Performance Evaluation of the Fast Spherical Harmonic Transform Algorithm in the Yin–He Global Spectral Model

FUKANG YIN, GUOLI WU, JIANPING WU, JUN ZHAO, AND JUNQIANG SONG

College of Meteorology and Oceanography, National University of Defense Technology, Changsha, China

(Manuscript received 2 May 2018, in final form 17 July 2018)

ABSTRACT

In this paper, we describe an implementation of the fast spherical harmonic transform (SHT) algorithm in the Yin–He global spectral model (YHGSM). A new analysis method is proposed to study the potential instability of interpolative decomposition and to evaluate the performance of fast SHT on the MilkyWay-2 supercomputer. The novel aspect of the proposed method is the use of the computational complexity analysis method by studying the properties of interpolative decompositions. Furthermore, three test cases are conducted to verify fast SHT in YHGSM. The results demonstrate that fast SHT is feasible and efficient for reducing the computational complexity and memory cost of SHT while still keeping enough accuracy.

1. Introduction

Spherical harmonic transform (SHT) is one of the most important dynamics algorithms in the fields of numerical weather prediction and climate modeling. The disproportionately growing computational cost of the Legendre transform (LT) with the increasing of resolution poses a formidable challenge to the high-resolution numerical weather prediction (NWP) spectral model. Therefore, several fast algorithms have been presented and investigated (Suda and Takami 2002; Kunis and Potts 2003; Suda 2004, 2005; Rokhlin and Tygert 2006; Tygert 2010a). However, it is difficult for these algorithms to remain both fast and stable until the butterfly algorithm presented by Tygert (2010b). Tygert’s algorithm has three advantages: 1) it is numerically stable, 2) the computational complexity is $O(N^2 \log N)$ in theory, and 3) the computational cost can be reduced by choosing lower desired accuracy. Thereafter, the fast SHT algorithm based on butterfly matrix compression has been successfully used for astrophysical applications (Seljebotn 2012), NWP, and climate models (Wedi et al. 2013).

Legendre transform can be expressed as a matrix–vector product $u = Kg$, where matrix $K$ is the representation of the discrete Legendre transform and vector $g$ is the function to be transformed. The main idea of the butterfly algorithm initially proposed by Michielssen and Boag (1996) and later extended by O’Neil et al. (2010) is that a dense matrix–vector multiplication is transformed into a set of sparse matrix–vector multiplication by using butterfly factorization and butterfly multiply algorithm. The butterfly algorithm factorizes a complementary low-rank matrix $K$ of size $N \times N$ into the product of $O(\log N)$ sparse matrices, each with $O(N)$ nonzero entries, and then the matrix–vector multiplication of $Kg$ has nearly optimal operation and the memory complexity of $O(N \log N)$ (Li and Yang 2017).

From above, it can be found that computational complexity and memory complexity of the butterfly algorithm depend on the number of nonzero entries generate in the factorization. An asymptotic scaling of $O(N^2 \log^2 N)$ and $O(N^2 \log^3 N)$ is obtained in Wavemoth (Seljebotn 2012) for astrophysical applications and in high-resolution spectral models for NWP (Wedi et al. 2013), respectively. Although the butterfly algorithm has had success in the above two applications, it still lacks detailed analysis about computational complexity; for example, why $O(N^2 \log^2 N)$ is obtained by Seljebotn (2012), but $O(N^2 \log^3 N)$ by Wedi et al. (2013).

Motivated by the above research, we focus our work solely on providing an optimized implementation and
analysis. The novel aspect is the performance evaluation of SHT in the perspective of the properties of interpolation decompositions such as the rank, maximum absolute value, and number of nonzero entries of projection or interpolation matrix. The remainder of the paper is organized as follows. In section 2, we introduce interpolative decomposition and describe a sparse matrix storage format and corresponding matrix multiplication algorithm for spherical harmonic transform. In section 3, we present a detailed discussion of the developed algorithm and perform comprehensive sets of tests comparing them against an original implementation of the transforms in the Yin–He global spectral model (YHGSM). Finally the conclusions are given in section 4.

2. Fast spherical harmonic transform algorithms based on butterfly matrix compression

Tygert (2010b) first described the fast SHT algorithm based on butterfly matrix compression. A further illustration was presented by Seljebotn (2012). Key ideas are the compression of the low rank matrix (Cheng et al. 2005) and the butterfly algorithm (O’Neil et al. 2010). Martinsson et al. (2008) developed a software package for low-rank approximation of matrices via interpolative decompositions. Although Seljebotn (2012) and Wedi et al. (2013) have successfully implemented the algorithm of Tygert (2010b) in astrophysics and NWP, respectively, some unsolved issues still remain. These issues include the choice of parameters such as Cmax and desired accuracy, the stability of interpolative decomposition (ID), and the estimation of the computational cost in practical application for fast SHT.

In this section, we will describe how to implement the sparse representation used in Wavemoth (Seljebotn 2012) for fast SHT on an NWP model YHGSM. A detailed description of the algorithm can be seen in O’Neil et al. (2010), Tygert (2010b), Seljebotn (2012), and Wedi et al. (2013). Here we briefly outline the most important features.

a. Interpolative decomposition

This section reviews the basic theorem for interpolative decomposition presented by Martinsson et al. (2008) and Martinsson and Tygert (2013), which will be used in subsequent sections.

Theorem 1. For positive integers $k$, $m$, and $n$ with $k \leq m$ and $k \leq n$, and any $m \times n$ matrix $A$, there exist a $k \times n$ matrix $P$, and an $m \times k$ matrix $B$ whose columns constitute a subset of the columns of $A$, such that

1) The rearrangement of some columns in matrix $P$ makes up a $k \times k$ identity matrix.
2) Each element of matrix $P$ has an absolute value less than or equal to a reasonably small positive real number $\beta$.
3) $\|P\| \leq \sqrt{\beta^2k(n-k)} + 1$ (where $\|P\|$ is the spectral norm of $P$).
4) The least (i.e., the $k$th greatest) singular value of $P$ is at least 1.
5) $BP = A$ when $k = m$ or $k = n$.
6) $\|BP - A\| \leq \sqrt{\beta^2k(n-k)} + 1\sigma_{k+1}$ when $k < m$ and $k < n$, where $\sigma_{k+1}$ is the $(k+1)$th greatest singular value of $A$, and $\|BP - A\|$ is the spectral norm of $BP - A$.

