Advances in Computational Mathematics **21:** 59–105, 2004.
© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

Towards safe and effective high-order Legendre transforms with applications to FFTs for the 2-sphere

D.M. Healy Jr. [a,*], P.J. Kostelec [b] and D. Rockmore [b,**]

[a] *Department of Mathematics, University of Maryland, College Park, MD 20742, USA*
[b] *Department of Mathematics, Dartmouth College, Hanover, NH 03755, USA*

Received 10 December 2001; accepted 28 October 2002
Communicated by J. Ward

This paper proposes and investigates a method for reducing potential numerical instability in the fast spherical harmonic transform algorithms proposed previously in [5,11]. The key objective of this study is a numerically reliable fast algorithm for computing the discrete Legendre transform (DLT); that is, the projection of sampled data onto the associated Legendre functions within a specified range of degrees. A simple divide-and-conquer approach derives from a factorization of high-degree Legendre functions into Legendre functions of lower-degree, exploiting the fact that the complexity of projection onto Legendre functions decreases with decreasing degree. Combining the resulting fast DLT algorithms for each relevant order of associated Legendre function results in an $O(N \log^2 N)$ algorithm for computing the spherical harmonic expansion of a function sampled at N points on the sphere. While fast DLT algorithms of this form are exact in exact arithmetic, actual (finite precision) implementations of the earlier variants display instabilities which generally grow with the the order of the associated Legendre function in the transform. Here we return to the basic algorithm, present a slight modification of the general schema and examine the error mechanism for the higher-order cases. This study suggests a new approach to the high-order DLTs achieved by substitution of a simple alternative factorization of the associated Legendre functions in place of that used previously. This technique improves stability significantly for a wide range of useful problem sizes and may be used with any of the variants of the basic algorithm previously proposed. We present a description of the use of the new Legendre decomposition in the basic fast algorithm along with numerical experiments demonstrating the large advances in stability and efficiency of the new approach over the previous results.

Keywords: fast transform algorithms, spherical harmonics

AMS subject classification: 65T, 42C10, 44A15

1. Introduction

For two centuries Fourier analysis has provided a tool of fundamental importance in a wide variety of application areas, through its perspective of symmetry-adapted rep-

* Partially supported by ONR.
** Partially supported by Sante Fe Institute, AFOSR, and NSF.

resentations which simplify many interesting problems. The same basic mathematical properties of the Fourier harmonic basis functions which provide a diagonalized representation of translation and convolution operators also provide insights into efficient numerical computation of Fourier transforms from sample data. This happy combination has contributed to the enormous impact of Fourier methods on science and technology.

The key is that Fourier harmonics are natural with respect to symmetries of both the time and frequency domain: The harmonic e^{ibx} scales under a translation as a simple phase: $\mathrm{e}^{ib(x+u)} = \mathrm{e}^{ibu}\mathrm{e}^{ibx}$ for the same reason that harmonics of high frequencies can be decomposed as products of harmonics of lower frequencies: $\mathrm{e}^{\mathrm{i}(b+c)x} = \mathrm{e}^{\mathrm{i}bx}\mathrm{e}^{\mathrm{i}cx}$. These properties play a role in numerical harmonic analysis, in the now "classical" Fast Fourier Transform (FFT), first discovered by Gauss and later rediscovered and popularized by Cooley and Tukey (see [12] for a nice outline of much of the history). This family of algorithms utilizes divide-and-conquer strategies, based on the above-mentioned algebraic properties of the harmonics, in order to efficiently compute Fourier coefficients of a band-limited function on the circle, an Abelian group. Its effective implementation has contributed to a wealth of advances in many fields, most noticeably digital signal processing (see, e.g., [7,21]).

An interesting generalization of this line of inquiry is the development of efficient and reliable algorithms to compute expansions of functions defined on non-Abelian groups in terms of the irreducible representation matrix coefficients, the basis that respects the symmetries of the accompanying symmetry group (see, e.g., [1,5,11,14–16,23, and references therein]).

A natural first step is the development of efficient algorithms for the calculation of Fourier expansions for the (non-Abelian) rotation group $SO(3)$ and its familiar homogeneous space, the 2-sphere. This amounts to development of efficient algorithms for the calculation of spherical harmonic expansions from sample data, an expansion in terms of the basis which respects the rotational symmetries of the 2-sphere. This problem has been identified as an important computational issue in many areas of applied science for example, including astronomy, computer vision, medical imaging, biology, statistical analysis of directional data, chemistry, and the fields of numerical weather prediction and global circulation modeling (see references in [11]).

One approach to the efficient computation of spherical harmonic expansions is an algebraic one, effectively relying on the Clebsch–Gordon relations (see, e.g., [30]) to obtain recurrence relations expressing spherical harmonics as combinations of spherical harmonics of lower-degree. These play a role reminiscent of the decomposition of Abelian harmonics in the classical FFT mentioned above, permitting the construction of similar fast algorithms in the new setting of the sphere.

This is the method which we revisit here, one originally described in the papers [5,11]. We begin by dissecting the two-dimensional spherical Fourier transform into a collection of one-dimensional Discrete Legendre Transforms (DLTs). For a given "bandwidth" $B > 0$ (cf. section 2) these are the sums of the form

$$\hat{\mathbf{s}}(\ell, m) = \sum_{k=0}^{2B-1} P_\ell^m(\cos\theta_k)[\mathbf{s}]_k, \quad |m| \leqslant \ell = 0, 1, \ldots, B-1, \tag{1}$$

where P_ℓ^m is the associated Legendre function of degree ℓ and order m, $\theta_k = \pi(k+1/2)/(2B)$ and $\mathbf{s}$ is a data vector with kth component $[\mathbf{s}]_k$, obtained from the samples of the original function which we wish to transform.

