mathematics

MDPI

Article

A High Accurate and Stable Legendre Transform Based on Block Partitioning and Butterfly Algorithm for NWP

Fukang Yin *, Jianping Wu, Junqiang Song and Jinhui Yang

College of Meteorology and Oceanography, National University of Defense Technology, Changsha 410073, China; wjp@nudt.edu.cn (J.W.); junqiang@nudt.edu.cn (J.S.); yangjinhui@nudt.edu.cn (J.Y.)

* Correspondence: yinfukang@nudt.edu.cn; Tel.: +86-15084976786

Received: 21 September 2019; Accepted: 10 October 2019; Published: 14 October 2019

Abstract: In this paper, we proposed a high accurate and stable Legendre transform algorithm, which can reduce the potential instability for a very high order at a very small increase in the computational time. The error analysis of interpolative decomposition for Legendre transform is presented. By employing block partitioning of the Legendre-Vandermonde matrix and butterfly algorithm, a new Legendre transform algorithm with computational complexity $O(N\log^2 N/\log\log N)$ in theory and $O(N\log^3 N)$ in practical application is obtained. Numerical results are provided to demonstrate the efficiency and numerical stability of the new algorithm.

Keywords: Legendre transform; block partitioning; interpolative decomposition; butterfly algorithm

1. Introduction

Legendre transform (LT) plays an important part in many scientific applications, such as astrophysical [1], numerical weather prediction and climate models [2,3]. Fast Legendre transform attracts considerable interest amongst the scientific computing and numerical simulation. Scientists have paid very serious attention to develop fast Legendre transform algorithms [4–11]. The validity and reliability of these algorithms depend on whether they can keep fast, stable and high accuracy.

The butterfly algorithm [12,13] is an effective multilevel technique to compress a matrix that satisfies a complementary low-rank property. It 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. Hence, dense matrix-vector multiplication can be transformed into a set of sparse matrix-vector multiplication in $O(N\log N)$ operations [14]. LT using butterfly algorithm has the advantages of high accuracy, stability and low computational complexity.

As one of the most widely used butterfly algorithms, Tygert's algorithm (2010) [11] has been successfully implemented in IFS of ECMWF [2], YHGSM [15–17] of NUDT [3] and astrophysical [1]. In the applications of numerical weather prediction and climate models, which need spectral harmonic transform (SHT) many times for each time step, only one precomputation is needed in the first-time step, then the results are stored in memory and reused in each transform. Though Tygert's algorithm (2010) is slow in terms of precomputation: $O(N^2)$ for LT and $O(N^3)$ for SHT, it does not have much impact on total performance. However, some unsolved issues still remain. The main issue is the potential instability of interpolative decomposition (ID) [18] for very high order Legendre transform. That said, Tygert [11] points out that the reason why the butterfly procedure works so well for associated Legendre functions may be that the associated transforms nearly weighted averages of Fourier integral operators. There are no literatures to prove that the pre-computations will compress the appropriate $N\times N$ matrix enough to enable application of the matrix to vectors using only $O(N\log N)$ floating-point

operations(flops). Full numerical stability has been demonstrated both empirically and theoretically for fast Fourier transform (FFT) using butterfly algorithm. It is difficult to give complete and rigorous proofs of interpolative decomposition for Legendre transform as Fourier transform.

Non-oscillatory phase functions method opens up new avenues for special function transforms. The solutions of some kinds of second order differential equations can be accurately represented by non-oscillatory phase functions [19,20]. It has been proved that Legendre's differential equation [21] and its generalization Jacobi's differential equation [22] admit a non-oscillatory phase function. So non-oscillatory phase functions can be used to the expansions [22], the calculation of the roots [23] and transform [24] of special functions. Jacobi transform by non-oscillatory phase functions shows an optimal computational complexity $O(N\log^2 N/\log\log N)$ in reference [24]. However, Legendre transform algorithm in ButterflyLab [25], which adopts interpolative butterfly factorization (IBF) [14,26] and non-oscillatory phase functions method to evaluate the Legendre polynomials [24], does not show high accuracy as Fourier transform using IBF. Therefore, Fast Legendre transform (FLT) based on IBF and non-oscillatory phase functions and its extension to the associated Legendre functions need further study.

Recently, fast Legendre transform algorithm based on FFT deserved more attentions for its optimal computational complexity $O(N\log^2 N/\log\log N)$. Hale and Townsend [27] firstly presented a fast Chebyshev-Legendre transform, and then developed a non-uniform discrete cosine transform which use a Taylor series expansion for Chebyshev polynomials about equally-spaced points in the frequency domain. Finally, Hale and Townsend [28] got an $O(N\log^2 N/\log\log N)$ Legendre transform algorithm. In the near future, fast polynomial transforms [29] based on Toeplitz and Hankel matrices will be presented to accelerate the Chebyshev-Legendre transform. Although FFT-based LT has the attractive computational complexity, it needs too many times FFT, which makes FFT-based LT only become more computationally efficient than LT using Dgemv when N is greater than or equal to 5000. Since the computation of associated-Legendre-Vandermonde matrices is completed in the pre-computation step, it will become worse on the occasion of multiple use of FLT such as NWP, in which only once computation of associated-Legendre-Vandermonde matrices is needed for many times spectral harmonic transform (SHT).

Motivated and inspired by the ongoing research in these areas, we present a theoretical method to analyze the error of LT using butterfly algorithm, and then provide a numerically stability Legendre transform algorithm based on block partitioning and butterfly algorithm. The novel aspect is the mitigation of the potential instability of LT using butterfly algorithm at a very small increase of computational cost.

2. Mathematical Preliminaries

In this section, we introduce the theorem that Legendre polynomials on equally-spaced grid can be expressed as a weighted linear combination of Chebyshev polynomials, and a partitioning of Legendre-Vandermonde matrix $\mathbf{P}_N\left(x_N^{cheb}\right)$ ($x = \underline{x}_N^{cheb} = \cos\left(\underline{\theta}_N^{cheb}\right)$). For more details, see references [27,28].

According to Stieltjes's theory [30], Legendre polynomials can be expressed as following asymptotic formula when $n \to \infty$

$$P_n(\cos\theta) = C_n \sum_{m=0}^{M-1} h_{m,n} \frac{\cos\left(\left(m+n+\frac{1}{2}\right)\theta - \left(m+\frac{1}{2}\right)\frac{\pi}{2}\right)}{(2\sin\theta)^{m+1/2}} + R_{M,n}(\theta) \tag{1}$$

where $\theta = \cos^{-1} x$, $\theta \in (0, \pi)$ and

$$C_n = \frac{4}{\pi}\prod_{j=1}^{n}\frac{j}{j+1/2} = \sqrt{\frac{4}{\pi}}\frac{\Gamma(n+1)}{\Gamma(n+3/2)} \tag{2}$$

$$h_{m,n} = \begin{cases} 1, & m = 0, \\ \prod_{j=1}^{m} \frac{(j-1/2)^2}{j(n+j+1/2)}, & m > 0. \end{cases} \tag{3}$$

The error term in Equation (1) can be bounded by

$$\left|R_{M,n}(\theta)\right| \leq C_n h_{M,n} \frac{2}{(2\sin\theta)^{M+1/2}} \tag{4}$$

Hale and Townsend [27] rewrote Equation (1) as a weighted linear combination of Chebyshev polynomials

$$P_n(\cos\theta) = C_n \sum_{m=0}^{M-1} h_{m,n}(u_m(\theta)T_n(\sin\theta) + v_m(\theta)T_n(\cos\theta)) + R_{M,n}(\theta) \tag{5}$$

with $T_n(\cos\theta) = \cos(n\theta)$, $T_n(\sin\theta) = \sin(n\theta)$ and

$$u_m(\theta) = \frac{\sin\left((m+1/2)\left(\frac{\pi}{2}-\theta\right)\right)}{(2\sin\theta)^{m+1/2}}, v_m(\theta) = \frac{\cos\left((m+1/2)\left(\frac{\pi}{2}-\theta\right)\right)}{(2\sin\theta)^{m+1/2}} \tag{6}$$

Let $x_k^{leg} = \cos\left(\theta_k^{leg}\right)$ and $\theta_0^{leg}, \cdots, \theta_{N-1}^{leg}$ are the transformed Legendre nodes, Equation (5) can be written as

$$P_n\left(x_k^{leg}\right) = C_n \sum_{m=0}^{M-1} h_{m,n}\left(u_m\left(\theta_k^{leg}\right)T_n\left(\sin\left(\theta_k^{leg}\right)\right) + v_m\left(\theta_k^{leg}\right)T_n\left(\cos\left(\theta_k^{leg}\right)\right)\right) + R_{M,n}\left(\theta_k^{leg}\right) \tag{7}$$

3. Error Analysis of Legendre Transform using Butterfly Algorithm

The transformed Legendre nodes $\theta_0^{leg}, \cdots, \theta_{N-1}^{leg}$ can be seen as a perturbation of an equally-spaced grid $\theta_0^*, \cdots, \theta_{N-1}^*$, i.e

$$\theta_k^{leg} = \theta_k^* + \delta\theta_k,\ 0 \leq k \leq N-1 \tag{8}$$

and then approximate each $x_k^{leg} = \cos\left(n\theta_k^{leg}\right)$ term by a truncated Taylor series expansion about θ_k^*. If $|\delta\theta_k|$ is small then only a few terms in the Taylor expansion are required.

