Fast algorithms for spherical harmonic expansions, II

Yale University Dept. of Computer Science technical report #1381

Mark Tygert

Program in Applied Mathematics, Yale University, A. K. Watson Hall, Room 110, 51 Prospect St., New Haven, CT 06511

Abstract

We provide an efficient algorithm for calculating, at appropriately chosen points on the two-dimensional surface of the unit sphere in $\mathbb{R}^3$, the values of functions that are specified by their spherical harmonic expansions (a procedure known as the inverse spherical harmonic transform). We also provide an efficient algorithm for calculating the coefficients in the spherical harmonic expansions of functions that are specified by their values at these appropriately chosen points (a procedure known as the forward spherical harmonic transform). The algorithms are numerically stable, and, if the number of points in our standard tensor-product discretization of the surface of the sphere is proportional to l^2, then the algorithms have costs proportional to $l^2 \ln(l)$ at any fixed precision of computations. Several numerical examples illustrate the performance of the algorithms.

Key words: fast, algorithm, spherical harmonic, transform, special function, FFT, recurrence, spectral

2000 MSC: 65T50, 65N35, 76M22

1. Introduction

Over the past several decades, the fast Fourier transform (FFT) and its variants (see, for example, [10]) have had an enormous impact across the sciences. The FFT is an efficient algorithm for computing, for any positive integer n and complex numbers $\beta_0, \beta_1, \ldots, \beta_{n-2}, \beta_{n-1}$, the complex numbers $\alpha_0, \alpha_1, \ldots, \alpha_{n-2}, \alpha_{n-1}$ defined by

$$\alpha_j = \sum_{k=0}^{n-1} \beta_k \, e_k(x_j) \tag{1}$$

for $j = 0, 1, \ldots, n-2, n-1$, where $e_0, e_1, \ldots, e_{n-2}, e_{n-1}$ are the functions defined on $[-1, 1]$ by

$$e_k(x) = \exp\left(\frac{\pi \, i \, (2k - n) \, x}{2}\right) \tag{2}$$

for $k = 0, 1, \ldots, n-2, n-1$, and $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$ are the real numbers defined by

$$x_k = \frac{2k - n}{n} \tag{3}$$

for $k = 0, 1, \ldots, n-2, n-1$. The FFT is efficient in the sense that there exists a reasonably small positive real number C such that, for any positive integer $n \geq 10$, the FFT requires at most $C \, n \ln(n)$ floating-point operations to compute $\alpha_0, \alpha_1, \ldots, \alpha_{n-2}, \alpha_{n-1}$ in (1) from $\beta_0, \beta_1, \ldots, \beta_{n-2}, \beta_{n-1}$. In contrast, evaluating the sum in (1) separately for every $j = 0, 1, \ldots, n-2, n-1$ costs at least n^2 operations in total.

It is desirable to have an analogue of the FFT for functions defined on the two-dimensional surface of the unit sphere in $\mathbb{R}^3$, in the following sense. The spherical harmonic expansion of a bandlimited function f on the surface of the sphere has the form

$$f(\theta, \varphi) = \sum_{k=0}^{2l-1} \sum_{m=-k}^{k} \beta_k^m \, \overline{P}_k^{|m|}(\cos(\theta)) \, e^{im\varphi}, \tag{4}$$

where (θ, φ) are the standard spherical coordinates on the two-dimensional surface of the unit sphere in $\mathbb{R}^3$, $\theta \in (0, \pi)$ and $\varphi \in (0, 2\pi)$, and $\overline{P}_k^{|m|}$ is the normalized associated Legendre function of degree k and order $|m|$, defined on $(-1, 1)$ via the formula

$$\overline{P}_k^{|m|}(x) = \sqrt{\frac{2k+1}{2} \frac{(k-|m|)!}{(k+|m|)!}} \; \sqrt{1-x^2}^{\,|m|} \; \frac{d^{|m|}}{dx^{|m|}} P_k(x), \tag{5}$$

where P_k is the Legendre polynomial of degree k (see, for example, Chapter 8 of [1]). (Please note that the superscript m in β_k^m denotes an index, rather than a power.) Obviously, the expansion (4) contains $4l^2$ terms. The complexity of the function f determines l.

In many scientific endeavors, particularly those using spectral methods for the numerical solution of partial differential equations, we need to evaluate the coefficients β_k^m in an expansion of the form (4) for a function f given by a table of its values at a collection of appropriately chosen nodes on the two-dimensional surface of the unit sphere. Conversely, given the coefficients β_k^m in (4), we often need to evaluate f at a collection of points on the surface of the sphere. The former is known as the forward spherical harmonic transform, and the latter is known as the inverse spherical harmonic transform. A standard discretization of the surface of the sphere is the "tensor product," consisting of all pairs of the form (θ_k, φ_j), with $\cos(\theta_0)$, $\cos(\theta_1)$, ..., $\cos(\theta_{2l-2})$, $\cos(\theta_{2l-1})$ being the Gauss-Legendre quadrature nodes of degree $2l$, that is,

$$-1 < \cos(\theta_0) < \cos(\theta_1) < \ldots < \cos(\theta_{2l-2}) < \cos(\theta_{2l-1}) < 1 \tag{6}$$

and

$$\overline{P}_{2l}^{0}(\cos(\theta_k)) = 0 \tag{7}$$

for $k = 0, 1, \ldots, 2l-2, 2l-1$, and with $\varphi_0, \varphi_1, \ldots, \varphi_{4l-3}, \varphi_{4l-2}$ being equispaced on the interval $(0, 2\pi)$, that is,

$$\varphi_j = \frac{2\pi \left(j + \frac{1}{2}\right)}{4l-1} \tag{8}$$

for $j = 0, 1, \ldots, 4l-3, 4l-2$. This leads immediately to numerical schemes for both the forward and inverse spherical harmonic transforms with costs proportional to l^3.

Indeed, given a function f defined on the two-dimensional surface of the unit sphere by (4), we can rewrite (4) in the form

$$f(\theta, \varphi) = \sum_{m=-2l+1}^{2l-1} e^{im\varphi} \sum_{k=|m|}^{2l-1} \beta_k^m \, \overline{P}_k^{|m|}(\cos(\theta)). \tag{9}$$

For a fixed value of θ, each of the sums over k in (9) contains no more than $2l$ terms, and there are $4l-1$ such sums (one for each value of m); since the inverse spherical harmonic transform involves $2l$ values θ_0, θ_1, ..., θ_{2l-2}, θ_{2l-1}, the cost of evaluating all sums over k in (9) is proportional to l^3. Once all sums over k have been evaluated, each sum over m may be evaluated for a cost proportional to l (since each of them contains $4l-1$ terms), and there are $(2l)(4l-1)$ such sums to be evaluated (one for each pair (θ_k, φ_j)), leading to costs proportional to l^3 for the evaluation of all sums over m in (9). The cost of the evaluation of the whole inverse spherical harmonic transform (in the form (9)) is the sum of the costs for the sums over k and the sums over m, and is also proportional to l^3; a virtually identical calculation shows that the cost of evaluating of the forward spherical harmonic transform is also proportional to l^3.

A trivial modification of the scheme described in the preceding paragraph uses the FFT to evaluate the sums over m in (9), roughly halving the operation count of the whole procedure. Several other careful considerations (see, for example, [2] and [14]) are able to reduce the costs by 50% or so, but there is no simple trick for reducing the costs of the whole spherical harmonic transform (either forward or inverse) below l^3. The present paper presents algorithms for both forward and inverse spherical harmonic transforms with costs proportional to $l^2 \ln(l)$ at any fixed precision of computations. Thus, our scheme provides an analogue of the FFT for functions defined on the surface of the sphere.

Specifically, the present paper provides an algorithm for evaluating a sum over k in (9) at $\theta = \theta_0$, θ_1, $\dots$, θ_{2l-2}, θ_{2l-1}, given the coefficients $\beta^m_{|m|}$, $\beta^m_{|m|+1}$, $\dots$, β^m_{2l-2}, β^m_{2l-1}, for a fixed m, with costs proportional to $l \ln(l)$. Moreover, the present paper provides an algorithm for the inverse procedure of determining the coefficients $\beta^m_{|m|}, \beta^m_{|m|+1}, \dots, \beta^m_{2l-2}, \beta^m_{2l-1}$ from the values of a sum over k in (9) at $\theta = \theta_0, \theta_1, \dots, \theta_{2l-2}, \theta_{2l-1}$, with costs proportional to $l \ln(l)$. FFTs or fast discrete sine and cosine transforms can be used to handle the sums over m in (9) efficiently.

The similarity between the transforms of the present paper and the discrete Fourier transform in (1) is perhaps most transparent in the following (equivalent) formulation. Given any integers l and m with $|m| \le 2l - 1$ and real numbers $\beta^m_{|m|}$, $\beta^m_{|m|+1}$, $\dots$, β^m_{2l-2}, β^m_{2l-1}, one of the algorithms of the present paper computes rapidly the real numbers α^m_{-l}, α^m_{1-l}, $\dots$, α^m_{l-2}, α^m_{l-1} defined by

$$\alpha^m_j = \sum_{k=|m|}^{2l-1} \beta^m_k \, f^m_k(z_j) \tag{10}$$

for $j = -l, 1-l, \dots, l-2, l-1$, where $f^m_{|m|}$, $f^m_{|m|+1}$, $\dots$, f^m_{2l-2}, f^m_{2l-1} are the functions defined on $(-1, 1)$ by

$$f^m_k(x) = \overline{P}^{|m|}_k(x) \tag{11}$$

for $k = |m|, |m|+1, \dots, 2l-2, 2l-1$, and z_{-l}, z_{1-l}, $\dots$, z_{l-2}, z_{l-1} are the real numbers defined by

$$z_k = \cos(\theta_{k+l}) \tag{12}$$

for $k = -l, 1-l, \dots, l-2, l-1$, where θ_{k+l} is from (7). (Incidentally, it follows from (12) that

$$z_k = -z_{-k-1} \tag{13}$$

for $k = -l, 1-l, \dots, l-2, l-1$.) The present paper also provides an algorithm for computing rapidly $\beta^m_{|m|}$, $\beta^m_{|m|+1}$, $\dots$, β^m_{2l-2}, β^m_{2l-1}, given α^m_{-l}, α^m_{1-l}, $\dots$, α^m_{l-2}, α^m_{l-1}.