The results in [5,11] provide basic tools for algorithms which improve the asymptotic complexity of the complete set of Legendre transforms from $\mathrm{O}(N^{3/2})$ (achieved using a basic separation of variables) to $\mathrm{O}(N \log^2 N)$ $(N = B^2)$. These fast algorithms use a divide-and-conquer approach, akin to that used in many of the usual (Abelian) FFT algorithms (see, e.g., [3,29]).

In the case of the DLT, the divide-and-conquer strategy derives from a factorization of the Legendre functions obtained from certain of their recurrence relations. This permits the problem of computing projections onto Legendre functions to be decomposed into smaller subproblems of a similar form. The subproblems are solved recursively, and their solutions are combined to solve the original problem. The advantage derives from the more efficient solution of the smaller subproblems, combined with an aggregate low splitting cost. Analysis and experimentation with numerical implementations demonstrated considerable variability in the numerical stability of these various transforms, which in general tails off as m increases. While the $m = 0$ (Legendre polynomial) transform admits a numerically benign fast implementation, significant difficulties arise for transforms corresponding to larger values of m. Simple variations of the basic algorithms allow a trade of computational speed for stability [11,22] resulting in stable FFTs for the 2-sphere. While faster than naive approaches for a large range of useful problem sizes, these approaches are constrained by the need to deal with the poor behavior of the Legendre decomposition, and leave significant room for improvement.

In this paper, we examine the error mechanism in the basic algorithms. This study suggests a way of employing the basic ideas of our original fast algorithm with new, simple, alternative factorizations of the associated Legendre functions. Over a wide range of problem sizes this improves stability enormously without sacrificing computational efficiency. We present a description of the requisite algorithmic modifications as well as numerical experiments demonstrating the improved stability.

Algebraic approaches of this sort are but one way to approach the problem of fast spherical harmonic expansions. Approximate and projection-based methods are also another avenue of research being actively pursued. In [13], associated Legendre projection algorithms based on the fast multipole method are developed. These methods are further refined in [31]. In [17], local trigonometric expansions are used to derive efficient associated Legendre transforms. Swarztrauber and Spotz [27] develop a projection algorithm that reduces significantly memory requirements and number of operations, i.e. requiring only half as many as compared with standard associated Legendre transforms. For a comparison and analysis of actual implementations of a variety of projection algorithms, including those mentioned here, the reader is strongly encouraged to read [25].

The organization of the remainder of the paper is as follows. In section 2 we briefly recall the notation and some necessary technical background material on Fourier analysis for the 2-sphere and and the basic ideas of the fast algorithm proposed in [5,11] for its computation. Section 3 describes the main features our fast Fourier transform for the 2-sphere. This algorithm has a natural formulation as a particular structured matrix factorization of matrix containing sampled Legendre functions. Section 4 presents our new ideas for dealing with numerical difficulties which can arise from a direct implementation of the algorithms in [5,11]. We present an alternative Legendre decomposition, indicate how we can incorporate it into our fast algorithms, and show experimental results which indicate that this new approach greatly reduces the numerical problems without cutting into efficiency for a range of useful problem sizes. We conclude in section 5 with a summary and brief discussion of future work.

2. Background on numerical Fourier analysis on the sphere

Here we summarize fundamental facts about Fourier analysis and synthesis on the sphere, as well as the the basic idea behind our efficient numerical approaches to it. Key steps are a discretization and subsequent fast transform for the discrete Legendre transform (DLT). The Legendre functions are trigonometric polynomials doubly indexed by increasing degree and order. The DLT takes spatial data to a transform domain indexed by the degree, with one transform for each order.

Each DLT computes a collection of discrete inner products; when computed directly, each inner product requires a number of operations equal to the number of sample points. A first order more efficient approach uses a cosine representation effected by the discrete cosine transform (DCT). In this setting the number of operations per inner product is equal to the degree of the target Legendre function. A further reduction comes from the use of a three-term recurrence which the Legendre functions satisfy. This permits higher-degree transforms of a given order to be written as linear combinations of lower-degree transforms which are more attractive computationally. Iteration of this idea provides the heart of the efficient algorithm.

Details of this approach have already appeared [5,11]. For this reason we present only those aspects necessary for the main purpose of this paper, which is to show that certain numerical instabilities in this recurrence-based approach can be ameliorated. We encourage the interested reader to consult the earlier papers for further details, and to obtain a working software package [20].

2.1. *Fourier analysis on the 2-sphere and numerical computation*

As usual, S^2 denotes the 2-sphere or unit sphere in $\mathbb{R}^3$. In the standard coordinates any $\omega \in S^2$ is described by an angle θ, $0 \leqslant \theta \leqslant \pi$ measured down from the z-axis and an angle ϕ, $0 \leqslant \phi < 2\pi$, measured counterclockwise off the x-axis, in the plane transverse to the z-axis. Let $L^2(S^2)$ denote the Hilbert space of square integrable functions on S^2. In coordinates, the usual inner product is given by

$$\langle f, h\rangle = \int_0^{\pi}\left[\int_0^{2\pi} f(\theta,\phi)\overline{h(\theta,\phi)}\,\mathrm{d}\phi\right]\sin\theta\,\mathrm{d}\theta.$$

The well-known (see, e.g., [30]), *spherical harmonics* provide an orthonormal basis for $L^2(S^2)$. For any nonnegative integer ℓ and integer m with $|m| \leqslant \ell$, the (ℓ, m)-*spherical harmonic* Y_ℓ^m is a harmonic homogeneous polynomial of degree ℓ. The harmonics of degree ℓ span a subspace of $L^2(S^2)$ of dimension $2\ell + 1$ which is invariant under the rotations of S^2. This symmetry property is the key to the broad utility of the spherical harmonic basis.