The Taylor series expansion of $T_n(\cos(\theta+\delta\theta)) = \cos(n(\theta+\delta\theta))$ about $\theta \in [0,\pi]$ can be expressed as

$$\begin{aligned} \cos(n(\theta+\delta\theta)) &= \cos(n\theta) + \sum_{l=1}^{\infty} \cos^{(l)}(n\theta)\frac{(n\delta\theta)^l}{l!} \\ &= \cos(n\theta) + \sum_{l=1}^{\infty} (-1)^{\lfloor (l+1)/2 \rfloor}\Phi_l(n\theta)\frac{(n\delta\theta)^l}{l!} \end{aligned} \tag{9}$$

where

$$\Phi_l(\theta) = \begin{cases} \cos(\theta), & l \quad even \\ \sin(\theta), & l \quad odd \end{cases} \tag{10}$$

Similarly, $T_n(\sin(\theta+\delta\theta)) = \sin(n(\theta+\delta\theta))$ about $\theta \in [0,\pi]$ can be expressed as

$$\begin{aligned} \sin(n(\theta+\delta\theta)) &= \sin(n\theta) + \sum_{l=1}^{\infty} \sin^{(l)}(n\theta)\frac{(n\delta\theta)^l}{l!} \\ &= \sin(n\theta) + \sum_{l=1}^{\infty} (-1)^{\lfloor l/2 \rfloor}\Psi_l(n\theta)\frac{(n\delta\theta)^l}{l!} \end{aligned} \tag{11}$$

where

$$\Psi_l(\theta) = \begin{cases} \cos(\theta), & l \quad odd \\ \sin(\theta), & l \quad even \end{cases} \tag{12}$$

Substituting θ_k^* for θ in Equation (5), one can get

$$\begin{aligned} P_n\left(\cos\left(\theta_k^*\right)\right) = \quad & C_n T_n\left(\sin\left(\theta_k^*\right)\right)\sum_{m=0}^{M-1} h_{m,n} u_m\left(\theta_k^*\right) \\ & + C_n T_n\left(\cos\left(\theta_k^*\right)\right)\sum_{m=0}^{M-1} h_{m,n} v_m\left(\theta_k^*\right) + R_{M,n}\left(\theta_k^*\right) \end{aligned} \tag{13}$$

The Taylor series expansion of $P_n\left(\cos\left(\theta_k^{leg}\right)\right)$ about θ_k^* can be expressed as

$$P_n\left(\cos\left(\theta_k^{leg}\right)\right) = P_n\left(\cos\left(\theta_k^* + \delta\theta_k\right)\right) = \sum_{l=0}^{\infty} P_n^{(l)}\left(\cos\left(\theta_k^*\right)\right)\frac{\left(\delta\theta_k\right)^l}{l!} \tag{14}$$

According to Equation (13), $P_n^{(l)}\left(\cos\left(\theta_k^*\right)\right)$ $(l > 0)$ can be written as

$$\begin{aligned} P_n^{(l)}\left(\cos\left(\theta_k^*\right)\right) = \quad & C_n \sum_{m=0}^{M-1} h_{m,n}\left\{u_m\left(\theta_k^*\right) T_n^{(l)}\left(\sin\left(\theta_k^*\right)\right) + u_m^{(l)}\left(\theta_k^*\right) T_n\left(\sin\left(\theta_k^*\right)\right)\right\} \\ & + C_n \sum_{m=0}^{M-1} h_{m,n}\left\{v_m\left(\theta_k^*\right) T_n^{(l)}\left(\cos\left(\theta_k^*\right)\right) + v_m^{(l)}\left(\theta_k^*\right) T_n\left(\cos\left(\theta_k^*\right)\right)\right\} \\ & + R_{M,n}^{l}\left(\theta_k^*\right) \end{aligned} \tag{15}$$

Substituting Equation (15) into Equation (14), one can obtain

$$\begin{aligned} P_n\left(x_k^{leg}\right) = \quad & C_n \sum_{m=0}^{M-1} h_{m,n}\left\{u_m\left(\theta_k^*\right) T_n\left(\sin\left(\theta_k^*\right)\right) + v_m\left(\theta_k^*\right) T_n\left(\cos\left(\theta_k^*\right)\right)\right\} \\ & + C_n \sum_{l=1}^{\infty} \frac{\left(\delta\theta_k\right)^l}{l!} \sum_{m=0}^{M-1} h_{m,n}\left\{u_m\left(\theta_k^*\right) T_n^{(l)}\left(\sin\left(\theta_k^*\right)\right) + u_m^{(l)}\left(\theta_k^*\right) T_n\left(\sin\left(\theta_k^*\right)\right)\right\} \\ & + C_n \sum_{l=1}^{\infty} \frac{\left(\delta\theta_k\right)^l}{l!} \sum_{m=0}^{M-1} h_{m,n}\left\{v_m\left(\theta_k^*\right) T_n^{(l)}\left(\cos\left(\theta_k^*\right)\right) + v_m^{(l)}\left(\theta_k^*\right) T_n\left(\cos\left(\theta_k^*\right)\right)\right\} \\ & + \sum_{l=0}^{\infty} R_{M,n}^{(l)}\left(\theta_k^*\right)\frac{\left(\delta\theta_k\right)^l}{l!} \end{aligned} \tag{16}$$

Because

$$\begin{aligned} \sum_{l=0}^{\infty} \frac{\left(\delta\theta_k\right)^l}{l!} T_n\left(\sin\left(\theta_k^*\right)\right)\sum_{m=0}^{M-1} h_{m,n} u_m^{(l)}\left(\theta_k^*\right) &= T_n\left(\sin\left(\theta_k^*\right)\right)\sum_{m=0}^{M-1} h_{m,n}\sum_{l=0}^{\infty} u_m^{(l)}\left(\theta_k^*\right)\frac{\left(\delta\theta_k\right)^l}{l!} \\ &= T_n\left(\sin\left(\theta_k^*\right)\right)\sum_{m=0}^{M-1} h_{m,n} u_m\left(\theta_k^* + \delta\theta_k\right) \\ &= T_n\left(\sin\left(\theta_k^*\right)\right)\sum_{m=0}^{M-1} h_{m,n} u_m\left(\theta_k^{leg}\right) \end{aligned} \tag{17}$$

and

$$\begin{aligned} \sum_{l=0}^{\infty} \frac{\left(\delta\theta_k\right)^l}{l!} T_n\left(\cos\left(\theta_k^*\right)\right)\sum_{m=0}^{M-1} h_{m,n} v_m^{(l)}\left(\theta_k^*\right) &= T_n\left(\cos\left(\theta_k^*\right)\right)\sum_{m=0}^{M-1} h_{m,n}\sum_{l=0}^{\infty} v_m^{(l)}\left(\theta_k^*\right)\frac{\left(\delta\theta_k\right)^l}{l!} \\ &= T_n\left(\cos\left(\theta_k^*\right)\right)\sum_{m=0}^{M-1} h_{m,n} v_m\left(\theta_k^* + \delta\theta_k\right) \\ &= T_n\left(\cos\left(\theta_k^*\right)\right)\sum_{m=0}^{M-1} h_{m,n} v_m\left(\theta_k^{leg}\right) \end{aligned} \tag{18}$$

Similarly, we have

$$\begin{aligned}\sum_{l=0}^{\infty}\frac{(\delta\theta_k)^l}{l!}\sum_{m=0}^{M-1}h_{m,n}u_m\left(\theta_k^*\right)T_n^{(l)}\left(\sin\left(\theta_k^*\right)\right) &= \sum_{m=0}^{M-1}h_{m,n}u_m\left(\theta_k^*\right)\sum_{l=0}^{\infty}T_n^{(l)}\left(\sin\left(\theta_k^*\right)\right)\frac{(\delta\theta_k)^l}{l!}\\ &= \sum_{m=0}^{M-1}h_{m,n}u_m\left(\theta_k^*\right)T_n\left(\sin\left(\theta_k^{leg}\right)\right)\end{aligned} \tag{19}$$

and

$$\begin{aligned}\sum_{l=0}^{\infty}\frac{(\delta\theta_k)^l}{l!}\sum_{m=0}^{M-1}h_{m,n}v_m\left(\theta_k^*\right)T_n^{(l)}\left(\cos\left(\theta_k^*\right)\right) &= \sum_{m=0}^{M-1}h_{m,n}v_m\left(\theta_k^*\right)\sum_{l=0}^{\infty}T_n^{(l)}\left(\cos\left(\theta_k^*\right)\right)\frac{(\delta\theta_k)^l}{l!}\\ &= \sum_{m=0}^{M-1}h_{m,n}v_m\left(\theta_k^*\right)T_n\left(\cos\left(\theta_k^{leg}\right)\right)\end{aligned} \tag{20}$$

