrate the algorithm in the particularly simple case of the Fourier transform of size $n = 2^m$. We define the function f via the formula

$$f(x) = \sum_{k=1}^{n} \alpha_k e^{i\omega_k x}, \tag{5.1}$$

where α_1, α_2, …, α_{n-1}, $\alpha_n \in \mathbb{C}$ and the frequencies ω_1, ω_2, …, ω_{n-1}, $\omega_n \in [0, 2\pi]$ are equispaced. Suppose that we would like to evaluate the function f at n equispaced nodes x_1, x_2, …, x_{n-1}, $x_n \in [a, b]$, that is we would like to apply to the vector $(\alpha_1, \alpha_2, \ldots, \alpha_{n-1}, \alpha_n)^\top$ the $n \times n$ matrix S defined via the formula

$$S = \begin{pmatrix} e^{i\omega_1 x_1} & e^{i\omega_2 x_1} & \cdots & e^{i\omega_{n-1} x_1} & e^{i\omega_n x_1} \\ e^{i\omega_1 x_2} & e^{i\omega_2 x_2} & \cdots & e^{i\omega_{n-1} x_2} & e^{i\omega_n x_2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ e^{i\omega_1 x_{n-1}} & e^{i\omega_2 x_{n-1}} & \cdots & e^{i\omega_{n-1} x_{n-1}} & e^{i\omega_n x_{n-1}} \\ e^{i\omega_1 x_n} & e^{i\omega_2 x_n} & \cdots & e^{i\omega_{n-1} x_n} & e^{i\omega_n x_n} \end{pmatrix}. \tag{5.2}$$

For any pair of subintervals $\Omega \subset [0, 2\pi]$ and $X \subset [a, b]$ we define $S(\Omega, X)$ to be the submatrix of S given by the intersection of those columns corresponding to $\omega_k \in \Omega$ and those rows corresponding to $x_k \in X$.

In the precomputation stage of the present algorithm, we compress the matrix S. This allows us, in the application stage, to evaluate f defined in (5.1) at the nodes x_1, x_2, …, x_{n-1}, x_n in $\mathcal{O}(n \log(n))$ operations.

PRECOMPUTATION

Level 0 On level 0, we split the interval $[0, 2\pi]$ into 2^L subintervals of length $2\pi/(2^L)$. Specifically, we define the interval $\Omega_{0,k}$ via the formula

$$\Omega_{0,k} = [2\pi\,(k-1)\,2^{-L}, 2\pi\,k\,2^{-L}], \tag{5.3}$$

for every integer k such that $1 \leq k \leq 2^L$. We observe that, due to Remark 4.2, if

$$L = \log_2(b-a) \tag{5.4}$$

then the matrices $S(\Omega_{0,k}, [a, b])$ have constant rank; we will be referring to this rank as r, so that

$$r = \mathcal{O}(1). \tag{5.5}$$

The matrices $S(\Omega_{0,k}, [a, b])$ are illustrated in Figure 5. We compute an interpolative decompostion (see Lemma 3.1) of every matrix $S(\Omega_{0,k}, [a, b])$. That is, for every matrix $S(\Omega_{0,k}, [a, b])$, we compute a column skeleton matrix $B_{0,k}$ which contains $\sim r$ columns of $S(\Omega_{0,k}, [a, b])$ and an interpolation matrix $P_{0,k}$ which contains coefficients expressing every column of $S(\Omega_{0,k}, [a, b])$ as a linear combination of the columns of $B_{0,k}$, that is,

$$S(\Omega_{0,k}, [a, b]) = B_{0,k}P_{0,k}, \tag{5.6}$$

for $1 \leq k \leq 2^L$.

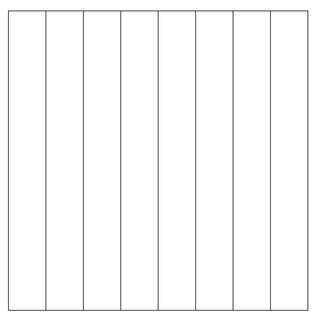

Figure 5: Level 0

Remark 5.1 The number of levels L defined in (5.4) is of the order $\log_2(n)$, where n is the number of nodes. Indeed, the number of nodes n is proportional to the length $b - a$ of the interval $[a, b]$.

Level 1 On Level 1, we split the interval $[0, 2\pi]$ into 2^{L-1} subintervals of length $2\pi/(2^{L-1})$. Each of the subintervals on Level 1 is obtained by merging two neighboring intervals on Level 0. Specifically, we define the interval $\Omega_{1,k}$ via the formula

$$\Omega_{1,k} = [2\pi\,(k-1)\,2^{-(L-1)}, 2\pi\,k\,2^{-(L-1)}], \tag{5.7}$$

for every integer k such that $1 \leq k \leq 2^{L-1}$. Splitting the interval $[a, b]$ in two, we denote by $X_{2,1}$ the first half of $[a, b]$ and by $X_{2,2}$ the second half of $[a, b]$. As described in Observation 5.2 the matrices $S(\Omega_{1,k}, X_{1,j})$ all have rank $\sim r$. The matrices $S(\Omega_{1,k}, X_{1,j})$ are illustrated in Figure 6. We compute an interpolative decomposition (see Lemma 3.1) of each matrix $S(\Omega_{1,k}, X_{1,j})$ on Level 1. Specifically, for each matrix $S(\Omega_{1,k}, X_{1,j})$, we compute a column skeleton matrix $B_{1,j,k}$ which contains $\sim r$ columns of $S(\Omega_{1,k}, X_{1,j})$; together, these columns span the range of $S(\Omega_{1,k}, X_{1,j})$. Further, we compute an interpolation matrix $P_{1,j,k}$ containing coefficients which express halves of columns in skeleton matrices on Level 0 as linear combinations of columns in $B_{1,j,k}$. Specifically,

$$\begin{pmatrix} B_{0,2k-1} & B_{0,2k} \end{pmatrix}^{+} = B_{1,1,k} P_{1,1,k} \tag{5.8}$$

and

$$\begin{pmatrix} B_{0,2k-1} & B_{0,2k} \end{pmatrix}^{-} = B_{1,2,k} P_{1,2,k}, \tag{5.9}$$

for $1 \leq k \leq 2^{L-1}$, where for any matrix X, the top half of X is denoted by X^{+} and the bottom half of X is denoted by X^{-}.

The interpolation matrices $P_{1,j,k}$ on Level 1 all have size $\sim (r \times 2r)$ (see Remark 5.3)

Observation 5.2 All submatrices $S(\Omega_{l,k}, X_{l,j})$ on all levels have approximately the same rank, namely $\sim r$. Indeed, on each Level l such that $1 \leq l \leq L$, we consider subintervals $\Omega_{l,k} \subset [0, 2\pi]$ obtained by combining two neighboring subintervals on the previous level $l-1$. Moreover, on each Level l such that $1 \leq l \leq L$, we consider subintervals $X_{l,j} \subset [a, b]$ obtained by splitting a subinterval on the previous level in half. Therefore, all rectangles on all levels have the same area. Definitions (5.3) and (5.4) yield that the rectangles $\Omega_{0,k} \times [a, b]$ on Level 0 have area 2π. Due to Remark 4.2, then, all the matrices $S(\Omega_{l,k}, X_{l,j})$ on all levels have rank $\mathcal{O}(1)$. However, in practice, not all the matrices $S(\Omega_{l,k}, X_{j,k})$ have exactly the same rank; their ranks are similar and are denoted by $\sim r$.

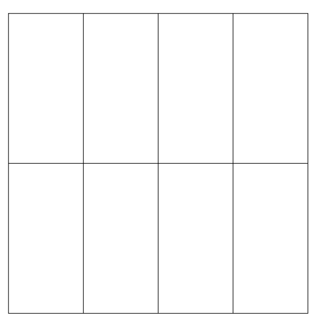

Figure 6: Level 1

Remark 5.3 The interpolation matrices on Level l (for $1 \leq l \leq L$) all have size $\sim (r \times 2r)$. Indeed all submatrices on all levels have the same rank $\sim r$ as the submatrices on Level 0 (see Observation 5.2). Each submatrix $S(\Omega_{l,k}, X_{l,j})$ on Level l is either the top half or the bottom half of two adjacent submatrices on Level $l-1$; we refer to these submatrices on

Level $l-1$ as the "parents" of $S(\Omega_{l,k}, X_{l,j})$. For each of the $\sim 2r$ columns in the skeleton matrices corresponding to the parents of $S(\Omega_{l,k}, X_{l,k})$ the interpolation matrix $P_{l,j,k}$ on Level l contains coefficients expressing that column as a linear combination of the columns in the skeleton matrix $B_{l,j,k}$.

Level 2 On Level 2, we split the interval $[0, 2\pi]$ into 2^{L-2} subintervals of length $2\pi/2^{L-2}$. Each of the subintervals on Level 2 is obtained by merging two neighboring intervals on Level 1. Specifically, we define the interval $\Omega_{2,k}$ via the formula

$$\Omega_{2,k} = [2\pi\,(k-1)\,2^{-(L-2)}, 2\pi\,k\,2^{-(L-2)}], \tag{5.10}$$

for every integer k such that $1 \leq k \leq 2^{L-2}$. Splitting the interval $[a, b]$ in four, we define $X_{2,1}$ to be the first quarter of $[a, b]$, $X_{2,2}$ to be the second quarter of $[a, b]$, $X_{2,3}$ to be the third quarter of $[a, b]$, and $X_{2,4}$ to be the fourth quarter of $[a, b]$. As described in Observation 5.2, the matrices $S(\Omega_{2,k}, X_{2,j})$ all have rank $\sim r$. The matrices $S(\Omega_{2,k}, X_{2,j})$ are illustrated in Figure 7. We compute an interpolative decomposition (see Lemma 3.1) of each matrix $S(\Omega_{2,k}, X_{2,j})$ on Level 2. Specifically, for each matrix $S(\Omega_{2,k}, X_{2,j})$, we compute a column skeleton matrix $B_{2,j,k}$ which contains $\sim r$ columns of $S(\Omega_{2,k}, X_{2,j})$; together, these columns span the range of $S(\Omega_{2,k}, X_{2,j})$. Further, we compute an interpolation matrix $P_{2,j,k}$ containing coefficients which express halves of columns in skeleton matrices on Level 1 as linear combinations of columns of $B_{2,j,k}$. Specifically, for $1 \leq k \leq 2^{L-2}$,

$$\begin{pmatrix} B_{1,\lfloor (j+1)/2 \rfloor, 2k-1} & B_{1,\lfloor (j+1)/2 \rfloor, 2k} \end{pmatrix}^{+} = B_{2,j,k} P_{2,j,k} \tag{5.11}$$

when $j = 1$ or $j = 3$, and

$$\begin{pmatrix} B_{1,\lfloor (j+1)/2 \rfloor, 2k-1} & B_{1,\lfloor (j+1)/2 \rfloor, 2k} \end{pmatrix}^{-} = B_{2,j,k} P_{2,j,k} \tag{5.12}$$

when $j = 2$ or $j = 4$, where for any matrix X, the top half of X is denoted by X^{+} and the bottom half of X is denoted by X^{-}. The interpolation matrices $P_{2,j,k}$ on Level 2 all have size $\sim (r \times 2r)$ (see Remark 5.3).