In the coordinates (θ, ϕ), Y_ℓ^m has a factorization,

$$Y_\ell^m(\theta,\phi) = k_{\ell,m}P_\ell^m(\cos\theta)\mathrm{e}^{\mathrm{i}m\phi} \tag{2}$$

where P_ℓ^m is the *associated Legendre function* of degree ℓ and order m and $k_{\ell,m}$ is a normalization constant. The associated Legendre functions of fixed order m satisfy a characteristic *three-term recurrence* with respect to the degree ℓ,

$$(\ell - m + 1)P_{\ell+1}^m(x) - (2\ell+1)xP_\ell^m(x) + (\ell+m)P_{\ell-1}^m(x) = 0 \tag{3}$$

which may be used to generate Legendre functions of higher-degree from those of lower-degree, starting from an initial condition at $\ell = m$:

$$P_m^m(\cos\theta) = \sin^m\theta, \qquad P_{m-1}^m(\cos\theta) = 0.$$

This is a fundamental characterization which will be critical for the algorithms developed in this paper.

The *Fourier transform of a function on the 2-sphere* amounts to its L^2-projection onto spherical harmonics. The expansion of any function $f \in L^2(S^2)$ in terms of spherical harmonics is written

$$f = \sum_{\ell\geqslant 0}\sum_{|m|\leqslant\ell}\hat{f}(\ell,m)Y_\ell^m$$

and $\hat{f}(\ell, m)$ denotes the (ℓ, m)-*Fourier coefficient*, equal to $\langle f, Y_\ell^m\rangle$.

The separation of variables according to (2) shows that the computation of the spherical harmonic transform can be reduced to a regular (i.e., Abelian) Fourier transform in the azimuth coordinate ϕ followed by a projection onto the associated Legendre functions

$$\hat{f}(\ell,m) = \langle f, Y_\ell^m\rangle = k_{\ell,m}\int_0^{\pi}\left[\int_0^{2\pi}\mathrm{e}^{-\mathrm{i}m\phi}f(\theta,\phi)\,\mathrm{d}\phi\right]P_\ell^m(\cos\theta)\sin\theta\,\mathrm{d}\theta. \tag{4}$$

In analogy with the case of functions on the circle, we say that $f \in L^2(S^2)$ is *band-limited* with *band-limit* or *bandwidth* $B \geqslant 0$ if $\hat{f}(\ell, m) = 0$ for all $\ell \geqslant B$. For band-limited functions we have a simple quadrature (sampling) result which reduces the integrals (4) to finite weighted sums of a sampled data vector obtained from the integrand.

Theorem 1 (Cf. [5, theorem 3]). Let $f \in L^2(S^2)$ have bandwidth B. Then for each $|m| \leqslant \ell < B$,

$$\hat{f}(\ell, m) = \frac{\sqrt{2\pi}}{2B} \sum_{j=0}^{2B-1} \sum_{k=0}^{2B-1} a_j^{(B)} f(\theta_j, \phi_k) \mathrm{e}^{-\mathrm{i}m\phi_k} P_\ell^m(\cos\theta_j) \tag{5}$$

where the sample points are chosen from the equiangular grid: $\theta_j = \pi(2j+1)/4B$, $\phi_k = 2\pi k/2B$; and the weights $a_j^{(B)}$ play a role analogous to the $\sin\theta$ factor in the integrals (cf. figure 1).

Remark. There are several such sampling schemes available. For instance, it is actually possible to give a sampling theorem which uses only B samples in the θ coordinate.

The *Fourier transform* of a function f of bandwidth B is the collection of its Fourier coefficients,

$$\{\hat{f}(\ell, m) \mid 0 \leqslant |m| \leqslant \ell < B\}.$$

Our objective is fast, numerically reliable computation of these coefficients from the samples of f.

2.2. *Complexity of the discrete Legendre transform*

The preceding shows that the Fourier transform of $f \in L^2(S^2)$ of bandwidth B may be computed by the sums (5) for all $0 \leqslant |m| \leqslant \ell < B$ which we call the *discrete Fourier transform*, or DFT of f. Notice that direct computation of each $\hat{f}(\ell, m)$ uses $4B^2$ operations so that computation of all Fourier coefficients in this way would require $\mathrm{O}(B^4)$ operations.

More efficient algorithms begin with a separation of variables approach. First summing over the k index gives the inner exponential sums which depend only the indices j and m. This may be done efficiently for all m between $-B$ and B via the FFT (see,

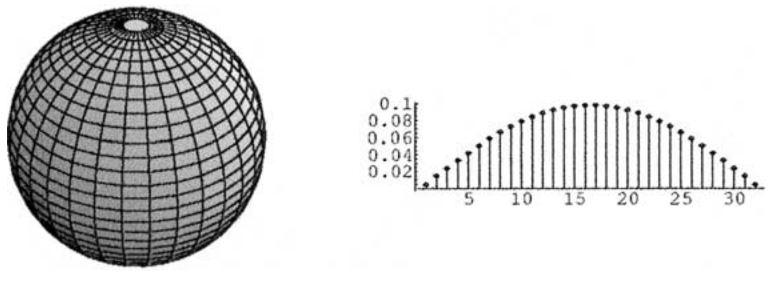

Figure 1. Quadrature nodes (at grid vertices) and sample weights from theorem 1.

e.g., [7]). The computation is completed by performing the requisite *discrete Legendre transforms* (DLTs), which for a data vector $\mathbf{f}$ and each given order m, $|m| \leqslant B$, is defined as the set of sums

$$\sum_{k=0}^{2B-1} [\mathbf{f}]_k P_\ell^m(\cos\theta_k) = \langle \mathbf{f}, \mathbf{P}_\ell^m \rangle; \quad \ell = |m|, |m+1|, \ldots, B-1, \tag{6}$$

for an arbitrary input vector $\mathbf{f}$ with kth component $[\mathbf{f}]_k$. Here we have introduced a discrete inner product notation and the convention that $\mathbf{P}_\ell^m$ denotes the vector comprised of the appropriate samples of the function $P_\ell^m(\cos\theta)$:

$$\mathbf{P}_\ell^m = \begin{pmatrix} P_\ell^m(\cos\theta_0) \\ \vdots \\ P_\ell^m(\cos\theta_{2B-1}) \end{pmatrix}.$$

We may also say that (6) *computes the projection of* $\mathbf{f}$ *onto* $\mathbf{P}_\ell^m$.