Substituting Equations (17)–(20) into Equation (16), one can get

$$\begin{aligned}P_n\left(x_k^{leg}\right) = \;& C_n\sum_{m=0}^{M-1}h_{m,n}\left(u_m\left(\theta_k^{leg}\right)T_n\left(\sin\left(\theta_k^*\right)\right)+v_m\left(\theta_k^{leg}\right)T_n\left(\cos\left(\theta_k^*\right)\right)\right)\\ &+C_n\sum_{m=0}^{M-1}h_{m,n}\left\{u_m\left(\theta_k^*\right)T_n\left(\sin\left(\theta_k^{leg}\right)\right)+v_m\left(\theta_k^*\right)T_n\left(\cos\left(\theta_k^{leg}\right)\right)\right\}\\ &-C_n\sum_{m=0}^{M-1}h_{m,n}\left\{u_m\left(\theta_k^*\right)T_n\left(\sin\left(\theta_k^*\right)\right)+v_m\left(\theta_k^*\right)T_n\left(\cos\left(\theta_k^*\right)\right)\right\}\\ &+R_{M,n}\left(\theta_k^{leg}\right)\end{aligned} \tag{21}$$

Then

$$\begin{aligned}P_n\left(x_k^{leg}\right) = \;& C_n\sum_{m=0}^{M-1}h_{m,n}\left(u_m\left(\theta_k^{leg}\right)T_n\left(\sin\left(\theta_k^*\right)\right)+v_m\left(\theta_k^{leg}\right)T_n\left(\cos\left(\theta_k^*\right)\right)\right)\\ &+C_n\sum_{m=0}^{M-1}h_{m,n}\left\{u_m\left(\theta_k^*\right)T_n\left(\sin\left(\theta_k^{leg}\right)\right)+v_m\left(\theta_k^*\right)T_n\left(\cos\left(\theta_k^{leg}\right)\right)\right\}\\ &-P_n\left(x_k^*\right)+R_{M,n}\left(\theta_k^*\right)+R_{M,n}\left(\theta_k^{leg}\right)\end{aligned} \tag{22}$$

By truncating the second term in the right hand side of Equation (19), it can be approximated as

$$\begin{aligned}P_n\left(x_k^{leg}\right) = \;& C_n\sum_{m=0}^{M-1}h_{m,n}\left(u_m\left(\theta_k^{leg}\right)T_n\left(\sin\left(\theta_k^*\right)\right)+v_m\left(\theta_k^{leg}\right)T_n\left(\cos\left(\theta_k^*\right)\right)\right)\\ &+C_n\sum_{l=1}^{L}\frac{(\delta\theta_k)^l}{l!}\sum_{m=0}^{M-1}h_{m,n}\left\{u_m\left(\theta_k^*\right)T_n^{(l)}\left(\sin\left(\theta_k^*\right)\right)+v_m\left(\theta_k^*\right)T_n^{(l)}\left(\cos\left(\theta_k^*\right)\right)\right\}\\ &+R_{M,n}\left(\theta_k^{leg}\right)+R_{L,M,n,\delta\theta}\end{aligned} \tag{23}$$

and then

$$\begin{aligned}P_n\left(x_k^{leg}\right) = \;& C_n\sum_{m=0}^{M-1}h_{m,n}\left(u_m\left(\theta_k^{leg}\right)T_n\left(\sin\left(\theta_k^*\right)\right)+v_m\left(\theta_k^{leg}\right)T_n\left(\cos\left(\theta_k^*\right)\right)\right)\\ &+C_n\sum_{l\ odd}^{L}\frac{(n\delta\theta_k)^l}{l!}\sum_{m=0}^{M-1}h_{m,n}\left((-1)^{\lfloor\frac{l}{2}\rfloor}u_m\left(\theta_k^*\right)T_n\left(\cos\theta_k^*\right)+(-1)^{\lfloor\frac{(l+1)}{2}\rfloor}v_m\left(\theta_k^*\right)T_n\left(\sin\theta_k^*\right)\right)\\ &+C_n\sum_{l\ even}^{L}\frac{(n\delta\theta_k)^l}{l!}\sum_{m=0}^{M-1}h_{m,n}\left((-1)^{\lfloor\frac{l}{2}\rfloor}u_m\left(\theta_k^*\right)T_n\left(\sin\theta_k^*\right)+(-1)^{\lfloor\frac{(l+1)}{2}\rfloor}v_m\left(\theta_k^*\right)T_n\left(\cos\theta_k^*\right)\right)\\ &+R_{M,n}\left(\theta_k^{leg}\right)+R_{L,M,n,\delta\theta}\end{aligned} \tag{24}$$

Equation (24) can be expressed in the following compact form

$$\begin{aligned}P_n\left(x_k^{leg}\right)\approx\left(U_n+V_n\right) \;& +\sum_{l\ odd}^{L}\frac{(n\delta\theta_k)^l}{l!}\left((-1)^{\lfloor\frac{l}{2}\rfloor}U_{nc}+(-1)^{\lfloor\frac{(l+1)}{2}\rfloor}V_{ns}\right)\\ &+\sum_{l\ even}^{L}\frac{(n\delta\theta_k)^l}{l!}\left((-1)^{\lfloor\frac{l}{2}\rfloor}U_{ns}+(-1)^{\lfloor\frac{(l+1)}{2}\rfloor}V_{nc}\right)\end{aligned} \tag{25}$$

where

$$\begin{aligned} U_n &= C_n \sum_{m=0}^{M-1} h_{m,n} u_m\left(\theta_k^{leg}\right) T_n\left(\sin\left(\theta_k^*\right)\right),\ V_n = C_n \sum_{m=0}^{M-1} h_{m,n} v_m\left(\theta_k^{leg}\right) T_n\left(\cos\left(\theta_k^*\right)\right) \\ U_{ns} &= C_n \sum_{m=0}^{M-1} h_{m,n} u_m\left(\theta_k^*\right) T_n\left(\sin\left(\theta_k^*\right)\right),\ V_{ns} = C_n \sum_{m=0}^{M-1} h_{m,n} v_m\left(\theta_k^*\right) T_n\left(\sin\left(\theta_k^*\right)\right) \\ U_{nc} &= C_n \sum_{m=0}^{M-1} h_{m,n} u_m\left(\theta_k^*\right) T_n\left(\cos\left(\theta_k^*\right)\right),\ V_{nc} = C_n \sum_{m=0}^{M-1} h_{m,n} v_m\left(\theta_k^*\right) T_n\left(\cos\left(\theta_k^*\right)\right) \end{aligned} \tag{26}$$

So, the computation of Legendre-Vandermonde matrix can be written as

$$\begin{aligned} \mathbf{P}_{\mathrm{N}}\left(\underline{x}_N^{leg}\right) = (\mathbf{U}_{\mathrm{N}} + \mathbf{V}_{\mathrm{N}}) \quad &+ \sum_{l\ odd}^{L} \frac{(n\delta\theta_k)^l}{l!}\left((-1)^{\lfloor \frac{l}{2} \rfloor}\mathbf{U}_{\mathrm{c}} + (-1)^{\lfloor \frac{(l+1)}{2} \rfloor}\mathbf{V}_{\mathrm{s}}\right) \\ &+ \sum_{l\ even}^{L} \frac{(n\delta\theta_k)^l}{l!}\left((-1)^{\lfloor \frac{l}{2} \rfloor}\mathbf{U}_{\mathrm{s}} + (-1)^{\lfloor \frac{(l+1)}{2} \rfloor}\mathbf{V}_{\mathrm{c}}\right) \\ &+ \mathbf{R}_{\mathrm{total}} \end{aligned} \tag{27}$$

The numerical stability of ID can be analyzed by Equation (27). Since the butterfly algorithm works well for equispaced Fourier series, Legendre transform using butterfly algorithm is numerical stability with the error of $\mathbf{R}_{\mathrm{total}}$. When L tends to infinity, the error is $R_{M,n}\left(\theta_k^{leg}\right)$.

