# Rapid Application of the Spherical Harmonic Transform via Interpolative Decomposition Butterfly Factorization

James Bremer
Department of Mathematics, University of California, Davis, California, USA

Ze Chen
Department of Mathematics, National University of Singapore, Singapore

Haizhao Yang
Department of Mathematics, Purdue University, USA

April 22, 2020

**Abstract**

We describe an algorithm for the application of the forward and inverse spherical harmonic transforms. While existing methods have total running times (including all precomputations) which grow at least as fast as $\mathcal{O}\left(N^3\right)$, where $N$ is the degree of the transform, the computational complexity of our method is $\mathcal{O}\left(N^2 \log^3(N)\right)$. It is based on a new method for rapidly computing the Legendre Transform (LT) by hierarchically applying the interpolative decomposition butterfly factorization (IDBF). Numerical results are provided to demonstrate the effectiveness and numerical stability of the new framework.

**Keywords.** Spherical harmonic transform, Legendre transform, block partitioning, butterfly factorization, interpolative decomposition, randomized algorithm

## 1 Introduction

### 1.1 Problem statement

This paper is concerned with the efficient application of the forward and inverse spherical harmonic transforms (SHT). These transformations play an important role in many scientific computing applications, including in the fields of numerical weather prediction and climate modeling [36, 33, 35], and are significant components in many numerical algorithms. The forward SHT of degree $N$ maps the coefficients in the expansion

$$f(\theta,\phi) = \sum_{k=0}^{2N-1} \sum_{m=-k}^{k} \beta_{k,m} \overline{P}_k^{|m|}(\cos(\theta)) e^{im\phi}, \tag{1}$$

where $\overline{P}_k^m(x)$ denotes the $L^2$ normalized associated Legendre function of order $m$ and degree $k$, to the values of the function $f(\theta,\phi)$ at a grid discretization nodes formed from the tensor product of an $\mathcal{O}(N)$ point trapezoidal quadrature rule in the $\phi$ variable and an $\mathcal{O}(N)$ point Gauss-Legendre quadrature in variable $x = \cos(\theta)$. The inverse SHT is, of course, the mapping which takes the values of the function $f(\theta,\phi)$ at the discretization nodes to the coefficients in the expansion (1).

If we let

$$g(m,\theta) = \sum_{k=|m|}^{2N-1} \beta_{k,m} \overline{P}_k^{|m|}(\cos(\theta)), \tag{2}$$

then

$$f(\theta,\phi) = \sum_{m=-2N+1}^{2N-1} g(m,\theta) e^{im\phi}. \tag{3}$$

Assuming the values of $g(m,\theta)$ are known for each $m = -2N+1, \ldots, 2N-1$ and each of the $\mathcal{O}(N)$ relevant values of $\theta$, the function $f(\theta,\phi)$ can be calculated at the $\mathcal{O}\left(N^2\right)$ nodes of the discretization grid in $\mathcal{O}\left(N^2 \log(N)\right)$ operations by applying $\mathcal{O}(N)$ fast Fourier transforms.

The mapping which, for a fixed $m$, takes the values of the coefficients in the expansion (2) to the values of $g(m,\theta)$ at the $\mathcal{O}(N)$ relevant values of $\theta$ is known as the Legendre transform (LT). The straightforward evaluation of a single LT takes $\mathcal{O}\left(N^2\right)$ operations, and $\mathcal{O}(N)$ such transforms must be applied in order to evaluate the SHT. This has motivated a signicant amount of research aimed at accelerating the Legendre Transform [22, 30, 31, 36, 33, 27, 18, 26]. Most existing methods for applying it rapidly are divided into an expensive precomputation phase and a much faster application phase. And while there are robust algorithms for the LT whose application phase runs in time which is nearly linear in $N$, to the best of our knowledge, there is no existing accurate and stable method which has nearly linear computational complexity when the costs of all calculations, including precomputations, are included.

In this paper, we propose a new method for applying the LT whose total complexity (including both the precomputation and application phases) is $\mathcal{O}\left(N \log^3(N)\right)$, and which leads to an SHT whose running time grows as $\mathcal{O}\left(N^2 \log^3(N)\right)$. This is achieved by hierarchically applying the butterfly algorithm (BF) [14, 17, 19, 13, 20, 15] and randomized low-rank approximation to speed up the calculation of the LT.

Butterfly algorithms are a collection of techniques for rapidly applying the matrices which result from discretizing oscillatory integral operators. They exploit the fact that these matrices have the complementary low-rank property (see [14] for a definition). A large class of special function transforms are integral operators of the appropriate type [19], and consequently can be applied rapidly using buttefly algorithms. Indeed, in the special case $m = 0$, the LT can be applied via standard butterfly algorithms in $\mathcal{O}(N \log(N))$ time. These results do not, however, extend to the case $m > 0$. In that event, the associated Legendre functions are not oscillatory on the entire

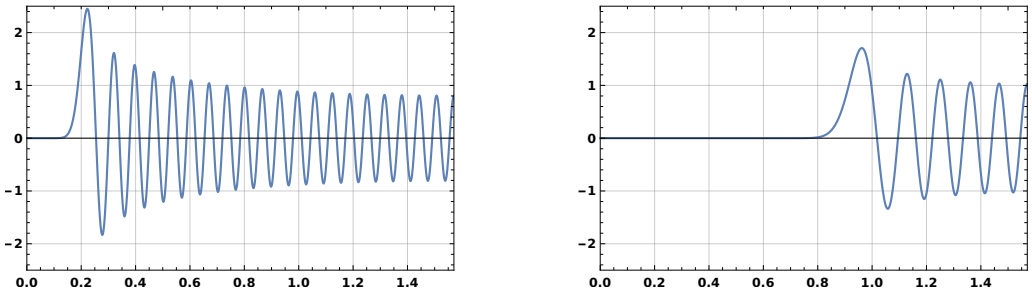



Figure 1: On the left is a plot of the $L^2$ normalized associated Legendre function $\tilde{P}_n^m(\cos(\theta))$ in the case $n = 100$ and $m = 20$. On the right is a plot of $\tilde{P}_n^m(\cos(\theta))$ when $n = 100$ and $m = 80$.

domain of interest. Instead, $\tilde{P}_n^m(\cos(\theta))$ is nonoscillatory on the interval

$$\left(0, \arcsin\left(\frac{\sqrt{m^2-1/4}}{n+1/2}\right)\right) \tag{4}$$

and oscillatory on

$$\left(\arcsin\left(\frac{\sqrt{m^2-1/4}}{n+1/2}\right), \frac{\pi}{2}\right) \tag{5}$$

(see Figure 1 plots of $\tilde{P}_n^m(\cos(\theta))$ for two different pairs of the parameters $n$ and $m$). As a consequence, the integral operator associated with the Legendre Transform when $m > 0$ is not of the purely oscillatory type whose discretizations have the complementary low rank property and are therefore amenable to rapid application via butterfly algorithms.

In order to overcome this difficulty we apply the following methodology:

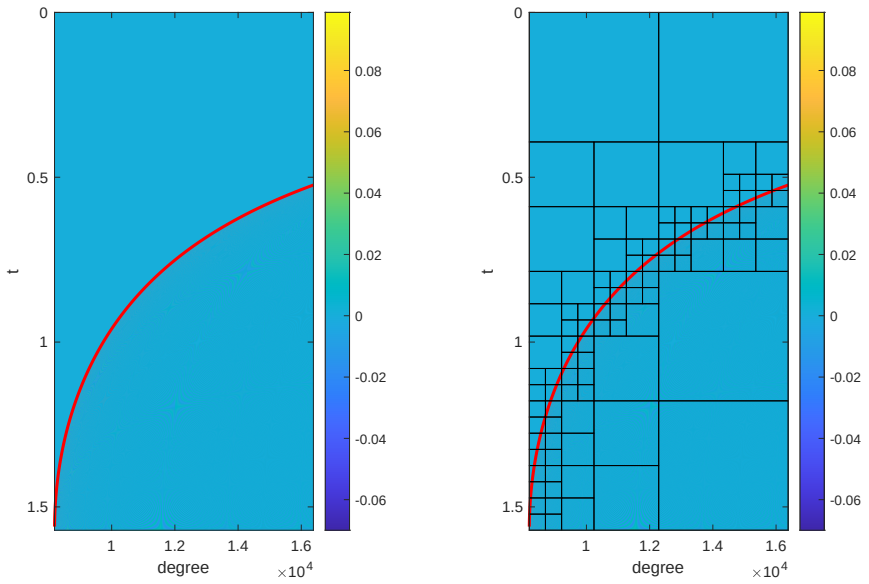
- We hierarchically partition the transformation matrix into purely oscillatory and purely nonoscillatory blocks (see Figure 2 (b)).
- In the purely nonoscillatory blocks, the corresponding matrix is numerically low-rank and hence its application to a vector can be accelerated to obtain linear scaling by randomized low-rank approximation algorithms.
- The matrices corresponding to purely oscillatory blocks admit complementary low-rank structures, the application of which to a vector can be accelerated via butterfly algorithms. We use the relatively new interpolative decomposition butterfly factorization (IDBF) [20], which yields nearly linear scaling in the degree $N$ of the LT transform.