We refer the reader to [11] and its compilation of references for prior work on fast algorithms for spherical harmonic expansions, as well as to [8] for an alternative approach that is suitable for certain applications. The present paper utilizes techniques based on recurrence relations, techniques that are substantially more efficient than the similar ones in [11]. We gave an overview of these techniques in [16]; the present paper gives details regarding their application to computing spherical harmonic transforms rapidly. Large-scale spherical harmonic transforms such as those accelerated by the present paper are probably most commonly used today in numerical weather simulations and other geophysical computations (see, for example, [12] and [13]).

The present paper has the following structure: Section 2 summarizes a number of facts from numerical and mathematical analysis, used in Section 3. Section 3 constructs fast algorithms. Section 4 describes the results of several numerical tests of the algorithms of Section 3.

Throughout the present paper, we denote the normalized associated Legendre function of degree k and order m by $\overline{P}^m_k$, as in (5).

2. Preliminaries

In this section, we summarize certain widely known facts from numerical and mathematical analysis, used in Section 3. Subsection 2.1 summarizes properties of the fast multipole method. Subsection 2.2 describes certain properties of divide-and-conquer methods for diagonalizing self-adjoint tridiagonal matrices and applying their matrices of normalized eigenvectors. Subsection 2.3 summarizes basic properties of normalized associated Legendre functions. Subsection 2.4 provides tools for the analysis and synthesis of linear combinations of normalized associated Legendre functions. Subsection 2.5 provides tools for the interpolation of linear combinations of normalized associated Legendre functions. Subsection 2.6 describes the Prüfer transformation for classical Sturm-Liouville problems (concerning the eigenfunctions of self-adjoint second-order linear differential operators).

2.1. *The fast multipole method*

This subsection summarizes properties of fast algorithms developed in [9] and [18] for applying scaled Cauchy matrices.

For any positive integers l and n, and real numbers u_0, u_1, ..., u_{n-2}, u_{n-1}, v_0, v_1, ..., v_{l-2}, v_{l-1}, x_0, x_1, ..., x_{n-2}, x_{n-1}, and y_0, y_1, ..., y_{l-2}, y_{l-1}, we define

$$t_j = \sum_{k=0}^{n-1} \frac{v_j\, u_k}{y_j - x_k} \tag{14}$$

for $j = 0, 1, \ldots, l-2, l-1$.

As described in [9] and [18], there exist an algorithm and a positive real number C such that, for any positive integers l and n, real numbers $u_0, u_1, \ldots, u_{n-2}, u_{n-1}, v_0, v_1, \ldots, v_{l-2}, v_{l-1}, x_0, x_1, \ldots, x_{n-2}, x_{n-1}$, $y_0, y_1, \ldots, y_{l-2}, y_{l-1}$, and positive real number $\varepsilon \le 1/10$, the algorithm computes $t_0, t_1, \ldots, t_{l-2}, t_{l-1}$ defined in (14) with a precision of computations ε, using at most

$$C\,(l+n)\,(\ln(1/\varepsilon))^2 \tag{15}$$

floating-point operations.

Remark 2.1 In practice, the algorithms described in [18] require only a small fraction of the memory required by the algorithm described in [9]. Also, the constant C in (15) is effectively smallest for the algorithm described in [9] only when it is used with fixed choices of x_j and y_k in (14) for many different choices of u_j and v_k. When the numbers x_j and y_k are fairly uniformly distributed, the "simple exponential-expansion FMM" algorithm described in Section 4 of [18] is often the most efficient of all the algorithms described in [9] and [18].

2.2. *Divide-and-conquer spectral methods*

This subsection summarizes properties of fast algorithms introduced in [6] and [7] for spectral representations of tridiagonal real self-adjoint matrices. Specifically, there exists an algorithm such that, for any tridiagonal real self-adjoint matrix T, (firstly) the algorithm computes the eigenvalues of T, (secondly) the algorithm computes any eigenvector of T, (thirdly) the algorithm applies a square matrix U consisting of normalized eigenvectors of T to any arbitrary column vector, and (fourthly) the algorithm applies U^{T} to any arbitrary column vector, all using a number of floating-point operations proportional to $n\,(\ln(n))\,(\ln(1/\varepsilon))^3$, where n is the positive integer for which T and U are $n \times n$, and ε is the precision of computations. The following is a more precise formulation.

For any positive integer n, self-adjoint $n \times n$ matrix T, and real $n \times 1$ column vector v, we define $\|T\|$ to be the largest of the absolute values of the eigenvalues of T, δ_T to be the minimum value of the distance $|\lambda - \mu|$ between any two distinct eigenvalues λ and μ of T, and

$$\|v\| = \sqrt{\sum_{k=0}^{n-1} (v_k)^2}, \tag{16}$$

where v_0, v_1, ..., v_{n-2}, v_{n-1} are the entries of v; we say that v is *normalized* to mean that $\|v\| = 1$. As originated in [7], there exist an algorithm and a positive real number C such that, for any positive real number $\varepsilon \le 1/10$, positive integer $n \ge 10$, tridiagonal real self-adjoint $n \times n$ matrix T with n distinct eigenvalues, real unitary $n \times n$ matrix U whose columns are n normalized eigenvectors of T, and real $n \times 1$ column vector v,

1. the algorithm computes to within absolute precision $\|T\|\,\varepsilon$ the n eigenvalues of T, using at most
$$C\,n\,(\ln(n))\,(\ln(1/\varepsilon))^3 \tag{17}$$
floating-point operations,
2. the algorithm computes to within absolute precision $\|T\|\,\|v\|\,\varepsilon/\delta_T$ the n entries of the matrix-vector product $U\,v$, using at most
$$C\,n\,(\ln(n))\,(\ln(1/\varepsilon))^3 \tag{18}$$
operations,
3. the algorithm computes to within absolute precision $\|T\|\,\|v\|\,\varepsilon/\delta_T$ the n entries of the matrix-vector product $U^{\mathrm{T}}\,v$, using at most
$$C\,n\,(\ln(n))\,(\ln(1/\varepsilon))^3 \tag{19}$$
operations, and,
4. after the algorithm performs some precomputations which are particular to T at a cost of at most
$$C\,n\,(\ln(n))\,(\ln(1/\varepsilon))^3 \tag{20}$$
operations, the algorithm computes to within absolute precision $\|T\|\,\varepsilon/\delta_T$ the $k\,n$ entries of any k user-specified normalized eigenvectors of T, using at most
$$C\,k\,n\,(\ln(1/\varepsilon))^2 \tag{21}$$
operations, for any positive integer k.

Remark 2.2 We omitted distracting factors of very small powers of n in the precisions mentioned in the present subsection.

Remark 2.3 In the second item of the present subsection, the algorithm in fact requires at most
$$C\,k\,n\,(\ln(n))\,(\ln(1/\varepsilon))^2 \tag{22}$$
operations in order to compute the matrix-vector products $U\,v^0$, $U\,v^1$, $\dots$, $U\,v^{k-2}$, $U\,v^{k-1}$, for any positive integer k, and real $n \times 1$ column vectors v^0, v^1, $\dots$, v^{k-2}, v^{k-1}, after the algorithm performs some precomputations which are particular to T at a cost of at most
$$C\,n\,(\ln(n))\,(\ln(1/\varepsilon))^3 \tag{23}$$
operations. Moreover, we can improve the precisions to which the algorithm calculates $U\,v^0$, $U\,v^1$, $\dots$, $U\,v^{k-2}$, $U\,v^{k-1}$, by performing more expensive precomputations (using higher-precision floating-point arithmetic or precomputation algorithms whose costs are not proportional to $n\,\ln(n)$, for example). For the numerical results reported in Section 4, we improve the precisions thus, by performing precomputations in extended-precision arithmetic. Similar considerations apply to the third item of the present subsection.

Remark 2.4 There exist similar algorithms when the eigenvalues of T are not all distinct.

Remark 2.5 The algorithm of the present subsection utilizes the fast multipole method (described, for example, in Subsection 2.1).

2.3. *Basic properties of normalized associated Legendre functions*

This subsection discusses several classical facts concerning normalized associated Legendre functions. All of these facts follow trivially from results contained, for example, in [1] or [15].

The following lemma states that the normalized associated Legendre functions of order m are orthonormal on $(-1, 1)$.

Lemma 2.6 *Suppose that m is a nonnegative integer.*

Then,

$$\int_{-1}^{1} dx\, \overline{P}_k^m(x)\, \overline{P}_l^m(x) = \begin{cases} 1, & k = l \\ 0, & k \neq l \end{cases} \tag{24}$$

for $k, l = m,\ m+1,\ m+2,\ \ldots.$

The following lemma states that the normalized associated Legendre functions satisfy a certain self-adjoint second-order linear (Sturm-Liouville) differential equation.