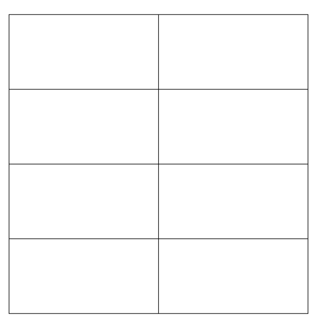

Figure 7: Level 2

Level L Continuing the process, we arrive finally at Level L. On Level L, we split the interval $[a, b]$ into 2^{L} subintervals of length $(b-a)/2^{L}$. Specifically, we define the interval $X_{L,j}$ via the formula

$$X_{L,j} = [a + (b-a)\,2^{-L}\,(j-1), a + (b-a)\,2^{-L}\,j], \tag{5.13}$$

for every integer j such that $1 \leq j \leq 2^L$. As described in Observation 5.2, the matrices $S([0,2\pi], X_{L,j})$ all have rank $\sim r$. The matrices $S([0,2\pi], X_{L,j})$ are illustrated in Figure 8. We compute an interpolative decomposition (see Lemma 3.1) of each matrix $S([0,2\pi], X_{L,j})$ on Level L. Specifically, for each matrix $S([0,2\pi], X_{L,j})$, we compute a column skeleton matrix $B_{L,j}$ which contains $\sim r$ columns of $S([0,2\pi], X_{L,j})$; together, these columns span the range of $S([0,2\pi, X_{L,j})$. Further, we compute an interpolation matrix $P_{L,j}$ containing coefficients which express halves of columns in skeleton matrices on Level $L-1$ as linear combinations of columns of $B_{L,j}$. Specifically, for $1 \leq j \leq 2^L$,

$$\begin{pmatrix} B_{L-1,\lfloor (j+1)/2 \rfloor,1} & B_{L-1,\lfloor (j+1)/2 \rfloor,2} \end{pmatrix}^{+} = B_{L,j} P_{L,j} \tag{5.14}$$

when j is odd, and

$$\begin{pmatrix} B_{L-1,\lfloor (j+1)/2 \rfloor,1} & B_{L-1,\lfloor (j+1)/2 \rfloor,2} \end{pmatrix}^{-} = B_{L,j} P_{L,j} \tag{5.15}$$

when j is even, where for any matrix X, the top half of X is denoted by X^+ and the bottom half of X is denoted by X^-. Because the matrices $S([0,2\pi], X_{L,j})$ on Level L and the matrices $S(\Omega_{L-1,k}, X_{L-1,j})$ on Level $L-1$ all have rank $\sim r$, the interpolation matrices $P_{L,j}$ on Level L all have size $\sim (r \times 2r)$ (see Remark 5.3).

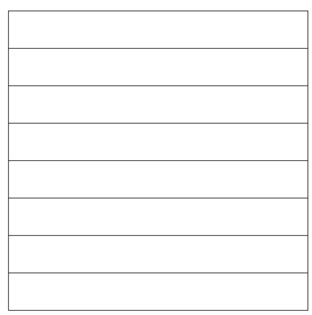

Figure 8: Level L

APPLICATION

The inputs to this stage of the algorithm are n coefficients $\alpha_1, \alpha_2, \ldots, \alpha_{n-1}, \alpha_n$ and the results of the precomputation described above. We would like to apply the matrix S defined in (5.2) to the vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{n-1}, \alpha_n)^\top$. That is, we would like to evaluate the linear combination of the columns of S with coefficients $\alpha_1, \alpha_2, \ldots, \alpha_{n-1}, \alpha_n$. This is equivalent to evaluating the function f defined in (5.1) at each of the nodes $x_1, x_2, \ldots, x_{n-1}, x_n$. For any subinterval $\Omega \subset [0,2\pi]$, we define $\alpha(\Omega)$ to be the entries of α corresponding to the frequencies $\omega_k \in \Omega$.

Step 0 For each $k = 1, 2, \ldots, 2^L - 1, 2^L$, we apply the interpolation matrix $P_{0,k}$, defined in (5.6), to the vector $\alpha(\Omega_{0,k})$ obtaining the vector $\beta_{0,k}$. The vector $\beta_{0,k}$ consists of $\sim r$ coefficients representing the effect at all the nodes $x_1, x_2, \ldots, x_{n-1}, x_n \in [a,b]$ of the $n/2^L$ frequencies $\alpha(\Omega_{0,k})$.

Step 1 Applying the interpolation matrices $P_{1,j,k}$ defined in (5.8) and (5.9), we calculate the vectors $\beta_{1,j,k}$ via the formula

$$\beta_{1,j,k} = P_{1,j,k} \begin{pmatrix} \beta_{0,2k-1} \\ \beta_{0,2k} \end{pmatrix}, \tag{5.16}$$

for each pair of integers j, k such that $1 \le k \le 2^{L-1}$ and $1 \le j \le 2$, where the vectors $\beta_{0,2k}$ and $\beta_{0,2k-1}$ were computed in Step 0. The vector $\beta_{1,j,k}$ consists of $\sim r$ coefficients representing the effect at the $n/2$ nodes $x_m \in X_{1,j}$ of the $n/2^{L-1}$ frequencies $\alpha(\Omega_{1,k})$.

Step 2 Applying the interpolation matrices $P_{2,j,k}$ defined in (5.11) and (5.12), we calculate the vectors $\beta_{2,j,k}$ via the formula

$$\beta_{2,j,k} = P_{2,j,k} \begin{pmatrix} \beta_{1,\lfloor (j+1)/2 \rfloor,2k-1} \\ \beta_{1,\lfloor (j+1)/2 \rfloor,2k} \end{pmatrix}, \tag{5.17}$$

for each pair of integers j and k such that $1 \le k \le 2^{L-2}$ and $1 \le j \le 4$, where the vectors $\beta_{1,\lfloor (j+1)/2 \rfloor,2k-1}$ and $\beta_{1,\lfloor (j+1)/2 \rfloor,2k}$ were computed in Step 1. The vector $\beta_{2,j,k}$ consisits of $\sim r$ coefficients representing the effect at the $n/4$ nodes $x_m \in X_{2,j}$ of the $n/2^{L-2}$ frequencies $\alpha(\Omega_{2,k})$.

Step L Continuing the process, we arrive at Step L. Applying the interpolation matrices $P_{L,j}$ defined in (5.14) and (5.15), we calculate the vectors $\beta_{L,j}$ via the formula

$$\beta_{L,j} = P_{L,j} \begin{pmatrix} \beta_{L-1,\lfloor (j+1)/2 \rfloor,1} \\ \beta_{L-1,\lfloor (j+1)/2 \rfloor,2} \end{pmatrix}, \tag{5.18}$$

for every integer j such that $1 \le j \le 2^L$, where the vectors $\beta_{L-1,\lfloor (j+1)/2 \rfloor,1}$ and $\beta_{L-1,\lfloor (j+1)/2 \rfloor,2}$ were calculated in Step $L-1$. Finally, we apply the matrix $B_{L,j}$ to the vector $\beta_{L,j}$ obtaining the product

$$B_{L,j}\,\beta_{L,j} = S([0,2\pi], X_{L,j})\alpha. \tag{5.19}$$

Equation (5.19) states that the vector $\beta_{L,j}$ represents the effect at the $n/2^L$ nodes $x_m \in X_{L,j}$ of all n frequencies $\alpha_1, \alpha_2, \dots, \alpha_{n-1}, \alpha_n \in [0, 2\pi]$. In other words, linearly combining the $\sim r$ columns of $B_{L,j}$ with coefficients $\beta_{L,j}$ yields the same vector of length $n/2^L$ as linearly combining the n columns of $S([0,2\pi], X_{L,j})$ with coefficients given by the entries of the vector α. Thus, $B_{L,j}\beta_{L,j}$ is the j^{th} block of $n/2^L$ entries in the vector $S\alpha$.

Remark 5.4 The algorithm of the present paper exhibits the same performance when applied to the transposed matrix $S^\top$, since all relevant submatrices of $S^\top$ satisfy the requisite bound on their ranks (see Remark 4.2).

Remark 5.5 We have described the algorithm in the illustrative case of the equispaced Fourier transform of size $n = 2^m$. The description is similar for other sizes of matrices and other transorms; it is therefore omitted.

5.2 CPU requirements

5.2.1 Precomputation

On Level 0, we compute the 2^L interpolative decompositions of the $n \times (n/2^L)$ matrices of rank $\sim r$ on the left hand side of (5.6); this takes a total of $\mathcal{O}(rn^2)$ operations. It then follows from (5.5) that we require $\mathcal{O}(n^2)$ operations on Level 0. On each Level l, for $1 \le l \le L$, we compute interpolative decompositions of the $(n/2^l)\times \sim 2r$ matrices

$$\begin{pmatrix} B_{l,\lfloor (j+1)/2 \rfloor, 2k-1} & B_{l,\lfloor (j+1)/2 \rfloor, 2k} \end{pmatrix}^{+} \tag{5.20}$$

and

$$\begin{pmatrix} B_{l,\lfloor (j+1)/2 \rfloor, 2k-1} & B_{l,\lfloor (j+1)/2 \rfloor, 2k} \end{pmatrix}^{-}; \tag{5.21}$$

these matrices have rank r. There are 2^L such matrices on each level. In total this requires of the order

$$2^L \sum_{l=0}^{L} \frac{2r^2 n}{2^l} \le 2^{L+2} r^2 n \tag{5.22}$$

operations. It follows from (5.5) and Remark 5.1 that $2^{L+2}r^2n = \mathcal{O}(n^2)$; the precomputation therefore takes $\mathcal{O}(n^2)$ operations.