Using this approach, we are able to apply each LT in $\mathcal{O}\left(N\log^3(N)\right)$ operations, and the SHT of degree $N$ in $\mathcal{O}\left(N^2\log^3(N)\right)$ operations. This includes the cost of both the precomputation and application phases of these algorithms. To the best of our knowledge, ours is the first stable and accurate method for applying the SHT whose running time — when all phases of the algorithm are accounted for — grows more slowly than $\mathcal{O}\left(N^3\right)$.

The scheme relies heavily on the algorithm of [6] for the numerical evaluation of the associated Legendre functions. That algorithm allows each evaluation to be performed in time independent of the parameters $n$ and $m$. If standard methods for calculating $\tilde{P}_n^m$, which have running times which grow with the parameters $n$ and $m$, were used instead, the running time of our algorithm for applying the LT would no longer scale as $\mathcal{O}\left(N\log^3(N)\right)$.

### 1.2 Related works

Many algorithms for special function transforms have been proposed. One class of such algorithms, which include [12, 4, 21, 3, 16, 24, 23], first compute the coefficients in a Chebyshev expansion of a function $f$ and then apply connection matrices or a series of connection matrices to obtain the coefficients of the expansion of $f$ in terms of a different class of Jacobi polynomials. There are a number of well-known techniques for computing the Chebyshev expansion in $\mathcal{O}\left(N\log\left(N\right)\right)$ operations, and, in certain cases, the connection matrices can be rapidly applied. For instance, the $N^{th}$ order Chebyshev-Legendre transform, which is the mapping which takes coefficients in the Chebyshev expansion of an $N^{th}$ degree polynomial to the coefficients in the Legendre expansion of the same polynomial, can be applied in $\mathcal{O}\left(N\right)$ operations using the fast multiple method [2]. However,




(a) (b)

Figure 2: An illustration of the partitioning process of an odd matrix for the forward matvec in the Legendre transform, when $N = 8192$ and order $m = 8192$. (a) The odd matrix with a piecewise continuous curve (red color) including turning points. (b) The hierarchically partitioned blocks of the odd matrix.

all existing algorithms for computing the LT in the case $m > 0$ using such an approach (e.g.,[24]) have total running times which grow at least as fast as $\mathcal{O}\left(N^2\right)$ including the precomputation.

Another class of algorithms for applying Jacobi transforms, which includes [8, 9], are based on the FFT and have computational complexity $\mathcal{O}\left(N \log^2(N) / \log(\log(N))\right)$. More recently, a method based on Toeplitz and Hankel matrices [29] is presented to accelerate the Chebyshev-Legendre transform with an observed complexity of $\mathcal{O}\left(N \log^2(N)\right)$. However, these algorithms only apply to the LT of particular orders (such as $m = 0$ or $m = 1$) and, at least in their current forms, not sufficient to accelerate the SHT.

It is shown in [31] that the application of the SHT can be accelerated by the BF. In particular, the algorithm in [31] applies the BF in [19] to evaluate the LTs in the SHT, each of which takes $\mathcal{O}\left(N^2\right)$ and $\mathcal{O}\left(N \log^3(N)\right)$ operations in the precomputation and application, respectively. This, of course, results in an SHT with a cost of $\mathcal{O}\left(N^3\right)$ for precomputation and $\mathcal{O}\left(N^2 \log^3(N)\right)$ cost for the application. Though the application of this fast SHT is nearly optimal, its precomputation might still be prohibitive when $N$ is large. More recently, a variant of [31] was proposed in [36] with the idea that partitioning the LT transformation matrix in an appropriate manner, e.g., see Figure 3, can lead to a smaller computational complexity in theory, which is $\mathcal{O}\left(N^2 \log^2(N) / \log(\log(N))\right)$, and an accuracy with one more digit better than that in [31]. However, the precomputation in [36] still takes $O(N^3)$ operations and the actual application complexity is still $\mathcal{O}\left(N^2 \log^3(N)\right)$. Another kind of methods based on the BF is introduced in [25], which derives a rapid transformation between spherical harmonic expansions and bivariate Fourier series via the BF and hierarchically off-diagonal low-rank matrix decompositions. Although it is rigorously proved that the application time is $\mathcal{O}\left(N^2 \log^2(N)\right)$, the total precomputation requires $\mathcal{O}\left(N^3 \log(N)\right)$ flops.

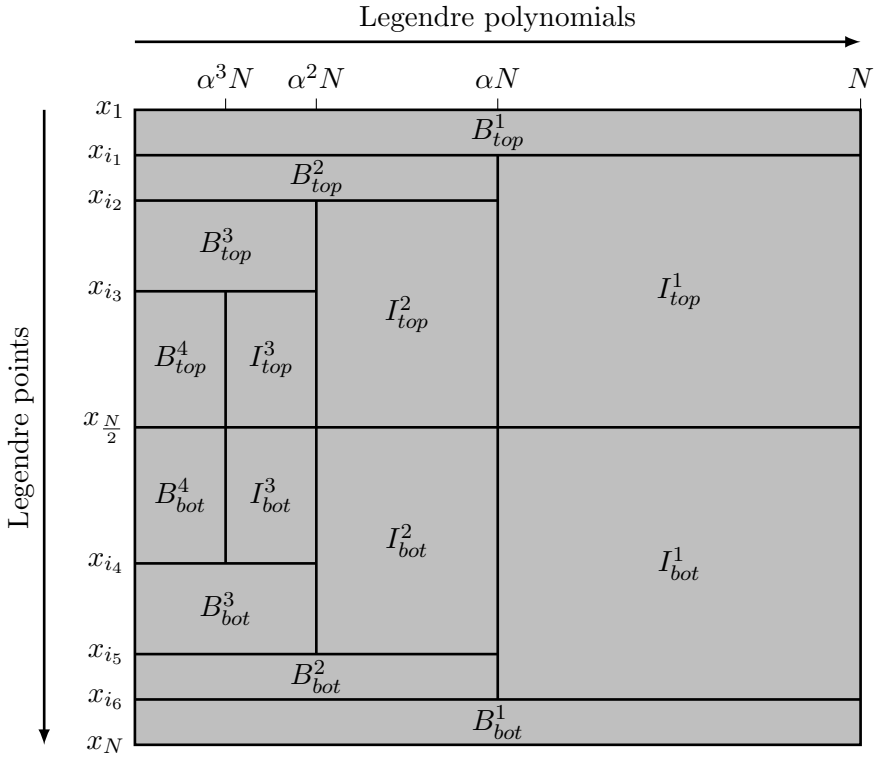



Figure 3: Block partitioning of the Legendre-Vandermonde matrix in [36], when $N = 1024$. $x_i$, $i = 1, 2, \ldots, N$, are the Legendre points. The parameters $i_1, i_2, \ldots, i_6$ and $\alpha$ are the computed partitioning coefficients, which are able to divide the Legendre-Vandermonde matrix into boundary parts (denoted by symbol $B$) and internal (denoted by symbol $I$) parts. The internal parts can be compressed by the BF while the boundary parts are directly computed for the corresponding matvecs.

### 1.3 Overview of this paper

The rest of the paper is organized as follows. Section 2 summarizes the spherical harmonic transforms. Section 3 revisits existing low-rank matrix factorization techniques. Section 4 proposes a new algorithm based on the previous factorization techniques, and analysis the complexity of the proposed algorithm. Finally, Section 5 describes several numerical examples to demonstrate the efficiency of the proposed algorithm in Section 4. For simplicity, we adopt MATLAB notations for the algorithm described in this paper: given row and column index sets $I$ and $J$, $K(I, J)$ is the submatrix with entries from rows in $I$ and columns in $J$; the index set for an entire row or column is denoted as " : ".

## 2 An overview of spherical harmonic transforms

The *spherical harmonic expansion* of a bandlimited function $f$ on the surface of the sphere has the form

$$f(\theta, \phi) = \sum_{k=0}^{2N-1} \sum_{m=-k}^{k} \beta_{k,m} \overline{P}_k^{|m|}(\cos(\theta)) e^{im\phi}, \tag{6}$$

where

$$\overline{P}_k^{|m|}(x) = \sqrt{\frac{2k+1}{2}\frac{(k-|m|)!}{(k+|m|)!}}\left(1-x^2\right)^{\frac{m}{2}}\frac{d^m}{dx^m}P_k(x), \text{ for } x \in (-1,1), \tag{7}$$

is the normalized associated Legendre function of degree $k$ and order $|m|$ (for example, Chapter 8 of [1]), with the Legendre polynomial of degree $k$,

$$P_k(x) = \frac{1}{2^k k!}\frac{d^k}{dx^k}\left((x^2-1)^k\right). \tag{8}$$

Parameters $(\theta, \phi)$ are the standard spherical coordinates on the two-dimensional surface of the unit sphere in $\mathbb{R}^3$, $\theta \in (0, \pi)$ and $\phi \in (0, 2\pi)$.

The first scenario for the Equation (6) is evaluating the coefficients $\beta_{k,m}$ when the function $f$ is given by a table of its values at a collection of appropriately chosen nodes on the two-dimensional surface of the unit sphere. This scenario is known as the *forward spherical harmonic transform*, which is applied for using spectral methods for the numerical solution of partial differential equations. Conversely, the second scenario is evaluating the function $f$ at a collection of points on the surface of the sphere when the coefficients $\beta_{k,m}$ are given, which is called by the *inverse spherical harmonic transform*.