This approach thus reduces the problem of a fast spherical harmonic transform to the efficient calculation of these DLTs. An obvious approach to evaluating the sums in (6) is to compute them successively for the various degrees and orders. Computed in this way, each of these sums requires $2B$ multiplications and $2B-1$ additions. Since there are $N = B^2$ of these sums required to compute the full Fourier transform, this implies a total direct computational complexity of at most $4N^{3/2} = 4B^3$ operations.[1] Note that even without a faster algorithm for the DLTs, the separation of variables has already produced a savings. We refer to this as the *direct algorithm*.

To do better requires a faster approach for the DLTs. In [5,11] we explain an efficient algorithm for effecting this computation, reducing the $O(N^2)$ complexity to $O(N\log^2 N)$. In brief, the basic idea has two main components: one is the use of the three-term recurrence (3) as a means to decompose a high-degree Legendre function as a linear combination of lower-degree Legendre functions. This permits us to write the projection of data onto a high-degree Legendre function in terms of projections onto low-degree Legendre functions. To derive computational advantage from this we make use of a second concept: the cosine domain representation of the Legendre functions. A polynomial in cosine has nonzero coefficients only up to its degree. Thus, in the cosine domain (i.e., data and Legendre samples both transformed via a *discrete cosine transform* (DCT), itself effected by an efficient algorithm) a DLT of degree m only requires m operations, instead of the original N. The overhead accrues slowly enough that overall savings are obtained.

For the sake of completeness we present a brief outline here for the important special case of the Legendre polynomial transform, order $m = 0$; the higher-order cases are similar. The order $m = 0$ version of the Legendre recurrence (3),

$$(\ell+1)P_{\ell+1}(\cos\theta) - (2\ell+1)\cos\theta\, P_\ell(\cos\theta) + (\ell)P_{\ell-1}(\cos\theta) = 0 \tag{7}$$

[1] We use the standard arithmetic complexity model which defines a single operation as a complex multiplication followed by a complex addition.

starting with $P_{-1} = 0$ and $P_0 = 1$ implies that $P_\ell(\cos\theta)$ is a trigonometric polynomial of degree ℓ. We exploit this by using cosine transform representations, as follows.

Let $\mathcal{C}_N$ denote the N-dimensional orthogonal DCT matrix (see, e.g., [7, p. 386]) of normalized samples of cosine functions:

$$(\mathcal{C}_N)_{j,k} = b(j)\cos(j\theta_k), \quad 0 \leqslant j, k \leqslant N-1,$$

where, as usual, $\theta_k = \pi(2k+1)/(2N)$, and the normalization factors are $b(0) = \sqrt{1/N}$ and $b(j) = \sqrt{2/N}$ for $j = 1, \ldots, N$. Applying this to a data N-vector $\mathbf{s}$ yields the product $\mathcal{C}_N\mathbf{s}$, a vector whose entries provide the coefficients in a cosine series expansion (of degree at most $N-1$) whose uniformly spaced samples give $\mathbf{s}$. The coefficients $\{[\mathcal{C}_N\mathbf{s}]_n \mid n = 0, \ldots, N-1\}$ of the cosine transform can be obtained efficiently (in at most $\frac{3}{2}N\log N$ operations for N a power of 2) by a fast DCT algorithm, which amounts to a clever factorization of the matrix $\mathcal{C}_N$ (see [26, and the references contained therein]).

The orthogonality of $\mathcal{C}$ $(= \mathcal{C}_N)$ implies $\langle\mathcal{C}\mathbf{s}, \mathcal{C}\mathbf{Q}\rangle = \langle\mathbf{s}, \mathbf{Q}\rangle$ for any vectors $\mathbf{Q}$ and $\mathbf{s}$. The computational advantage provided by computing the inner product using the DCT comes from the fact that a trigonometric polynomial has only as many cosine coefficients as its degree. Thus, if Q is a trigonometric polynomial of degree $n < N$, then $[\mathcal{C}\mathbf{Q}]_n = 0$ for $n > deg(Q)$, implying that at most $deg(Q) + 1 < N$ operations are needed to compute $\langle\mathcal{C}\mathbf{s}, \mathcal{C}\mathbf{Q}\rangle$ assuming $\mathcal{C}\mathbf{s}$ and $\mathcal{C}\mathbf{Q}$ are given.

Define the *critically sampled lowpass operator* (of bandwidth p), denoted $\mathcal{L}_p^N$ (for $p < N$), by

$$\mathcal{L}_p^N = \mathcal{C}_p^{-1}\mathcal{T}_p^N\mathcal{C}_N$$

where $\mathcal{T}_p^N$ is the *truncation operator* that only keeps the first p coordinates of a given input vector.

Lemma 1 [11, lemma 2]. Let Q be a trigonometric polynomial of degree p,

$$Q(\cos\theta) = \sum_{m=0}^{p}\gamma_m\cos m\theta,$$

and let $\mathbf{s}$ be any sequence of length N with $N \geqslant p$. Then

$$\langle\mathbf{s}, \mathbf{Q}\rangle = \sum_{k=0}^{N-1}[\mathbf{s}]_k Q(\cos\theta_k) = \sum_{j=0}^{p-1}\bigl[\mathcal{L}_p^N\mathbf{s}\bigr]_j Q\left(\cos\frac{N\theta_j}{p}\right) = \bigl\langle\mathcal{L}_p^N\mathbf{s}, \mathcal{L}_p^N\mathbf{Q}\bigr\rangle.$$

Note $\mathcal{L}_p^N\mathbf{Q}$ is simply Q sampled on the coarser p sample grid. If p is a power of 2 and these values of Q are prestored, the computation of the inner product thus requires p operations after the overhead of computing $\mathbf{s} \mapsto \mathcal{L}_p^N\mathbf{s}$ in at most $\frac{3}{2}N\log N + \frac{3}{2}p\log p$ operations.