Matrices $P$ and $B$ are named as the projection or interpolation matrix and skeleton matrix of the interpolative decomposition, respectively. The interpolative decomposition is “interpolative” and numerically stable because of property 2 and property 4 above, respectively. The Gu–Eisenstat algorithm (Gu and Eisenstat 1996) can be employed in interpolative decomposition to obtain matrices $B$ and $P$ satisfying properties 1–6 above. The interpolative decomposition constructed via the Gu–Eisenstat algorithm is provably numerically stable, but the Gu–Eisenstat algorithm requires about $\log_2(n)$ times more flops than the classical pivoted “QR” decomposition algorithms in the worst

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLTFLAG</td>
<td>LOGICAL</td>
<td>‘.TRUE.’: use FLT; ‘.FALSE.’: use DGEMM</td>
</tr>
<tr>
<td>PRECFLAG</td>
<td>LOGICAL</td>
<td>‘.TRUE.’: compute the butterfly compression representation of FLT; ‘.FALSE.’: read compression representation of FLT from specified files</td>
</tr>
<tr>
<td>PRE_SAVE</td>
<td>LOGICAL</td>
<td>‘.TRUE.’: save the butterfly compression representation of FLT to specified file</td>
</tr>
<tr>
<td>DIMTHRESH</td>
<td>INTERGER</td>
<td>Threshold value for using FLT</td>
</tr>
</tbody>
</table>
Therefore, Martinsson et al. (2008) used the classical pivoted QR decomposition algorithms instead of the Gu–Eisenstat algorithm in the ID software package. In interpolative decomposition, a desired or specified precision \( \varepsilon \) is used as the spectral-norm accuracy of the pivoted QR decomposition (i.e., \( \| A_{m \times n} - B_{m \times k} P_{k \times n} \| \leq \varepsilon \)). But round-off errors from specified precision of the pivoted QR decompositions may affect the numerical stability of the interpolative decomposition. The key idea of Martinsson et al. is to sacrifice the accuracy of interpolative decomposition for the ease of implementation and optimization of performance. This is a trade-off between numerical stability and computational cost. It should be noted that any problem with numerical stability could be fixed completely by replacing the ID package of Martinsson et al. with the Gu–Eisenstat algorithm (albeit at significantly increased cost for precomputations and for the complexity of implementation). To maximize performance improvement, we only focus on the Legendre transform algorithm using the ID package of Martinsson et al. in this paper.

According to properties 2 and 6 in Theorem 1, it can be found that the positive real number \( \beta \) appears in the upper bounds of spectral norm of \( P \) and \( BP - A \). It means that the accuracy of ID depends on parameters \( \beta \), \( n \), \( k \), and \( \sigma_{k+1} \). Moreover, the desired precision \( \varepsilon \) impacts on rank \( k \). It is obvious that rank \( k \) is larger when \( \varepsilon \) is smaller (the desired precision is improved). The values of \( \sigma_{k+1} \) and \( n(n - k) \) decrease with the increasing of rank \( k \). The columns of matrix \( B \) constitute a subset of the columns of \( A \). Let matrix \( \tilde{P} \) be the \( n(n - k) \) matrix that removes \( k \times k \) identity matrix from matrix \( P \). The maximum absolute value of \( \tilde{P} \) reflects the importance of the remainder columns to matrix \( A \). The smaller the maximum absolute value of \( \tilde{P} \) (i.e., maximum absolute value of \( P \)), the higher the accuracy of ID. Finally, the accuracy of ID depends on the desired precision \( \varepsilon \), the properties of matrix \( A \), and the dimension \( n \). A larger \( \beta \) will cause loss of accuracy. The “instability” may occur in the LT with low desired precision for high-resolution cases. A simple relationship is shown in the following equation:

\[
\| A_{m \times n} - B_{m \times k} P_{k \times n} \| \leq \varepsilon
\]

TABLE 2. List of the routines and their descriptions.

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flt_dir_precomputation</td>
<td>Completes the precomputation for the forward spherical harmonic transform</td>
</tr>
<tr>
<td>flt_inv_precomputation</td>
<td>Completes the precomputation for the backward spherical harmonic transform</td>
</tr>
<tr>
<td>flt_dir_idallocation</td>
<td>Allocates the space for the forward SHT</td>
</tr>
<tr>
<td>flt_inv_idallocation</td>
<td>Allocates the space for the backward SHT</td>
</tr>
<tr>
<td>flt_dir_idsaveledir</td>
<td>Saves matrices generated in ID and some auxiliary information used in the butterfly multiply algorithm to file for the forward SHT</td>
</tr>
<tr>
<td>flt_inv_idsaveleinv</td>
<td>Saves matrices generated in ID and some auxiliary information used in the butterfly multiply algorithm to file for the backward SHT</td>
</tr>
<tr>
<td>flt_dir_idreadleinv</td>
<td>Reads matrices generated in ID and some auxiliary information used in the butterfly multiply algorithm from the file for the forward SHT</td>
</tr>
<tr>
<td>flt_inv_idreadleinv</td>
<td>Reads matrices generated in ID and some auxiliary information used in the butterfly multiply algorithm from the file for the backward SHT</td>
</tr>
<tr>
<td>butterfly_extract_smatrix</td>
<td>Extracts matrices to be decomposed</td>
</tr>
<tr>
<td>butterfly_compression</td>
<td>Performs the butterfly matrix compression</td>
</tr>
<tr>
<td>butterfly_extract_vmatrix</td>
<td>Extracts the vector used in the butterfly multiply</td>
</tr>
<tr>
<td>butterfly_memory_matrix</td>
<td>Saves the results of the matrix–vector multiply</td>
</tr>
<tr>
<td>flt_dir_mxm</td>
<td>Performs the matrix–matrix multiply in the forward SHT</td>
</tr>
<tr>
<td>flt_inv_mxm</td>
<td>Performs the matrix–matrix multiply in the backward SHT</td>
</tr>
<tr>
<td>savecol</td>
<td>Saves matrices generated in ID and some auxiliary information used in the butterfly multiply algorithm</td>
</tr>
<tr>
<td>exact_sub_matrix</td>
<td>Extracts submatrices to be compressed on level 0</td>
</tr>
<tr>
<td>comb_l_and_r_neighbor</td>
<td>Extracts submatrices to be compressed for level ( &gt; 0 )</td>
</tr>
</tbody>
</table>