5.2.2 Application

In Step 0, for each integer k such that $1 \le k \le 2^L$, we calculate 2^L vectors $\beta_{0,k}$ of length $\sim r$ by applying the matrix $P_{0,k}$ of size $\sim r \times (n/2^L)$ to the vector $\alpha(\Omega_{0,k})$ of length $n/2^L$. For integers l, k, and j such that $1 \le l \le L$ and $1 \le k \le 2^{L-l}$, with $1 \le j \le 2^l$, we compute the vector $\beta_{l,j,k}$ of length $\sim r$ by applying the $\sim (r \times 2r)$ matrix $P_{l,j,k}$ to a vector of length $\sim 2r$. Finally, on level L, for each integer j such that $1 \le j \le 2^L$, we apply the column skeleton matrix $B_{L,j}$ (see (5.19)) having size $(n/2^L)\times \sim r$ to the vector $\beta_{L,j}$ of length $\sim r$. In total, the time taken to apply the matrix S to an arbitrary vector is $\mathcal{O}(rn + r^2 L 2^L)$. Combining Remark 5.1 and (5.5) then yields that we require $\mathcal{O}(rn + r^2L2^L) = \mathcal{O}(n\log(n))$ operations to apply the matrix S to an arbitrary vector α.

5.3 Memory requirements

5.3.1 Precomputation

On Level 0, we store 2^L interpolation matrices each having size $\sim r \times (n/2^L)$. On each Level l, for $1 \le l \le L$, we store 2^L interpolation matrices, each having size $\sim (r \times 2r)$ (see Remark 5.3). On Level L, we store an additional 2^L column skeleton matrices, each having size $(n/2^L)\times \sim r$. The total memory requirement for the precomputation is therefore $\mathcal{O}(rn + L2^Lr^2)$. Combining Remark 5.1 and (5.5) then yields that the total memory requirement for the precomputation is $\mathcal{O}(rn + L2^Lr^2) = \mathcal{O}(n\log(n))$.

5.3.2 Application

During the application stage of the present algorithm, the interpolation matrices $P_{l,j,k}$ (for $0 \le l \le L$) and the column skeleton matrices $B_{L,j}$ on Level L must be kept in memory; this requires $\mathcal{O}(n\log(n))$ memory, as described in Section 5.3.1. In addition, in Step 0, we store 2^L vectors $\beta_{0,k}$ of length $\sim r$. On each Level l such that $1 \le l \le l$, we store 2^L vectors $\beta_{l,j,k}$ (see (5.16), (5.17), and (5.18)) of length $\sim r$. This requires $\mathcal{O}(rL2^L)$ memory. Combining (5.5) with Remark 5.1 then yields that the memory requirement for the application stage is $\mathcal{O}(n\log(n))$.

5.4 Detailed description of the algorithm

This section contains a detailed description of the algorithm that was described informally in Section 5.1. Given an integer $n = 2^m$ and an $n \times n$ matrix S, such that any $p \times q$ contiguous submatrix of S has rank bounded by a constant times pq/n, we compute interpolative decompositions of submatrices of S. We then apply S to an arbitrary vector α rapidly, yielding $f = S\alpha$.

In this section we denote by $\alpha_{L,k}$ the k^{th} block of $n/2^L$ entries in α. Similarly, we denote by $f_{L,k}$ the k^{th} block of $n/2^L$ entries in f.

Initialization Step

Choose principal parameters and create dyadic hierarchy

1. Choose a positive real number ε. All interpolative decompositions in this algorithm are computed to precision ε.

2. Choose $C_{\max}$, the number of columns in each submatrix on Level 0.

   Comment [In what follows, we assume that the value of $C_{\max}$ chosen above is a positive integer power of 2. If this is not the case, the algorithm is similar and its description is therefore omitted.]

3. Choose the number of levels L in the hierarchy described in Section 5.1 according to the formula $L = \log_2(n/C_{\max})$.

Comment [Create the dyadic hierarchy of subblocks of the matrix S. On each of the $L+1$ levels of the hierarchy, there are 2^L submatrices. Retain the structure created for use in precomputation.]
do $l = 0, 1, \ldots, L-1, L$
    do $j = 1, 2, \ldots, 2^l - 1, 2^l$
        do $k = 1, 2, \ldots, 2^{L-l} - 1, 2^{L-l}$
            Define $S(\Omega_{l,k}, X_{l,j})$ to be the submatrix consisting of rows $(j-1)2^{m-l}+1$ through $j2^{m-l}$ of S and columns $(k-1)2^{m-L+l}+1$ through $k2^{m-L+l}$ of S.
        end do
    end do
end do

Precomputation Step

Precomputation

Comment [In this stage, all submatrices $S(\Omega_{l,k}, X_{l,j})$ of S are compressed using the interpolative decomposition described in Section 3.1.2.]

do $l = 0, 1, \ldots, L-1, L$

  do $j = 1, 2, \ldots, 2^l - 1, 2^l$

    do $k = 1, 2, \ldots, 2^{L-l} - 1, 2^{L-l}$

      if $l = 0$ **then**

1. Compute the interpolative decomposition

$$S(\Omega_{l,k}, X_{l,j}) = B_{0,k}\, P_{0,k}. \tag{5.23}$$

2. Store $P_{0,k}$.

      if ($l > 0$ **and** j is odd) **then**

1. Compute the interpolative decomposition

$$\left(B_{l-1,\lfloor\frac{j+1}{2}\rfloor,2k-1} B_{l-1,\lfloor\frac{j+1}{2}\rfloor,2k}\right)^{+} = B_{l,j,k}\, P_{l,j,k}. \tag{5.24}$$

2. Store $P_{l,j,k}$.
3. **if** $l = L$ **then** store $B_{l,j,k}$.

      if ($l > 0$ **and** j is even) **then**

1. Compute the interpolative decomposition

$$\left(B_{l-1,\lfloor\frac{j+1}{2}\rfloor,2k-1} B_{l-1,\lfloor\frac{j+1}{2}\rfloor,2k}\right)^{-} = B_{l,j,k}\, P_{l,j,k}. \tag{5.25}$$

2. Store $P_{l,j,k}$.
3. **if** $l = L$ **then** store $B_{l,j,k}$.

    end do

  end do

end do

Application Step

Application

Comment [Given an arbitrary vector α, we compute a vector f in this Step such that $f = S\alpha$ to precision ε, using the interpolative decompositions computed in Step P.]

do $l = 0, 1, \ldots, L-1, L$

do $j = 1, 2, \ldots, 2^l - 1, 2^l$

do $k = 1, 2, \ldots, 2^{L-l} - 1, 2^{L-l}$

if $l = 0$ **then** calculate and store

$$\beta_{0,k} = P_{0,k}\, \alpha_{L,k}. \tag{5.26}$$

if $l > 0$ **then** calculate and store

$$\beta_{l,j,k} = P_{l,j,k} \begin{pmatrix} \beta_{l-1,\lfloor \frac{j+1}{2} \rfloor, 2k-1} \\ \beta_{l-1,\lfloor \frac{j+1}{2} \rfloor, 2k} \end{pmatrix}. \tag{5.27}$$

if $l = L$ **then** calculate and store

$$f_{L,j} = B_{L,j}\, \beta_{L,j}. \tag{5.28}$$

end do

end do

end do

Comment [The vectors $f_{L,1}, f_{L,2}, \ldots, f_{L,2^L-1}, f_{L,2^L}$ are concatenated to form the vector f. The vector f satisfies $f = S\alpha$ to precision ε.]

5.5 An adaptive version of algorithm

In practice, any two different contiguous submatrices of S with the same number of entries usually have slightly different ranks to precision ε (see Observation 5.2). It is possible to modify the algorithm described in Sections 5.1 and 5.4 such that, for a given positive integer paramter $r_{\max}$, every submatrix for which we calculate an interpolative decomposition has rank at most $r_{\max}$. Specifically, if the submatrix $S(\Omega_{l,k}, X_{l,j})$ has rank to precision ε greater than $r_{\max}$, we do not compute the interpolative decomposition of $S(\Omega_{l,k}, X_{l,j})$ but partition $S(\Omega_{l,k}, X_{l,j})$ into its top half and its bottom half. Similarly, if T is any contigous submatrix of S encountered in the adaptive dyadic hierarchy such that the rank of S is greater than $r_{\max}$, we do not compute the interpolative decomposition of T, but partition T into its top half and its bottom half. We compute the interpolative decompositions of those contiguous submatrices encountered in the adaptive dyadic hieararchy whose numerical ranks are at most $r_{\max}$. Figure 9 illustrates one possible partition of the matrix A on Level 1 of the adaptive algorithm.

Observation 5.6 With an appropriate choice of $r_{\max}$, we have not yet encountered a case in which the adaptive algorithm applies a matrix more slowly than the non-adaptive algorithm. This appears to be due to a combination of more efficient CPU cache usage with decreased complexity of the algorithm.

6 Numerical Examples

In this section, we describe the results of several numerical tests of the algorithm described in Section 5. In the examples, we use the adaptive algorithm described in Section 5.5. We perform all computations to precision $\varepsilon = 10^{-10}$, unless specified otherwise.

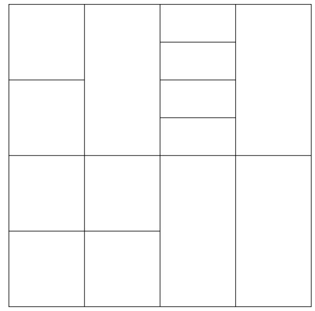

Figure 9: One possible partition of A on level 1 of the adaptive algorithm

We performed all computations using IEEE standard double-precision variables, whose mantissas have approximately one bit of precision less than 16 digits (so that the relative precision of the variables is approximately .2E–15). We ran all computations on one core of a 2.66 GHz Intel E6750 Core Duo microprocessor with 4MB of L2 cache and 4GB of RAM. We compiled the Fortran 77 code using the Lahey/Fujitsu Linux Express v6.2 compiler, with the optimization flag `--o2` enabled. The Lahey/Fujitsu Express v6.2 compiler can address only about 2GB of RAM per array.