In particular, lemma 1 applies to the various Legendre functions. To illustrate, the Legendre polynomial $Q = P_\ell$ is a trigonometric polynomial of degree ℓ, as is easily verified using the recurrence relation (7) with the initial conditions $P_0(\cos\theta) = 1$, $P_1(\cos\theta) = \cos\theta$. Consequently, for $\ell < n$, $[\mathcal{C}\mathbf{P}_\ell]_n = 0$ and the inner product sum $\langle \mathcal{C}\mathbf{P}_\ell, \mathcal{C}\mathbf{s}\rangle = \langle \mathbf{P}_\ell, \mathbf{s}\rangle$ can be computed as a sum of only $\ell + 1$ terms (instead of N).

If the cosine coefficients of the sampled Legendre polynomials ($[\mathcal{C}\mathbf{P}_j]_k$) are prestored, this cosine domain approach is a simple alternative to the direct computation of the Legendre transform. It has the same asymptotic complexity, but is faster for even moderate sized transforms, assuming the use of a fast DCT routine. This idea is originally due to Dilts [4]. A general formulation of this observation now follows:

Lemma 2 [11, lemma 3]. Let N be a power of 2 and $\mathbf{s}$ a vector of length N. Suppose $p_\ell(\cos\theta)$ ($\ell = 0, \ldots, N-1$) satisfies a recurrence $a_\ell p_{\ell+1}(\cos\theta) - b_\ell \cos(\theta) p_\ell(\cos\theta) + c_\ell p_{\ell-1}(\cos\theta)$ with initial conditions $p_{-1} = 0$ and $p_0 = 1$. Then assuming the prestorage of the DCT of the p_ℓ, for an arbitrary input $\mathbf{s}$, the collection of inner products $\langle \mathbf{p}_\ell, \mathbf{s}\rangle$ can be computed in at most $\frac{3}{2}N \log N + N(N+1)/2$ operations, versus N^2 required by direct computation.

Lemma 2 applied to the Legendre functions provides a *semi-naive* method for computing the DLT faster than the direct approach for even moderate problem sizes, although still requiring quadratic complexity.

In order to help turn this into a divide-and-conquer algorithm we need a way to make the “high-degree” Legendre transform coefficients (of degree at least $N/2$) as efficient as the low-degree (of degree less than $N/2$), for after application of the DCT the former all require at least $N/2$ operations, while the latter at most this many. This imbalance is illustrated schematically in figure 2.

The main point of our work towards a fast algorithm is to address this imbalance, reducing the set of high-degree projections to an equivalent set of low-degree projections. Lemma 2 suggests that it makes sense to use the recurrence to compute high-degree transforms in terms of lower-degree transforms since: (1) the lower-degree transforms are more efficient and (2) the overhead seems manageable.

2.3. Legendre function decompositions from the Legendre recurrence

The (one-step) recurrence (3) implies that $P_\ell^m(\cos\theta)$ can be written as a linear combination of $P_{\ell-1}^m$ and $P_{\ell-2}^m$. When iterated, this provides an expression of P_ℓ^m in terms of a linear combination of other pairs of lower-degree associated Legendre functions. Taken together with lemma 1, this decomposition provides a divide and conquer approach to the DLT. A closer analysis demonstrates that, for certain orders m, this basic decomposition also yields a source of numerical instability. In order to see this and address the problem, we revisit the basic decomposition of Legendre functions and recast it with an eye to the analysis of section 4.

For each given order m, and $r \geqslant m$, the three-term recurrence satisfied by the associated Legendre functions may be cast in matrix form:

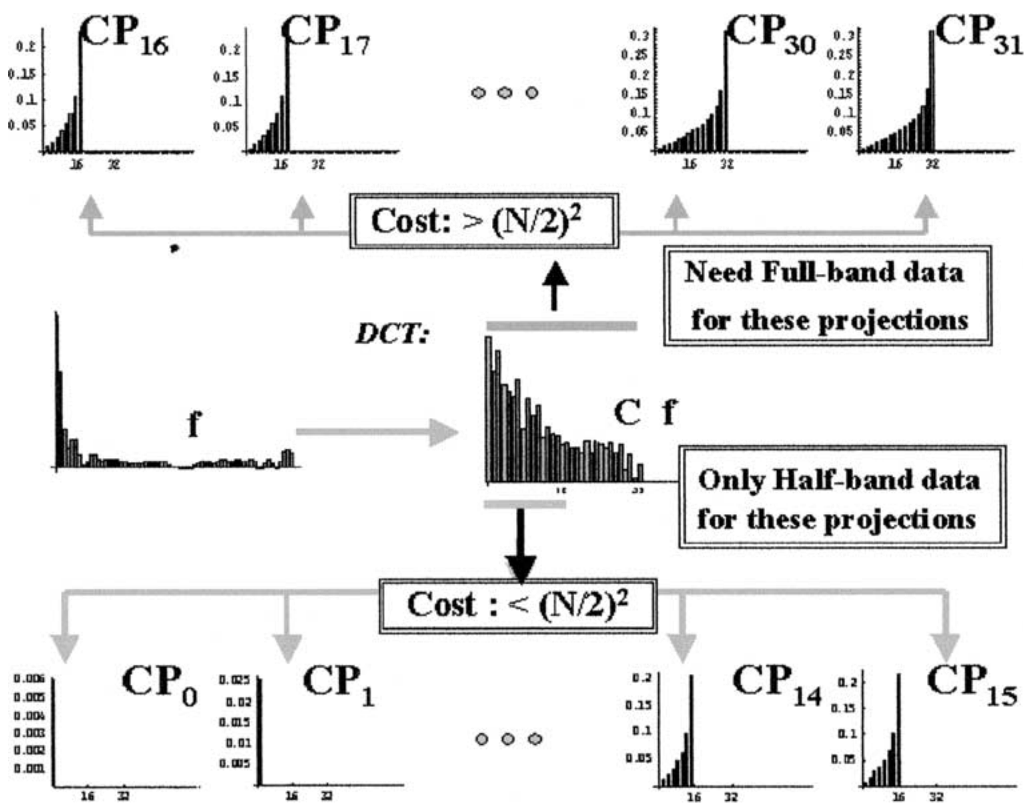
CP16
CP17
CP30
CP31
Cost: > (N/2)²
Need Full-band data for these projections
DCT:
f
C f
Only Half-band data for these projections
Cost : < (N/2)²
CP0
CP1
CP14
CP15