Lemma 1. *For any $L \geq 1$ and $n \geq 0$ [27]*

$$R_{L,n,\delta\theta} := \max_{\theta\in[0,\pi]} \left| \cos(n(\theta+\delta\theta)) - \sum_{l=0}^{L-1} \cos^{(l)}(n\theta) \frac{(\delta\theta)^l}{l!} \right| \leq \frac{(n|\delta\theta|)^L}{L!} \tag{28}$$

Lemma 2. *For any $L \geq 1$ and $n \geq 0$, the error bound of Equation (24) is*

$$R \leq \frac{2C_n h_{M,n}}{L!} \left(\frac{(n|\delta\theta_k|)^L}{\left(2\sin\theta_k^{cheb}\right)^{M+\frac{1}{2}}} + \frac{1}{\left(2\sin\theta_k^{leg}\right)^{M+\frac{1}{2}}} \right) \tag{29}$$

Proof of Lemma 2.

$$\begin{aligned} \left|R_{M,L,n,\delta\theta_k}\right| &= \left| \sum_{l=0}^{L} (-1)^{\lfloor \frac{(l+1)}{2} \rfloor} C_n \sum_{m=0}^{M-1} h_{m,n}\left(u_m\left(\theta_k^*\right)\Psi_l\left(n\theta_k^*\right) + v_m\left(\theta_k^*\right)\Phi_l\left(n\theta_k^*\right)\right) \frac{(n\delta\theta_k)^l}{l!} \right| \\ &\leq \left| \frac{(n|\delta\theta_k|)^L}{L!} C_n \sum_{m=0}^{M-1} h_{m,n} \left| \left(u_m\left(\theta_k^*\right)\Psi_l\left(n\theta_k^*\right) + v_m\left(\theta_k^*\right)\Phi_l\left(n\theta_k^*\right)\right) \right| \right| \\ &\leq C_n h_{M,n} \frac{2}{\left(2\sin\theta_k^*\right)^{M+1/2}} \frac{(n|\delta\theta_k|)^L}{L!} \end{aligned} \tag{30}$$

Finally, one can get the total upper error bound

$$R = \left| R_{M,L,n,\delta\theta_k} + R_{M,n}\left(\theta_k^{leg}\right) \right| \leq \frac{2C_n h_{M,n}}{L!} \left(\frac{(n|\delta\theta_k|)^L}{\left(2\sin\theta_k^{cheb}\right)^{M+\frac{1}{2}}} + \frac{1}{\left(2\sin\theta_k^{leg}\right)^{M+\frac{1}{2}}} \right) \tag{31}$$

□

4. Legendre Transform Based on Block Partitioning and Butterfly Algorithm

In this section, we will propose a Legendre transform based on block partitioning and butterfly algorithm. The main idea is separate the matrix $\mathbf{P}_{\mathrm{N}}\left(x_N^{leg}\right)$ into block $\mathbf{P}_{\mathrm{N}}^{\mathrm{REC}}\left(x_N^{leg}\right)$ and K sub-matrices $\mathbf{P}_{\mathrm{N}}^{(k)}\left(x_N^{leg}\right)$ and then factorize K sub-matrices $\mathbf{P}_{\mathrm{N}}^{(k)}\left(x_N^{leg}\right)$ by butterfly algorithm. The butterfly algorithm can factorize a complementary low-rank matrix of size $N \times N$ into the product of $O(\log N)$ sparse matrices, each with O(N) nonzero entries. What's more, butterfly algorithm works well for $\mathbf{P}_{\mathrm{N}}^{(k)}\left(x_N^{leg}\right)$ as mentioned in Section 3. Hence, the total of nonzero entries after factorization can be approximate to $O(N\log^2 N/\log\log N)$ by controlling the number of nonzero entries in $\mathbf{P}_{\mathrm{N}}^{\mathrm{REC}}\left(x_N^{leg}\right)$. Finally, one can get a Legendre transform with $O(N\log^2 N/\log\log N)$ computational complexity.

It can be found that the matrix $\mathbf{P}_{\mathrm{N}}\left(x_N^{leg}\right)$ can be considered as a perturbation of matrix $\mathbf{P}_{\mathrm{N}}\left(x_N^{cheb}\right)$ from Equation (24). The block partitioning of $\mathbf{P}_{\mathrm{N}}\left(x_N^{leg}\right)$ can be performed by using the same method as $\mathbf{P}_{\mathrm{N}}\left(x_N^{cheb}\right)$ in the paper of Hale and Townsend [27]. Therefore, the matrix $\mathbf{P}_{\mathrm{N}}\left(x_N^{leg}\right)$ is partitioned as

$$\mathbf{P}_{\mathrm{N}}\left(x_N^{leg}\right) = \mathbf{P}_{\mathrm{N}}^{\mathrm{REC}}\left(x_N^{leg}\right) + \sum_{k=1}^{K} \mathbf{P}_{\mathrm{N}}^{(k)}\left(x_N^{leg}\right) \tag{32}$$

This partitioning separates the matrix $\mathbf{P}_{\mathrm{N}}\left(x_N^{leg}\right)$ into block $\mathbf{P}_{\mathrm{N}}^{\mathrm{REC}}\left(x_N^{leg}\right)$ and K sub-matrices $\mathbf{P}_{\mathrm{N}}^{(k)}\left(x_N^{leg}\right)$. Block $\mathbf{P}_{\mathrm{N}}^{\mathrm{REC}}\left(x_N^{leg}\right)$ contains the columns and rows of $\mathbf{P}_{\mathrm{N}}\left(x_N^{leg}\right)$, which cannot be computed by using Equation (24).

$$\mathbf{P}_{\mathrm{N}}^{\mathrm{REC}}\left(x_N^{leg}\right)_{ij} = \begin{cases} \mathbf{P}_{\mathrm{N}}\left(x_N^{leg}\right)_{ij}, & 1 \le \min(i, N-i+1) \le j_M, \\ \mathbf{P}_{\mathrm{N}}\left(x_N^{leg}\right)_{ij}, & 1 \le j \le n_M, \\ 0, & otherwsie \end{cases} \tag{33}$$

where

$$n_M = \left\lfloor \frac{1}{2}\left(\varepsilon \frac{\pi^{3/2}\Gamma(M+1)}{4\Gamma(M+1/2)}\right)^{\frac{-1}{M+\frac{1}{2}}} \right\rfloor \tag{34}$$

and

$$j_M = \left\lfloor \frac{N+1}{\pi} \sin^{-1}\left(\frac{n_M}{N}\right) \right\rfloor \tag{35}$$

$$\mathbf{P}_{\mathrm{N}}^{(k)}\left(x_N^{leg}\right)_{ij} = \begin{cases} \mathbf{P}_{\mathrm{N}}\left(x_N^{leg}\right)_{ij}, & i_k \le i \le N - j_k, \quad \alpha^k N \le j \le \alpha^{k-1} N \\ 0, & otherwsie \end{cases} \tag{36}$$

where $\alpha = O(1/\log N)$ and $i_k = \left\lfloor \frac{N+1}{\pi} \sin^{-1}\left(\frac{n_M}{\alpha^k N}\right) \right\rfloor$.

Legendre transform can be expressed as

$$\mathbf{P}_{\mathrm{N}}\left(\underline{x}_N^{leg}\right)\underline{c}_N^{leg} = \mathbf{P}_{\mathrm{N}}^{\mathrm{REC}}\left(\underline{x}_N^{leg}\right)\underline{c}_N^{leg} + \sum_{k=1}^{K} \mathbf{P}_{\mathrm{N}}^{(k)}\left(\underline{x}_N^{leg}\right)\underline{c}_N^{leg} \tag{37}$$

Nonzero entries of $\mathbf{P}_{\mathrm{N}}^{(k)}\left(x_N^{leg}\right)$ can be accurately expressed by the asymptotic formula, which means that the butterfly compression to $\mathbf{P}_{\mathrm{N}}^{(k)}\left(x_N^{leg}\right)$ is stable and accurate. Instead of the FFT method, the butterfly algorithm is employed to compute the matrix-vector product $\mathbf{P}_{\mathrm{N}}^{(k)}\left(x_N^{leg}\right)\underline{c}_N^{leg}$. This is because the butterfly algorithm works well for $\mathbf{P}_{\mathrm{N}}^{(k)}\left(x_N^{leg}\right)$ as mentioned in Section 3. So $\sum_{k=1}^{K} \mathbf{P}_{\mathrm{N}}^{(k)}\left(\underline{x}_N^{leg}\right)\underline{c}_N^{leg}$ can be computed in $O(KN\log N)$ operations. By restricting $\mathbf{P}_{\mathrm{N}}^{\mathrm{REC}}\left(\underline{x}_N^{leg}\right)$ has fewer than $O(KN\log N)$ nonzero entries, the matrix-vector product $\mathbf{P}_{\mathrm{N}}^{\mathrm{REC}}\left(\underline{x}_N^{leg}\right)\underline{c}_N^{leg}$ can be computed in $O(KN\log N)$ operations. Finally,

the optimal computational cost is achieved. Let $n_m = \min(n_M, N-1)$, the parameters α and K are defined as

$$\alpha = \begin{cases} \min(1/\log(N/n_m), 1/2), & \text{for small } N \\ 1/\log(N/n_m), & \text{for large } N \end{cases}$$

and $K = O(\log N/\log\log N)$, respectively. In the practical application, only parameters N, n_m, α and K are used to obtain information such as starting row/column index and offset for all blocks.