In general, a standard discretization of the surface of the sphere is the "tensor product", consisting of all pairs of the form $(\theta_l, \phi_j)$, with $\cos(\theta_0), \cos(\theta_1), \ldots, \cos(\theta_{2N-2}), \cos(\theta_{2N-1})$ being the Gauss-Legendre quadrature nodes of degree $2N$, that is,

$$-1 < \cos(\theta_0) < \cos(\theta_1) < \cdots < \cos(\theta_{2N-2}) < \cos(\theta_{2N-1}) < 1 \tag{9}$$

and

$$\overline{P}_{2N}^0(\cos(\theta_l)) = 0, \text{ for } l = 0, 1, \ldots, 2N-2, 2N-1, \tag{10}$$

and with $\phi_0, \phi_1, \ldots, \phi_{4N-3}, \phi_{4N-2}$ being equispaced on the interval $(0, 2\pi)$, that is,

$$\phi_j = \frac{2\pi(j+\frac{1}{2})}{4N-1}, \text{ for } j = 0, 1, \ldots, 4N-3, 4N-2. \tag{11}$$

This standard discretization results in that the direct computation of both the forward or the inverse SHT takes $\mathcal{O}\left(N^3\right)$ operations and is prohibitive in large-scale computation.

When a function $f$ defined on the two-dimensional surface of the unit sphere in (6) is given, the form of spherical harmonic expansion can be rewritten as

$$f(\theta, \phi) = \sum_{m=-2N+1}^{2N-1} g(\theta, m)e^{im\phi}, \tag{12}$$

where

$$g(\theta, m) = \sum_{k=|m|}^{2N-1} \beta_{k,m}\overline{P}_k^{|m|}(\cos(\theta)). \tag{13}$$

According to the standard discretization, the inverse SHT involves $2N$ values of the parameter $\theta$: $\theta_0, \theta_1, \ldots, \theta_{2N-2}, \theta_{2N-1}$. Then, the complexity of evaluating the *Legendre transform* $g(\theta, m)$ with a fixed order $m$ in (13), is proportional to $2(2N-|m|)N$, which is $\mathcal{O}\left(N^2\right)$ on average. Thus, it costs $\mathcal{O}\left(N^3\right)$ operations for evaluating all sums over different order $m$ in (13). Once $g(\theta, m)$ are given by $\theta = \theta_0, \theta_1, \ldots, \theta_{2N-2}, \theta_{2N-1}$ and $m = -2N+1, -2N+2, \ldots, 2N-2, 2N-1$, the fast Fourier transform (FFT) [32] can evaluate (12) in $\mathcal{O}\left(N^2 \log(N)\right)$ operations. Therefore, the computational

complexity for the whole process of the inverse SHT is bounded by $\mathcal{O}\left(N^3\right)$. A virtually identical calculation shows that the cost of evaluating the forward SHT is also $\mathcal{O}\left(N^3\right)$.

In this paper, we will focus on the LTs in (13) with different order $m$'s and propose a fast algorithm with nearly linear computational complexity for evaluating the LTs with a fixed order $m$. Finally, both the forward and the inverse SHT can be evaluated with an $\mathcal{O}\left(N^2 \log^3(N)\right)$ scaling.

# 3 Preliminaries

In this section, we summarize certain facts and algorithms from mathematical and numerical analysis, which will be repeatedly applied in our method in Section 4. Subsection 3.1 outlines the linear scaling interpolative decomposition (ID) method, which is an important tool for the interpolative decomposition butterfly factorization (IDBF) in Subsection 3.2. Subsection 3.3 describes the low-rank approximation by randomized sampling.

## 3.1 Linear scaling Interpolative Decomposition (ID)

This subsection introduces the linear scaling ID method which has been proposed in [20].

Suppose $K \in \mathbb{C}^{m\times n}$ has a numerical rank $k_\epsilon \ll \min\{m,n\}$. Let $s$ be an index set containing $tk$ rows of $K$ chosen from the Mock-Chebyshev grids as in [34, 11, 5] or randomly sampled points, $t$ is an oversampling parameter, and $k$ is an empirical estimation of $k_\epsilon$.

We apply the rank revealing thin QR to $K(s,:)$:

$$K(s,:)\Lambda = QR = Q[R_1\ R_2] \qquad \text{with} \qquad R_1 \in \mathbb{C}^{tk\times tk} \text{ and } R_2 \in \mathbb{C}^{tk\times(n-tk)}.$$

Define

$$T = (R_1(1:k,1:k))^{-1}[R_1(1:k,k+1:kt)\ R_2(1:k,:)] \in \mathbb{C}^{k\times(n-k)},$$

and $V = [I\ T]\Lambda^* \in \mathbb{C}^{k\times n}$. Let $q$ be the index set with $|q| = k$ such that

$$K(s,q) = QR_1(1:k,1:k),$$

then $q$ and $V$ satisfy

$$K(s,:) \approx K(s,q)V \tag{14}$$

with an approximation error by the QR truncation. By the approximation power of Lagrange interpolation with Mock-Chebyshev points or randomly selected points, we have

$$K \approx K(:,q)V \tag{15}$$

with an approximation error coming from the QR truncation and the Lagrange interpolation. Hence, $K(:,q)$ are important columns of $K$ such that they can be "interpolated" back to $K$ via a *column interpolation matrix* $V$. In this sense, $q$ is called the *skeleton* index set, and the rest of indices are called *redundant* indices. This column ID requires only $\mathcal{O}\left(nk^2\right)$ operations and $\mathcal{O}\left(nk\right)$ memories and is denoted as *cID* for short.

Similarly, a row ID with $\mathcal{O}\left(mk^2\right)$ operations and $\mathcal{O}\left(mk\right)$ memories, denoted as *rID*, can be constructed via

$$K \approx \Lambda[I\ T]^* K(q,:) := UK(q,:) \tag{16}$$

with a *row interpolation matrix* $U$.

### 3.2 Interpolative Decomposition Butterfly Factorization (IDBF)

A nearly linear scaling algorithm for constructing the low-rank factorization of a complementary low-rank matrix $K \in \mathbb{C}^{M \times N}$ with $M \approx N$ has been proposed in [20]. In this subsection, we revisit the algorithm IDBF in [20], which will be repeatedly applied.

Let us recall the definition of complementary low-rank matrices in [14]. For such a matrix, we construct two trees $T_X$ and $T_\Omega$ for point sets $X$ and $\Omega$, respectively, assuming that both trees have the same depth $L = \mathcal{O}(\log(N))$, with the top-level being level 0 and the bottom one being level $L$ (see Figure 4 for an illustration). Such a matrix $K$ of size $N \times N$ is said to satisfy the *complementary low-rank property* if for any level $\ell$, any node $A$ in $T_X$ at level $\ell$, and any node $B$ in $T_\Omega$ at level $L - \ell$, the submatrix $K(A, B)$, obtained by restricting $K$ to the rows indexed by the points in $A$ and the columns indexed by the points in $B$, is numerically low-rank. See Figure 5 for an illustration of low-rank submatrices in a 1D complementary low-rank matrix of size $16 \times 16$ with uniform point distributions in $X$ and $\Omega$.

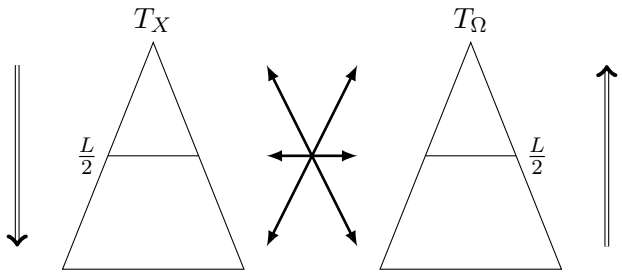



Figure 4: Trees of the row and column indices. Left: $T_X$ for the row indices $X$. Right: $T_\Omega$ for the column indices $\Omega$. The interaction between $A \in T_X$ and $B \in T_\Omega$ starts at the root of $T_X$ and the leaves of $T_\Omega$.

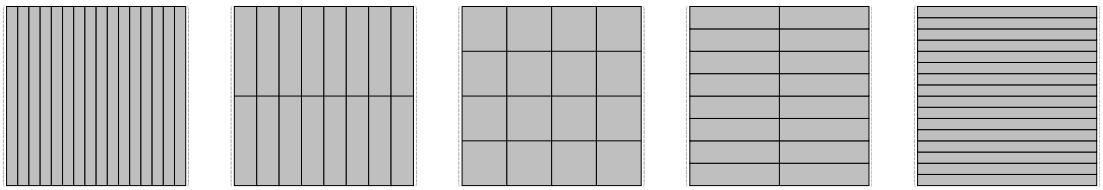


Figure 5: Hierarchical decomposition of the row and column indices of a $16 \times 16$ matrix. The dyadic trees $T_X$ and $T_\Omega$ have roots containing 16 rows and 16 columns respectively, and their leaves containing only a single row or column. The partition above indicates the complementary low-rank property of the matrix, and assumes that each submatrix is rank-1.