Figure 2. Semi-naive calculation transforms input to the cosine transform domain, where it is projected onto the cosine transformed Legendre vectors. The low-degree inner products each require less than $N/2$ operations, but the high-degree inner products each require greater than $N/2$ operations.

$$\begin{aligned}\bigl(P^m_{r+1}(\cos\theta),\, P^m_r(\cos\theta)\bigr) &= \bigl(P^m_r(\cos\theta),\, P^m_{r-1}(\cos\theta)\bigr)\begin{pmatrix}\dfrac{2r+1}{r-m+1}\cos\theta & 1\\ -\dfrac{r+m}{r-m+1} & 0\end{pmatrix}\\ &= \bigl(P^m_r(\cos\theta),\, P^m_{r-1}(\cos\theta)\bigr)\,{}^r\Pi^m_1(\cos\theta) \qquad (8)\end{aligned}$$

with the initial condition at $r = m$: $(P^m_m(\cos\theta), P^m_{m-1}(\cos\theta)) = (\sin^m\theta, 0)$, and where we have introduced the concise notation ${}^r\Pi^m_1(\cos\theta)$ for the matrix (actually a matrix-valued trigonometric polynomial) which advances the degree of the Legendre functions one step, starting from degree r.

Repeating this process n times produces the matrix effecting an n-step advance starting at r:

$$\bigl(P^m_{r+n}(\cos\theta),\, P^m_{r+n-1}(\cos\theta)\bigr) = \bigl(P^m_r(\cos\theta),\, P^m_{r-1}(\cos\theta)\bigr)\,{}^r\Pi^m_n(\cos\theta) \qquad (9)$$

with recursive definition

$$ {}^r\Pi^m_{n+1} = {}^r\Pi^m_n \; {}^{r+n}\Pi^m_1. \tag{10} $$

From this it follows that for any choice of ℓ, r, the function ${}^r\Pi^m_\ell$ generalizes in a useful way the ℓth degree associated Legendre functions. In particular, (10) expresses the fact that ${}^r\Pi^m_n$ is a matrix trigonometric polynomial satisfying the same recurrence as the order m associated Legendre functions themselves. Introducing notation for the matrix entries:

$$ {}^r\Pi^m_n(\cos\theta) = \begin{pmatrix} {}^r\overline{P}^m_n(\cos\theta) & {}^r\overline{P}^m_{n-1}(\cos\theta) \\ {}^r\underline{P}^m_n(\cos\theta) & {}^r\underline{P}^m_{n-1}(\cos\theta) \end{pmatrix} $$

an explicit rewriting of the recurrence (10)

$$ \begin{pmatrix} {}^r\overline{P}^m_{n+1}(\cos\theta) & {}^r\overline{P}^m_n(\cos\theta) \\ {}^r\underline{P}^m_{n+1}(\cos\theta) & {}^r\underline{P}^m_n(\cos\theta) \end{pmatrix} = \begin{pmatrix} {}^r\overline{P}^m_n(\cos\theta) & {}^r\overline{P}^m_{n-1}(\cos\theta) \\ {}^r\underline{P}^m_n(\cos\theta) & {}^r\underline{P}^m_{n-1}(\cos\theta) \end{pmatrix} \times \begin{pmatrix} \dfrac{2(r+n)+1}{(n+r)-m+1}\cos\theta & 1 \\ -\dfrac{(n+r)+m}{(n+r)-m+1} & 0 \end{pmatrix} $$

shows that ${}^r\overline{P}^m_{n+1}(\cos\theta)$ and ${}^r\underline{P}^m_{n+1}(\cos\theta)$ are (at most) nth order trigonometric polynomials, both of which are solutions of an r-lagged form of the Legendre function recurrence

$$ \begin{aligned} &\big([n+r]-m+1\big)p_{n+1}(\cos\theta) - \big(2[n+r]+1\big)\cos\theta p_n(\cos\theta) \\ &\quad + \big([n+r]+m\big)p_{n-1}(\cos\theta) = 0 \end{aligned} \tag{11} $$

and are determined by the initial conditions ${}^r\overline{P}^m_0 = 1, {}^r\overline{P}^m_{-1} = 0$, and ${}^r\underline{P}^m_0 = 0$, ${}^r\underline{P}^m_{-1} = 1$, respectively. These initial conditions come from the fact that ${}^r\Pi^m_0(\cos\theta)$ is the identity matrix, as seen by setting $n = 0$ in the recurrence (9).