The columns labeled "n" in the following tables list the size of the matrix to which the algorithm described in Section 5 was applied. All matrices in these examples are square, unless specified otherwise. The columns labeled "Precomputation" list the times taken in seconds for the initialization and precomputation steps of the algorithm described in Section 5. The columns labeled "Direct evaluation" list the times taken in seconds for a direct matrix-vector multiplication. For large matrices, the times taken for a direct matrix-vector multiplication were estimated and are in parentheses. The columns labeled "Fast evaluation" list the times taken to apply the matrix using the algorithm described in Section 5. The columns labeled "l^2 error" contain the relative errors between the solution obtained via the algorithm described in Section 5 and the solution obtained via a direct matrix-vector multiplication. The columns labeled "MB used" list the amount of memory in megabytes required by the algorithm for precomputation and evaluation. In these examples, we apply each matrix with n columns to the same vector $w^{(n)} = v^{(n)}/\|v^{(n)}\|$, where $v^{(n)}$ is a vector containing n independent random entries chosen uniformly at random from the interval $[0, 1]$.

Remark 6.1 It should be noted that no effort has been made to optimize the running time of the precomputations stage in the butterfly algorithm, either algorithmically or in the implementation. Thus, while the times listed below under the heading "Fast evaluation" are a reasonable indication of the algorithm's behavior, those listed under "Precompuatation" should be regarded as slower than necessary.

n	Precomputation	Direct evaluation	Fast evaluation	l^2 error	MB used
256	.54E+00	.85E-04	.86E-04	.69E-11	.43E+00
512	.10E+01	.34E-03	.26E-03	.11E-10	.13E+01
1024	.30E+01	.17E-02	.76E-03	.10E-10	.34E+01
2048	.91E+01	.68E-02	.23E-02	.80E-11	.90E+01
4096	.31E+02	.27E-01	.58E-02	.82E-11	.22E+02
8192	.12E+03	.11E+00	.14E-01	.92E-11	.54E+02
16384	.51E+03	.44E+00	.33E-01	.93E-11	.13E+03
32768	.23E+04	(.17E+01)	.79E-01	.92E-11	.30E+03
65536	.10E+05	(.70E+01)	.18E+00	.11E-10	.70E+03
131072	.45E+05	(.28E+02)	.43E+00	.13E-10	.17E+04

Table 1: Times, errors, and memory usage for the Legendre transform with $r_{\max} = 72$.

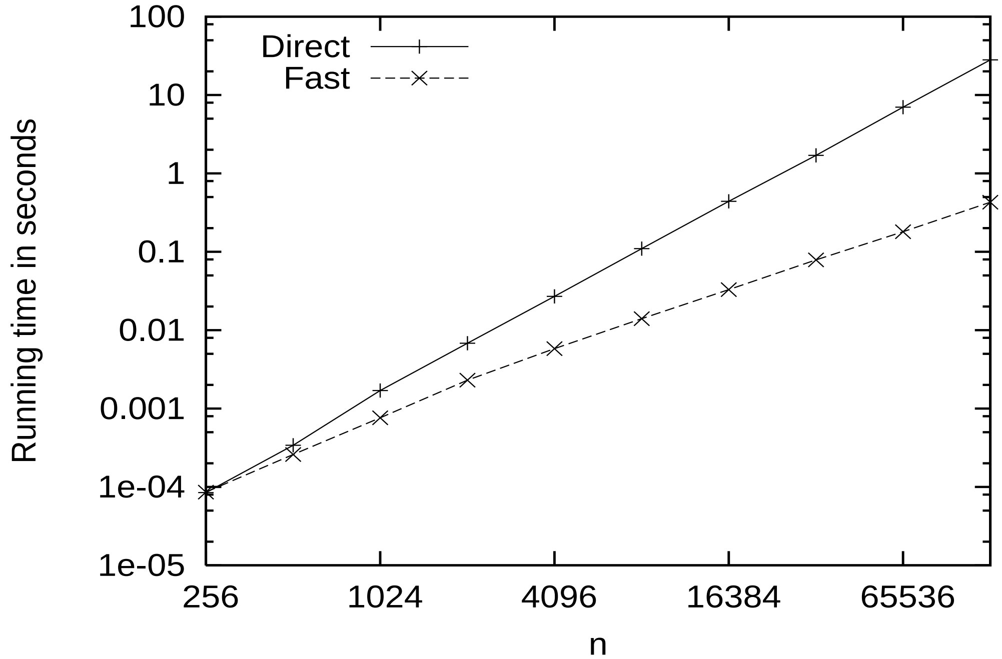

Figure 10: Comparison of the algorithm of Section 5 with direct calculation for evaluating the Legendre transform.

6.1 The Legendre transform

Table 1 and Figure 10 display the results of applying the algorithm described in Section 5 to the change of basis matrix T_P from the standard basis to the basis of Legendre polynomials. We chose the value $r_{\max} = 72$ to optimize the running times in Table 1. The parameter $r_{\max}$ is described in Section 5.5.

6.2 The Laguerre transform

Table 2 and Figure 11 display the results of applying the algorithm of Section 5 to the change of basis matrix T_L from the standard basis to the basis of Laguerre polynomials. We chose the value $r_{\max} = 83$ to optimize the running times in Table 2. The parameter $r_{\max}$ is described in Section 5.5.

6.3 The Hermite transform

Table 3 and Figure 12 display the results of applying the algorithm of Section 5 to the change of basis matrix T_H from the standard basis to the basis of Hermite polynomials. We chose the value $r_{\max} = 90$ to optimize the running times in Table 3. The parameter $r_{\max}$ is described in Section 5.5.

6.4 The non-equispaced Fourier transform

Table 4 and Figure 13 display the results of applying the algorithm described in Section 5 to the matrix T_F defined in equation (3.8), where the nodes x_j are chosen uniformly at random from the interval $[0, 2\pi]$ and the frequencies ω_j are chosen uniformly at random from the interval $[-n, n]$. We chose the parameter $r_{\max} = 73$ to optimize the running times in Table 4. The parameter $r_{\max}$ is described in Section 5.5.

6.5 The Fourier-Bessel transform

Tables 5-10 and Figures 14-19 display the results of applying the algorithm described in Section 5 to the matrix T_m^n defined in equation (3.44) where the real numbers y_j are the Gaussian quadrature nodes associated with Legendre polynomials on the interval $[0, 1]$ defined in Formula 25.4.30 in [1]. The value of R used in the definition of the function $\tilde{J}$ in (3.26) is $R = 1$.

In Table 5, we chose the value $r_{\max} = 93$ to optimize the running time of the algorithm described in Section 5 for the application of the Fourier-Bessel series transform of order $m = n/4$. This same value, $r_{\max} = 93$, is used in Tables 5-10. The parameter $r_{\max}$ is described in Section 5.5.

n	Precomputation	Direct evaluation	Fast evaluation	l^2 error	MB used
256	.52E+00	.85E-04	.83E-04	.18E-10	.42E+00
512	.12E+01	.34E-03	.24E-03	.32E-10	.12E+01
1024	.40E+01	.17E-02	.70E-03	.13E-09	.33E+01
2048	.15E+02	.68E-02	.23E-02	.15E-09	.88E+01
4096	.66E+02	.27E-01	.56E-02	.17E-09	.21E+02
8192	.30E+03	.11E+00	.13E-01	.23E-09	.52E+02
16384	.14E+04	.44E+00	.32E-01	.31E-09	.12E+03
32768	.62E+04	(.17E+01)	.75E-01	.39E-09	.29E+03
65536	.27E+05	(.70E+01)	.17E+00	.58E-09	.68E+03
131072	.12E+06	(.28E+02)	.42E+00	.80E-09	.16E+04

Table 2: Times, errors, and memory usage for the Laguerre transform with $r_{\max} = 83$.

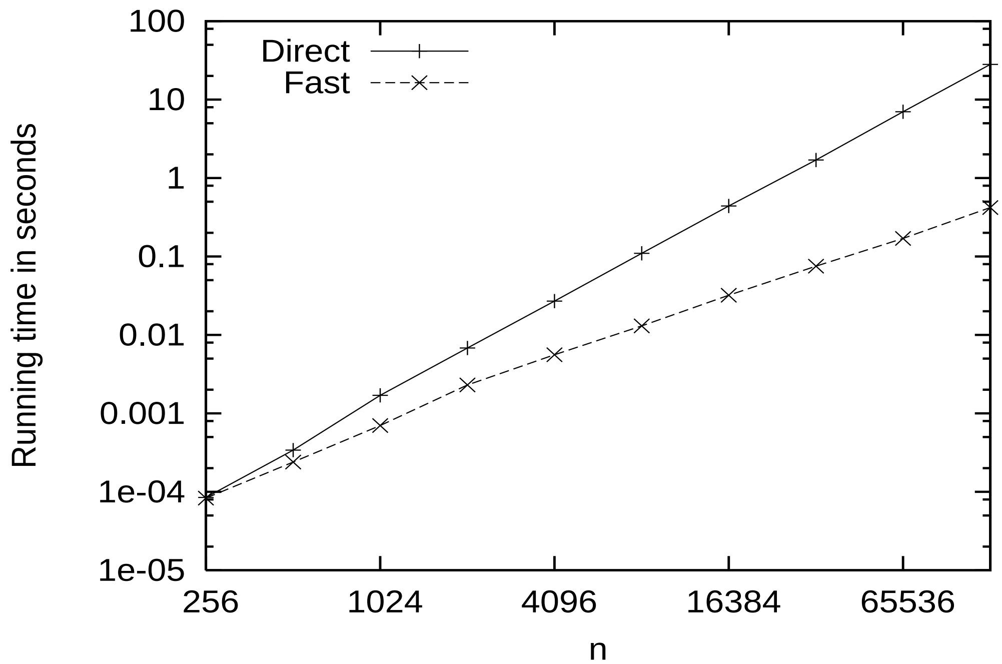

Figure 11: Comparison of the algorithm of Section 5 with direct calculation for evaluating the Laguerre transform.

n	Precomputation	Direct evaluation	Fast evaluation	l^2 error	MB used
256	.55E+00	.85E-04	.88E-04	.38E-11	.45E+00
512	.13E+01	.34E-03	.25E-03	.15E-10	.13E+01
1024	.42E+01	.17E-02	.71E-03	.22E-10	.34E+01
2048	.16E+02	.68E-02	.23E-02	.28E-10	.89E+01
4096	.70E+02	.27E-01	.57E-02	.33E-10	.22E+02
8192	.31E+03	.11E+00	.13E-01	.34E-10	.53E+02
16384	.15E+04	.44E+00	.32E-01	.38E-10	.13E+03
32768	.67E+04	(.17E+01)	.75E-01	.46E-10	.30E+03
65536	.30E+05	(.70E+01)	.17E+00	.51E-10	.69E+03
131072	.13E+06	(.28E+02)	.40E+00	.58E-10	.16E+04

Table 3: Times, errors, and memory usage for the Hermite transform with $r_{\max} = 90$.