Lemma 2.7 *Suppose that m is a nonnegative integer.*

Then,

$$-\frac{d}{dx}\left((1-x^2)\,\frac{d}{dx}\overline{P}_l^m(x)\right) + \left(\frac{m^2}{1-x^2} - l(l+1)\right)\overline{P}_l^m(x) = 0 \tag{25}$$

for any $x \in (-1,1)$, and $l = m,\ m+1,\ m+2,\ \ldots.$

The following lemma states that the normalized associated Legendre function of order m and degree $m+2n$ has exactly n zeros inside $(0,1)$, and, moreover, that the normalized associated Legendre function of order m and degree $m+2n+1$ also has exactly n zeros inside $(0,1)$.

Lemma 2.8 *Suppose that m and n are nonnegative integers with $n > 0$.*

Then, there exist precisely n real numbers $x_0,\ x_1,\ \ldots,\ x_{n-2},\ x_{n-1}$ such that

$$0 < x_0 < x_1 < \ldots < x_{n-2} < x_{n-1} < 1 \tag{26}$$

and

$$\overline{P}_{m+2n}^m(x_k) = 0 \tag{27}$$

for $k = 0,\ 1,\ \ldots,\ n-2,\ n-1.$

Moreover, there exist precisely n real numbers $y_0,\ y_1,\ \ldots,\ y_{n-2},\ y_{n-1}$ such that

$$0 < y_0 < y_1 < \ldots < y_{n-2} < y_{n-1} < 1 \tag{28}$$

and

$$\overline{P}_{m+2n+1}^m(y_k) = 0 \tag{29}$$

for $k = 0,\ 1,\ \ldots,\ n-2,\ n-1.$

The case $m = 0$ of the preceding lemma is particularly important in applications; the following lemma restates part of the preceding lemma for the case $m = 0$.

Lemma 2.9 *Suppose that l is a positive integer.*

Then, there exist precisely l real numbers $z_0,\ z_1,\ \ldots,\ z_{l-2},\ z_{l-1}$ such that

$$0 < z_0 < z_1 < \ldots < z_{l-2} < z_{l-1} < 1 \tag{30}$$

and

$$\overline{P}_{2l}^0(z_k) = 0 \tag{31}$$

for $k = 0,\ 1,\ \ldots,\ l-2,\ l-1.$

Suppose that m and n are nonnegative integers with $n > 0$. Then, we define real numbers $\rho_0,\ \rho_1,\ \ldots,\ \rho_{n-2},\ \rho_{n-1},\ \sigma_0,\ \sigma_1,\ \ldots,\ \sigma_{n-2},\ \sigma_{n-1}$, and σ_n via the formulae

$$\rho_k = \frac{2\,(2m+4n+1)}{(1-x_k^2)\left(\frac{d}{dx}\overline{P}_{m+2n}^m(x_k)\right)^2} \tag{32}$$

for $k = 0,\ 1,\ \ldots,\ n-2,\ n-1$, where $x_0,\ x_1,\ \ldots,\ x_{n-2},\ x_{n-1}$ are from (27),

$$\sigma_k = \frac{2\,(2m+4n+3)}{(1-y_k^2)\left(\frac{d}{dx}\overline{P}_{m+2n+1}^m(y_k)\right)^2} \tag{33}$$

for $k = 0, 1, \ldots, n-2, n-1$, where $y_0, y_1, \ldots, y_{n-2}, y_{n-1}$ are from (29), and

$$\sigma_n = \frac{2m + 4n + 3}{\left(\frac{d}{dx}\overline{P}^m_{m+2n+1}(0)\right)^2}. \tag{34}$$

The following lemma describes what are known as Gauss-Jacobi quadrature formulae corresponding to associated Legendre functions.

Lemma 2.10 *Suppose that m and n are nonnegative integers with $n > 0$.*
Then,

$$\int_{-1}^{1} dx\, (1-x^2)^m\, p(x) = \sum_{k=0}^{n-1} \rho_k\, p(x_k) \tag{35}$$

for any even polynomial p of degree at most $4n-2$, where $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$ are from (27), and $\rho_0, \rho_1, \ldots, \rho_{n-2}, \rho_{n-1}$ are defined in (32).
Furthermore,

$$\int_{-1}^{1} dx\, (1-x^2)^m\, p(x) = \sigma_n\, p(0) + \sum_{k=0}^{n-1} \sigma_k\, p(y_k) \tag{36}$$

for any even polynomial p of degree at most $4n$, where $y_0, y_1, \ldots, y_{n-2}, y_{n-1}$ are from (29), and $\sigma_0, \sigma_1, \ldots, \sigma_{n-1}, \sigma_n$ are defined in (33) and (34).

Suppose that l is a positive integer. Then, we define real numbers $w_0, w_1, \ldots, w_{l-2}, w_{l-1}$ via the formula

$$w_k = \frac{2\,(4l+1)}{(1-z_k^2)\,\left(\frac{d}{dx}\overline{P}^0_{2l}(z_k)\right)^2} \tag{37}$$

for $k = 0, 1, \ldots, l-2, l-1$, where $z_0, z_1, \ldots, z_{l-2}, z_{l-1}$ are from (31).

The case $m = 0$ of the preceding lemma is particularly important in applications; the following lemma restates part of the preceding lemma for the case $m = 0$.

Lemma 2.11 *Suppose that l is a positive integer.*
Then,

$$\int_{-1}^{1} dx\, p(x) = \sum_{k=0}^{l-1} w_k\, p(z_k) \tag{38}$$

for any even polynomial p of degree at most $4l-2$, where $z_0, z_1, \ldots, z_{l-2}, z_{l-1}$ are from (31), and $w_0, w_1, \ldots, w_{l-2}, w_{l-1}$ are defined in (37).

Suppose that m is a nonnegative integer. Then, we define real numbers $c_m, c_{m+1}, c_{m+2}, \ldots$ and $d_m, d_{m+1}, d_{m+2}, \ldots$ via the formulae

$$c_l = \sqrt{\frac{(l-m+1)(l-m+2)(l+m+1)(l+m+2)}{(2l+1)\,(2l+3)^2\,(2l+5)}} \tag{39}$$

for $l = m, m+1, m+2, \ldots$, and

$$d_l = \frac{2l(l+1) - 2m^2 - 1}{(2l-1)(2l+3)} \tag{40}$$

for $l = m, m+1, m+2, \ldots$.

The following lemma states that the normalized associated Legendre functions of order m satisfy a certain three-term recurrence relation.

Lemma 2.12 *Suppose that m is a nonnegative integer.*
Then,

$$x^2\, \overline{P}^m_l(x) = d_l\, \overline{P}^m_l(x) + c_l\, \overline{P}^m_{l+2}(x) \tag{41}$$

for any $x \in (-1, 1)$ and $l = m$ or $l = m + 1$, and

$$x^2\, \overline{P}_l^m(x) = c_{l-2}\, \overline{P}_{l-2}^m(x) + d_l\, \overline{P}_l^m(x) + c_l\, \overline{P}_{l+2}^m(x) \tag{42}$$

for any $x \in (-1, 1)$ and $l = m + 2$, $m + 3$, $m + 4$, $\dots$, where c_m, c_{m+1}, c_{m+2}, $\dots$ are defined in (39), and d_m, d_{m+1}, d_{m+2}, $\dots$ are defined in (40).

2.4. *Analysis and synthesis of linear combinations of normalized associated Legendre functions*

This subsection provides tools for the analysis and synthesis of linear combinations of normalized associated Legendre functions via the theory of tridiagonal matrices. The methods of the present subsection have been in wide use for numerical purposes at least since [5] appeared.

Suppose that m and n are nonnegative integers with $n > 0$. We define S to be the tridiagonal real self-adjoint $n \times n$ matrix with the entry

$$S_{j,k} = \begin{cases} c_{m+2j-2}, & k = j - 1 \\ d_{m+2j}, & k = j \\ c_{m+2j}, & k = j + 1 \\ 0, & \text{otherwise (when } k < j - 1 \text{ or } k > j + 1) \end{cases} \tag{43}$$

for $j, k = 0, 1, \dots, n-2, n-1$, where c_m, c_{m+2}, $\dots$, c_{m+2n-4}, c_{m+2n-2} and d_m, d_{m+2}, $\dots$, d_{m+2n-4}, d_{m+2n-2} are defined in (39) and (40). We define T to be the tridiagonal real self-adjoint $n \times n$ matrix with the entry

$$T_{j,k} = \begin{cases} c_{m+2j-1}, & k = j - 1 \\ d_{m+2j+1}, & k = j \\ c_{m+2j+1}, & k = j + 1 \\ 0, & \text{otherwise (when } k < j - 1 \text{ or } k > j + 1) \end{cases} \tag{44}$$

for $j, k = 0, 1, \dots, n - 2, n - 1$, where c_{m+1}, c_{m+3}, $\dots$, c_{m+2n-3}, c_{m+2n-1} and d_{m+1}, d_{m+3}, $\dots$, d_{m+2n-3}, d_{m+2n-1} are defined in (39) and (40). We define U to be the real $n \times n$ matrix with the entry

$$U_{j,k} = \frac{\overline{P}_{m+2j}^m(x_k)}{\sqrt{\sum_{i=0}^{n-1} \left(\overline{P}_{m+2i}^m(x_k)\right)^2}} \tag{45}$$

for $j, k = 0, 1, \dots, n - 2, n - 1$, where $x_0, x_1, \dots, x_{n-2}, x_{n-1}$ are from (27). We define V to be the real $n \times n$ matrix with the entry