Given an $N \times N$ matrix $K$, or equivalently an $\mathcal{O}(1)$ algorithm to evaluate an arbitrary entry of $K$, as well as an adaptive rank $r_k$ for ID, IDBF aims at constructing a data-sparse representation of $K$ using the ID of low-rank submatrices in the complementary low-rank structure as follows:

$$K \approx U^L U^{L-1} \cdots U^h S^h V^h \cdots V^{L-1} V^L, \tag{17}$$

where the depth $L = \mathcal{O}(\log(N))$ is assumed to be even without loss of generality, $h = \frac{L}{2}$ is a middle level index, and all factors are sparse matrices with $\mathcal{O}(N)$ nonzero entries. The factorization and application of IDBF requires only $\mathcal{O}(N \log(N))$ memories and time.

### 3.3 Low-rank approximation by randomized sampling

We revisit an existing low-rank matrix factorization with linear complexity in this subsection.

For $A \in \mathbb{C}^{m\times n}$, a rank-$r$ approximate singular value decomposition (SVD) of $A$ is defined as

$$A \approx U\Sigma V^T, \tag{18}$$

where $U \in \mathbb{C}^{m\times r}$ is orthogonal, $\Sigma \in \mathbb{R}^{r\times r}$ is diagonal, and $V \in \mathbb{C}^{n\times r}$ is orthogonal, and $r = \mathcal{O}(1)$ independent of the matrix size $m$ and $n$ with a prefactor depending only on the approximation error $\epsilon$. Previously, [7, 10] have proposed efficient randomized tools to compute approximate SVDs for numerically low-rank matrices. The method in [7] is more attractive because it only requires $\mathcal{O}(1)$ randomly sampled rows and columns of $A$ for constructing (18) with $\mathcal{O}(m+n)$ operations and memories complexity, and it is observed that $|A(i,j) - (U\Sigma V^T)(i,j)| = \mathcal{O}(\epsilon)$ in a probabilistic sense, where $1 \le i \le m$ and $1 \le j \le n$.

The method is denoted as Function `randomizedSVD` and is presented in Algorithm 1. Assuming the whole low-rank matrix $A$ is known, the input of Function `randomizedSVD` is $A$, $\mathcal{O}(1)$ randomly sampled row indices $\mathcal{R}$ and column indices $\mathcal{C}$, as well as a rank parameter $r_\epsilon$ based on the error $\epsilon$. Equivalently, it can also be assumed that $A(\mathcal{R},:)$ and $A(:,\mathcal{C})$ are known as the inputs. Let $r$ be an empirical estimation of $r_\epsilon$, then the outputs are three matrices $U \in \mathbb{C}^{m\times r}$, $\Sigma \in \mathbb{R}^{r\times r}$, and $V \in \mathbb{C}^{n\times r}$ satisfying (18). In Function `randomizedSVD`, for simplicity, given any matrix $K \in \mathbb{C}^{s\times t}$, Function `qr(`$K$`)` performs a pivoted QR decomposition $K(:,P) = QR$, where $P$ is a permutation vector of the $t$ columns, $Q$ is a unitary matrix, and $R$ is an upper triangular matrix with positive diagonal entries in decreasing order. Function `randperm(`$m$`,`$r$`)` denotes an algorithm that randomly selects $r$ different samples in the set $\{1, 2, \ldots, m\}$. If necessary, we can add an over sampling parameter $q$ such that we sample $rq$ rows and columns and only generate a rank $r$ truncated SVD in Line 10 in Algorithm 1. Larger $q$ results in better stability of Algorithm 1.

1 **Function** $[U, \Sigma, V] \leftarrow$ `randomizedSVD(`$A, \mathcal{R}, \mathcal{C}, r$`)`
2 $[m, n] \leftarrow \mathrm{size}(A)$
3 $P \leftarrow$ `qr(`$A(\mathcal{R},:)$`)` ; $\Pi_{col} \leftarrow P(1:r)$ // $A(\mathcal{R}, P) = QR$
4 $P \leftarrow$ `qr(`$A(:,\mathcal{C})^T$`)` ; $\Pi_{row} \leftarrow P(1:r)$ // $A(P, \mathcal{C}) = R^T Q^T$
5 $Q \leftarrow$ `qr(`$A(:,\Pi_{col})$`)` ; $Q_{col} \leftarrow Q(:,1:r)$ // $A(P, \Pi_{col}) = QR$
6 $Q \leftarrow$ `qr(`$A(\Pi_{row},:)^T$`)` ; $Q_{row} \leftarrow Q(:,1:r)$ // $A(\Pi_{row}, P) = R^T Q^T$
7 $S_{row} \leftarrow$ `randperm(`$m, r$`)` ; $I \leftarrow [\Pi_{row}, S_{row}]$
8 $S_{col} \leftarrow$ `randperm(`$n, r$`)` ; $J \leftarrow [\Pi_{col}, S_{col}]$
9 $M \leftarrow (Q_{col}(I,:))^\dagger A(I,J) \left(Q_{row}^T(:,J)\right)^\dagger$ // $(\cdot)^\dagger$ : `pseudo-inverse`
10 $[U_M, \Sigma_M, V_M] \leftarrow \mathrm{svd}(M)$
11 $U \leftarrow Q_{col}U_M$ ; $\Sigma \leftarrow \Sigma_M$ ; $V \leftarrow Q_{row}V_M$

**Algorithm 1:** Randomized sampling for a rank-$r$ approximate SVD with $\mathcal{O}(m+n)$ operations, such that $A \approx U\Sigma V^T$.

## 4 Block partitioning algorithm

In this section, we propose a nearly linear block partitioning algorithm based on IDBF and low-rank approximation by randomized sampling for evaluating LTs in three steps. Several classical facts concerning normalized associated Legendre functions will be discussed in each step when necessary.

### 4.1 Odd and even Legendre transform matrices

First of all, the costs of both the forward and the inverse SHT can be reduced by 50% by the property of the normalized associated Legendre functions.

Let us combine (7) and (8) to yield

$$\begin{cases} \overline{P}_k^m(x) = -\overline{P}_k^m(-x), & k-m \text{ is odd}, \\ \overline{P}_k^m(x) = \overline{P}_k^m(-x), & k-m \text{ is even}. \end{cases} \tag{19}$$

Then, the LT in (13) can be split into two groups as

$$\begin{aligned} g(\theta, m) &= \sum_{k=|m|}^{2N-1} \beta_{k,m} \overline{P}_k^{|m|}(\cos(\theta)) \\ &= \sum_{k=0}^{2N-|m|-1} \beta_{k+|m|,m} \overline{P}_{k+|m|}^{|m|}(\cos(\theta)) \\ &= \sum_{0 \le k \le 2N-|m|-1,\, k \text{ is odd}} \beta_{k+|m|,m} \overline{P}_{k+|m|}^{|m|}(\cos(\theta)) \\ &\quad + \sum_{0 \le k \le 2N-|m|-1,\, k \text{ is even}} \beta_{k+|m|,m} \overline{P}_{k+|m|}^{|m|}(\cos(\theta)) \\ &=: g_1(\theta, m) + g_2(\theta, m). \end{aligned} \tag{20}$$

We denote the group of odd functions as $g_1(\theta, m)$ and the group of even functions as $g_2(\theta, m)$. Once applying the standard discretization for the parameter $\theta$ in (20), $g_1(\theta, m)$ is a linear combination of odd functions and $g_2(\theta, m)$ is a linear combination of even functions, such that

$$\begin{cases} g_1(\theta_l, m) = -g_1(\theta_{2N-1-l}, m), \\ g_2(\theta_l, m) = g_2(\theta_{2N-1-l}, m), \end{cases} \tag{21}$$

according to the properties of the Gauss-Legendre quadrature nodes, when $l = 0, 1, \ldots, 2N-1$. Therefore, we only need to evaluate $g(\theta_l, m)$ by $g_1(\theta_l, m) + g_2(\theta_l, m)$ for $l = 0, 1, \ldots, N-1$, then $g(\theta_{2N-1-l}, m)$ can be obtained by

$$\begin{aligned} g(\theta_{2N-1-l}, m) &= g_1(\theta_{2N-1-l}, m) + g_2(\theta_{2N-1-l}, m) \\ &= -g_1(\theta_l, m) + g_2(\theta_l, m). \end{aligned} \tag{22}$$

After reducing the costs of both the forward and the inverse SHT by 50%, we are able to evaluate $g_1(\theta_l, m)$ and $g_2(\theta_l, m)$ by matvec for $l = 0, 1, \ldots, N-1$. For simplicity, we denote the matrices for these two matvecs as odd matrix and even matrix, and the size of them are $N \times \left(N - \lceil \frac{|m|}{2} \rceil\right)$ and $N \times \left(N - \lfloor \frac{|m|}{2} \rfloor\right)$, respectively. The symbol $\lceil x \rceil$ means the smallest integer greater than or equal to $x$, and the symbol $\lfloor x \rfloor$ means the largest integer less than or equal to $x$.

### 4.2 Partitioning matrices into blocks

Secondly, the odd and the even matrix can be separated into the oscillatory region and non-oscillatory region by a piecewise continuous curve.

In the article [6], a formula of the turning point of the normalized associated Legendre functions is introduced as follows,

$$t_{k,m}^{*} = \arcsin\left(\frac{\sqrt{m^2 - 1/4}}{k + 1/2}\right), \tag{23}$$

for $k = m, m+1, m+2, \ldots$. For a fixed degree $k \geq 0$ and a fixed order $m \leq k$, the turning point can separate the normalized associated Legendre functions $\overline{P}_k^{|m|}(\cos(\theta))$ into oscillatory region and non-oscillatory region. The interval of the oscillatory region for $\theta$ is $(t_{k,m}^{*}, \frac{\pi}{2})$, and the interval of the non-oscillatory region for $\theta$ is $(0, t_{k,m}^{*})$.