Figure 1 shows the partitioning of the Legendre-Vandermonde matrix for $N = 1024$. The Legendre-Vandermonde matrix is divided into boundary (denoted by symbol **B**) and internal (denoted by symbol **P**) parts. The boundary parts include the elements which cannot be accurately expressed by the asymptotic formula. There are 2(K+1) sub-matrices of **B** and 2K sub-matrices of **P**. According to the symmetric or anti-symmetric property of Legendre polynomials, only K+1 sub-matrices of **B** and K sub-matrices of **P** on the top are used. Algorithm 1 presents a summary of Legendre transform algorithm using block butterfly algorithm. Direct computation part and butterfly multiplication part is cost $O(KN\log N)$ operations, respectively.

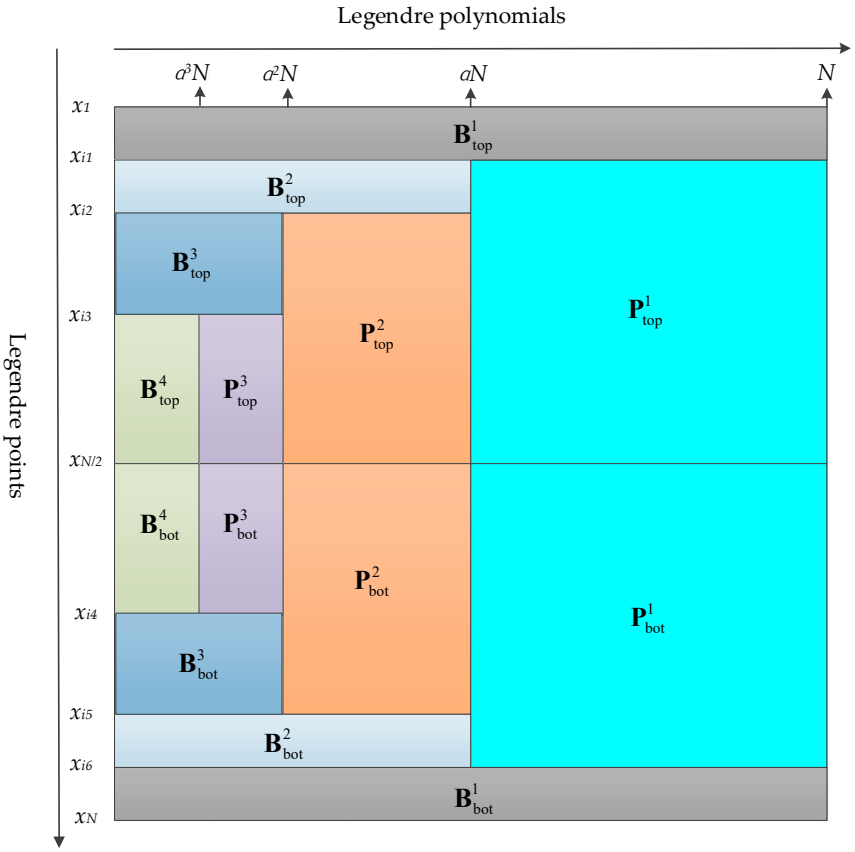

Figure 1. Partitioning of the Legendre-Vandermonde matrix for $N = 1024$ (in which matrix **B** is the boundary parts can't be accurately expressed by the asymptotic formula while matrix **P** is the internal parts can. There are 2(K + 1) sub-matrices of **B** and 2K sub-matrices of **P**. According to the symmetric or anti-symmetric property of Legendre polynomials, we only need to consider K + 1 sub-matrices of **B** and K sub-matrices of **P** on top).

Algorithm 1: Block Butterfly Algorithm for Legendre Transform.

Input N and $\underline{c}_N^{leg}$ to compute $\underline{y}_N^{leg} = \mathbf{P}_N\left(\underline{x}_N^{leg}\right)\underline{c}_N^{leg}$

Pre-computation Part

Block Partitioning: $\mathbf{B}_{top}^1, \mathbf{B}_{top}^2, \cdots, \mathbf{B}_{top}^{k+1}$ and $\mathbf{P}_{top}^1, \mathbf{P}_{top}^2, \cdots, \mathbf{P}_{top}^k$

extract symmetric part $\mathbf{B}_{tops}^1, \mathbf{B}_{tops}^2, \cdots, \mathbf{B}_{tops}^{k+1}$, $\mathbf{P}_{tops}^1, \mathbf{P}_{tops}^2, \cdots, \mathbf{P}_{tops}^k$ and anti-symmetric part $\mathbf{B}_{topa}^1, \mathbf{B}_{topa}^2, \cdots, \mathbf{B}_{topa}^{k+1}$ $\mathbf{P}_{topa}^1, \mathbf{P}_{topa}^2, \cdots, \mathbf{P}_{topa}^k$

for i=1,2,…,k

call butterfly_compression($\mathbf{P}_{tops}^i$) ! Symmetric Part

call butterfly_compression($\mathbf{P}_{topa}^i$) ! Anti-Symmetric Part

end for

Direct Computation Part:

for i=1,2,…,k+1

call dgemv($\mathbf{B}_{tops}^i$) ! Symmetric Part

call dgemv($\mathbf{B}_{topa}^i$) ! Anti-Symmetric Part

end for

Butterfly Multiplication Part:

for i=1,2,…,k

call butterfly_multiply() ! Symmetric Part

call butterfly_multiply() ! Anti-Symmetric Part

end for

Combine the results of symmetric and anti-symmetric part to get $\underline{y}_N^{leg}$

Parameters CMAX and EPS need for butterfly matrix compression are still needed in block butterfly algorithm. CMAX is the number of columns in each sub-matrix on level 0, EPS is desired precision in interpolative decomposition [3]. A dimensional thresh value DIMTHESH [3] is also needed in Legendre transform calls to activate FLT when wavenumber (m) less and equal to NSMAX-2DIMTHESH+3 (NSMAX is truncation order). Block butterfly algorithm is equivalent to Tygert's algorithm (2010) when no block partition is used, so two dimensional thresh values could be introduced to include Tygert's algorithm (2010) and LT using DGEMM for further reducing the computational complexity. To facilitate comparison with Tygert's algorithm, only one dimensional thresh value is used and set to 200 in the rest of the paper.

5. Results

In this section, all tests are performed on the MilkyWay-2 super computer (see Liao et al. [31] for more details), which installed in NUDT. Each compute node possesses 64GB of memory. The CPU model name is Intel(R)Xeon(R) CPU E5-2692V2 @2.2GHz. A private 32KB L1 instruction cache, a 32KB L1 data cache, a 256KB L2 cache, and a 30720KB L3 cache are used. ID software package developed by Martinsson et al. [32] for low rank approximation of matrices is employed to perform interpolative decompositions for all tests. ID package can be downloaded from Mark Tygert's homepage [33]. Hereafter, LT using matrix-matrix multiplication, Tygert's algorithm (2010) and block butterfly algorithm are named as LT0, LT1 and LT2, respectively.

Figures 2–4 show the errors of LT with CMAX = 64 in log10 form for EPS = 1.0E-05, EPS = 1.0E-07 and EPS = 1.0E-10, respectively. Abbreviations "MAX ERR" and "RMS ERR" in Figures 2–4 denote maximum error and root-mean-square error, respectively. It can be found that both maximum error and root-mean-square error of LT2 are improved by about one order magnitude than LT1. The results

show that the proposed method is effective in improving the accuracy of Legendre transform using butterfly algorithm.