FIG. 2. Call relationships of butterfly_compression, flt_dir_precomputation, and flt_dir_mxm.
To facilitate reading, the same symbols, such as Cmax, L, and ε, defined in the algorithm described in the literature (Tygert 2010b; Cheng et al. 2005; Seljebotn 2012; Wedi et al. 2013) are adopted in the paper. The parameter Cmax is the number of columns in each submatrix on level 0. The number of level L is determined by the formula $L = \log_2\left(\frac{N}{C_{\text{max}}}\right)$ ($N$ is the dimension of matrix to be compressed). Therefore, the factor in the asymptotic costs of LT is $\log_2\left(\frac{N}{C_{\text{max}}}\right)$. Moreover, the asymptotic cost of LT also depends on the number of nonzero entries in interpolation matrices, which is affected with the performance of ID. In addition, ID with high desired precision ε is more time consuming than that with low desired precision ε. However, there is still a lack of research on the relationship between the max absolute value of projection matrix and the desired precision ε nor the relationship between the number of nonzero entries in ID and the desired precision ε.

**FIG. 3.** Ranks of projection matrices with INVEPS = DIREPS = $1.0 \times 10^{-7}$ for resolution T7999.

**FIG. 4.** Ranks of projection matrices with INVEPS = DIREPS = $1.0 \times 10^{-10}$ for resolution T7999.
Given an $N \times N$ matrix $A$, and the number of level $L = \lfloor \log_2(N/C_{\text{max}}) \rfloor$ (where $\lfloor \cdot \rfloor$ is integral function), each level needs $\lfloor N/C_{\text{max}} \rfloor$ IDs. The total number of IDs is $\lfloor N/C_{\text{max}} \rfloor \times \lfloor \log_2(N/C_{\text{max}}) \rfloor$ for butterfly matrix compression. The columns of matrix $B$ constitute a subset of the columns of $A$, it can be assumed that the skeleton matrices $B$ generated in butterfly matrix compression are accuracy, the error brings up with the projection matrices $P$. So the accuracy of the ID depends on the maximal value of $b$ for all projection matrices $P$.

It needs all projection matrices in different levels (0, 1, ..., $L$) to perform the LT by using the algorithm of butterfly multiplication. Thus, it is difficult to determine the function of the error in butterfly compression and those in IDs. The maximum error of one level depends on the maximum error of IDs for all blocks. So it is may be reasonable to use the maximum absolute values of projection matrices to study the accuracy of LT. In summary, performance evaluation of SHT in the perspective of properties of interpolation decompositions such as the rank, max absolute value, and number of nonzero entries of projection or interpolation matrix is valuable and feasible.

### b. A sparse matrix storage format and multiplication algorithm for the projection matrix $P$

For the fast Legendre transform algorithm, it needs to compute and store the interpolation matrix $P$ of the columns of submatrix $S$ in the precomputing step. Moreover, the matrix premultiplying operation of $P$ is needed in the butterfly multiply step. The matrix $P$ may lead to greater computational complexity and larger storage requirements. However, the matrix $P$ is sparse, and most importantly, contains an identity matrix. These features can be used to decrease the storage and computational complexity (Seljebotn 2012; Martinsson and Tygert 2013). Seljebotn (2012) points out that it can roughly halve the storage size and FLOP count. Although a sparse matrix

---

**Fig. 5.** Maximum absolute values of projection matrices for Legendre–Vandermonde matrix with $\epsilon = 10^{-7}$. 
Fig. 6. Maximum absolute values of projection matrices for resolution T7999: (a) reduced Gaussian grid and (b) full Gaussian grid. In (a) and (b), from top to bottom are the results with $C_{\text{max}} = 64$ and $\text{INVEPS} = \text{DIREPS} = 1.0 \times 10^{-7}$, $C_{\text{max}} = 64$ and $\text{INVEPS} = \text{DIREPS} = 1.0 \times 10^{-10}$, $C_{\text{max}} = 128$ and $\text{INVEPS} = \text{DIREPS} = 1.0 \times 10^{-7}$, and $C_{\text{max}} = 128$ and $\text{INVEPS} = \text{DIREPS} = 1.0 \times 10^{-10}$, respectively.
storage format has already been implemented to perform SHTs in Wavemoth (Seljebotn 2012), more detailed description and analysis is still needed.

In the ID package of Martinsson et al. (2008), the ID of an arbitrary (generally dense) matrix to a specified precision is computed by routine iddp_id. In the routine iddp_id, the output parameter list reserves the permutation information of projection or interpolation matrix P. It can get a permutation matrix of P according to the output parameter list, that is, P = [P_{11}, P_{12}] in which P_{11} = [P(:, list(1)), ... , P(:, list(krank))] is a krank \times krank identity matrix and P_{12} is a krank \times (n - krank) matrix.