The turning point is an important tool for motivating us to propose a block partitioning algorithm, which is attributed to the curve consisting of the turning points. For example, Figure 2 (a) shows an example of an odd matrix with a piecewise continuous curve in the forward matvec of LT.

Next, we hierarchically partition the odd and the even matrix. Since the shape of the matrices are not square when $m \neq 0$, we carefully partition the row indices of the corresponding matrices into $b$ almost equal size parts, where

$$b = \begin{cases} \lfloor \frac{N}{N - \lceil \frac{|m|}{2} \rceil} \rceil & \text{for odd matrices,} \\ \lfloor \frac{N}{N - \lfloor \frac{|m|}{2} \rfloor} \rceil & \text{for even matrices,} \end{cases} \tag{24}$$

such that each partitioned block approximates to a set of square matrices. The symbol $\lfloor x \rceil$ means the nearest integer to $x$. For the partitioned blocks, we store them in $\mathcal{B}_s$, $s = 1, 2, \ldots, b$. However, because of the existence of the turning point, $\mathcal{B}_s$ is likely to be a high-rank matrix, and it is not easy to be constructed in the complementary low-rank structure. In the next step, we hierarchically partition the row and column indices of $\mathcal{B}_s$ into $2 \times 2$ parts if there are some turning points included in $\mathcal{B}_s$, and let the four submatrices be $\mathcal{B}_{s,t}$, $t = 1, 2, 3, 4$. This procedure can be repeated for all blocks with turning points until the length of the row or column of the submatrix is less than a parameter $n_0$, i.e., maximum partition level is $L = \log\left(\frac{N}{n_0}\right)$. For each partition level $l$ of $\mathcal{B}_s$, the curve of turning points goes through no more than $2^l - 1$ submatrices. Therefore, this procedure takes at most $\mathcal{O}(N)$ operations to partition the odd and the even matrices into submatrices, since

$$\mathcal{O}\left(\sum_{l=1}^{L}\left(2^l - 1\right)\right) \sim \mathcal{O}\left(\frac{2N}{n_0} - \log\left(\frac{N}{n_0}\right) - 2\right) \sim \mathcal{O}(N). \tag{25}$$

For example, Figure 2 (b) shows an example of an odd matrix which is partitioned into blocks by this procedure.

### 4.3 Factorization and application of the blocks

Finally, there are three kinds of partitioned blocks that have been generated: oscillatory blocks, non-oscillatory blocks, and the blocks including turning points. We deal with these different kinds of blocks by different methods:

1. For an oscillatory block $\mathcal{B}^o$, IDBF is able to construct a complementary low-rank structure as

$$\mathcal{B}^o \approx U^L U^{L-1} \cdots U^h S^h V^h \cdots V^{L-1} V^L, \tag{26}$$

where $L = \mathcal{O}(\log(N))$, $h = \frac{L}{2}$ and it takes only $\mathcal{O}(N \log(N))$ operations. After that, the factorization is well prepared for the application of the corresponding matvec with nearly linear complexity.

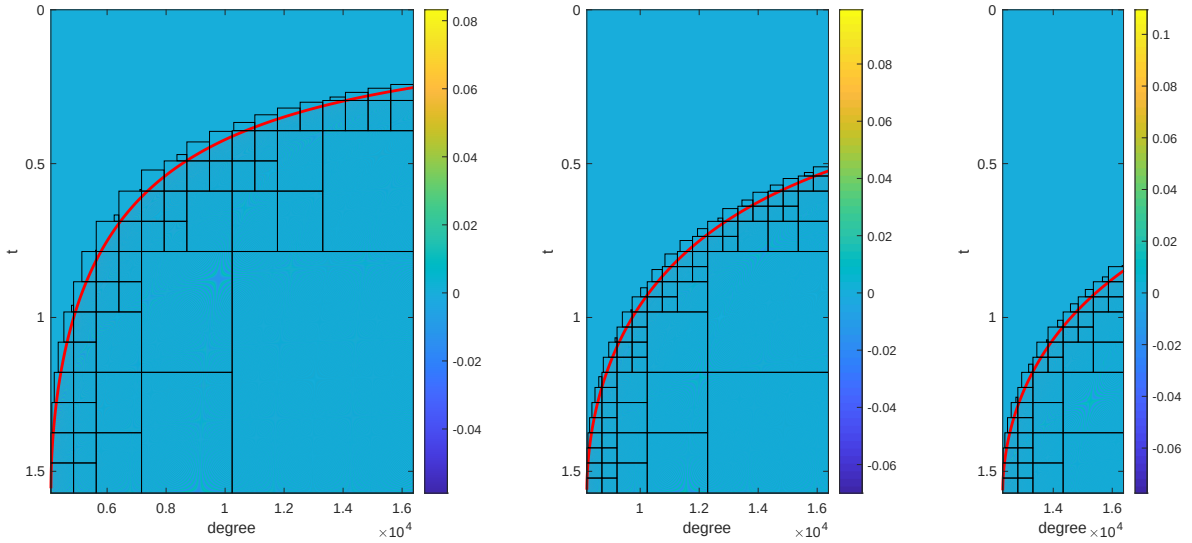
2. For a non-oscillatory block $\mathcal{B}^n$, it may contain some entries with computational absolute small values, which are useless for the matvec. Thus, we cut out the region of $\mathcal{B}^n$ with $\mathcal{O}(N)$ operations to obtain a smaller matrix $\mathcal{B}^{n'}$, whose absolute values of the boundary are not less than a tolerance, e.g. the floating-point relative accuracy (≈2.2204e-16). Next, low-rank approximation by randomized sampling is applied to construct a low-rank factorization as

$$\mathcal{B}^{n'} \approx U\Sigma V^T, \tag{27}$$

with $\mathcal{O}(N \log(N))$ operations. The corresponding matvec can be evaluated with nearly linear complexity.

3. For a block $\mathcal{B}^t$ including turning points, a similar method in the previous situation is used to cut out the region for omitting a proportion of almost zero parts and obtain a smaller matrix $\mathcal{B}^{t'}$. Then, the corresponding matvec is evaluated directly.

Figures 6 shows the procedure of shrinking the blocks to smaller matrices for different kinds of examples of odd matrices in the forward matvec of LT. Empty blocks with $0 \times 0$ size are omitted in the figure and will not be utilized in the application step.




(a) $m = 4096$ (b) $m = 8192$ (c) $m = 16384$

Figure 6: The visualization of the results of the whole partitioning procedure for different kinds of odd matrices in the forward matvec of Legendre transform. The size of the matrices is $N = 8192$. The orders of them are 4096, 8192 and 16384 from the left panel to the right panel.

The process of matvec of the even matrix is similar to that of the odd matrix. Eventually, LT in (13) can be computed by (20)-(22) according to the results of the matvecs.

Furthermore, the following lemma 4.1 in [28] states that the normalized associated Legendre functions of order $m$ are *orthonormal* on $(-1, 1)$ with respect to the standard inner product. Thus, the inverse matvec in LT can be simply evaluated by the transpose of the matrix of the forward matvec.

**Lemma 4.1.** *Suppose that $m$ is a non-negative integer. Then,*

$$\int_{-1}^{1} \overline{P}_k^{|m|}(x)\overline{P}_l^{|m|}(x)dx = \begin{cases} 1, & k = l, \\ 0, & k \neq l, \end{cases} \tag{28}$$

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

### 4.4 Computational complexity analysis

Let us analyze the complexity of the whole factorization process of a $N \times N$ square matrix $\mathcal{B}$, which is including turning points. Recall that the maximum partition level is $L = \log\left(\frac{N}{n_0}\right)$, and the curve of turning points goes through no more than $2^l - 1$ submatrices for each partition level $l$. Since the complexity of matvec generated by $\mathcal{B}^t$ is higher than that of $\mathcal{B}^o$ and $\mathcal{B}^n$, we assume that the curve goes through $2^l - 1$ submatrices in each level $l$. Then, the complexity of the whole process is bounded by follows

$$\begin{aligned}
&\mathcal{O}\left(n_0^2(2^L-1) + \sum_{l=1}^{L}(2^{l+1}-3)(2^{L-l}n_0)\log\left(2^{L-l}n_0\right)\right) \\
&\leq \mathcal{O}\left(n_0^2 \times 2^L + \sum_{l=1}^{L}(2^{L+1}n_0)\log\left(2^{L-l}n_0\right)\right) \\
&= \mathcal{O}\left(n_0 N\right) + \mathcal{O}\left(N\log\left(\frac{N}{n_0}\right)\left(\log\left(\frac{N}{n_0}\right)-1\right) + 2N\log\left(n_0\right)\log\left(\frac{N}{n_0}\right)\right) \\
&\sim \mathcal{O}\left(N\log^2\left(N\right)\right).
\end{aligned} \tag{29}$$

We can use the same method to prove that the theoretical complexity of the application step is also bounded by $\mathcal{O}\left(N\log^2\left(N\right)\right)$. Therefore, when $N$ increases and $b$ in (24) is fixed, i.e., $m \sim \mathcal{O}\left(N\right)$, the factorization and application of the proposed algorithm will be nearly linear scaling. Once the matvec of the odd matrix is computed with an $\mathcal{O}\left(N\log^2\left(N\right)\right)$ complexity, it is simple to conclude that the total performance of the matvec in LT is nearly linear scaling. Similarly, it is simple to prove that the complexity of the inverse matvec in LT is bounded by $\mathcal{O}\left(N\log^2\left(N\right)\right)$ as well. In the theoretical analysis, we assume that the numerical rank of all low-rank matrices is $\mathcal{O}\left(1\right)$. However, in the numerical results, we observe that the numerical rank grows like $\mathcal{O}\left(\log N\right)$ to maintain a fixed numerical approximation accuracy. Hence, the actual computational complexity of our method for an LT is $\mathcal{O}\left(N\log^3\left(N\right)\right)$.