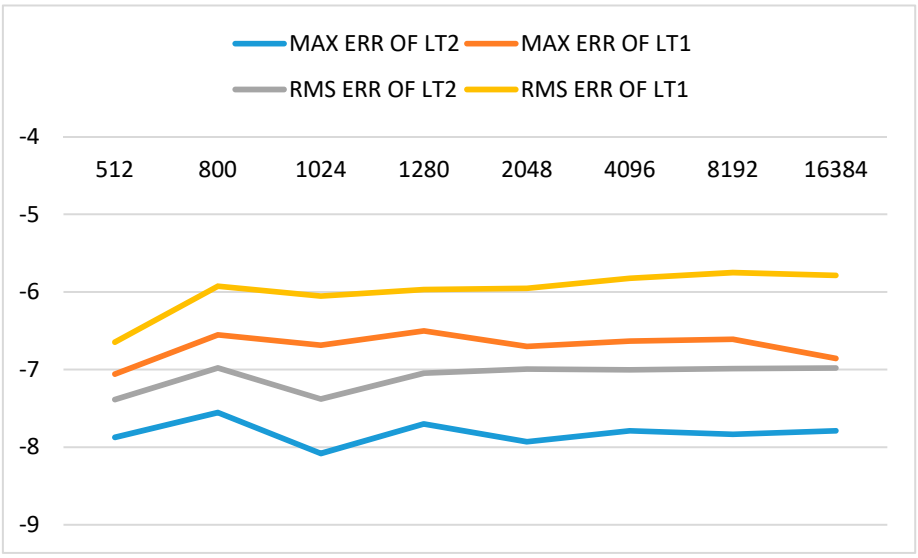

Figure 2. Errors of LT in log10 form with EPS = 1.0E-05 and CMAX = 64 (LT1 is the butterfly algorithm and LT2 is the proposed method. Abbreviations "MAX ERR" and "RMS ERR" denote the maximum error and root-mean-square error, respectively. EPS is desired precision in interpolative decomposition, CMAX is the number of columns in each sub-matrix on level 0).

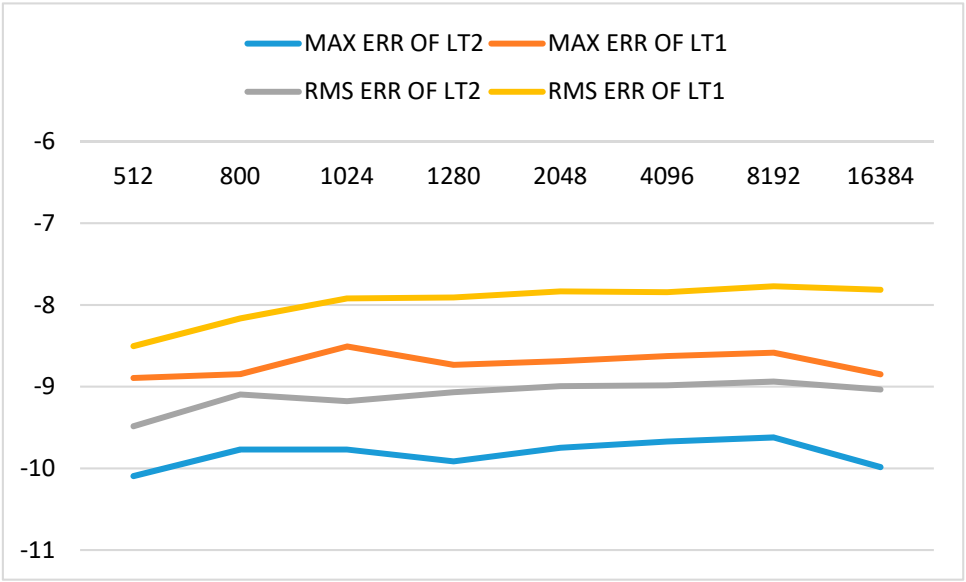

Figure 3. Errors of LT in log10 form with EPS = 1.0E-07 and CMAX = 64 (LT1 is the butterfly algorithm and LT2 is the proposed method. Abbreviations "MAX ERR" and "RMS ERR" denote the maximum error and root-mean-square error, respectively. EPS is desired precision in interpolative decomposition, CMAX is the number of columns in each sub-matrix on level 0).

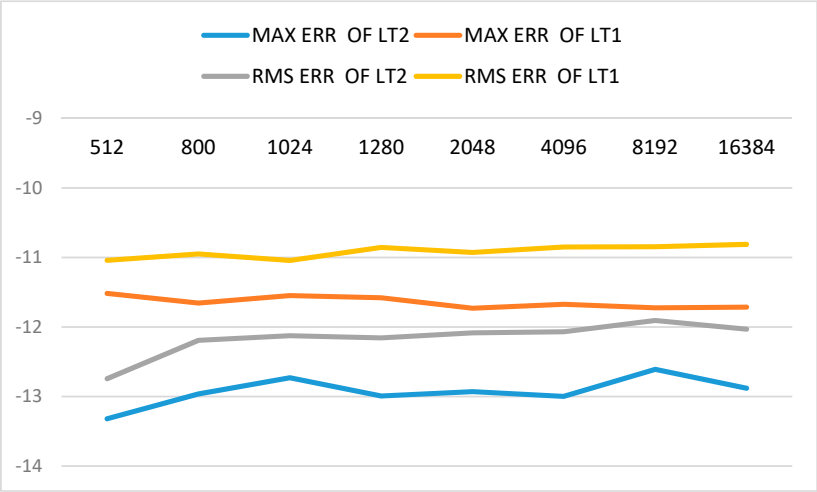

Figure 4. Errors of LT in log10 form with EPS = 1.0E-10 and CMAX = 64 (LT1 is the butterfly algorithm and LT2 is the proposed method. Abbreviations "MAX ERR" and "RMS ERR" denote the maximum error and root-mean-square error, respectively. EPS is desired precision in interpolative decomposition, CMAX is the number of columns in each sub-matrix on level 0).

Figure 5 shows the computational time for different Legendre transform algorithms. The speedup and loss speedup of LT2 with CMAX = 64 are demonstrated in Figures 6 and 7, respectively. Loss speedup which measures the relative performance penalty is defined as the speedup of LT1 minus the speedup of LT2 and divided by the speedup of LT0. From Figures 5 and 6, LT2 begins to be faster than LT0 when N = 2048 and achieves more than 26%, 22%, 17% reduction in elapsed time for EPS = 1.0E-5, EPS = 1.0E-7 and EPS = 1.0E-10. LT2 has achieved more than 17%, 63%, 75% and 86% reduction in elapsed time for a run of N2048, N4096, N8192 and N16384, respectively. In Figure 7, the loss speedup of LT2 relative to LT1 is less than 21%, 11%, 7% and 4% for N = 2048, 4096, 8192 and 16,384, respectively. Moreover, the loss speedup of LT2 relative to LT1 decreases rapidly with the increase of N. According to the results of Yin [3], the potential instability of interpolative decomposition only exists in the case of very high order. So, the presented method can alleviate the potential instability of interpolative decomposition at a very small computational burden.

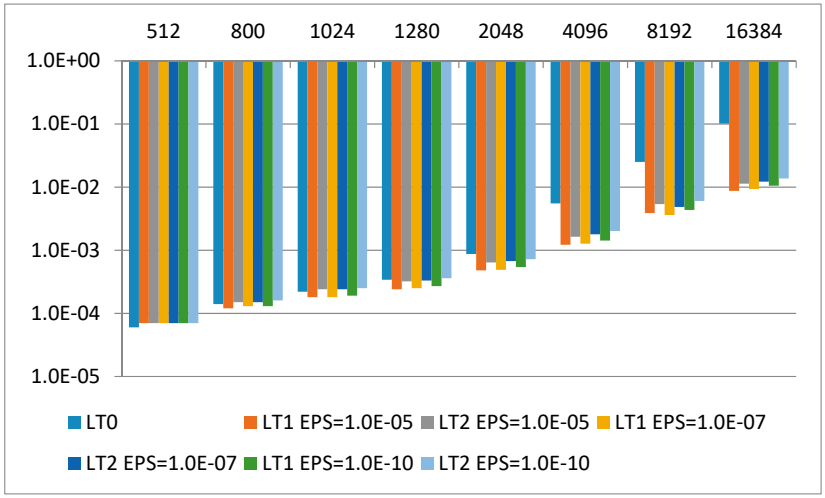

Figure 5. Computational time for different Legendre transform algorithms with CMAX = 64 (LT0 is the algorithm using DGEMM, LT1 is the butterfly algorithm and LT2 is the proposed method, EPS is desired precision in interpolative decomposition, CMAX is the number of columns in each sub-matrix on level 0, Unit: Second).