$$V_{j,k} = \frac{\overline{P}_{m+2j+1}^m(y_k)}{\sqrt{\sum_{i=0}^{n-1} \left(\overline{P}_{m+2i+1}^m(y_k)\right)^2}} \tag{46}$$

for $j, k = 0, 1, \dots, n - 2, n - 1$, where $y_0, y_1, \dots, y_{n-2}, y_{n-1}$ are from (29). We define Λ to be the diagonal real $n \times n$ matrix with the entry

$$\Lambda_{j,k} = \begin{cases} (x_j)^2, & k = j \\ 0, & k \neq j \end{cases} \tag{47}$$

for $j, k = 0, 1, \dots, n - 2, n - 1$, where $x_0, x_1, \dots, x_{n-2}, x_{n-1}$ are from (27). We define Γ to be the diagonal real $n \times n$ matrix with the entry

$$\Gamma_{j,k} = \begin{cases} (y_j)^2, & k = j \\ 0, & k \neq j \end{cases} \tag{48}$$

for $j, k = 0, 1, \ldots, n-2, n-1$, where $y_0, y_1, \ldots, y_{n-2}, y_{n-1}$ are from (29). We define A to be the diagonal real $n \times n$ matrix with the entry

$$A_{j,k} = \begin{cases} \sqrt{\sum_{i=0}^{n-1} \left(\overline{P}_{m+2i}^{m}(x_j) \right)^2}, & k = j \\ 0, & k \neq j \end{cases} \tag{49}$$

for $j, k = 0, 1, \ldots, n-2, n-1$, where $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$ are from (27). We define B to be the diagonal real $n \times n$ matrix with the entry

$$B_{j,k} = \begin{cases} \sqrt{\sum_{i=0}^{n-1} \left(\overline{P}_{m+2i+1}^{m}(y_j) \right)^2}, & k = j \\ 0, & k \neq j \end{cases} \tag{50}$$

for $j, k = 0, 1, \ldots, n-2, n-1$, where $y_0, y_1, \ldots, y_{n-2}, y_{n-1}$ are from (29).

The following lemma states that U is a matrix of normalized eigenvectors of the tridiagonal real self-adjoint matrix S, and that Λ is a diagonal matrix whose diagonal entries are the eigenvalues of S (which, according to (26), are distinct). The lemma also states, similarly, that V is a matrix of normalized eigenvectors of the tridiagonal real self-adjoint matrix T, and that Γ is a diagonal matrix whose diagonal entries are the eigenvalues of T (which, according to (28), are distinct).

Lemma 2.13 *Suppose that m and n are nonnegative integers with $n > 0$.*

Then,

$$U^{\mathrm{T}}\, S\, U = \Lambda, \tag{51}$$

where S is defined in (43), U is defined in (45), and Λ is defined in (47). Moreover, U is real and unitary.

Furthermore,

$$V^{\mathrm{T}}\, T\, V = \Gamma, \tag{52}$$

where T is defined in (44), V is defined in (46), and Γ is defined in (48). Moreover, V is real and unitary.

Proof. Combining (41), (42), and (27) yields that

$$S\, U = U\, \Lambda. \tag{53}$$

Combining (53), (45), (47), and (26) yields that U is a real matrix of normalized eigenvectors of S, with distinct corresponding eigenvalues. Therefore, since eigenvectors corresponding to distinct eigenvalues of a real self-adjoint matrix are orthogonal, U is orthogonal. Applying U^{T} from the left to both sides of (53) yields (51). The remaining statements of the lemma follow similarly. □

The following lemma expresses in matrix notation the analysis and synthesis of linear combinations of normalized associated Legendre functions.

Lemma 2.14 *Suppose that m and n are nonnegative integers with $n > 0$, and α, β, μ, and ν are real $n \times 1$ column vectors, such that α has the entry*

$$\alpha_j = \sum_{k=0}^{n-1} \beta_k\, \overline{P}_{m+2k}^{m}(x_j) \tag{54}$$

for $j = 0, 1, \ldots, n-2, n-1$, where $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$ are from (27), and μ has the entry

$$\mu_j = \sum_{k=0}^{n-1} \nu_k\, \overline{P}_{m+2k+1}^{m}(y_j) \tag{55}$$

for $j = 0, 1, \ldots, n-2, n-1$, where $y_0, y_1, \ldots, y_{n-2}, y_{n-1}$ are from (29).

Then,

$$\alpha = A\,U^{\mathrm{T}}\beta \tag{56}$$

and

$$\beta = U\,A^{-1}\,\alpha, \tag{57}$$

where U is defined in (45), A is defined in (49), and $A\,U^{\mathrm{T}}\beta$ and $U\,A^{-1}\,\alpha$ are matrix-matrix-vector products. Furthermore,

$$\mu = B\,V^{\mathrm{T}}\nu \tag{58}$$

and

$$\nu = V\,B^{-1}\,\mu, \tag{59}$$

where V is defined in (46), B is defined in (50), and $B\,V^{\mathrm{T}}\nu$ and $V\,B^{-1}\,\mu$ are matrix-matrix-vector products.

Proof. Combining (45) and (49) yields (56). According to Lemma 2.13, U is real and unitary. Therefore, applying the product $U\,A^{-1}$ from the left to both sides of (56) yields (57). The remaining statements of the lemma follow similarly. □

2.5. *Interpolation of linear combinations of normalized associated Legendre functions*

This subsection provides tools for the interpolation of linear combinations of normalized associated Legendre functions. The methods of the present subsection are taken from [8] and [17].

The following lemma provides what is known as a Christoffel-Darboux identity for the normalized associated Legendre functions.

Lemma 2.15 *Suppose that m and n are nonnegative integers with $n > 0$.*
Then,

$$\sum_{k=0}^{n-1} \overline{P}_{m+2k}^{m}(x)\,\overline{P}_{m+2k}^{m}(y) = \frac{c_{m+2n-2}}{x^2-y^2}\left(\overline{P}_{m+2n}^{m}(x)\,\overline{P}_{m+2n-2}^{m}(y) - \overline{P}_{m+2n-2}^{m}(x)\,\overline{P}_{m+2n}^{m}(y)\right) \tag{60}$$

for any $x, y \in (-1,1)$ such that $x \neq y$, where c_{m+2n-2} is defined in (39).
Furthermore,

$$\sum_{k=0}^{n-1} \overline{P}_{m+2k+1}^{m}(x)\,\overline{P}_{m+2k+1}^{m}(y) = \frac{c_{m+2n-1}}{x^2-y^2}\left(\overline{P}_{m+2n+1}^{m}(x)\,\overline{P}_{m+2n-1}^{m}(y) - \overline{P}_{m+2n-1}^{m}(x)\,\overline{P}_{m+2n+1}^{m}(y)\right) \tag{61}$$

for any $x, y \in (-1,1)$ such that $x \neq y$, where c_{m+2n-1} is defined in (39).

Proof. Combining (41) and (42) yields that

$$\begin{aligned}\sum_{k=0}^{n-1} \left(x^2\,\overline{P}_{m+2k}^{m}(x)\right)\,\overline{P}_{m+2k}^{m}(y) - \sum_{k=0}^{n-1} \overline{P}_{m+2k}^{m}(x)\,\left(y^2\,\overline{P}_{m+2k}^{m}(y)\right) \\ = c_{m+2n-2}\left(\overline{P}_{m+2n}^{m}(x)\,\overline{P}_{m+2n-2}^{m}(y) - \overline{P}_{m+2n-2}^{m}(x)\,\overline{P}_{m+2n}^{m}(y)\right).\end{aligned} \tag{62}$$

Dividing both sides of (62) by $x^2 - y^2$ yields (60). The remainder of the lemma follows similarly. □

The following lemma provides a formula which interpolates an even linear combination of n normalized associated Legendre functions of order m from its values at the positive zeros of $\overline{P}_{m+2n}^{m}$ to its values at arbitrary points in $(-1,1)$. Furthermore, the lemma provides a formula which interpolates an odd linear combination of n normalized associated Legendre functions of order m from its values at the positive zeros of $\overline{P}_{m+2n+1}^{m}$ to its values at arbitrary points in $(-1,1)$.

Lemma 2.16 *Suppose that m and n are nonnegative integers with $n > 0$. Suppose further that β_0, β_1, ..., β_{n-2}, β_{n-1} and ν_0, ν_1, ..., ν_{n-2}, ν_{n-1} are real numbers, and f and g are the functions defined on $(-1,1)$ via the formulae*

$$f(y) = \sum_{k=0}^{n-1} \beta_k \, \overline{P}_{m+2k}^m(y) \tag{63}$$

and

$$g(x) = \sum_{k=0}^{n-1} \nu_k \, \overline{P}_{m+2k+1}^m(x). \tag{64}$$

Then,

$$f(y) = c_{m+2n-2} \, \overline{P}_{m+2n}^m(y) \sum_{k=0}^{n-1} \frac{1}{y^2 - x_k^2} \, \rho_k \, \overline{P}_{m+2n-2}^m(x_k) \, f(x_k) \tag{65}$$

for any $y \in (-1,1)$, where c_{m+2n-2} is defined in (39), x_0, x_1, ..., x_{n-2}, x_{n-1} are from (27), and ρ_0, ρ_1, ..., ρ_{n-2}, ρ_{n-1} are defined in (32).

Furthermore,

$$g(x) = c_{m+2n-1} \, \overline{P}_{m+2n+1}^m(x) \sum_{k=0}^{n-1} \frac{1}{x^2 - y_k^2} \, \sigma_k \, \overline{P}_{m+2n-1}^m(y_k) \, g(y_k) \tag{66}$$

for any $x \in (-1,1)$, where c_{m+2n-1} is defined in (39), y_0, y_1, ..., y_{n-2}, y_{n-1} are from (29), and σ_0, σ_1, ..., σ_{n-2}, σ_{n-1} are defined in (33).