Let P and \beta be a krank \times n and n \times nflev matrices, respectively. Then the product P_{krank} \times \beta_{n \times nflev} can be rewritten as

\[
P_{krank \times n} \beta_{n \times nflev} = \begin{bmatrix} I_{krank \times krank} & R_{krank \times (n - krank)} \end{bmatrix} \beta
\]

\[
= \begin{bmatrix} \beta_{list(1)} \\ \beta_{list(2)} \\ \vdots \\ \beta_{list(krank)} \end{bmatrix} + R_{krank \times (n - krank)} \begin{bmatrix} \beta_{list(krank + 1)} \\ \beta_{list(krank + 2)} \\ \vdots \\ \beta_{list(n)} \end{bmatrix}
\] (1)

Fig. 7. Compression ratio of projection matrices. The compression ratio is the data size that becomes X\% smaller size than original size.
It covers a krank $\times n$ matrix multiplying a $n \times \text{nflev}$ matrix to a krank $\times n$ matrix multiplying a $n \times \text{nflev}$ matrix, and reduces the number of float operations from $O(\text{krank} \times n \times \text{nflev})$ to $O(\text{krank} \times (n - \text{krank}) \times \text{nflev})$. The matrix product of $P$ and $\mathbf{b}$ can be seen as a sum of two parts, where the first part consists of krank rows of $\mathbf{b}$ indicated by list$(1: \text{krank})$, while the second part can be obtained by the remainder $n$-krank columns of $P$ multiplied by the remainder rows of $\mathbf{b}$ indicated by list$(\text{krank} + 1:n)$. To help make things clearer to the reader, we give an example illustrating the algorithm.

Example 1: If the projection matrix
\[ P = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0.5 & 0 & 0 \\ 0 & 0.5 & 1 & 0 \end{bmatrix} \]

with list$ = [4 \ 1 \ 3 \ 2]$, then the permutation matrix is
\[ P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.5 \\ 0 & 0 & 1 & 0.5 \end{bmatrix}. \tag{2} \]

To save memory, the last column $[0, 0.5, 0.5]^T$ needs only to be stored and parameter list is used to reconstruct $P$. The $i$th ($i \leq \text{krank}$) elements in parameter list means that the $(i, \text{list}(i))$th element is 1 (see Fig. 1).

<table>
<thead>
<tr>
<th>Resolution</th>
<th>T255</th>
<th>T511</th>
<th>T799</th>
<th>T1023</th>
<th>T1279</th>
<th>T2047</th>
<th>T3999</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVEPS $=$ DIREPS $= \times 10^{-7}$</td>
<td>157</td>
<td>1195</td>
<td>3775</td>
<td>7545</td>
<td>12436</td>
<td>43038</td>
<td>218117</td>
</tr>
<tr>
<td>INVEPS $=$ DIREPS $= \times 10^{-10}$</td>
<td>157</td>
<td>1201</td>
<td>3859</td>
<td>7728</td>
<td>12951</td>
<td>45047</td>
<td>233231</td>
</tr>
</tbody>
</table>
Multiplying a matrix $\mathbf{b}$ by $\mathbf{P}$, one has

$$\mathbf{B} = \mathbf{Pb}.$$

The matrix product $\mathbf{Pb}$ can be rewritten as

$$\mathbf{B} = \mathbf{Pb} = \mathbf{\hat{P}}\mathbf{\hat{b}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.5 \\ 0 & 0 & 1 & 0.5 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 4 & 8 \\ 1 & 5 \\ 3 & 7 \end{bmatrix} = \begin{bmatrix} 4 & 8 \\ 1 & 5 \\ 3 & 7 \end{bmatrix} \begin{bmatrix} 2 & 6 \\ 4 & 10 \end{bmatrix}.$$

where $\mathbf{\hat{b}}$ is the row permutation matrix of $\mathbf{b}$. Figure 1 depicts the compressed storage format for projection matrix and the multiplication algorithm for projection matrix $\mathbf{P}$ considering compressed storage format is shown in algorithm 1. In Fig. 1, green blocks denote the elements of projection matrix $\mathbf{P}$ with value 1, red blocks denote the columns that need to be stored, and orange blocks represent vector list.

c. Key subroutines of fast spherical harmonic transform algorithm

In this section, we will describe how to implement the fast Legendre transform algorithm in YHGSM (Wu et al.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>T255</th>
<th>T511</th>
<th>T799</th>
<th>T1023</th>
<th>T1279</th>
<th>T2047</th>
<th>T3999</th>
<th>T7999</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIREPS</td>
<td>INVEPS</td>
<td>$=10^{-7}$</td>
<td>$=10^{-10}$</td>
<td>$=10^{-10}$</td>
<td>$=10^{-10}$</td>
<td>$=10^{-10}$</td>
<td>$=10^{-10}$</td>
<td>$=10^{-10}$</td>
</tr>
<tr>
<td>DIMT</td>
<td>0</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>DIREPS</td>
<td>INVEPS</td>
<td>$=10^{-7}$</td>
<td>$=10^{-10}$</td>
<td>$=10^{-10}$</td>
<td>$=10^{-10}$</td>
<td>$=10^{-10}$</td>
<td>$=10^{-10}$</td>
<td>$=10^{-10}$</td>
</tr>
<tr>
<td>Cmax</td>
<td>064</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>Cmax</td>
<td>064</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>DGEMM</td>
<td>228.76</td>
<td>1387.26</td>
<td>5263.56</td>
<td>10887.40</td>
<td>21013.16</td>
<td>84619.96</td>
<td>854367.46</td>
<td>9393950.70</td>
</tr>
<tr>
<td>Proc Num</td>
<td>64 x 1</td>
<td>64 x 1</td>
<td>64 x 1</td>
<td>64 x 1</td>
<td>64 x 1</td>
<td>64 x 1</td>
<td>64 x 1</td>
<td>64 x 1</td>
</tr>
<tr>
<td>DIREPS</td>
<td>INVEPS</td>
<td>$=10^{-7}$</td>
<td>$=10^{-10}$</td>
<td>$=10^{-10}$</td>
<td>$=10^{-10}$</td>
<td>$=10^{-10}$</td>
<td>$=10^{-10}$</td>
<td>$=10^{-10}$</td>
</tr>
<tr>
<td>Cmax</td>
<td>064</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>Cmax</td>
<td>064</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>DGEMM</td>
<td>228.76</td>
<td>1387.26</td>
<td>5263.56</td>
<td>10887.40</td>
<td>21013.16</td>
<td>84619.96</td>
<td>854367.46</td>
<td>9393950.70</td>
</tr>
<tr>
<td>Proc Num</td>
<td>4 x 16</td>
<td>4 x 16</td>
<td>32 x 16</td>
<td>32 x 16</td>
<td>32 x 16</td>
<td>32 x 16</td>
<td>32 x 16</td>
<td>32 x 16</td>
</tr>
</tbody>
</table>

FIG. 9. Memory cost of FLT scaled by $N^2 \log^4 N$ with Cmax = 128.