## 5 Numerical results

This section presents several numerical examples to demonstrate the efficiency of the proposed framework. All implementations are in MATLAB® on a server computer with a single thread and 3.2 GHz CPU, and are available in the ButterflyLab (`https://github.com/ButterflyLab/ButterflyLab`).

For the forward matvec in LT, given an order $m$, we let $g^d(\theta)$ and $g^b(\theta)$ denote the results given by the direct matvec and the proposed algorithm, respectively. The accuracy of applying fast algorithms for the forward matvec is estimated by the relative error defined as follows:

$$\epsilon^{fwd} = \sqrt{\frac{\sum_{\theta\in S_1}|g^b(\theta)-g^d(\theta)|^2}{\sum_{\theta\in S_1}|g^d(\theta)|^2}}, \tag{30}$$

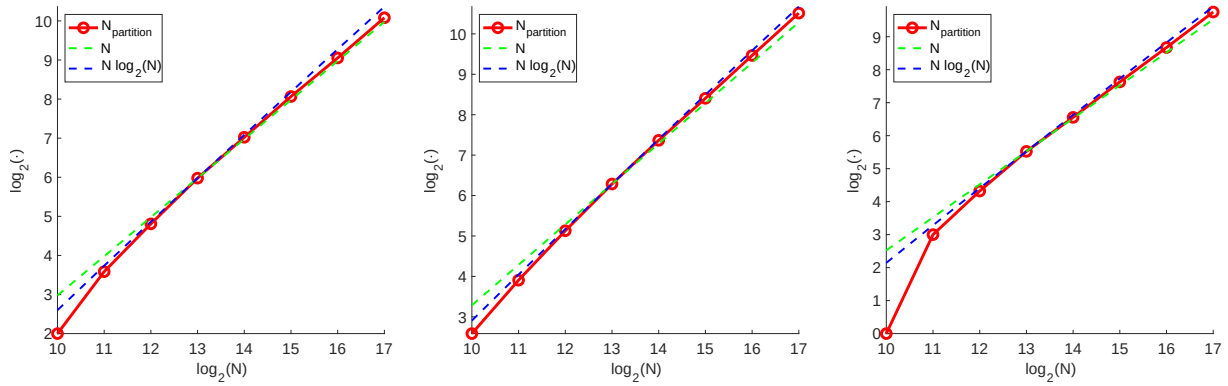
where $S_1$ is an index set containing 256 randomly sampled row indices of the non-zero part in the odd matrix or the even matrix. Then, let $a^d(k)$ and $a^b(k)$ denote the results given by the direct inverse matvec in LT and the proposed algorithm, respectively. The definition of the error $\epsilon^{inv}$ is similar to that of the forward matvec as follows:

$$\epsilon^{inv} = \sqrt{\frac{\sum_{k\in S_2}|a^b(k)-a^d(k)|^2}{\sum_{k\in S_2}|a^d(k)|^2}}, \tag{31}$$

where $S_2$ is an index set containing 256 randomly sampled row indices of the odd matrix or the even matrix.

In all of our examples, the tolerance parameter $\epsilon$ is set to $10^{-10}$, the minimum length $n_0$ for the partitioned block is set to 512, and the rank parameter $r$ for randomized SVD in low-rank phase matrix factorization is set to 30. The adaptive rank $r_k$ for IDBF will be set variously for different cases.

**Number of the blocks:** Our first experiment is to verify the number of the remaining blocks, which are obtained by Section 4.2 and cut out by Section 4.3. Figure 7 visualizes the results of this experiment for different grid sizes $N$ and order $m$ is set to be $0.5N$, $N$ and $1.5N$, respectively. It shows that the number of the remaining blocks scales nearly linearly when the problem size increases.




(a) $m = 0.5N$ (b) $m = N$ (c) $m = 1.5N$

Figure 7: The visualization of the number of the remaining blocks with different orders. $N$ is the size of the matrix, and order $m$ is set to be $0.5N$, $N$ and $1.5N$ from the left panel to the right panel.

**Selection of Mock-Chebyshev points or randomly selected points:** Next, we use Mock-Chebyshev points and randomly selected points for evaluating IDs in the IDBF process, respectively. The results of these two situations are compared in Figure 8 when the order parameter $m$ is set to be equal to $N$. The adaptive rank $r_k$ for IDBF is set to be 50, 100 and 150, respectively. It shows that the accuracy of results will increase as well when the rank parameter $r_k$ increases. It is obvious that the accuracy of IDs with Mock-Chebyshev points is higher and more stable than the accuracy of IDs with randomly selected points when the problem size increases. Note that $r_k = 150$ is good enough for IDs with Mock-Chebyshev points. Thus, for the rest of experiments, we will use Mock-Chebyshev points as grids to compute IDs in IDBF, and the adaptive rank $r_k$ for IDBF will be fixed at 150.

**Legendre transforms with different orders:** Finally, we apply the proposed algorithm to evaluate the matvec in LT for different orders. Then, the accuracy and efficiency will be verified through numerical results.

Figure 9 illustrates that the accuracy of the proposed algorithm for different orders stays almost of the same order, though the accuracy becomes slightly worse as the problem size increases. The

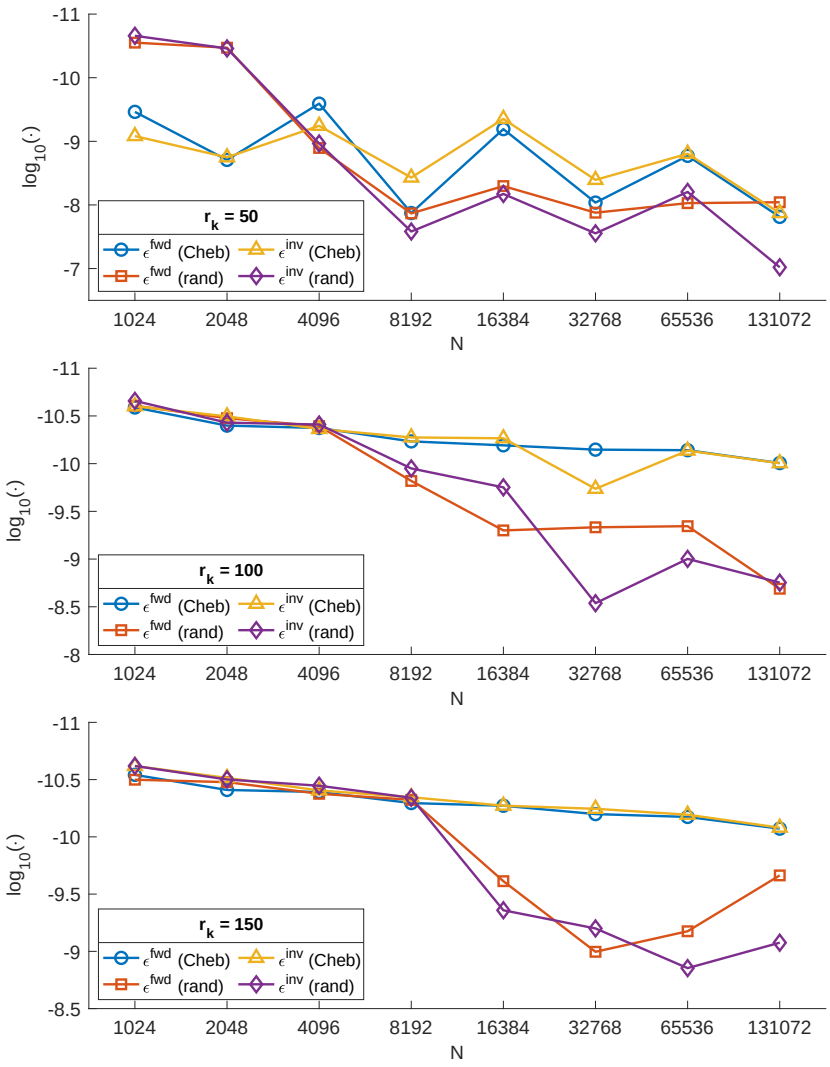



Figure 8: Numerical results for the error of Legendre transform with different kinds of grids in IDs in IDBF. $N$ is the size of the matrix, and order $m$ is set to be $N$ in each case. The adaptive rank $r_k$ for IDBF is set to be 50, 100 and 150 from the top panel to the bottom panel. "Cheb" and "rand" represent IDs with Mock-Chebyshev points and randomly selected points, respectively.

slightly increasing error is due to the randomness of the proposed algorithm in Subsection 3.3. As the problem size increases, the probability of capturing the low-rank matrix with a fixed rank parameter becomes smaller.