Proof. Combining (63) and (24) yields that

$$\sum_{k=0}^{n-1} \int_{-1}^{1} dx \, f(x) \, \overline{P}_{m+2k}^m(x) \, \overline{P}_{m+2k}^m(y) = f(y) \tag{67}$$

for any $y \in (-1,1)$. Applying (35) to the integral in the left-hand side of (67) — as permitted by (63) and (5) — yields that

$$f(y) = \sum_{k=0}^{n-1} \sum_{j=0}^{n-1} \rho_j \, f(x_j) \, \overline{P}_{m+2k}^m(x_j) \, \overline{P}_{m+2k}^m(y) \tag{68}$$

for any $y \in (-1,1)$. Combining (68), (60), and (27) yields (65). The remainder of the lemma follows similarly, using in addition the fact that $\overline{P}_{m+2n-1}^m(0) = 0 = \overline{P}_{m+2n+1}^m(0)$ (after all, $\overline{P}_{m+2n-1}^m$ and $\overline{P}_{m+2n+1}^m$ are odd due to (5)). □

The following lemma provides a formula which interpolates an even linear combination of n normalized associated Legendre functions of order m from its values at the positive zeros of $\overline{P}_{2l}^0$ to its values at the positive zeros of $\overline{P}_{m+2n}^m$. Here, l is an integer such that $m + 2n \le 2l$. Furthermore, the lemma provides a formula which interpolates an odd linear combination of n normalized associated Legendre functions of order m from its values at the positive zeros of $\overline{P}_{2l}^0$ to its values at the positive zeros of $\overline{P}_{m+2n+1}^m$.

Lemma 2.17 *Suppose that l, m, and n are nonnegative integers, such that $n > 0$ and $m + 2n \le 2l$. Suppose further that β_0, β_1, ..., β_{n-2}, β_{n-1} and ν_0, ν_1, ..., ν_{n-2}, ν_{n-1} are real numbers, and f and g are the functions defined on $(-1,1)$ via the formulae*

$$f(x) = \sum_{k=0}^{n-1} \beta_k \, \overline{P}_{m+2k}^m(x) \tag{69}$$

and

$$g(y) = \sum_{k=0}^{n-1} \nu_k \, \overline{P}_{m+2k+1}^m(y). \tag{70}$$

Then,

$$f(x_j) = c_{m+2n-2}\, \overline{P}^m_{m+2n-2}(x_j) \sum_{k=0}^{l-1} \frac{1}{z_k^2 - x_j^2}\, w_k\, \overline{P}^m_{m+2n}(z_k)\, f(z_k) \tag{71}$$

for $j = 0, 1, \dots, n-2, n-1$, *where* c_{m+2n-2} *is defined in (39),* $x_0, x_1, \dots, x_{n-2}, x_{n-1}$ *are from (27),* $z_0, z_1, \dots, z_{l-2}, z_{l-1}$ *are from (31), and* $w_0, w_1, \dots, w_{l-2}, w_{l-1}$ *are defined in (37).*

Furthermore,

$$g(y_j) = c_{m+2n-1}\, \overline{P}^m_{m+2n-1}(y_j) \sum_{k=0}^{l-1} \frac{1}{z_k^2 - y_j^2}\, w_k\, \overline{P}^m_{m+2n+1}(z_k)\, g(z_k) \tag{72}$$

for $j = 0, 1, \dots, n-2, n-1$, *where* c_{m+2n-1} *is defined in (39),* $y_0, y_1, \dots, y_{n-2}, y_{n-1}$ *are from (29),* $z_0, z_1, \dots, z_{l-2}, z_{l-1}$ *are from (31), and* $w_0, w_1, \dots, w_{l-2}, w_{l-1}$ *are defined in (37).*

Proof. Combining (69) and (24) yields that

$$\sum_{k=0}^{n-1} \int_{-1}^{1} dy\, f(y)\, \overline{P}^m_{m+2k}(x)\, \overline{P}^m_{m+2k}(y) = f(x) \tag{73}$$

for any $x \in (-1, 1)$. Applying (38) to the integral in the left-hand side of (73) — as permitted by (69) and (5) — yields that

$$f(x) = \sum_{k=0}^{n-1} \sum_{j=0}^{l-1} w_j\, f(z_j)\, \overline{P}^m_{m+2k}(x)\, \overline{P}^m_{m+2k}(z_j) \tag{74}$$

for any $x \in (-1, 1)$. Combining (74), (60), and (27) yields (71). The remainder of the lemma follows similarly. □

2.6. *The Prüfer transformation*

This subsection describes a certain reformulation of classical Sturm-Liouville problems (those concerning the eigenfunctions of self-adjoint second-order linear differential operators). The methods of the present subsection are taken from [4].

Straightforward calculations yield the following lemma, reformulating a certain self-adjoint second-order (Sturm-Liouville) differential equation as a pair of coupled first-order differential equations. This reformulation is classically known as the Prüfer transformation (see, for example, [4]).

Lemma 2.18 *Suppose that* a *and* b *are real numbers with* $a < b$, *and* f, p, q, r, *and* θ *are functions of* $x \in (a, b)$, *such that* f *is twice differentiable,* p *is differentiable,* $p(x) > 0$, $q(x) > 0$,

$$\frac{d}{dx}\left(p(x)\, \frac{d}{dx} f(x)\right) + q(x)\, f(x) = 0, \tag{75}$$

$$r(x) = \sqrt{\left(\sqrt{p(x)}\, \frac{d}{dx} f(x)\right)^2 + \left(\sqrt{q(x)}\, f(x)\right)^2}, \tag{76}$$

and

$$\theta(x) = \tan^{-1}\left(\frac{\sqrt{p(x)}\, \frac{d}{dx} f(x)}{\sqrt{q(x)}\, f(x)}\right) \tag{77}$$

for any $x \in (a, b)$.

Then, for any $x \in (a, b)$, $f(x) = 0$ *if and only if*

$$\theta(x) = \frac{\pi}{2} + k\pi \tag{78}$$

for some integer k. Furthermore,

$$\frac{d}{dx} r(x) = \frac{r(x)}{2} \left(\frac{\frac{d}{dx} q(x)}{q(x)} \cos^2(\theta(x)) - \frac{\frac{d}{dx} p(x)}{p(x)} \sin^2(\theta(x)) \right) \tag{79}$$

and

$$\frac{d}{dx} \theta(x) = -\sqrt{\frac{q(x)}{p(x)}} - \left(\frac{\frac{d}{dx} q(x)}{q(x)} + \frac{\frac{d}{dx} p(x)}{p(x)} \right) \frac{\sin(2\,\theta(x))}{4} \tag{80}$$

for any $x \in (a, b)$.

Moreover, if θ is strictly decreasing as a function of x, then x and r are parameterizable as functions of θ, with

$$\frac{dx}{d\theta} = -\left(\sqrt{\frac{q(x(\theta))}{p(x(\theta))}} + \left(\frac{\frac{d}{dx} q(x(\theta))}{q(x(\theta))} + \frac{\frac{d}{dx} p(x(\theta))}{p(x(\theta))} \right) \frac{\sin(2\theta)}{4} \right)^{-1} \tag{81}$$

and

$$\frac{dr}{d\theta} = \frac{r}{2} \left(\frac{\frac{d}{dx} p(x(\theta))}{p(x(\theta))} \sin^2(\theta) - \frac{\frac{d}{dx} q(x(\theta))}{q(x(\theta))} \cos^2(\theta) \right) \cdot \left(\sqrt{\frac{q(x(\theta))}{p(x(\theta))}} + \left(\frac{\frac{d}{dx} q(x(\theta))}{q(x(\theta))} + \frac{\frac{d}{dx} p(x(\theta))}{p(x(\theta))} \right) \frac{\sin(2\theta)}{4} \right)^{-1}. \tag{82}$$

Remark 2.19 The variables r in (76) and θ in (77) are polar coordinates in the (phase) plane having $\sqrt{q}\, f$ as the abscissa ("x-coordinate") and $\sqrt{p}\, \frac{d}{dx} f$ as the ordinate ("y-coordinate"). Please note that (80) and (81) do not involve r.

3. Description of the algorithms

In this section, we describe the algorithms of the present paper.

We describe all processing as it pertains to even or odd functions. To process a function h defined on $(-1, 1)$ which is neither even nor odd, we first separate h into its even and odd parts f and g, defined via the formulae

$$f(x) = \frac{h(x) + h(-x)}{2} \tag{83}$$

and

$$g(x) = \frac{h(x) - h(-x)}{2}. \tag{84}$$

All subsequent processing then concerns f or g, each of which is uniquely defined by its values on $(0, 1)$. Combining (83) and (84) yields that

$$h = f + g. \tag{85}$$

In the present section, we assume that f has the form

$$f(x) = \sum_{k=0}^{n-1} \beta_k\, \overline{P}^m_{m+2k}(x) \tag{86}$$

for any $x \in (-1, 1)$, where m and n are nonnegative integers with $n > 0$, and β is a real $n \times 1$ column vector. Combining (86) and (5) yields that f is a linear combination of even functions, and hence $f(-x) = f(x)$ for any $x \in (-1, 1)$. Similarly, we assume that g has the form

$$g(y) = \sum_{k=0}^{n-1} \nu_k\, \overline{P}^m_{m+2k+1}(y) \tag{87}$$

for any $y \in (-1, 1)$, where again m and n are nonnegative integers with $n > 0$, and ν is a real $n \times 1$ column vector. Combining (87) and (5) yields that g is a linear combination of odd functions, and hence $g(-y) = -g(y)$ for any $y \in (-1, 1)$.