For this reason we call these functions *lagged Legendre functions*. In the special case of $r = m$ (which is the minimal possible lag for order m Legendre functions), we have ${}^m\overline{P}^m_\ell(\cos\theta)\sin^m\theta = P^m_{m+\ell}(\cos\theta)$ so that the lagged Legendre functions coincide with the associated Legendre functions up to a weighting factor, for $\ell \geqslant 0$. In this case, (10) captures the usual Legendre recurrence for P^m_k, $k \geqslant m$:

$$ \begin{aligned} \begin{pmatrix} P^m_{k+1}(\cos\theta) & P^m_k(\cos\theta) \\ * & * \end{pmatrix} &= {}^m\Pi^m_{k-m+1}(\cos\theta)\sin^m\theta \\ &= \big[{}^m\Pi^m_{k-m}(\cos\theta)\; {}^k\Pi^m_1(\cos\theta)\big]\sin^m\theta \\ &= \begin{pmatrix} P^m_k(\cos\theta) & P^m_{k-1}(\cos\theta) \\ * & * \end{pmatrix} {}^k\Pi^m_1(\cos\theta), \end{aligned} $$

subsuming (8). This demonstrates that the associated Legendre functions and their recurrence are special cases of the lagged Legendre functions and recurrence.

It is easy to generalize (10) and obtain the general decomposition of high-degree lagged Legendre functions into products of those of lower-degree

$$ {}^{r}\Pi^{m}_{n+\ell}(\cos\theta) = {}^{r}\Pi^{m}_{n}(\cos\theta)\,{}^{r+n}\Pi^{m}_{\ell}(\cos\theta). \tag{12}$$

In particular, for the minimal possible lag $r = m$ the matrix entries of this decomposition give a decomposition of associated Legendre functions in terms of lower-degree Legendre functions:

$$P^{m}_{m+n+k} = P^{m}_{m+n}\,{}^{m+n}\overline{P}^{m}_{k} + P^{m}_{m+n-1}\,{}^{m+n}\underline{P}^{m}_{k}. \tag{13}$$

This plays a fundamental role in obtaining a divide-and-conquer fast DLT, analogous to the role of the familiar decomposition $\mathrm{e}^{\mathrm{i}(m+n)x} = \mathrm{e}^{\mathrm{i}mx}\mathrm{e}^{\mathrm{i}nx}$ for the usual Fourier transform on the circle.

2.4. Balanced split of the DLT using the Legendre decomposition

Our goal is efficient computation of the DLT (6), a collection of projections of data of bandwidth B onto the sampled order m Legendre functions $\langle \mathbf{f}, \mathbf{P}^{m}_{\ell}\rangle$ with degrees $\ell = |m|, |m+1|, \ldots, B-1$. We have seen previously that the high-degree projections account for most of the computational effort. The decomposition (13) can now be used to effect a decomposition or splitting in order to “demodulate” all of the high-degree Legendre functions in these projections down to low-degree (lagged) Legendre functions. This is compensated in the inner product computations by providing the corresponding “up modulation” to the input data vector. The resulting low-degree projections may be computed at a reduced cost, as we will now see. Again, for concreteness we focus on the Legendre polynomial case, so $m = 0$ in the various definitions of lagged Legendre functions.

Let $0 \leqslant \ell < N/2$. For input vector $\mathbf{f}$, application of (13) gives the decomposition

$$\begin{aligned}\langle \mathbf{f}, \mathbf{P}_{N/2+\ell}\rangle &= \big\langle \mathbf{f}, \big(\mathbf{P}_{N/2}\,{}^{N/2}\overline{\mathbf{P}}_{\ell} + \mathbf{P}_{N/2-1}\,{}^{N/2}\underline{\mathbf{P}}_{\ell}\big)\big\rangle \\ &= \big\langle \mathbf{f}\mathbf{P}_{N/2}, {}^{N/2}\overline{\mathbf{P}}_{\ell}\big\rangle + \big\langle \mathbf{f}\mathbf{P}_{N/2-1}, {}^{N/2}\underline{\mathbf{P}}_{\ell}\big\rangle. \end{aligned}\tag{14}$$

Here $\mathbf{f}\mathbf{P}_{N/2}$ and and $\mathbf{f}\mathbf{P}_{N/2-1}$ denote the modulation of the input vector by Legendre vectors, effected by multiplication using the componentwise or Hadamard product. We also continue our convention and use boldface to denote the vectors of sample values for the ${}^{r}\overline{P}_{\ell}$ and ${}^{r}\underline{P}_{\ell}$

$${}^{r}\overline{\mathbf{P}}_{\ell} = \begin{pmatrix} {}^{r}\overline{P}_{\ell}(\cos\theta_0) \\ \vdots \\ {}^{r}\overline{P}_{\ell}(\cos\theta_{N-1}), \end{pmatrix}, \qquad {}^{r}\underline{\mathbf{P}}_{\ell} = \begin{pmatrix} {}^{r}\underline{P}_{\ell}(\cos\theta_0) \\ \vdots \\ {}^{r}\underline{P}_{\ell}(\cos\theta_{N-1}) \end{pmatrix}, \quad \theta_k = \frac{\pi(2k+1)}{2N}. \tag{15}$$

Equation (14) shows that any higher-degree inner product in the DLT can instead be computed as an inner product of modulated data against (precomputed) sampled values of the trig polynomials ${}^{N/2}\underline{\mathbf{P}}_{\ell}$ and ${}^{N/2}\overline{\mathbf{P}}_{\ell}$. Each of these lagged Legendre functions