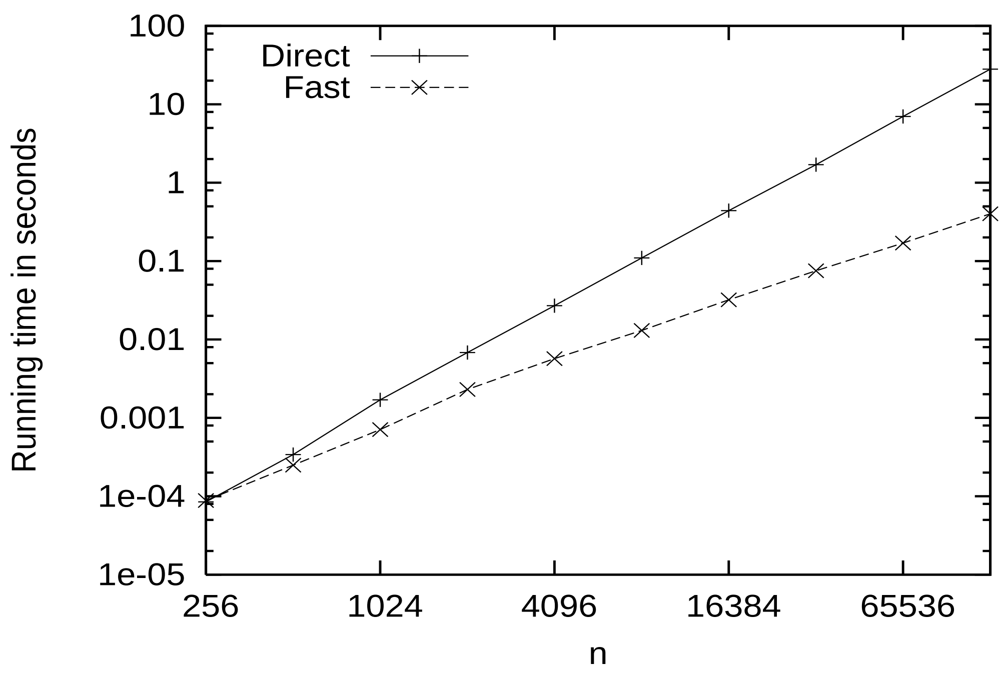

Figure 12: Comparison of the algorithm of Section 5 with direct calculation for evaluating the Hermite transform.

n	Precomputation	Direct evaluation	Fast evaluation	l^2 error	MB used
256	.42E+00	.21E-03	.21E-03	.76E-09	.93E+00
512	.11E+01	.88E-03	.59E-03	.28E-09	.25E+01
1024	.36E+01	.38E-02	.18E-02	.17E-08	.64E+01
2048	.49E+02	.15E-01	.45E-02	.26E-09	.16E+02
4096	.50E+02	.61E-01	.11E-01	.10E-08	.38E+02
8192	.20E+03	.24E+00	.25E-01	.15E-08	.87E+02
16384	.86E+03	(.97E+00)	.58E-01	.22E-08	.20E+03
32768	.36E+04	(.39E+01)	.13E+00	.13E-08	.45E+03
65536	.15E+05	(.19E+02)	.30E+00	.10E-08	.10E+04

Table 4: Times, errors, and memory usage for the non-equispaced Fourier transform with $r_{\max} = 73$.

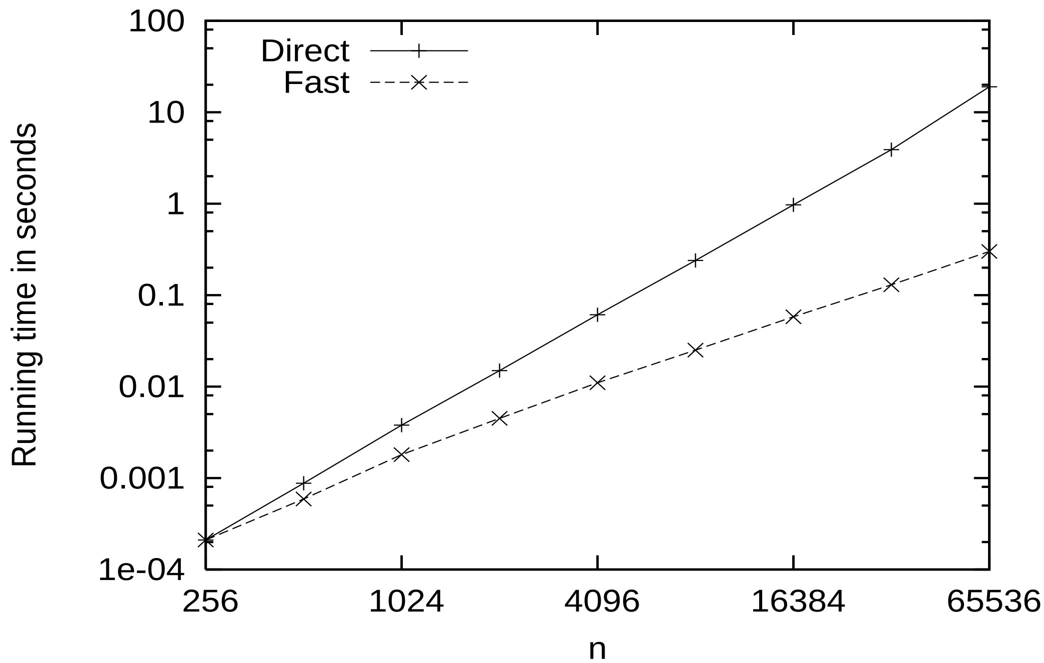

Figure 13: Comparison of the algorithm of Section 5 with direct calculation for evaluating the non-equispaced Fourier transform.

n	$\frac{n}{2}-m-C$	Precomputation	Direct evaluation	Fast evaluation	l^2 error	MB used
512	118	.10E+01	.77E-04	.58E-04	.61E-11	.28E+00
1024	246	.19E+01	.33E-03	.16E-03	.87E-11	.76E+00
2048	502	.45E+01	.17E-02	.44E-03	.13E-10	.21E+01
4096	1014	.11E+02	.69E-02	.13E-02	.15E-10	.52E+01
8192	2038	.29E+02	.28E-01	.33E-02	.17E-10	.13E+02
16384	4086	.91E+02	(.11E+00)	.79E-02	.21E-10	.31E+02
32768	8182	.32E+03	(.45E+00)	.18E-01	.22E-10	.73E+02
65536	16374	.13E+04	(.18E+01)	.42E-01	.23E-10	.17E+03
131072	32758	.55E+04	(.72E+01)	.95E-01	.28E-10	.41E+03

Table 5: Times, errors, and memory usage for calculating the first $n/2 - m - 10$ coefficients in the Fourier-Bessel expansion of order $m = n/4$, discretized at n nodes, that is, applying the real $\frac{n}{2} - m - C \times n$ matrix T_m^n defined in equation (3.44) with $r_{\max} = 93$ and $C = 10$.

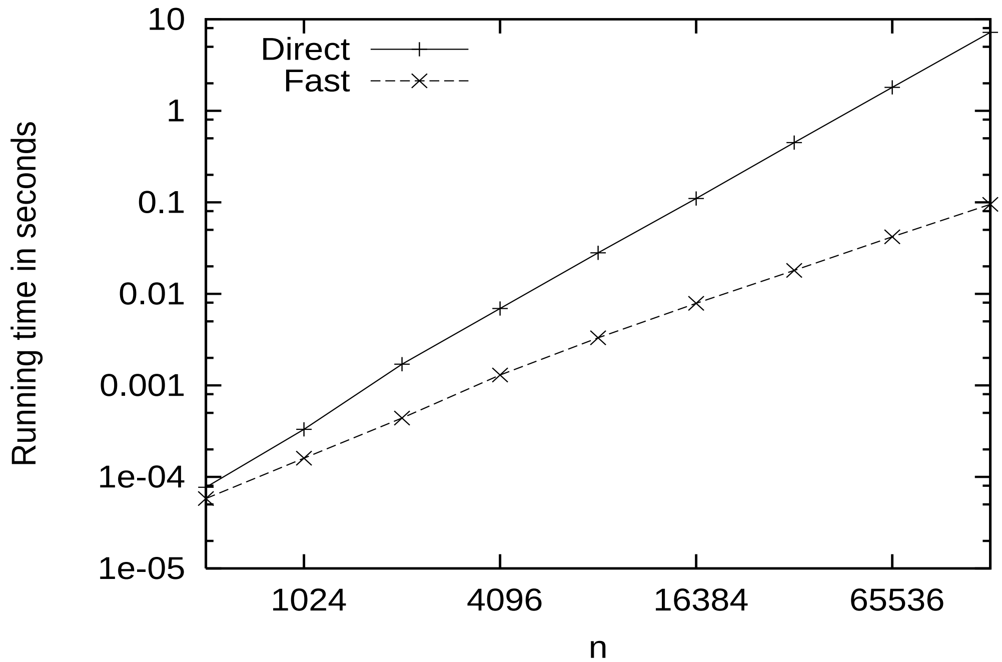

Figure 14: Comparison of the algorithm of Section 5 with direct calculation for evaluating the first $n/2 - m - 10$ coefficients in the Fourier-Bessel expansion of order $m = n/4$, discretized at n nodes, that is, applying the real $\frac{n}{2} - m - C \times n$ matrix T_m^n defined in equation (3.44) with $r_{\max} = 93$ and $C = 10$.

n	$\frac{n}{2} - m - C$	Precomputation	Direct evaluation	Fast evaluation	l^2 error	MB used
512	246	.10E+01	.24E-03	.12E-03	.67E-11	.61E+00
1024	502	.21E+01	.73E-03	.35E-03	.68E-11	.17E+01
2048	1014	.56E+01	.34E-02	.11E-02	.35E-11	.47E+01
4096	2038	.16E+02	.14E-01	.30E-02	.46E-11	.12E+02
8192	4086	.53E+02	.55E-01	.72E-02	.66E-11	.29E+02
16384	8182	.20E+03	(.22E+00)	.17E-01	.92E-11	.69E+02
32768	16374	.83E+03	(.88E+00)	.39E-01	.76E-11	.16E+03
65536	32758	.37E+04	(.35E+01)	.90E-01	.49E-10	.38E+03
131072	65526	.17E+05	(.14E+02)	.21E+00	.10E-09	.89E+03

Table 6: Times, errors, and memory usage for calculating the first $n/2 - 10$ coefficients in the Fourier-Bessel expansion or order $m = 0$, discretized at n nodes, that is, applying the real $\frac{n}{2} - C \times n$ matrix T_0^n defined in equation (3.44) with $r_{\max} = 93$ and $C = 10$.