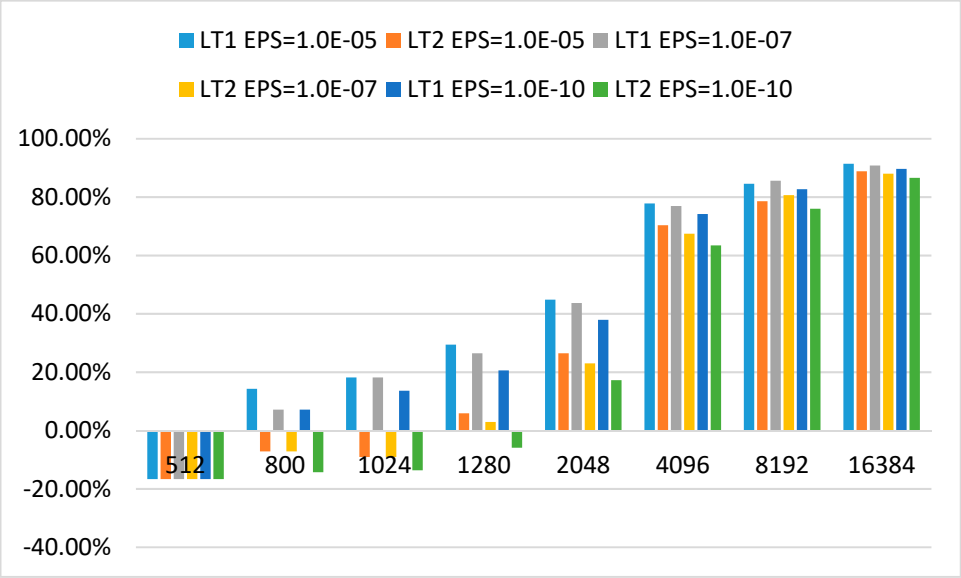

Figure 6. Speedup of LT1 and LT2 with CMAX = 64 compare to LT0 (LT0 is the algorithm using DGEMM, LT1 is the butterfly algorithm and LT2 is the proposed method, EPS is desired precision in interpolative decomposition, CMAX is the number of columns in each sub-matrix on level 0).

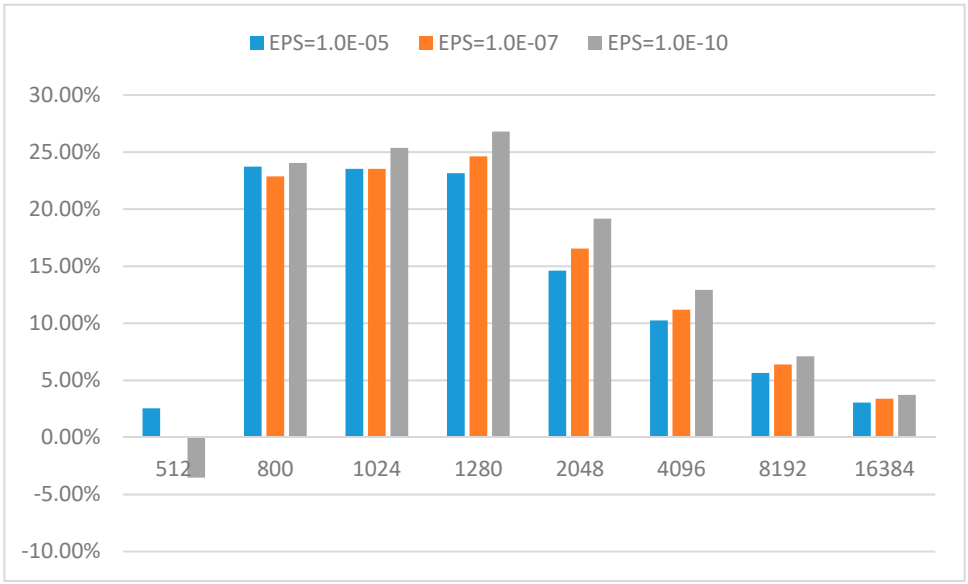

Figure 7. Loss speedup of LT2 with CMAX = 64 compare to LT1 (LT1 is the butterfly algorithm and LT2 is the proposed method, EPS is desired precision in interpolative decomposition, CMAX is the number of columns in each sub-matrix on level 0).

Figures 8 and 9 show the computational time of LT scaled by $N\log^3 N$ and $N\log^4 N$, respectively. It can be found that the computational complexity of LT2 appears to a little bigger than LT1. The boundary blocks which can't be accurately expressed by the asymptotic formula and the internal blocks with dimension less that dimensional thresh value result in the increase of the computational complexity. Although the results of LT2 are bigger than those of LT1, LT2 has a similar trend as LT1 in Figures 8 and 9. This means that LT2 has the same computational complexity $O(N\log^3 N)$ as LT1.

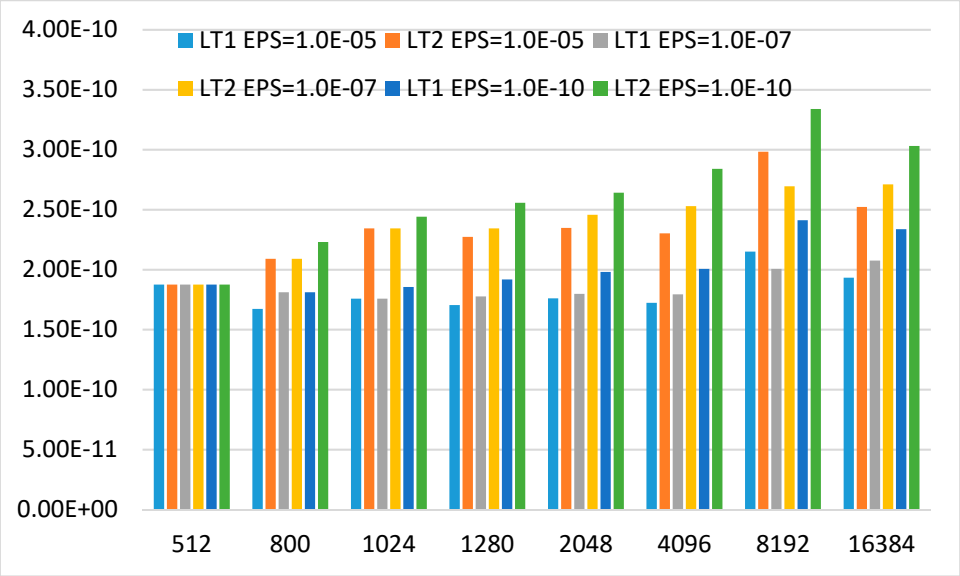

Figure 8. Computational time scaled by $N\log^3 N$ with CMAX = 64(LT1 is the butterfly algorithm and LT2 is the proposed method. EPS is desired precision in interpolative decomposition, CMAX is the number of columns in each sub-matrix on level 0).

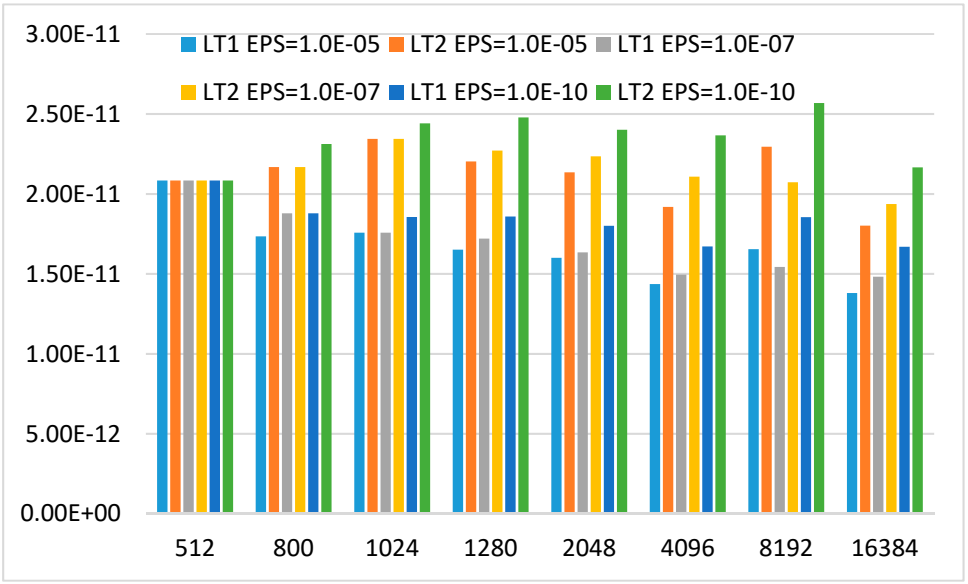

Figure 9. Computational time scaled by $N\log^4 N$ with CMAX = 64 (LT1 is the butterfly algorithm and LT2 is the proposed method. EPS is desired precision in interpolative decomposition, CMAX is the number of columns in each sub-matrix on level 0).

Legendre-Vandermond matrix is divided into boundary blocks and internal blocks. Boundary blocks which can't be accurately expressed by the asymptotic formula cause instability of interpolative decompositions and are not suitable for interpolation decomposition. The matrix-vector multiplication based on butterfly algorithm is faster than BLAS function DGEMV only when the dimension of matrix is greater than or equal to 512. Internal blocks with lower matrix dimension adopt direct matrix-vector multiplication instead of butterfly algorithm. The number of nonzero elements of boundary blocks, internal blocks which do not participate in interpolation decomposition cause the increase of the computational cost compare to Tygert's algorithm. Therefore, through reasonable

partitioning, the theoretical computational complexity of the proposed method can reach the optimal computational complexity $O(N\log^2 N/\log\log N)$.