In Figure 10, the visualization of the computational complexity shows that the factorization time and the application time of the forward matvec and the inverse matvec by the proposed algorithm scale nearly linearly, when the problem size increases. $T_{fac}^{fwd}$ and $T_{app}^{fwd}$ are the factorization time and the application time of the proposed algorithm for the forward matvec in LT, respectively. $T_{mat}^{fwd}$ and $T_{dir}^{fwd}$ is the time for constructing the normalized associated Legendre matrix and computing the matvec directly in the forward matvec in LT, respectively. The definitions of $T_{fac}^{inv}$, $T_{app}^{inv}$, $T_{mat}^{inv}$, and $T_{dir}^{inv}$ for the inverse matvec in LT are similar to that of the forward matvec.

Figure 11 demonstrates that the factorization time and the application time of the proposed

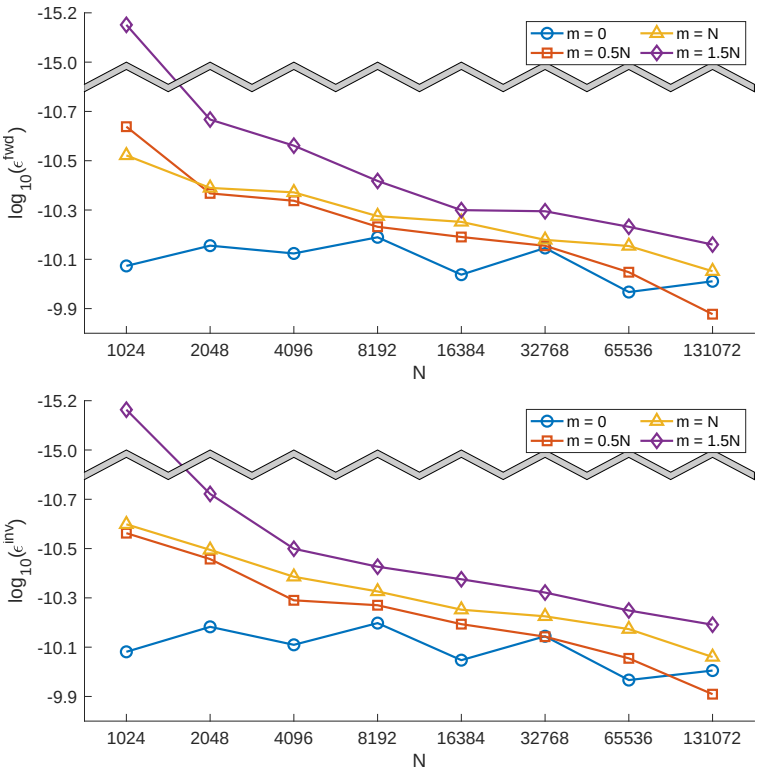



Figure 9: Numerical results for the error of Legendre transform with different orders. The top panel is the result of the error $\epsilon^{fwd}$ of the forward matvec in LT, and the bottom panel is the result of the error $\epsilon^{inv}$ of the inverse matvec. $N$ is the size of the matrix, and order $m$ is set to be 0, $0.5N$, $N$ and $1.5N$, respectively. The adaptive rank $r_k$ for IDBF is set to be 150.

algorithm compared with direct matvec. It shows that both precomputation time and matvec time are accelerated when the problem size increases. Especially, the speedup of the application time is more rapid.

## 6 Conclusion

This paper introduced a framework with theoretical complexity $\mathcal{O}\left(N\log^2(N)\right)$ for evaluating Legendre transform $g(\theta,m)=\sum_{k=|m|}^{2N-1}\beta_{k,m}\overline{P}_k^{|m|}(\cos(\theta))$, where $\overline{P}_k^{|m|}(x)$ is the normalized associated Legendre function. According to the properties of the normalized associated Legendre function, this paper proposed a block partitioning algorithm based on IDBF and low-rank approximation by randomized sampling for precomputation of the matvec, which can make good use of the non-zero numerical values. In practice, evaluating IDs with Mock-Chebyshev points in the IDBF process is more proper for the accuracy of the matvec. The numerical results show that the proposed algorithm can accelerate both factorization time and application time, and it requires only $\mathcal{O}\left(N\log^3(N)\right)$ operations to evaluate Legendre transform, in order to evaluate the spherical harmonic transform with a nearly quadratic scaling. Meanwhile, the proposed algorithm is able to remain highly accurate and stable.

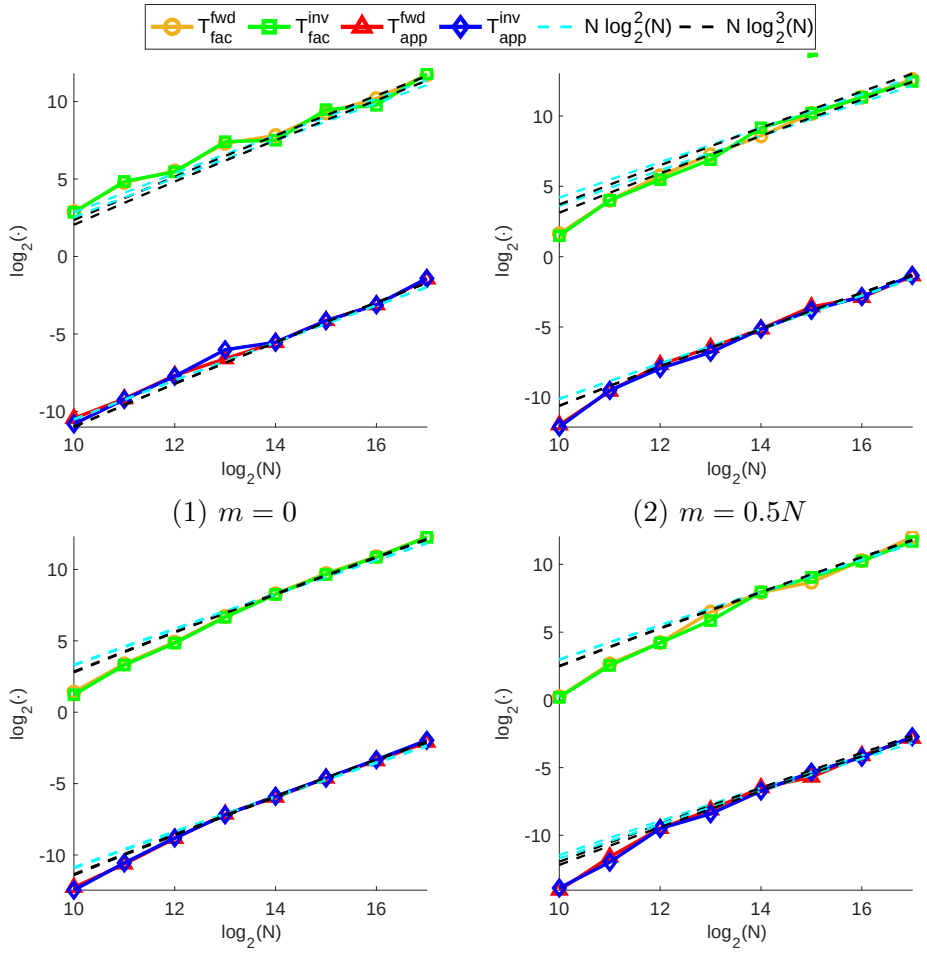



Figure 10: The visualization of the computational complexity with different orders. $N$ is the size of the matrix, and order $m$ is set to be 0, $0.5N$, $N$ and $1.5N$, respectively. 'Fac' and 'App' represent the factorization time and the application time, respectively. All times are in seconds.

**Acknowledgments.** The authors thank Yingzhou Li for his discussion on block partitioning the oscillatory region of Legendre transforms. J.B. was supported in part by NSF grant DMS-1418723. Z. C. was partially supported by the Ministry of Education in Singapore under the grant MOE2018-T2-2-147. H. Y. was partially supported by NSF under the grant award DMS-1945029.

## References

[1] M. Abramowitz and I. A. Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Dover Publications, New York, 9th dover printing with corrections edition, 1972.

[2] B. K. Alpert and V. Rokhlin. A Fast Algorithm for the Evaluation of Legendre Expansions. *Society for Industrial and Applied Mathematics. SIAM Journal on Scientific and Statistical Computing*, 12(1):158, 1991.