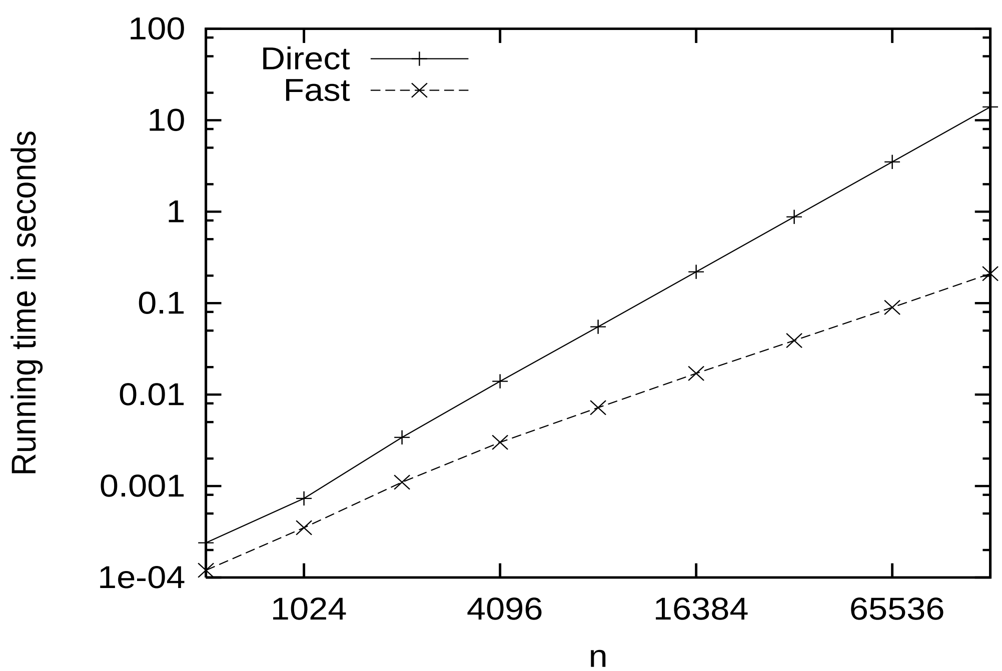

Figure 15: Comparison of the algorithm of Section 5 with direct calculation for evaluating the first $n/2 - 10$ coefficients in the Fourier-Bessel expansion of order $m = 0$, discretized at n nodes, that is, applying the real $\frac{n}{2} - m - C \times n$ matrix T_0^n defined in equation (3.44) with $r_{\max} = 93$ and $C = 10$.

n	$\frac{n}{2} - m - C$	Precomputation	Direct evaluation	Fast evaluation	l^2 error	MB used
512	245	.13E+01	.24E-03	.12E-03	.46E-11	.60E+00
1024	501	.26E+01	.73E-03	.35E-03	.15E-10	.17E+01
2048	1013	.66E+01	.34E-02	.11E-02	.56E-11	.47E+01
4096	2037	.18E+02	.14E-01	.30E-02	.90E-11	.12E+02
8192	4085	.57E+02	.55E-01	.72E-02	.10E-10	.29E+02
16384	8181	.21E+03	(.22E+00)	.17E-01	.10E-10	.69E+02
32768	16373	.85E+03	(.88E+00)	.39E-01	.78E-11	.16E+03
65536	32757	.38E+04	(.35E+01)	.90E-01	.79E-09	.38E+03
131072	65525	.17E+05	(.14E+02)	.21E+00	.42E-09	.90E+03

Table 7: Times, errors, and memory usage for calculating the first $n/2 - 11$ coefficients in the Fourier-Bessel expansion of order $m = 1$, discretized at n nodes, that is, applying the real $\frac{n}{2} - 1 - C \times n$ matrix T_1^n defined in equation (3.44) with $r_{\max} = 93$ and $C = 10$.

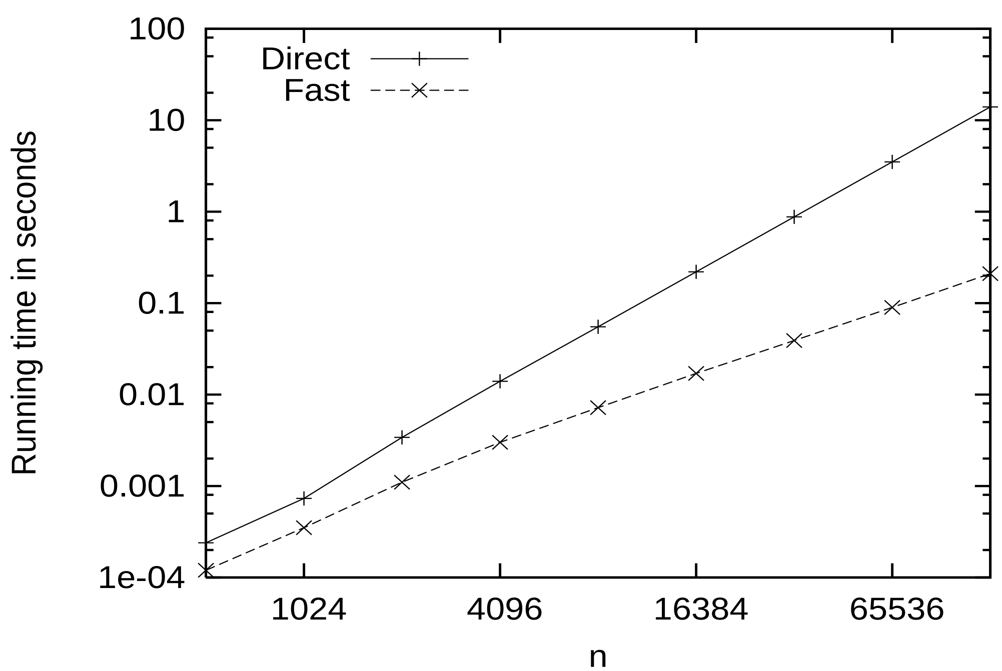

Figure 16: Comparison of the algorithm of Section 5 with direct calculation for evaluating the first $n/2 - 11$ coefficients in the Fourier-Bessel expansion of order $m = 1$, discretized at n nodes, that is, applying the real $\frac{n}{2} - m - C \times n$ matrix T_1^n defined in equation (3.44) with $r_{\max} = 93$ and $C = 10$.

n	$\frac{n}{2} - m - C$	Precomputation	Direct evaluation	Fast evaluation	l^2 error	MB used
512	146	.10E+01	.95E-04	.69E-04	.77E-11	.34E+00
1024	402	.22E+01	.54E-03	.27E-03	.13E-10	.13E+01
2048	914	.61E+01	.30E-02	.97E-03	.15E-10	.42E+01
4096	1938	.19E+02	.13E-01	.28E-02	.15E-10	.11E+02
8192	3986	.69E+02	.53E-01	.70E-02	.15E-10	.28E+02
16384	8082	.27E+03	(.21E+00)	.17E-01	.22E-10	.68E+02
32768	16274	.11E+04	(.87E+00)	.39E-01	.23E-10	.16E+03
65536	32658	.51E+04	(.35E+01)	.90E-01	.17E-10	.38E+03
131072	65426	.22E+05	(.14E+02)	.21E+00	.22E-10	.90E+03

Table 8: Times, errors, and memory usage for calculating the first $n/2 - 110$ coefficients in the Fourier-Bessel expansion of order $m = 100$, discretized at n nodes, that is, applying the real $\frac{n}{2} - 100 - C \times n$ matrix T^n_{100} defined in equation (3.44) with $r_{\max} = 93$ and $C = 10$.