Subsection 3.1 constructs the underlying algorithms which Subsection 3.2 uses to compute fast associated Legendre transforms (thus allowing us to compute fast spherical harmonic transforms, as elaborated in Section 1). Subsection 3.3 discusses certain auxiliary precomputations needed by the algorithms of Subsection 3.2.

3.1. *Analysis and synthesis of linear combinations of normalized associated Legendre functions*

Suppose that m and n are nonnegative integers with $n > 0$, and α and β are real $n \times 1$ column vectors, such that α has the entry

$$\alpha_j = f(x_j) = \sum_{k=0}^{n-1} \beta_k \, \overline{P}_{m+2k}^{m}(x_j) \tag{88}$$

for $j = 0, 1, \ldots, n-2, n-1$, where $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$ are the positive zeros of $\overline{P}_{m+2n}^{m}$ from (27), and f is defined in (86). This subsection constructs algorithms which compute α in (88) rapidly given β, and, vice versa, compute β rapidly given α.

According to (56) and (57),

$$\alpha = A\, U^{\mathrm{T}} \beta \tag{89}$$

and

$$\beta = U\, A^{-1}\, \alpha, \tag{90}$$

where U is defined in (45), A is defined in (49), and $A\, U^{\mathrm{T}} \beta$ and $U\, A^{-1}\, \alpha$ are matrix-matrix-vector products.

Since A defined in (49) is diagonal, we can apply A and A^{-1} for a cost proportional to n to an arbitrary real $n \times 1$ vector. According to (51), U in (89) and (90) is a matrix of normalized eigenvectors of the self-adjoint tridiagonal matrix S defined in (43), so we can apply U and U^{T} for a cost proportional to $n \ln(n)$ to an arbitrary real $n \times 1$ vector using items 2 and 3 of Subsection 2.2. Thus, we can use (89) to compute α in (88) for a cost proportional to $n \ln(n)$ given β, and, vice versa, use (90) to compute β for a cost proportional to $n \ln(n)$ given α.

Via similar procedures, we can compute μ in (55) for a cost proportional to $n \ln(n)$ given ν, and, vice versa, compute ν for a cost proportional to $n \ln(n)$ given μ.

3.2. *Interpolation of linear combinations of normalized associated Legendre functions*

Given nonnegative integers m and n with $n > 0$, the vector α defined in (88) provides the values of a function f at the positive zeros $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$ of $\overline{P}_{m+2n}^{m}$, where f is defined in (86). Subsection 3.1 provides an algorithm for computing α rapidly. To compute the inverse spherical harmonic transform, however, we need to calculate the values of f at the positive zeros $z_0, z_1, \ldots, z_{l-2}, z_{l-1}$ of $\overline{P}_{2l}^{0}$ from (31) (or, equivalently, from (7) and (12)), where l is an integer such that $m + 2n \le 2l$. The present subsection provides efficient schemes for performing this task, in addition to related tasks needed for computing the forward spherical harmonic transform, given the output of the algorithms of Subsection 3.1.

We can interpolate from the values of f at $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$, obtained via the algorithm of Subsection 3.1, to the values of f at $z_0, z_1, \ldots, z_{l-2}, z_{l-1}$ by means of (65), that is,

$$f(z_j) = c_{m+2n-2}\, \overline{P}_{m+2n}^{m}(z_j) \sum_{k=0}^{n-1} \frac{1}{z_j^2 - x_k^2}\, \rho_k\, \overline{P}_{m+2n-2}^{m}(x_k)\, f(x_k) \tag{91}$$

for $j = 0, 1, \ldots, l-2, l-1$, where c_{m+2n-2} is defined in (39), and $\rho_0, \rho_1, \ldots, \rho_{n-2}, \rho_{n-1}$ are defined in (32).

Similarly, we can interpolate from the values of f at $z_0, z_1, \ldots, z_{l-2}, z_{l-1}$ to the values of f at $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$ by means of (71), that is,

$$f(x_j) = c_{m+2n-2}\,\overline{P}^m_{m+2n-2}(x_j)\,\sum_{k=0}^{l-1} \frac{1}{z_k^2 - x_j^2}\,w_k\,\overline{P}^m_{m+2n}(z_k)\,f(z_k) \tag{92}$$

for $j = 0, 1, \ldots, n-2, n-1$, where c_{m+2n-2} is defined in (39), and $w_0, w_1, \ldots, w_{l-2}, w_{l-1}$ are defined in (37).

Using the fast multipole method summarized in Subsection 2.1 in conjunction with (91), we can compute for a cost proportional to $l+n$ the values of f at $z_0, z_1, \ldots, z_{l-2}, z_{l-1}$ from the values of f at $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$ (much like in [8] and [17]). Similarly, using the fast multipole method in conjunction with (92), we can compute for a cost proportional to $l+n$ the values of f at $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$ from the values of f at $z_0, z_1, \ldots, z_{l-2}, z_{l-1}$.

Via analogous procedures, we can interpolate for a cost proportional to $l+n$ between the values of a function g at $z_0, z_1, \ldots, z_{l-2}, z_{l-1}$ and the values of g at the positive zeros $y_0, y_1, \ldots, y_{n-2}, y_{n-1}$ of $\overline{P}^m_{m+2n+1}$ from (29), where g is defined in (87).

3.3. *Auxiliary precomputations*

In order to use (91) and (92), we have to precompute the positive zeros $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$ of $\overline{P}^m_{m+2n}$, the positive zeros $z_0, z_1, \ldots, z_{l-2}, z_{l-1}$ of $\overline{P}^0_{2l}$, the values of $\overline{P}^m_{m+2n-2}$ at $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$, the values of $\overline{P}^m_{m+2n}$ at $z_0, z_1, \ldots, z_{l-2}, z_{l-1}$, the numbers $\rho_0, \rho_1, \ldots, \rho_{n-2}, \rho_{n-1}$ defined in (32), and the numbers $w_0, w_1, \ldots, w_{l-2}, w_{l-1}$ defined in (37). This subsection describes algorithms which perform these precomputations.

To find the positive zeros $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$ of $\overline{P}^m_{m+2n}$, we integrate the ordinary differential equation (ODE) in formula (81) via an explicit predictor-corrector method like that in [3], thus obtaining the zeros of $\overline{P}^m_{m+2n}$ when (78) is satisfied. In (81), we take $p(x) = 1 - x^2$ and $q(x) = (m+2n)(m+2n+1) - \frac{m^2}{1-x^2}$, in accordance with the combination of (25) and (75), with $f(x) = \overline{P}^m_{m+2n}(x)$.

To find the numbers $\rho_0, \rho_1, \ldots, \rho_{n-2}, \rho_{n-1}$ defined in (32), we calculate r defined in (76) by integrating the ODE (82) (together with the ODE (81)) via an explicit predictor-corrector method like that in [3]. We then obtain $\rho_0, \rho_1, \ldots, \rho_{n-2}, \rho_{n-1}$ via the formula

$$\rho_k = \frac{2\,(2m+4n+1)}{\left(r(x_k)\right)^2} \tag{93}$$

for $k = 0, 1, \ldots, n-2, n-1$, which follows from (32), (76), and (27), with $f(x) = \overline{P}^m_{m+2n}(x)$. In (76), (81), and (82), we again take $p(x) = 1 - x^2$ and $q(x) = (m+2n)(m+2n+1) - \frac{m^2}{1-x^2}$, in accordance with the combination of (25) and (75), with $f(x) = \overline{P}^m_{m+2n}(x)$.

Via similar procedures, we can compute the positive zeros $z_0, z_1, \ldots, z_{l-2}, z_{l-1}$ of $\overline{P}^0_{2l}$ and the numbers $w_0, w_1, \ldots, w_{l-2}, w_{l-1}$ defined in (37). We can compute the values of $\overline{P}^m_{m+2n-2}$ at $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$ and the values of $\overline{P}^m_{m+2n}$ at $z_0, z_1, \ldots, z_{l-2}, z_{l-1}$ by integrating ODEs of the form (75) together with ODEs of the form (81), again via an explicit predictor-corrector method like that in [3].

We can compute similarly the positive zeros $y_0, y_1, \ldots, y_{n-2}, y_{n-1}$ of $\overline{P}^m_{m+2n+1}$, as well as the other numbers needed to interpolate functions of the form (87).

Since we use an explicit predictor-corrector method like that in [3], we find that we can perform all computations needed by the algorithm of the present subsection for a cost proportional to $l+n$. However, we found it expedient to perform the precomputations in extended-precision arithmetic, in order to compensate for the small loss of precision incurred during the precomputations.

Remark 3.1 We used the ODE integration scheme of [3] primarily for convenience; other highly accurate integration schemes would probably suffice.

4. Numerical results

In this section, we describe the results of several numerical tests of the algorithms of the present paper.

Tables 1, 2, and 3 report on the analysis and synthesis schemes described in Subsection 3.1, while Tables 4, 5, and 6 report on the interpolation schemes described in Subsection 3.2. Computing the forward or inverse spherical harmonic transform requires the schemes of both subsections, as well as fast discrete sine and cosine transforms (see Sections 1 and 3 for details). In each table, n is the size of the transform, m is the order of the normalized associated Legendre functions used in the transform, t_{fast} is the time in seconds required to apply the algorithm of the present paper once to calculate the application of a matrix to one input vector, $t_{\text{precomps.}}$ is the time in seconds required to precompute all data needed in order to execute the algorithm of the present paper, and $\varepsilon_{\text{r.m.s.}}$ is the root-mean-square of the difference of the output vector calculated by the algorithm of the present paper from the directly calculated output vector, divided by the root-mean-square of the input vector. We should note that we made no attempt to optimize the precomputations.