6. Conclusions

In this paper, a high accurate and stable Legendre transform algorithm is proposed. A block partitioning based on asymptotic formula is employed to mitigate the potential instability of Legendre transform using butterfly algorithm. The Legendre-Vandermonde matrix is divided into one block $\mathbf{P}_{\mathrm{N}}^{\mathrm{REC}}\left(\underline{x}_N^{leg}\right)$ and K sub-matrices $\mathbf{P}_{\mathrm{N}}^{(k)}\left(x_N^{leg}\right)$. Instead of FFT method, butterfly algorithm is employed to compute $\mathbf{P}_{\mathrm{N}}^{(k)}\left(x_N^{leg}\right)\underline{c}_N^{leg}$. Numerical results demonstrate that the proposed method improves stability by about one order magnitude than Tygert's algorithm (2010), while only sacrifices less than 7% speedup for very high order ($N \geq 4096$) Legendre transform.

Although the computational time of proposed method is a little bigger than Tygert's algorithm, it has the same computational complexity $O(N\log^3 N)$ as Tygert's algorithm. Moreover, the proposed method is equivalent to Tygert's algorithm when no block partition is used. In the application of NWP, an additional dimensional thresh value could be introduced to include Tygert's algorithm (2010) for further reducing the computational complexity.

In the future, we will study the more optimal block partition method to improve the computational performance, while still keeping stability and making a detailed analysis in regard to the spectral harmonic transform using the proposed method for very high resolution—especially its performance in the reduction of potential numerical instability for resolution T7999.

Author Contributions: Conceptualization, F.Y. and J.W.; Formal analysis, F.Y.; Funding acquisition, F.Y.; Methodology, F.Y. and J.S.; Supervision, J.S.; Validation, J.W. and J.Y.; Writing—original draft, F.Y.; Writing—review & editing, J.Y.

Funding: This research was funded 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).

Acknowledgments: The author acknowledges Yingzhou Li (Duke University) and Haizhao Yang (National University of Singapore) for providing ButterflyLab for reference. The author would also like to thank two anonymous reviewers for their insightful and constructive comments, which help to improve the quality of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Seljebotn, D.S. Wavemoth—Fast spherical harmonic transforms by butterfly matrix compression. *Astrophys. J.* **2012**, *199*, 1–12. [CrossRef]
2. Wedi, N.P.; Hamrud, M.; Mozdzynski, G. A Fast Spherical Harmonics Transform for Global NWP and Climate Models. *Mon. Weather Rev.* **2013**, *141*, 3450–3461. [CrossRef]
3. Yin, F.; Wu, G.; Wu, J.; Zhao, J.; Song, J. Performance evaluation of the fast spherical harmonic transform algorithm in the yin–he global spectral model. *Mon. Weather Rev.* **2018**, *146*, 3163–3182. [CrossRef]
4. Suda, R.; Takami, M. A fast spherical harmonics transform algorithm. *Math. Comput.* **2002**, *71*, 703–715. [CrossRef]
5. Kunis, S.; Potts, D. Fast spherical Fourier algorithms. *J. Comput. Appl. Math.* **2003**, *161*, 75–98. [CrossRef]
6. Suda, R. Stability analysis of the fast Legendre transform algorithm based on the fast multipole method. In *Proceedings of the Estonian Academy of Sciences Physics*; Tallinn Book Printers Ltd.: Tallinn, Estonia, 2004; Volume 53, pp. 107–115.
7. Suda, R. Fast Spherical Harmonic Transform Routine FLTSS Applied to the Shallow Water Test Set. *Mon. Weather Rev.* **2005**, *133*, 634–648. [CrossRef]
8. Rokhlin, V.; Tygert, M. Fast Algorithms for Spherical Harmonic Expansions. *SIAM J. Sci. Comput.* **2005**, *27*, 1903–1928. [CrossRef]
9. Tygert, M. Recurrence relations and fast algorithms. *Appl. Comput. Harmon. Anal.* **2006**, *28*, 121–128. [CrossRef]

10. Tygert, M. Fast algorithms for spherical harmonic expansions, II. *J. Comput. Phys.* **2008**, *227*, 4260–4279. [CrossRef]
11. Tygert, M. Short Note: Fast algorithms for spherical harmonic expansions, III. *J. Comput. Phys.* **2010**, *229*, 6181–6192. [CrossRef]
12. Michielssen, E.; Boag, A. A multilevel matrix decomposition algorithm for analyzing scattering from large structures. *IEEE Trans. Antenn. Propag.* **1996**, *44*, 1086–1093. [CrossRef]
13. O'Neil, M.; Woolfe, F.; Rokhlin, V. An algorithm for the rapid evaluation of special function transforms. *Appl. Comput. Harmon. Anal.* **2010**, *28*, 203–226. [CrossRef]
14. Li, Y.; Yang, H. Interpolative Butterfly Factorization. *SIAM J. Sci. Comput.* **2017**, *39*, A503–A531. [CrossRef]
15. Wu, J.P.; Zhao, J.; Song, J.Q.; Zhang, W.M. Preliminary design of dynamic framework for global non-hydrostatic spectral model. *Comput. Eng. Des.* **2011**, *32*, 3539–3543.
16. Yang, J.; Song, J.; Wu, J.; Ren, K.; Leng, H. A high-order vertical discretization method for a semi-implicit mass-based non-hydrostatic kernel. *Q. J. R. Meteorol. Soc.* **2015**, *141*, 2880–2885. [CrossRef]
17. Yang, J.; Song, J.; Wu, J.; Ying, F.; Peng, J.; Leng, H. A semi-implicit deep-atmosphere spectral dynamical kernel using a hydrostatic-pressure coordinate. *Q. J. R. Meteorol. Soc.* **2017**, *143*, 2703–2713. [CrossRef]
18. Cheng, H.; Gimbutas, Z.; Martinsson, P.G.; Rokhlin, V. On the compression of low rank matrices. *SIAM J. Sci. Comput.* **2005**, *26*, 1389–1404. [CrossRef]
19. Heitman, Z.; Bremer, J.; Rokhlin, V. On the existence of nonoscillatory phase functions for second order ordinary differential equations in the high-frequency regime. *J. Comput. Phys.* **2015**, *290*, 1–27. [CrossRef]
20. Bremer, J.; Rokhlin, V. Improved estimates for nonoscillatory phase functions. *Discrete Cont. Dyn.-Am.* **2016**, *36*, 4101–4131. [CrossRef]
21. Bremer, J.; Rokhlin, V. On the nonoscillatory phase function for Legendre's differential equation. *J. Comput. Phys.* **2017**, *350*, 326–342. [CrossRef]
22. Bremer, J.; Yang, H. Fast algorithms for Jacobi expansions via nonoscillatory phase functions. *IMA J. Numer. Anal.* **2018**, arXiv:1803.03889.
23. Glaser, A.; Liu, X.; Rokhlin, V. A fast algorithm for the calculation of the roots of special functions. *SIAM J. Sci. Comput.* **2019**, *29*, 1420–1438. [CrossRef]
24. James, B.; Qiyuan, P.; Haizhao, Y. Fast Algorithms for the Multi-dimensional Jacobi Polynomial Transform. *arXiv* **2019**, arXiv:1901.07275.
25. ButterflyLab. Available online: https://github.com/ButterflyLab/ButterflyLab (accessed on 14 August 2019).
26. Candès, E.; Demanet, L.; Ying, L. A Fast Butterfly Algorithm for the Computation of Fourier Integral Operators. *Multiscale Model. Simul.* **2009**, *7*, 1727–1750. [CrossRef]
27. Hale, N.; Townsend, A. A fast, simple, and stable Chebyshev-Legendre transform using an asymptotic formula. *SIAM J. Sci. Comput.* **2014**, *36*, 148–167. [CrossRef]
28. Hale, N.; Townsend, A. A fast FFT-based discrete Legendre transform. *IMA J. Numer. Anal.* **2015**, *36*, 1670–1684. [CrossRef]
29. Townsend, A.; Webby, M.; Olver, S. Fast polynomial transforms based on Toeplitz and Hankel matrices. *Math. Comput.* **2018**, *87*, 1913–1934. [CrossRef]
30. Stieltjes, T.J. Sur les polynômes de Legendre. *Ann. Fac. Sci. Toulouse* **1890**, *4*, G1–G17. [CrossRef]
31. Liao, X.; Xiao, L.; Yang, C.; Lu, Y. MilkyWay-2 supercomputer: System and application. *Front. Comput. Sci.* **2014**, *8*, 345–356. [CrossRef]
32. Martinsson, P.G.; Rokhlin, V.; Shkolnisky, Y.; Tygert, M. ID: a software package for low rank approximation of matrices via interpolative decompositions, version 0.2. Available online: http://cims.nyu.edu/~{}tygert/id_doc.pdf (accessed on 4 August 2017).
33. Mark Tygert's Homepage. Available online: http://tygert.com/software.html (accessed on 14 August 2019).