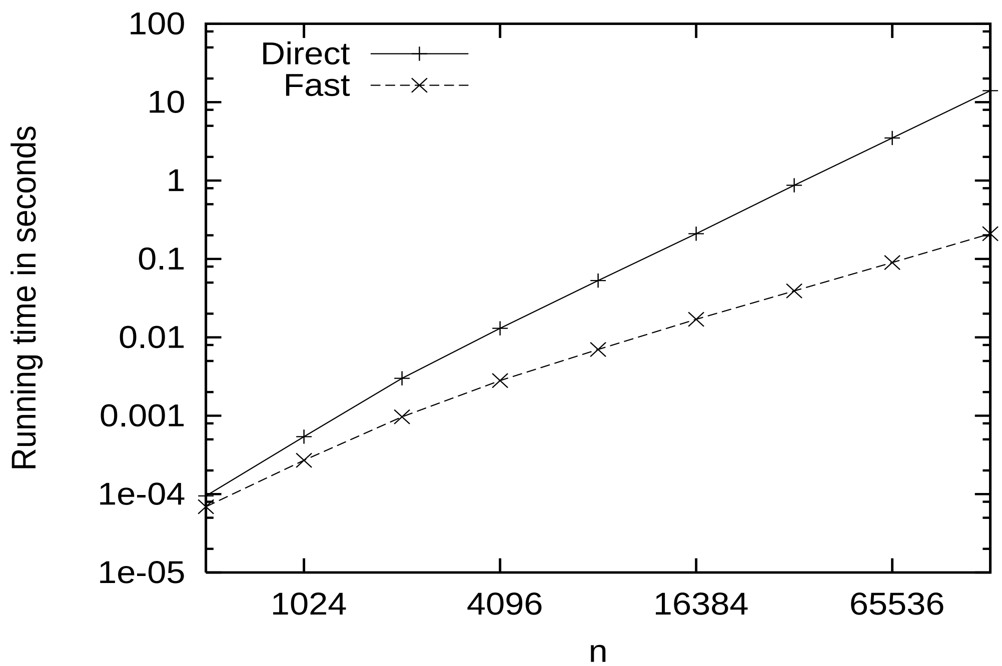

Figure 17: Comparison of the algorithm of Section 5 with direct calculation for evaluating the first $n/2 - 110$ coefficients in the Fourier-Bessel expansion of order $m = 100$, discretized at n nodes, that is, applying the real $\frac{n}{2} - m - C \times n$ matrix T^n_{100} defined in equation (3.44) with $r_{\max} = 93$ and $C = 10$.

n	$\frac{n}{2}-m-C$	ε	Precomputation	Direct evaluation	Fast evaluation	l^2 error	MB used
8192	4086	10^{-4}	.44E+02	.55E-01	.55E-02	.10E-04	.22E+02
8192	4086	10^{-6}	.47E+02	.55E-01	.62E-02	.17E-06	.25E+02
8192	4086	10^{-8}	.50E+02	.55E-01	.67E-02	.13E-08	.27E+02
8192	4086	10^{-10}	.53E+02	.55E-01	.73E-02	.66E-11	.29E+02
8192	4086	10^{-12}	.54E+02	.55E-01	.82E-02	.91E-13	.32E+02
8192	4086	10^{-14}	.11E+03	.55E-01	.54E-01	.49E-14	.21E+03
8192	4086	10^{-16}	.13E+03	.55E-01	.68E-01	.46E-14	.27E+03

Table 9: Times, errors, and memory usage for calculating the first $n/2-10$ coefficients in the Fourier-Bessel expansion of order $m=0$, discretizing at n nodes, that is, applying the real $\frac{n}{2}-C \times n$ matrix T_0^n defined in equation (3.44) with $r_{\max}=93$, $C=10$, $n=8192$, and various precisions ε.

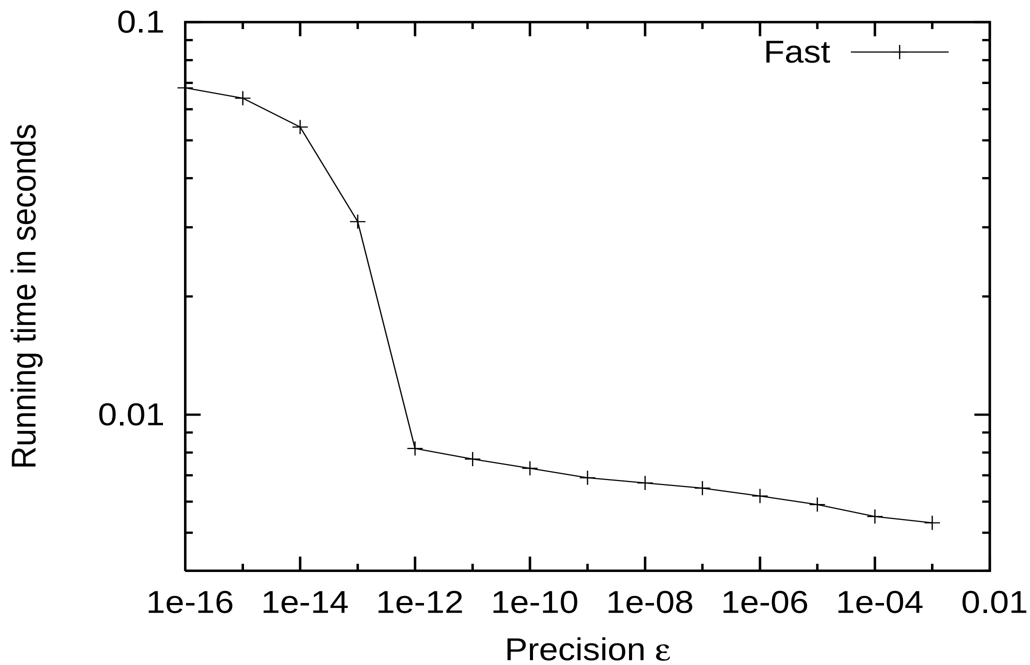

Figure 18: Times and errors for calculating the first $n/2-10$ coefficients in the Fourier-Bessel expansion of order $m=0$, discretizing at n nodes, that is, applying the real $\frac{n}{2}-C\times n$ matrix T_0^n defined in equation (3.44) with $r_{\max}=93$, $C=10$, $n=8192$, and various precisions ε.

n	$\frac{n}{2}-m-C$	ε	Precomputation	Direct evaluation	Fast evaluation	l^2 error	MB used
16384	8182	10^{-4}	.16E+03	(.22E+00)	.13E-01	.11E-04	.22E+02
16384	8182	10^{-6}	.17E+03	(.22E+00)	.14E-01	.85E-07	.25E+02
16384	8182	10^{-8}	.18E+03	(.22E+00)	.16E-01	.92E-09	.27E+02
16384	8182	10^{-10}	.20E+03	(.22E+00)	.17E-01	.92E-11	.29E+02
16384	8182	10^{-12}	.23E+03	(.22E+00)	.26E-01	.13E-12	.32E+02
16384	8182	10^{-14}	.52E+03	(.22E+00)	.23E+00	.57E-14	.21E+03
16384	8182	10^{-16}	.59E+03	(.22E+00)	.27E+00	.55E-14	.27E+03

Table 10: Times, errors, and memory usage for calculating the first $n/2-10$ coefficients in the Fourier-Bessel expansion of order $m=0$, discretizing at n nodes, that is, applying the real $\frac{n}{2}-C \times n$ matrix T_0^n defined in equation (3.44) with $r_{\max}=93$, $C=10$, $n=16384$, and various precisions ε.

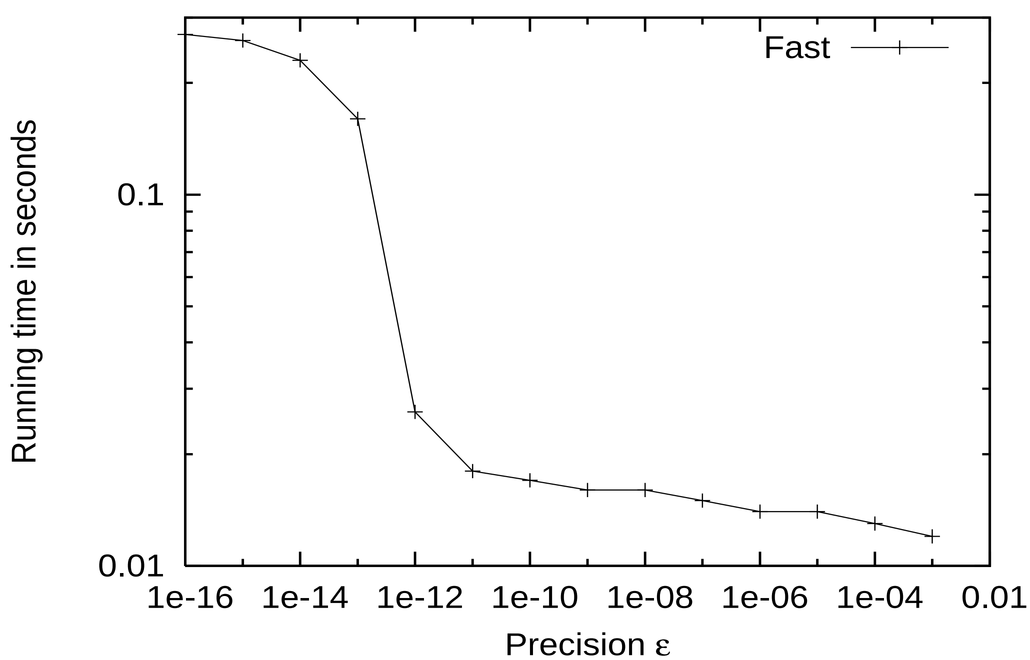

Figure 19: Times and errors for calculating the first $n/2-10$ coefficients in the Fourier-Bessel expansion of order $m=0$, discretizing at n nodes, that is, applying the real $\frac{n}{2}-C \times n$ matrix T_0^n defined in equation (3.44) with $r_{\max}=93$, $C=10$, $n=16384$, and various precisions ε.

n	Precomputation	Direct evaluation	Fast evaluation	l^2 error	MB used
256	.32E+00	.85E-04	.42E-04	.17E-10	.21E+00
512	.65E+00	.34E-03	.99E-04	.40E-10	.50E+00
1024	.21E+01	.17E-02	.25E-03	.28E-10	.13E+01
2048	.97E+01	.68E-02	.59E-03	.42E-10	.30E+01
4096	.58E+02	.27E-01	.16E-02	.47E-10	.67E+01
8192	.42E+03	.11E+00	.36E-02	.47E-10	.15E+02
16384	.32E+04	.44E+00	.79E-02	.52E-10	.34E+02

Table 11: Times, errors, and memory usage for evaluating Bessel function expansions by applying the matrix E_J defined in equation (3.11), with $r_{\max} = 86$.