FIG. 10. Computational time for FLT for difference zonal wavenumber $m$. The unit is in the number of clock cycles.
YHGSM is a global spectral NWP model developed by the college of meteorology and oceanography in National University of Defense Technology (NUDT). The dynamical core of YHGSM is hydrostatic in a reduced Gaussian grid (Hortal and Simmons 1991; Courtier and Naughton 1994). A two-time-level, semi-implicit, semi-Lagrangian integration scheme (Bénard 2003, 2004) is used for the time stepping, and spherical harmonic functions expansion is adopted in the horizontal direction and a finite-element discretization is employed in the vertical direction. YHGSM possesses two levels of parallelism: one is intranode parallelism using MPI, and the other is inner-node parallelism using OpenMP. The parallelization of single Legendre transform and FFT is not considered in YHGSM. Some standard tests for YHGSM have been performed by Yang et al. (2015, 2017).

In YHGSM, the original Legendre transform is performed by Basic Linear Algebra Subprograms (BLAS) matrix–matrix multiplication function DGEMM. (BLAS was downloaded from the website http://www.netlib.org/blas/ and compiled for YHGSM.) Hereafter, fast Legendre transform (FLT) is referred to when using the algorithm of Tygert (2010b). A switch is added in actual direct and inverse Legendre transform calls to activate FLT. Variables used to control FLT are shown in Table 1. We need to add an initialization subroutine to complete the initialization of FLT as listing 1 and modify subroutine lt_dir_trans/lt_inv_trans (lt_dir_trans/lt_inv_trans is the subroutine of forward/backward Legendre transform) as in listing 2 and 3 before using FLT. Table 2 gives the list of the routines and descriptions of their functionality. Figure 2 shows call relationships of key subroutines involved in FLT.

3. Numerical results

In this section, we first confirm that the algorithm of Tygert (2010b) satisfies the properties of interpolative factorization by studying the rank, max absolute value, and number of nonzero entries of projection matrices, and second, we make a detailed analysis to verify the performance of fast SHT on YHGSM. All numerical experiments were performed on the MilkyWay-2 supercomputer [see Liao et al. (2014) for more details] installed in NUDT. Each compute node possesses 64 GB of memory. The CPU model name is Intel(R) Xeon(R) CPU E5-2692 V2 @2.2 GHz. A private 32 KB L1 instruction cache, a 32 KB L1 data cache, a 256 KB L2 cache, and a 30720 KB L3 cache are used.

The unit of time for all tests is in the number of clock cycles obtained by function system_clock( ).

Fig. 11. Speedup for LT part of SHT compare to DGEMM. Results of (top) Table 4 (top) and (bottom) Table 4 (bottom).

Fig. 12. Computational time for LT part of SHT in Table 4 (top) scaled by (top) \( N^2 \log^2 N \) and (bottom) \( N^2 \log^3 N \).
Cmax is the number of columns in each submatrix on level 0 (O’Neil et al. 2010) and EPS is the desired/preaccuracy precision that is specified in interpolative decompositions (i.e., the spectral-norm accuracy of the pivoted QR decomposition) (Martinsson and Tygert 2013). In addition, notations DIREPS and INVEPS are used for desired precision in direct and inverse Legendre transform, respectively. Hereafter, fast Legendre transform using the algorithm of Tygert (2010b) is named as FLT.

\[ a. \text{ Interpolative decomposition property of fast spherical harmonic transform} \]

1) THE RANKS OF PROJECTION MATRICES

Figure 3 shows the ranks of projection matrices with INVEPS = DIREPS = 1.0 \times 10^{-7} for resolution T7999. Similar results with INVEPS = DIREPS = 1.0 \times 10^{-10} are also shown in Fig. 4. The min/max rank in Figs. 3 and 4 is the minimal/maximum rank in all projection matrices for zonal wavenumber \( m \). It is clear that the algorithm of Tygert (2010b) satisfies the interpolative decomposition property. The preaccuracy DIREPS/INVEPS has nearly no impact on ranks of projection matrices for large zonal wavenumber \( m \). However, high preaccuracy DIREPS/INVEPS will result in larger ranks for small zonal wavenumber \( m \).

2) MAXIMUM ABSOLUTE VALUE OF PROJECTION MATRICES

The potential numerical instability of ID depends on properties of the matrix to be decomposed. Figure 5 shows maximum absolute values of projection matrices with preaccuracy 10^{-7} for Legendre–Vandermonde (LV) matrices (Legendre polynomial and Gauss–Legendre points). The maximum absolute values of projection matrices with preaccuracy 10^{-7} for LV matrices are less than constant 3, and similar results are also obtained for preaccuracy 10^{-4}, 10^{-6}, and 10^{-8}. It means that ID is stable for LV matrices.

---

**Fig. 13.** Results of wind fields for resolution T1279. The wind fields are computed from true wind fields by using one backward and one forward spherical harmonic transform.
Figure 6 shows the maximum absolute values of projection matrices for resolution T7999 in YHGSM. The decomposed matrices are the associated-Legendre–Vandermonde (ALV) matrix in direct SHT while the “product” of ALV matrix and weights of Gaussian–Legendre integration is in inverse SHT. Without loss of generality, the decomposed matrices are named as ALV matrices. It can be found that all the elements of projection matrices have an absolute value less than or equal to a reasonably small positive real number. But a relatively bigger value $1336.959\ 671\ 577\ 676\ 09$ at wavenumber $m = 60$ exists in the case of $C_{\text{max}} = 128$ and $\text{EPS} = 1.0 \times 10^{-7}$ for inverse transform of asymmetric part on a reduced Gaussian grid. Besides, a value $1376.669\ 994\ 807\ 441\ 07$ at wavenumber $m = 493$ exists in the case of $C_{\text{max}} = 128$ and $\text{EPS} = 1.0 \times 10^{-7}$ for inverse transform of asymmetric part on full Gaussian grid. Larger values only occur in the cases with small zonal wavenumber $m$. These unexpected results mean that ID may be numerically unstable for inverse transform at very high resolutions.