In each table, t_{direct} is the time in seconds required to apply a dense matrix to a vector via the conventional matrix-vector multiplication algorithm. We estimated the last two entries for t_{direct} by multiplying the third-to-last entry in each table by 4 and 16, since the large matrices required to generate those entries exceeded the memory addressable by the (32-bit addressing) Lahey-Fujitsu compiler. We indicate that these entries are estimates by enclosing them in parentheses. We individually describe the dimensionality of the matrix associated with each table in the description for each table below. Please note that the Lahey-Fujitsu compiler we used optimizes matrix-vector applications to make them particularly efficient (50–100% faster than matrix-vector applications effected using the f2c- and gcc-based fort77 compiler under its highest optimization setting, `-O3`).

Table 1 lists the results of computing from real numbers $\nu_0, \nu_1, \ldots, \nu_{n-2}, \nu_{n-1}$ the real numbers $\mu_0, \mu_1, \ldots, \mu_{n-2}, \mu_{n-1}$ defined via the formula

$$\mu_j = \sum_{k=0}^{n-1} \nu_k \, \overline{P}_{m+2k+1}^{m}(y_j) \tag{94}$$

for $j = 0, 1, \ldots, n-2, n-1$, where $y_0, y_1, \ldots, y_{n-2}, y_{n-1}$ are the zeros of $\overline{P}_{m+2n+1}^{m}$ from (29). Table 1 tests the transform with each of the numbers $\nu_0, \nu_1, \ldots, \nu_{n-2}, \nu_{n-1}$ being distributed uniformly on $(-1, 1)$, as obtained from a pseudorandom number generator. In Table 1, t_{direct} is the time in seconds required to apply a dense real $n \times n$ matrix to a real $n \times 1$ vector.

Table 2 lists the results of computing from real numbers $\beta_0, \beta_1, \ldots, \beta_{n-2}, \beta_{n-1}$ the real numbers $\alpha_0, \alpha_1, \ldots, \alpha_{n-2}, \alpha_{n-1}$ defined via the formula

$$\alpha_j = \sum_{k=0}^{n-1} \beta_k \, \overline{P}_{0+2k}^{0}(x_j) \tag{95}$$

for $j = 0, 1, \ldots, n-2, n-1$, where $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$ are the zeros of $\overline{P}_{0+2n}^{0}$ from (27). Table 2 tests the transform with each of the numbers $\beta_0, \beta_1, \ldots, \beta_{n-2}, \beta_{n-1}$ being distributed uniformly on $(-1, 1)$, as obtained from a pseudorandom number generator. In Table 2, t_{direct} is the time in seconds required to apply a dense real $n \times n$ matrix to a real $n \times 1$ vector.

Table 3 lists the results of computing from real numbers $\mu_0, \mu_1, \ldots, \mu_{n-2}, \mu_{n-1}$ the real numbers $\nu_0, \nu_1, \ldots, \nu_{n-2}, \nu_{n-1}$ satisfying

$$\mu_j = \sum_{k=0}^{n-1} \nu_k \, \overline{P}_{m+2k+1}^{m}(y_j) \tag{96}$$

for $j = 0, 1, \ldots, n-2, n-1$, where $y_0, y_1, \ldots, y_{n-2}, y_{n-1}$ are the zeros of $\overline{P}_{m+2n+1}^{m}$ from (29). Table 3 tests the transform with each of the numbers $\nu_0, \nu_1, \ldots, \nu_{n-2}, \nu_{n-1}$ being distributed uniformly on $(-1, 1)$, as obtained from a pseudorandom number generator. We calculated the input values for $\mu_0, \mu_1, \ldots, \mu_{n-2}, \mu_{n-1}$ using (96), calculating $\overline{P}_{m+2k+1}^{m}(y_j)$ for $j, k = 0, 1, \ldots, n-2, n-1$ via the recurrence relations (41) and (42) in extended-precision arithmetic (to compensate for the apparently mildly unstable recursion on the degrees of the normalized associated Legendre functions). In Table 3, t_{direct} is the time in seconds required to apply a dense real $n \times n$ matrix to a real $n \times 1$ vector.

Table 4 lists the results of computing from the values of a function f at the positive zeros $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$ of $\overline{P}^m_{m+2n}$ from (27) the values of f at the positive zeros $z_0, z_1, \ldots, z_{m/2+n-2}, z_{m/2+n-1}$ of $\overline{P}^0_{m+2n}$ from (31), where f is the function defined on $(-1, 1)$ via the formula

$$f(x) = \sum_{k=0}^{n-1} \beta_k \, \overline{P}^m_{m+2k}(x) \tag{97}$$

for some real numbers $\beta_0, \beta_1, \ldots, \beta_{n-2}, \beta_{n-1}$. Table 4 tests the transform with each of the numbers $\beta_0, \beta_1, \ldots, \beta_{n-2}, \beta_{n-1}$ being distributed uniformly on $(-1, 1)$, as obtained from a pseudorandom number generator. In Table 4, t_{direct} is the time in seconds required to apply a dense real $(\frac{m}{2} + n) \times n$ matrix to a real $n \times 1$ vector.

Table 5 lists the results of computing from the values of a function g at the positive zeros $y_0, y_1, \ldots, y_{n-2}, y_{n-1}$ of $\overline{P}^0_{0+2n+1}$ from (29) the values of g at the positive zeros $z_0, z_1, \ldots, z_{n-2}, z_{n-1}$ of $\overline{P}^0_{2n}$ from (31), where g is the function defined on $(-1, 1)$ via the formula

$$g(y) = \sum_{k=0}^{n-1} \nu_k \, \overline{P}^0_{0+2k+1}(y) \tag{98}$$

for some real numbers $\nu_0, \nu_1, \ldots, \nu_{n-2}, \nu_{n-1}$. Table 5 tests the transform with each of the numbers $\nu_0, \nu_1, \ldots, \nu_{n-2}, \nu_{n-1}$ being distributed uniformly on $(-1, 1)$, as obtained from a pseudorandom number generator. In Table 5, t_{direct} is the time in seconds required to apply a dense real $n \times n$ matrix to a real $n \times 1$ vector.

Table 6 lists the results of computing from the values of a function f at the positive zeros $z_0, z_1, \ldots, z_{m/2+n-2}, z_{m/2+n-1}$ of $\overline{P}^0_{m+2n}$ from (31) the values of f at the positive zeros $x_0, x_1, \ldots, x_{n-2}, x_{n-1}$ of $\overline{P}^m_{m+2n}$ from (27), where f is the function defined on $(-1, 1)$ via the formula

$$f(x) = \sum_{k=0}^{n-1} \beta_k \, \overline{P}^m_{m+2k}(x) \tag{99}$$

for some real numbers $\beta_0, \beta_1, \ldots, \beta_{n-2}, \beta_{n-1}$. Table 6 tests the transform with each of the numbers $\beta_0, \beta_1, \ldots, \beta_{n-2}, \beta_{n-1}$ being distributed uniformly on $(-1, 1)$, as obtained from a pseudorandom number generator. In Table 6, t_{direct} is the time in seconds required to apply a dense real $n \times (\frac{m}{2} + n)$ matrix to a real $(\frac{m}{2} + n) \times 1$ vector.

We wrote all code in Fortran 77, compiling it using the Lahey-Fujitsu compiler, with optimization flag `--o2` enabled. We ran all of the examples on a 2.8 GHz Pentium Xeon with 1 MB of L2 cache and 2 GB of RAM. We performed all precomputations in quadruple-precision arithmetic, as implemented in software by the Lahey-Fujitsu compiler. Aside from the precomputations, our code is compliant with the IEEE double-precision standard (so that the mantissas of variables have approximately one bit of precision less than 16 digits, yielding a relative precision of about .2e-15). For the fast multipole method (FMM) needed in the algorithms of Subsections 2.2 and 3.1, we used the FMM of [9]. For the FMM needed in the algorithm of Subsection 3.2, we used the "simple exponential-expansion FMM" algorithm described in Section 4 of [18].

Remark 4.1 The timings reported in Tables 1–6, as well as in our further experiments, appear to be consistent with the expected costs of the algorithms. In Tables 1–3, t_{fast} takes on values that are consistent with its expected values of a constant times $n \ln(n)$, for sufficiently large n. In Tables 4–6, t_{fast} takes on values that are consistent with its expected values of a constant times n, for sufficiently large n. In Tables 1–6, t_{direct} takes on values that are consistent with its expected values of a constant times n^2. In Tables 4–6, $t_{\text{precomps.}}$ takes on values that are consistent with its expected values of a constant times n, for sufficiently large n. In Tables 1–3, $t_{\text{precomps.}}$ takes on values that appear to scale as a constant times n^2. We were expecting $t_{\text{precomps.}}$ to scale as a constant times n^2 in Tables 1–3: to simplify our implementation, we used precomputations for the algorithm summarized in Subsection 2.2 that should scale as a constant times n^2. In principle, the methods of [7] and the last lemma of Section 2.2 in [16] can give rise to precomputations that would scale as a constant times $n \ln(n)$. We made no attempt to optimize the precomputations.