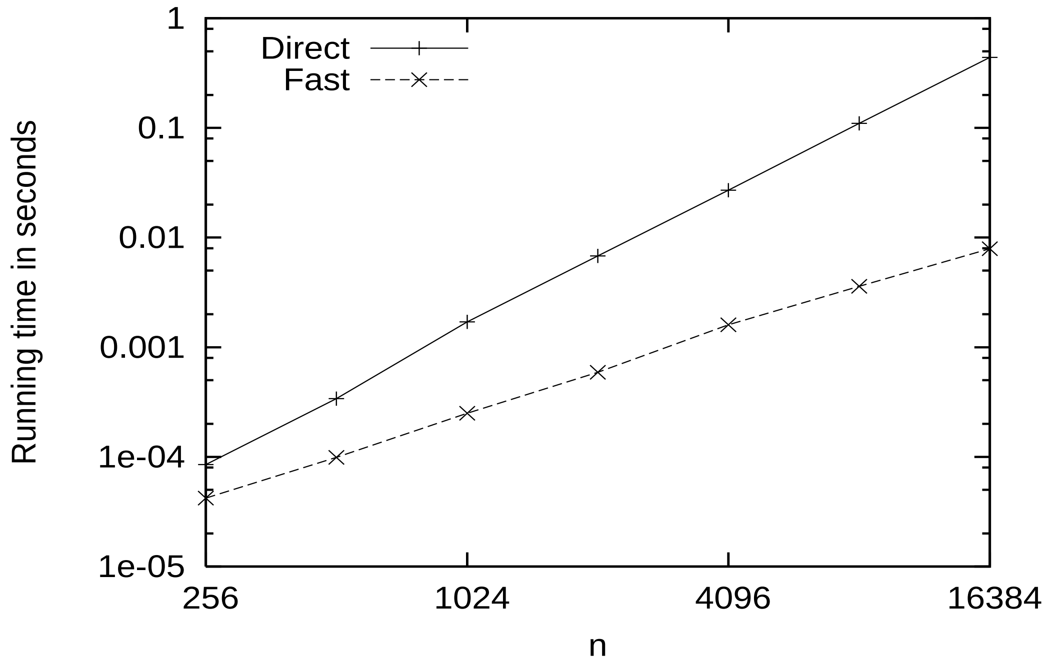

Figure 20: Comparison of the algorithm of Section 5 with direct calculation for evaluating Bessel function expansions by applying the matrix E_J defined in equation (3.11).

n	Precomputation	Direct evaluation	Fast evaluation	l^2 error	MB used
256	.75E+00	.21E-03	.61E-04	.37E-10	.24E+00
512	.14E+01	.88E-03	.14E-03	.35E-10	.53E+00
1024	.34E+01	.38E-02	.32E-03	.38E-10	.12E+01
2048	.86E+01	.15E-01	.70E-03	.50E-10	.27E+01
4096	.28E+02	.61E-01	.18E-02	.62E-10	.62E+01
8192	.13E+03	.24E+00	.36E-02	.54E-10	.15E+02
16384	.77E+03	(.97E+00)	.79E-02	.59E-10	.37E+02

Table 12: Times, errors, and memory usage for evaluating Hankel function expansions by applying the matrix $E_H^{(1)}$ defined in equation (3.12), with $r_{\max} = 38$.

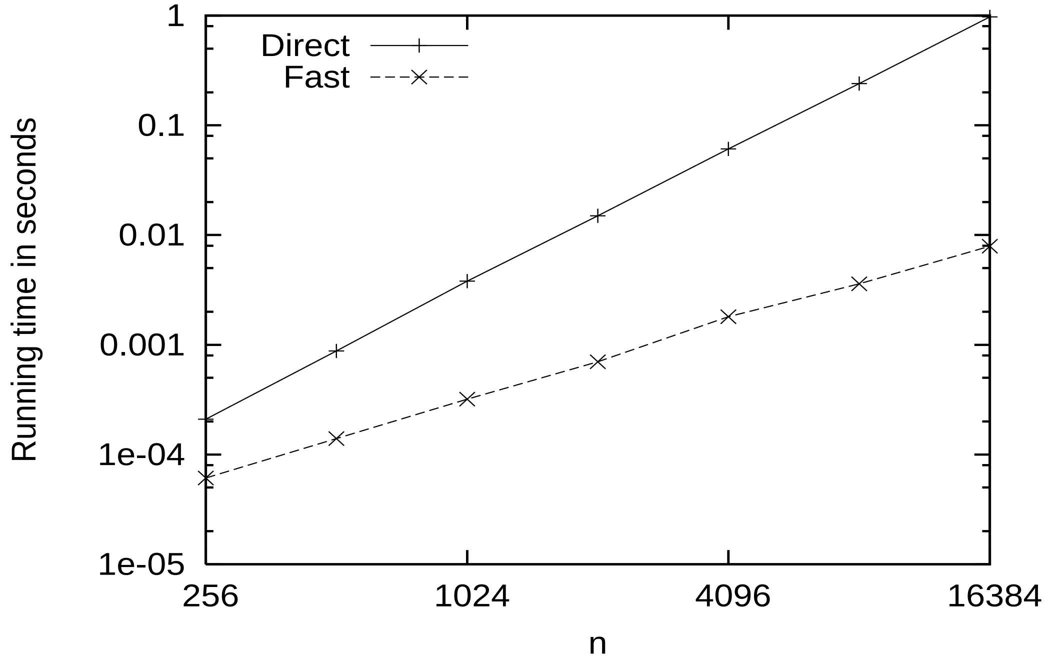

Figure 21: Comparison of the algorithm of Section 5 with direct calculation for evaluating Hankel function expansions by applying the matrix $E_H^{(1)}$ defined in equation (3.12).

6.6 Sums of Bessel and Hankel functions

Table 11 and Figure 20 display the results of applying the algorithm described in Section 5 to the matrix E_J defined in equation (3.11) where

$$x_j = n + \frac{2\pi}{3}(j-1), \tag{6.1}$$

for each integer j such that $1 \leq j \leq n$. We chose the value $r_{\max} = 86$ to optimize the running times in Table 11.

Table 12 and Figure 21 display the results of applying the algorithm described in Section 5 to the matrix $E_H^{(1)}$ defined in equation (3.12), where the nodes x_j are defined in (6.1). We chose the value $r_{\max} = 38$ to optimize the running times in Table 12. The parameter $r_{\max}$ is described in Section 5.5.

7 Conclusions and further work

We have presented an algorithm for the numerical computation of several special function transforms with asymptotic running time $\mathcal{O}(n \log(n))$, and asymptotic precomputation cost $\mathcal{O}(n^2)$. These running times have been proven in the case of the Fourier-Bessel transform. Numerical examples demonstrate a much wider applicability; analysis of these cases is in progress and will be reported at a later date. An implementation of the algorithm described in Section 5 for the acceleration of spherical harmonic transforms is currently under development.

Acknowledgments

We would like to thank Professors Eric Michielssen and Mark Tygert for many useful discussions.

References

[1] Milton Abramowitz and Irene Stegun. *Handbook of Mathematical Functions*. Applied Math. Series (National Bureau of Standards), Washington, DC, 1964.

[2] Hongwei Cheng, Zydrunas Gimbutas, P.-G. Martinsson, and Vladimir Rokhlin. On the compression of low rank matrices. *SIAM J. Sci. Comput.*, 26(4):1389–1404, 2005.

[3] Germund Dahlquist and Åke Björck. *Numerical Methods*. Dover Publications, Mineola, NY, 2003.

[4] Alok Dutt and Vladimir Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM Journal of Scientific Computing*, 14(6):1368–1393, 1993.

[5] Alok Dutt and Vladimir Rokhlin. Fast Fourier transforms for nonequispaced data. *Applied Computational Harmonic Analysis*, 2(1):85–100, 1995.

[6] George V. Frisk, Alan V. Oppenheim, and David R. Martinez. A technique for measureing the plane-wave reflection coefficient of the ocean bottom. *Journal of the Acoustical Society of America*, 68(2):602–612, 1980.

[7] I. S. Gradshteyn and I. M. Ryzhik. *Table of Integrals, Series, and Products*. Academic Press, San Diego, CA, 2000.

[8] Ming Gu and Stanley C. Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM J. Sci. Comput.*, 17(4):848–869, July 1996.

[9] William E. Higgins and David C. Munson. A Hankel transform approach to tomographic image reconstruction. *IEEE Transactions on Medical Imaging*, 7(1):59–72, March 1988.

[10] H. J. Landau and H. O. Pollak. Prolate spheroidal wave functions, Fourier analysis and uncertainty - III: the dimension of the space of essentially time- and band-limited signals. *The Bell System technical journal*, 41:1295–1336, 1962.

[11] P.-G. Martinsson, Vladimir Rokhlin, and Mark Tygert. On interpolation and integration in finite-dimensional spaces of bounded functions. *Comm. Appl. Math. Comput. Sci.*, 1:133–142, 2006.

[12] Eric Michielssen and Amir Boag. A multilevel matrix decomposition algorithm for analysing scattering from large structures. *IEEE Transactions on Antennas and Propagation*, 44:1086–1093, 1996.

[13] Jack Nachamkin and C. J. Maggiore. A Fourier Bessel transform method for efficiently calculating the magnetic field of solenoids. *Journal of Computational Physics*, 37:41–55, 1980.

[14] Frank Natterer and Frank Wübbeling. *Mathematical Methods in Image Reconstruction*. SIAM, Philadelphia, PA, 2001.

[15] Michael O'Neil and Vladimir Rokhlin. A new class of analysis-based fast transforms. Technical Report 1384, Yale University, Department of Computer Science, August 2007.

[16] Alan V. Oppenheim, George V. Frisk, and David R. Martinez. Computation of the Hankel transform using projections. *Journal of the Acoustical Society of America*, 68(2):523–529, 1980.

[17] Vladimir Rokhlin and Mark Tygert. Fast algorithms for spherical harmonic expansions. *SIAM Journal on Scientific Computing*, 27:1903–1928, 2006.

[18] Elias Stein and Guido Weiss. *Introduction to Fourier Analysis on Euclidean Spaces*. Princeton University Press, Princeton, NJ, 1971.

[19] Paul N. Swartztrauber. Spectral transform methods for solving the shallow-water equations on the sphere. *Monthly Weather Review*, 124:730–744, 1996.

[20] Paul N. Swartztrauber. Shallow-water flow on the sphere. *Monthly Weather Review*, 132:3010–3018, 2004.

[21] Paul N. Swartztrauber and W. F. Spotz. Generalized discrete spherical harmonic transforms. *Journal of Computational Physics*, 159:213–230, 2000.

[22] Mark Tygert. Fast algorithms for spherical harmonic expansions II. *Journal of Computational Physics*, 227:4260–4279.

[23] Mark Tygert. Recurrence relations and fast algorithms. Technical Report 1343, Yale University, Department of Computer Science, December 2005.

[24] Mark Tygert. Analogues for Bessel functions of the Christoffel-Darboux identity. Technical Report 1351, Yale University, Department of Computer Science, March 2006.

[25] George Watson. *A Treatise on the Theory of Bessel Functions.* Cambridge University Press, Cambridge, UK, 1945.