In addition, the maximum absolute values of projection matrices for LV matrices are less than those for ALV matrices in Fig. 6, especially for small zonal wavenumbers. The properties of projection matrices for ALV matrices are worse than those for LV matrices. The ID for LV matrices is stable while it may be unstable for ALV matrices with small zonal wavenumber $m$. The above results demonstrate that the instability of ID for SHT may depend on high-order associated Legendre polynomials (Yessad 2018) and the use of a reduced Gaussian grid. However, higher preaccuracy DIREPS/INVEPS and smaller $C_{\text{max}}$ contribute to the stability of interpolative decomposition.
Figure 7 shows the compression performance of the sparse matrix storage format for resolution T7999. Note that the sparse matrix storage format is not used for matrices whose dimension is less than Cmax. As Fig. 7 shows, the sparse matrix storage format can save more than 50% storage. Similar results with INVEPS = DIREPS = $1.0 \times 10^{-10}$ can also be obtained but are not shown here. The above results are consistent with the

![Figure 7](image_url)

**Table 5.** The wall-clock time of 50 time steps in YHGSM (unit: s).

<table>
<thead>
<tr>
<th>EPS</th>
<th>CMAX</th>
<th>T799L91</th>
<th>T1279L91</th>
<th>T2047L91</th>
<th>T3999L91</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Computational</td>
<td>Pretime</td>
<td>Computational</td>
<td>Pretime</td>
</tr>
<tr>
<td>1.0 $\times$ 10 $^{-7}$</td>
<td>64</td>
<td>99.215</td>
<td>7.845</td>
<td>256.303</td>
<td>59.497</td>
</tr>
<tr>
<td>1.0 $\times$ 10 $^{-10}$</td>
<td>128</td>
<td>100.458</td>
<td>7.162</td>
<td>258.707</td>
<td>51.693</td>
</tr>
<tr>
<td>1.0 $\times$ 10 $^{-10}$</td>
<td>64</td>
<td>100.314</td>
<td>7.896</td>
<td>258.555</td>
<td>61.075</td>
</tr>
<tr>
<td>1.0 $\times$ 10 $^{-10}$</td>
<td>128</td>
<td>100.575</td>
<td>7.215</td>
<td>259.071</td>
<td>52.209</td>
</tr>
<tr>
<td>DGEMM</td>
<td></td>
<td>108.012</td>
<td>2.158</td>
<td>312.470</td>
<td>5.250</td>
</tr>
<tr>
<td>Proc Num</td>
<td>40 nodes $\times$ 24 processors</td>
<td>64 nodes $\times$ 16 processors</td>
<td>80 nodes $\times$ 8 processors</td>
<td>128 nodes $\times$ 4 processors</td>
<td></td>
</tr>
</tbody>
</table>
The compression ratio in Fig. 7 is defined as the data size becomes X% smaller than the original size.

3) The Number of Nonzero Entries of Projection Matrices

Figure 8 shows the numbers of nonzero in butterfly compression representations for resolution T7999. As the results show, the number of nonzero has an asymptotic scaling of $O(N \log^3 N) \sim O(N \log^2 N)$ for all zonal wave-numbers. So fast Legendre transform using butterfly matrix–vector multiplication has nearly optimal operation complexity of $O(N \log^2 N) \sim O(N \log^2 N)$ in theory. It can also be found that the numbers of nonzero with Cmax = 64 are less than those obtained with Cmax = 128 and the numbers of nonzero with EPS = 1.0 × 10⁻⁷ are less...
than those obtained with \( \text{EPS} = 1.0 \times 10^{-10} \). It means that the minimal numbers of floating-point operations is obtained when parameter \( C_{\text{max}} \) is set to 64. The ID with high desired precision (small \( \varepsilon \)) has more numbers of nonzero than that with low desired precision (large \( \varepsilon \)). Therefore, the asymptotic cost of FLT also depends on desired precision \( \varepsilon \). It is recommended to use a lower desired precision \( \varepsilon \) when enough precision is obtained.

It can be found that both of the ranks and the numbers of nonzero with \( C_{\text{max}} = 64 \) are less than those with \( C_{\text{max}} = 128 \). It means that the computation cost of FLT with \( C_{\text{max}} = 64 \) may be less than those with \( C_{\text{max}} = 128 \). However, smaller \( C_{\text{max}} \) will increase the level \( L \) and the number of projection matrices in ID. Thus, more matrix–matrix multiplying operations are needed in butterfly matrix–matrix multiplication algorithm. The advantage due to less nonzero entries on computation cost may be offset by increasing the number of matrix–vector multiplication operations and also depends on the hardware in practical application.

**b. The performance of fast spherical harmonics transform**

1) **PRECOMPUTATIONAL COST**

Table 3 shows the memory cost of FLT for single spherical harmonic transform with \( C_{\text{max}} = 128 \). In Fig. 9 the memory costs are scaled by \( N^2 \log^4 N \). As the results show, the memory complexity of FLT is \( O(N^2 \log^4 N) \). Furthermore, we found that the memory costs for FLT with \( \text{INVEPS} = 1.0 \times 10^{-4} \) and \( \text{DIREPS} = 1.0 \times 10^{-7} \) (not shown here) are almost the same as those with \( \text{INVEPS} = \text{DIREPS} = 1.0 \times 10^{-7} \) for \( C_{\text{max}} = 128 \) and the memory costs of FLT with \( \text{INVEPS} = 1.0 \times 10^{-10} \) and \( \text{DIREPS} = 1.0 \times 10^{-10} \) are a little bigger than those with \( \text{INVEPS} = \text{DIREPS} = 1.0 \times 10^{-7} \) for \( C_{\text{max}} = 128 \). It means that memory costs of FLT are not sensitive to parameters \( \text{DIREPS} \) and \( \text{INVEPS} \). The adoption of high accuracy in butterfly compression only results in a slight increase in memory cost.