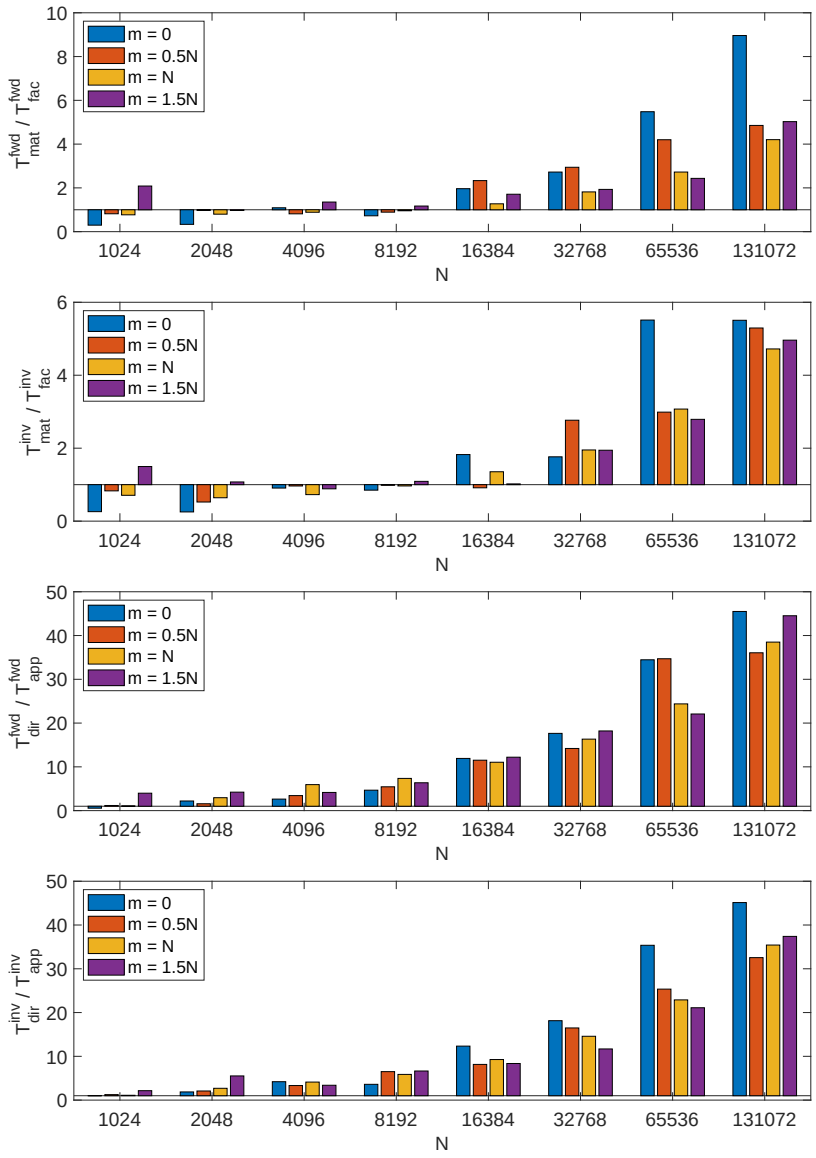



Figure 11: Numerical results for the speedup of Legendre transform with different orders. $N$ is the size of the matrix, and order $m$ is set to be 0, $0.5N$, $N$ and $1.5N$, respectively. The adaptive rank $r_k$ for IDBF is set to be 150. The results from the top panel to the bottom panel are $T_{mat}^{fwd}/T_{fac}^{fwd}$, $T_{mat}^{inv}/T_{fac}^{inv}$, $T_{dir}^{fwd}/T_{app}^{fwd}$ and $T_{dir}^{inv}/T_{app}^{inv}$.

[3] G. Andrews, R. Askey, and R. Roy. *Special Functions*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1999.

[4] R. Askey, S. for Industrial, A. Mathematics, and V. P. I. National Science Foundation. Regional Conference. 1974. *Orthogonal Polynomials and Special Functions*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1975.

[5] J. P. Boyd and F. Xu. Divergence (Runge Phenomenon) for least-squares polynomial approximation on an equispaced grid and Mock Chebyshev subset interpolation. *Applied Mathematics*

*and Computation*, 210(1):158 – 168, 2009.

[6] J. Bremer. An algorithm for the numerical evaluation of the associated Legendre functions that runs in time independent of degree and order. *Journal of Computational Physics*, 360:15 – 38, 2018.

[7] B. Engquist and L. Ying. A fast directional algorithm for high frequency acoustic scattering in two dimensions. *Communications in Mathematical Sciences*, 7(2):327–345, 06 2009.

[8] N. Hale and A. Townsend. A fast, simple, and stable Chebyshev–Legendre transform using an asymptotic formula. *SIAM Journal on Scientific Computing*, 36(1):A.148–A.167, 2014. Copyright - 2014, Society for Industrial and Applied Mathematics; Last updated - 2014-02-20.

[9] N. Hale and A. Townsend. A fast FFT-based discrete Legendre transform. *IMA Journal of Numerical Analysis*, 36(4):1670–1684, 11 2015.

[10] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[11] P. Hoffman and K. Reddy. Numerical Differentiation by High Order Interpolation. *SIAM Journal on Scientific and Statistical Computing*, 8(6):979–987, 1987.

[12] L.-k. Hua. *Harmonic analysis of functions of several complex variables in the classical domains*, volume v. 6.;v. 6;. American Mathematical Society, Providence, 1963.

[13] Y. Li and H. Yang. Interpolative butterfly factorization. *SIAM Journal on Scientific Computing*, 39(2):A503–A531, 2017.

[14] Y. Li, H. Yang, E. R. Martin, K. L. Ho, and L. Ying. Butterfly Factorization. *Multiscale Modeling & Simulation*, 13(2):714–732, 2015.

[15] Y. Liu, X. Xing, H. Guo, E. Michielssen, P. Ghysels, and X. S. Li. Butterfly factorization via randomized matrix-vector multiplications. *arXiv:2002.03400 [math.NA]*, 2020.

[16] P. Maroni and Z. da Rocha. Connection coefficients between orthogonal polynomials and the canonical sequence: an approach based on symbolic computation. *Numerical Algorithms*, 47(3):291–314, 2008.

[17] E. Michielssen and A. Boag. A multilevel matrix decomposition algorithm for analyzing scattering from large structures. *Antennas and Propagation, IEEE Transactions on*, 44(8):1086–1093, Aug 1996.

[18] M. J. Mohlenkamp. A fast transform for spherical harmonics. *The Journal of Fourier Analysis and Applications*, 5(2):159–184, 1999.

[19] M. O'Neil, F. Woolfe, and V. Rokhlin. An algorithm for the rapid evaluation of special function transforms. *Appl. Comput. Harmon. Anal.*, 28(2):203–226, 2010.

[20] Q. Pang, K. L. Ho, and H. Yang. Interpolative Decomposition Butterfly Factorization. *SIAM Journal of Scientific Computing*, To appear.

[21] D. Potts, G. Steidl, and M. Tasche. Fast algorithms for discrete polynomial transforms. *Mathematics of Computation of the American Mathematical Society*, 67(224):1577–1590, 1998.

[22] V. Rokhlin and M. Tygert. Fast Algorithms for Spherical Harmonic Expansions. *SIAM J. Sci. Comput.*, 27(6):1903–1928, Dec. 2005.

[23] J. Shen, Y. Wang, and J. Xia. Fast structured Jacobi-Jacobi transforms. *Mathematics of Computation of the American Mathematical Society*, 88(318):1743–1772, 2019;2018;.

[24] R. M. Slevinsky. On the use of Hahns asymptotic formula and stabilized recurrence for a fast, simple and stable Chebyshev-Jacobi transform. *IMA Journal of Numerical Analysis*, 38(1):102–124, 02 2017.

[25] R. M. Slevinsky. Fast and backward stable transforms between spherical harmonic expansions and bivariate Fourier series. *Applied and Computational Harmonic Analysis*, 47(3):585–606, 2019.

[26] R. Suda. Fast Spherical Harmonic Transform Routine FLTSS Applied to the Shallow Water Test Set. *Monthly Weather Review*, 133(3):634–648, 2005.

[27] R. Suda and M. Takami. A fast spherical harmonics transform algorithm. *Mathematics of Computation of the American Mathematical Society*, 71(238):703–715, 2002;2001;.

[28] G. Szegö. *Orthogonal Polynomials*. Number v. 23 in American Mathematical Society colloquium publications. American Mathematical Society, 1959.

[29] A. Townsend, M. Webb, and S. Olver. Fast polynomial transforms based on Toeplitz and Hankel matrices. *Mathematics of Computation of the American Mathematical Society*, 87(312):1913–1934, 2018;2017;.

[30] M. Tygert. Fast algorithms for spherical harmonic expansions, II. *Journal of Computational Physics*, 227(8):4260–4279, 2008.

[31] M. Tygert. Fast algorithms for spherical harmonic expansions, III. *Journal of Computational Physics*, 229(18):6181 – 6192, 2010.

[32] C. Van Loan. *Computational Frameworks for the Fast Fourier Transform*. Society for Industrial and Applied Mathematics, 1992.

[33] N. P. Wedi, M. Hamrud, and G. Mozdzynski. A Fast Spherical Harmonics Transform for Global NWP and Climate Models. *Monthly Weather Review*, 141(10):3450–3461, 2013.

[34] H. Yang. A unified framework for oscillatory integral transforms: When to use NUFFT or butterfly factorization? *Journal of Computational Physics*, 388:103–122, Jul 2019.

[35] F. Yin, G. Wu, J. Wu, J. Zhao, and J. Song. Performance Evaluation of the Fast Spherical Harmonic Transform Algorithm in the YinHe Global Spectral Model. *Monthly Weather Review*, 146(10):3163–3182, 2018.

[36] F. Yin, J. Wu, J. Song, and J. Yang. A High Accurate and Stable Legendre Transform Based on Block Partitioning and Butterfly Algorithm for NWP. *Mathematics*, 7, 10 2019.