Remark 4.2 When $n = 32{,}768$, the algorithm of the present paper appears to be around 16 times faster than the direct algorithm. The method of the present paper would seem to be preferable to the algorithm of [11] whenever the method of the present paper is available, as this method is much more efficient, particularly at smaller problem sizes. However, while the algorithm of the present paper does generalize to certain classes of special functions not treated by [11] (see, for example, [16]), the algorithm of the present paper does not generalize directly to spheroidal wave functions (which are amenable to the method of [11]).

Notwithstanding this, it is possible that the algorithms of the present paper will generalize in a non-obvious manner. For instance, combining a fast associated Laguerre transform algorithm with the fact that the eigenfunctions of the Fourier-Bessel transform are associated Laguerre functions would yield a fast Fourier-Bessel transform algorithm, as pointed out to us by Michael O'Neil and Vladimir Rokhlin during personal communication in early 2007. (Please note here that the Fourier-Bessel transform is also known as the Hankel transform.) As described in [16], the algorithms of the present paper do indeed appear to generalize to associated Laguerre functions.

Acknowledgements

We would like to thank Vladimir Rokhlin for his support, for his numerous technical and editorial suggestions, and for sharing his extensive library of codes.

References

[1] M. Abramowitz, I. A. Stegun (eds.), Handbook of Mathematical Functions, Dover Publications, New York, 1972.
[2] J. C. Adams, P. N. Swarztrauber, SPHEREPACK 3.0: A model development facility, Mon. Wea. Rev. 127 (1999) 1872–1878.
[3] A. Glaser, A new class of highly accurate solvers for ordinary differential equations, Ph.D. thesis, Yale University, V. Rokhlin (Ph.D. advisor) (May 2007).
[4] A. Glaser, X. Liu, V. Rokhlin, A fast algorithm for the calculation of the roots of special functions, Tech. Rep. 1367, Dept. of Computer Science, Yale University (June 2006).
[5] G. H. Golub, J. H. Welsch, Calculation of Gauss quadrature rules, Math. Comput. 23 (106) (1969) 221–230 and s1–s10.
[6] M. Gu, S. C. Eisenstat, A stable and efficient algorithm for the rank-1 modification of the symmetric eigenproblem, SIAM J. Matrix Anal. Appl. 15 (1994) 1266–1276.
[7] M. Gu, S. C. Eisenstat, A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem, SIAM J. Matrix Anal. Appl. 16 (1995) 172–191.
[8] R. Jakob-Chien, B. Alpert, A fast spherical filter with uniform resolution, J. Comput. Phys. 136 (2) (1997) 580–584.
[9] P.-G. Martinsson, V. Rokhlin, An accelerated kernel-independent fast multipole method in one dimension, Tech. Rep. 1353, Dept. of Computer Science, Yale University (May 2006).
[10] W. Press, S. Teukolsky, W. Vetterling, B. Flannery, Numerical Recipes, 2nd ed., Cambridge University Press, Cambridge, UK, 1992.
[11] V. Rokhlin, M. Tygert, Fast algorithms for spherical harmonic expansions, SIAM J. Sci. Comput. 27 (6) (2006) 1903–1928.
[12] P. N. Swarztrauber, Shallow-water flow on the sphere, Mon. Wea. Rev. To appear.
[13] P. N. Swarztrauber, Spectral transform methods for solving the shallow-water equations on the sphere, Mon. Wea. Rev. 124 (4) (1996) 730–744.
[14] P. N. Swarztrauber, W. F. Spotz, Generalized discrete spherical harmonic transforms, J. Comput. Phys. 159 (2) (2000) 213–230.
[15] G. Szegö, Orthogonal Polynomials, vol. 23 of Colloquium Publications, eleventh ed., American Mathematical Society, Providence, RI, 2003.
[16] M. Tygert, Recurrence relations and fast algorithms, Tech. Rep. 1343, Dept. of Computer Science, Yale University, also available in revised form as cs.CE/0609081 at http://www.arxiv.org/ (December 2005).
[17] N. Yarvin, V. Rokhlin, A generalized one-dimensional fast multipole method with application to filtering of spherical harmonics, J. Comput. Phys. 147 (2) (1998) 594–609.
[18] N. Yarvin, V. Rokhlin, An improved fast multipole method for potential fields on the line, SIAM J. Numer. Anal. 36 (2) (1999) 629–666.

Table 1

Times in seconds and relative errors for synthesis of odd degrees

n	m	t_{fast}	t_{direct}	$\frac{t_{\text{direct}}}{t_{\text{fast}}}$	$t_{\text{precomps.}}$	$\varepsilon_{\text{r.m.s.}}$
512	512	.81e-03	.97e-03	1.2	.36e+02	.18e-12
1024	1024	.24e-02	.37e-02	1.5	.19e+03	.21e-11
2048	2048	.65e-02	.14e-01	2.2	.77e+03	.26e-11
4096	4096	.16e-01	.56e-01	3.5	.34e+04	.52e-11
8192	8192	.40e-01	.27e-00	6.8	.12e+05	.39e-10
16384	16384	.93e-01	(.11e+01)	12	.59e+05	.19e-10
32768	32768	.21e-00	(.43e+01)	20	.20e+06	.46e-10

Table 2

Times in seconds and relative errors for synthesis of even degrees

n	m	t_{fast}	t_{direct}	$\frac{t_{\text{direct}}}{t_{\text{fast}}}$	$t_{\text{precomps.}}$	$\varepsilon_{\text{r.m.s.}}$
512	0	.82e-03	.97e-03	1.2	.33e+02	.14e-11
1024	0	.25e-02	.37e-02	1.5	.15e+03	.37e-11
2048	0	.66e-02	.14e-01	2.2	.65e+03	.48e-11
4096	0	.18e-01	.60e-01	3.5	.27e+04	.11e-10
8192	0	.40e-01	.27e-00	6.8	.13e+05	.28e-10
16384	0	.10e-00	(.11e+01)	11	.44e+05	.42e-10
32768	0	.22e-00	(.43e+01)	20	.19e+06	.81e-10

Table 3

Times in seconds and relative errors for analysis of odd degrees

n	m	t_{fast}	t_{direct}	$\frac{t_{\text{direct}}}{t_{\text{fast}}}$	$t_{\text{precomps.}}$	$\varepsilon_{\text{r.m.s.}}$
512	512	.82e-03	.93e-03	1.1	.37e+02	.12e-13
1024	1024	.25e-02	.36e-02	1.4	.17e+03	.51e-13
2048	2048	.65e-02	.14e-01	2.2	.83e+03	.67e-13
4096	4096	.16e-01	.56e-01	3.5	.34e+04	.68e-13
8192	8192	.40e-01	.27e-00	6.8	.12e+05	.15e-12
16384	16384	.93e-01	(.11e+01)	12	.52e+05	.27e-12
32768	32768	.24e-00	(.43e+01)	18	.22e+06	.18e-12

TABLE 4
Times in seconds and relative errors for interpolation from zeros of $\overline{P}^m_{m+2n}$ to zeros of $\overline{P}^0_{m+2n}$

n	m	t_{fast}	t_{direct}	$\frac{t_{\text{direct}}}{t_{\text{fast}}}$	$t_{\text{precomps.}}$	$\varepsilon_{\text{r.m.s.}}$
512	512	.89e-03	.14e-02	1.6	.20e+03	.60e-13
1024	1024	.19e-02	.58e-02	3.1	.39e+03	.66e-13
2048	2048	.37e-02	.24e-01	6.5	.79e+03	.94e-13
4096	4096	.74e-02	.93e-01	13	.16e+04	.17e-12
8192	8192	.15e-01	.42e-00	28	.32e+04	.31e-12
16384	16384	.30e-01	(.17e+01)	57	.64e+04	.60e-12
32768	32768	.60e-01	(.67e+01)	112	.13e+05	.12e-11

TABLE 5
Times in seconds and relative errors for interpolation from zeros of $\overline{P}^0_{2n+1}$ to zeros of $\overline{P}^0_{2n}$

n	m	t_{fast}	t_{direct}	$\frac{t_{\text{direct}}}{t_{\text{fast}}}$	$t_{\text{precomps.}}$	$\varepsilon_{\text{r.m.s.}}$
512	0	.66e-03	.88e-03	1.3	.17e+03	.42e-12
1024	0	.13e-02	.36e-02	2.8	.33e+03	.34e-11
2048	0	.27e-02	.15e-01	5.6	.66e+03	.63e-11
4096	0	.54e-02	.62e-01	11	.13e+04	.14e-10
8192	0	.11e-01	.27e-00	25	.27e+04	.18e-10
16384	0	.22e-01	(.11e+01)	50	.52e+04	.31e-10
32768	0	.44e-01	(.43e+01)	98	.11e+05	.90e-10

TABLE 6
Times in seconds and relative errors for interpolation from zeros of $\overline{P}^0_{m+2n}$ to zeros of $\overline{P}^m_{m+2n}$

n	m	t_{fast}	t_{direct}	$\frac{t_{\text{direct}}}{t_{\text{fast}}}$	$t_{\text{precomps.}}$	$\varepsilon_{\text{r.m.s.}}$
512	512	.81e-03	.14e-02	1.7	.20e+03	.45e-13
1024	1024	.15e-02	.54e-02	3.6	.39e+03	.55e-13
2048	2048	.32e-02	.22e-01	6.9	.79e+03	.78e-13
4096	4096	.65e-02	.92e-01	14	.16e+04	.14e-12
8192	8192	.15e-01	.41e-00	27	.32e+04	.28e-12
16384	16384	.35e-01	(.16e+01)	46	.64e+04	.55e-12
32768	32768	.83e-01	(.66e+01)	80	.13e+05	.11e-11