2) **COMPUTATIONAL COST**

To minimize memory usage, we only keep ALV matrices and their transpose with wavenumber \( (m) \) greater than \( \text{NSMAX} - 2 \times \text{Dimthresh} + 3 \) (\( \text{NSMAX} \) is truncation order) and ID matrices used in the butterfly algorithm with wavenumber \( (m) \) less than or equal to \( \text{NSMAX} - 2 \times \text{Dimthresh} + 3 \) in memory. In addition, BLAS function \( \text{DGEMM} \) is also used for those matrix–matrix multiplications whose dimension is less than \( C_{\text{max}} \) in FLT. Let \( \text{DIRPES} = \text{INVEPS} = \text{EPS} \) for all tests in this section.

Computational time for LT with difference zonal wavenumber \( (m) \) is shown in Fig. 10. We suggest that no
threshold should be used for Cmax = 128 and threshold 128 is adopted for Cmax = 64. Hereafter, both SHT and LT contain one direct transform and one inverse transform, and the computational time is the sum of execution time of all processes and the unit is in the number of system clock cycles, which divided by 10000 can get the unit in seconds. The DIMTHREASH is set to 128 for the remainder of tests in this section.

Table 4 lists the computational time for LT part of SHT using FLT and DGEMM. The parallelization of single Legendre transform is not considered in YHGSM. Since single Legendre transform is performed in the same node, no inter node communication is needed. All tests in Table 4 (top) are run on 64 nodes each with one compute process. To study the impact of memory footprint on the computational time, the test with resolution T7999 uses 128 nodes each with two compute process while other tests have 16 compute processes per node in Table 4 (bottom). It can be found that memory footprint indeed influences the computational time of Legendre transform especially LT using DGEMM. This is because FLT can save more than half the memory compared with LT using DGEMM. Figure 11 shows the speedup for LT part of SHT compared with DGEMM. It can be found that the computational time for FLT is less than DGEMM when the resolution is higher than or equal to T255 and the difference enlarges with increasing resolution. Furthermore, the computational time for FLT with Cmax = 64 is less than that with Cmax = 128 for EPS = 1.0 × 10^{-7} while this trend is not clearly for EPS = 1.0 × 10^{-10}. This may depend on the status of nodes and the architecture of hardware that is consistent with previous analysis in the end of section 3a(3).

Figure 12 shows the computational time for the LT part of SHT scaled by different complexities. Without regard to T7999 whose results are susceptible to memory and cache, it can be concluded that the computational complexity of FLT part of SHT is O(N^2 \log^3 N).

Two real fields on the spherical reduced Gaussian grid are used to demonstrate the correctness of fast SHT. Figure 13 compares the results of wind fields for resolution T1279. The wind fields in Fig. 13 are evaluated from real wind fields by using one backward SHT and one forward SHT. Figure 14 shows the errors of wind fields in Fig. 13. Figure 15 shows the errors of wind fields which obtained by using the same transform.
The results in Fig. 13 are in agreement with the real fields. Comparing Fig. 14 and Fig. 15, it can be found that fast SHT is as numerically stable as original SHT using DGEMM. Moreover, the results of FLT with $\text{EPS} = 1.0 \times 10^{-7}$ are closer to those of DGEMM.

Table 5 lists the wall-clock time for 50 time steps of YHGSM. The computational time with $\text{Cmax} = 64$ is significantly less than that with $\text{Cmax} = 128$ while the precomputational time is reversed. The precomputational time of T3999L91 amounts to that of about 60 time steps of YHGSM. Therefore, the parameters $\text{Cmax} = 64$ and $\text{EPS} = 1.0 \times 10^{-7}$ is recommended in YHGSM.

3) TEST CASES

In this section, three test cases including the zonal wavenumber-4 Rossby–Haurwitz wave (Williamson et al. 1992), the baroclinic instability wave (Jablonowski and Williamson 2006), and the true forecast on YHGSM are employed to validate the correctness and reliability of YHGSM using FLT. The accuracy of SHT is estimated by the following error functions:

$$\text{error}_{\text{MAX}} = \max_{i=1,\ldots,N} |u_{\text{exact}}(i) - u_{\text{approx}}(i)|$$

and

$$\text{error}_{\text{RMS}} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (u_{\text{exact}}(i) - u_{\text{approx}}(i))^2}.$$ 

The $u$ wind in the initial fields of the Rossby–Haurwitz wave test case is used to demonstrate the accuracy of SHT using FLT. Errors of the $u$ wind field are listed in Table 6. It can be found that the errors of SHT using FLT with $\text{EPS} = 1.0 \times 10^{-10}$ are less than those with $\text{EPS} = 1.0 \times 10^{-7}$ and the errors become greater with the increase of the resolution. The RMS error is smaller than the desired precision $\epsilon$ except for the cases with $\text{EPS} = 1.0 \times 10^{-10}$ and $\text{Cmax} = 64$ for resolutions T1023 and T1279. The RMS error is one order higher than the desired precision $\epsilon$ with increased resolution. No significant loss of precision was found in the case with $\text{Cmax} = 128$ and

300 times. The results in Fig. 13 are in agreement with the real fields. Comparing Fig. 14 and Fig. 15, it can be found that fast SHT is as numerically stable as original SHT using DGEMM. Moreover, the results of FLT with $\text{EPS} = 1.0 \times 10^{-10}$ are closer to those of DGEMM.

Table 5 lists the wall-clock time for 50 time steps of YHGSM. The computational time with $\text{Cmax} = 64$ is significantly less than that with $\text{Cmax} = 128$ while the precomputational time is reversed. The precomputational time of T3999L91 amounts to that of about 60 time steps of YHGSM using FLT. Therefore, the parameters $\text{Cmax} = 64$ and $\text{EPS} = 1.0 \times 10^{-7}$ is recommended in YHGSM.

3) TEST CASES

In this section, three test cases including the zonal wavenumber-4 Rossby–Haurwitz wave (Williamson et al. 1992), the baroclinic instability wave (Jablonowski and Williamson 2006), and the true forecast on YHGSM are employed to validate the correctness and reliability of YHGSM using FLT. The accuracy of SHT is estimated by the following error functions:

$$\text{error}_{\text{MAX}} = \max_{i=1,\ldots,N} |u_{\text{exact}}(i) - u_{\text{approx}}(i)|$$

and

$$\text{error}_{\text{RMS}} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (u_{\text{exact}}(i) - u_{\text{approx}}(i))^2}.$$ 

The $u$ wind in the initial fields of the Rossby–Haurwitz wave test case is used to demonstrate the accuracy of SHT using FLT. Errors of the $u$ wind field are listed in Table 6. It can be found that the errors of SHT using FLT with $\text{EPS} = 1.0 \times 10^{-10}$ are less than those with $\text{EPS} = 1.0 \times 10^{-7}$ and the errors become greater with the increase of the resolution. The RMS error is smaller than the desired precision $\epsilon$ except for the cases with $\text{EPS} = 1.0 \times 10^{-10}$ and $\text{Cmax} = 64$ for resolutions T1023 and T1279. The RMS error is one order higher than the desired precision $\epsilon$ with increased resolution. No significant loss of precision was found in the case with $\text{Cmax} = 128$ and
EPS = $1.0 \times 10^{-10}$ for resolution T7999. This may be due to the smoothness of test fields. Most of the energy of function is concentrated in the low-frequency part. In addition, the instability of FLT occurring in the computation involves high-order associated Legendre polynomials. Therefore, the influence of the instability to the precision of SHT is so small that it can be ignored.

Figure 16 shows log surface pressure and $u$ and $v$ component of wind at 500 hPa at day 10 from the YHGSM dynamic core at resolution T511L91. The agreement among the results of FLT with EPS = $1.0 \times 10^{-7}$, FLT with EPS = $1.0 \times 10^{-10}$, and DGEMM in Fig. 16 gives us confidence that the model using FLT is reliable and stable.

The wave is set in breaking at day 10 for the instability baroclinic wave test case. Figure 17 displays the evolution of the baroclinic wave from days 8 to 10. The results in Fig. 17 depict the similar common structure of temperature as those shown in Fig. 5h (Jablonowski and Williamson 2006). In addition, the results of FLT were in accordance with the results of DGEMM.

Figures 18–20 depict the evolving fields of temperature, as well as $u$-and $v$-wind components on the 500-hPa pressure level for a 10-day forecast on YHGSM using difference Legendre transform. In Figs. 18–20, from top to bottom are the results of days 0, 5, and 10, respectively. The first, second, and third columns are the results of using FLT with EPS = $1.0 \times 10^{-7}$, FLT with EPS = $1.0 \times 10^{-10}$, and DGEMM, respectively. Although, there are only a few subtle differences in some points, the results of FLT are generally consistent with those of DGEMM.

4. Conclusions

We have presented an implementation of the fast spherical harmonic algorithm on YHGSM and discussed its performance, overall efficiency, and scalability. There are five important points to be made here. 1) The number of nonzero elements in butterfly compression representation has an asymptotic scaling of $O(N \log^2 N) \sim O(N \log^{3.5} N)$, so the computational complexity of FLT is $O(N \log^2 N) \sim O(N \log^{3.5} N)$ in theory. 2) The computational complexity of LT part of SHT is $O(N^2 \log^3 N)$ in practical application. 3) The memory cost of LT part of SHT is $O(N^3 \log^4 N)$. 4) $C_{\text{max}} = 64$ and EPS = $1.0 \times 10^{-7}$ are suggested for YHGSM. 5) The instability of SHT for very high resolution may
depend on high-order associated Legendre polynomials and the use of reduced Gaussian grid. In addition, the influence of instability to precision of SHT is negligible for smooth and continuous fields. These results demonstrate that fast SHT is feasible and efficient in reducing the computational complexity and the memory cost of YHGSM.

Acknowledgments. This work is supported by the National Natural Science Foundation of China (Grant 41705078) and partly supported by the National Natural Science Foundation of China (Grants 61379022 and 41605070). The authors are very grateful to the reviewers for carefully reading the paper and for their comments and suggestions, which have improved the paper.

APPENDIX

Pseudocode for Key Subroutines of FLT

In the butterfly multiply step, FLT needs to perform the matrix premultiplying operation of projection matrix. Algorithm 1 is used to perform the sparse matrix multiplication for projection matrix. Algorithm 1 completes the initialization of FLT. Listing 1 completes the initialization of FLT. Listing 2 and 3 show how to implement FLT in YHGSM.


```c
if(krank<n)then
  call dgemm('n','n',krank,nflev,n-krank, 1.0d0, p,krank,&
  & beta(list(krank+1:n),:),n-krank,0.0, b,krank)
  b = b + beta(list(1:krank),:)
else
  call dgemm('n','n',krank,nflev,n, 1.0d0, p,krank,&
  & beta,n,0.0,b,krank)
endif
```

Listing 1. flt_initialization.

```c
if(FLTFLAG)then
  call flt_dir_idallocation()
  call flt_inv_idallocation()
  if(PRECFLAG)then
    call flt_dir_precomputation()
    call flt_inv_precomputation()
    if(PRE_SAVE)then
      call flt_dir_idsaveledir()
      call flt_inv_idsaveleinv()
    endif
  else
    call flt_dir_idreadledir()
    call flt_inv_idreadleinv()
  endif
else
  call flt_dir_idreadledir()
  call flt_inv_idreadleinv()
endif
```

Listing 2. lt_dir_trans

```c
if(FLTFLAG.and.(IM.LE.NSMAX-2*DIMTHREASH))then
  call flt_dir_mxm() !symmetric part
  call flt_dir_mxm() !asymmetric part
else
  call dgemm() !symmetric part
  call dgemm() !asymmetric part
endif
```

Listing 3. lt_inv_trans

```c
if(FLTFLAG.and.(IM.LE.NSMAX-2*DIMTHREASH+3))then
  call flt_inv_mxm() !symmetric part
  call flt_inv_mxm() !asymmetric part
else
  call dgemm() !symmetric part
  call dgemm() !asymmetric part
endif
```
REFERENCES