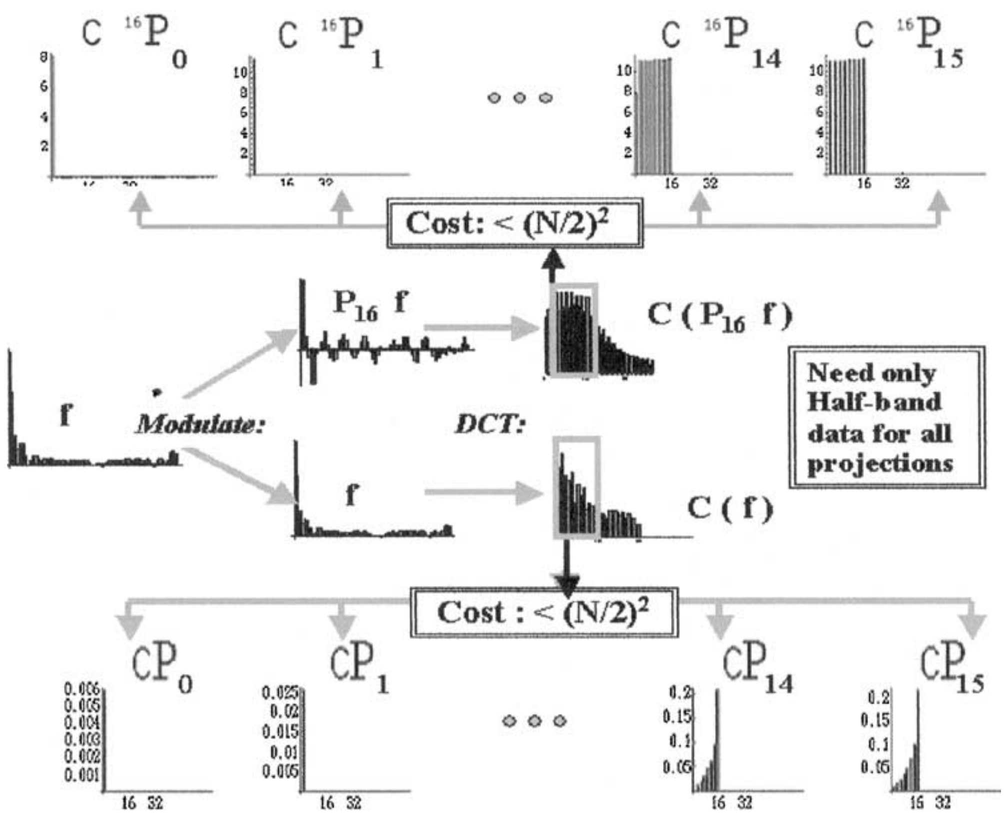

Figure 3. The balanced splitting approach replaces (cosine domain) projections of input data onto high-degree Legendre functions with projections of a modulated form of the data onto low-degree Legendre functions.

has degree at most ℓ, which is necessarily less than the degree $N/2$. Therefore, each of the higher-degree inner products in the original DLT can instead be computed with fewer than $N/2$ computations using stored DCT's of the low-degree lagged Legendre functions and of the modulated data. This contrasts favorably with the straight semi-naive algorithm, in which each of the higher-degree inner products requires more than $N/2$ computations. For N large enough, this advantage outweighs the added overhead of forming the modulates $\mathbf{f}\mathbf{P}_{N/2}$, $\mathbf{f}\mathbf{P}_{N/2-1}$ and then applying the DCT to each (with a total cost $2N + 3N\log N$).

This new approach balances the complexity of the high-degree and low-degree inner product computations of the DLT, as indicated schematically in figure 3. This contrasts to the imbalance in the standard semi-naive method as depicted in figure 2.

At this point we have seen that we can split the DLT into two half-sized problems and obtain a computational savings thereby. Because ${}^{N/2}\overline{\mathbf{P}}_\ell$ and ${}^{N/2}\underline{\mathbf{P}}_\ell$ also satisfy a Legendre recurrence, this procedure can be repeated. Following this through yields a fast divide-and-conquer scheme for performing a DLT. We put the whole thing together in the next section.

3. Fast Legendre transforms: theory and experiment

The decomposition of the Legendre functions (12), can be applied recursively to provide a full divide and conquer fast discrete Legendre transform algorithm. We begin this section by showing how this is done. Combining fast Legendre transforms for the various orders m enables fast algorithms for Fourier analysis, synthesis, and convolution on the two-sphere. We next summarize experimental timing and numerical stability results obtained from implementation of these algorithms, as reported in [11]. The results demonstrate a real efficiency advantage over previous algorithms, but nevertheless represent mixed success, as we find that the discrete Legendre transforms for the higher-order associated Legendre functions exhibit stability difficulties when implemented with the baseline fast algorithm. We briefly recall how these problems with higher-order Legendre transforms have in past been mitigated by modifications of the basic algorithm at some cost in speed. While the resulting spherical Fourier transform is reliable and and faster than previous implementations, the problems with the higher-order Legendre transforms motivate further study aimed at needed stability improvements. Significant progress in this direction will be reported in the section 4 of this paper.

3.1. A divide-and-conquer discrete Legendre transform

We begin by formalizing and generalizing the description of the "divide" portion of our algorithm: the process of splitting a discrete Legendre transform into two transforms, each involving half as many projections as the original transform. We then show how this process can be repeated on the resulting subtransforms, ultimately resulting in an efficient algorithm for the original DLT problem. We continue to illustrate this approach with the order $m = 0$ case.

To keep the notation under control it is helpful to work with the matrix versions of the sampled Legendre functions

$$ {}^{r}\mathbf{\Pi}_n = \begin{pmatrix} {}^{r}\overline{\mathbf{P}}_n & {}^{r}\overline{\mathbf{P}}_{n-1} \\ {}^{r}\underline{\mathbf{P}}_n & {}^{r}\underline{\mathbf{P}}_{n-1} \end{pmatrix}, $$

whose matrix entries are vectors of sampled Legendre functions (cf. equation (15)). This matrix valued function satisfies the fundamental decomposition property given by a sampled version of (12):

$$ {}^{r}\mathbf{\Pi}_{n+k} = {}^{r}\mathbf{\Pi}_n \, {}^{r+n}\mathbf{\Pi}_k. \tag{16} $$

Here we define the product of matrix-valued functions (matrices whose entries are sampled function vectors) as the usual multiplication of the 2×2 matrices, with the entries combined by pointwise or Hadamard product. For example, the (1,1) entry of the right-hand side of (16) is ${}^{r}\overline{\mathbf{P}}_n \, {}^{r+n}\overline{\mathbf{P}}_k + {}^{r}\overline{\mathbf{P}}_{n-1} \, {}^{r+n}\underline{\mathbf{P}}_k$, and a particular case is illustrated in figure 4.

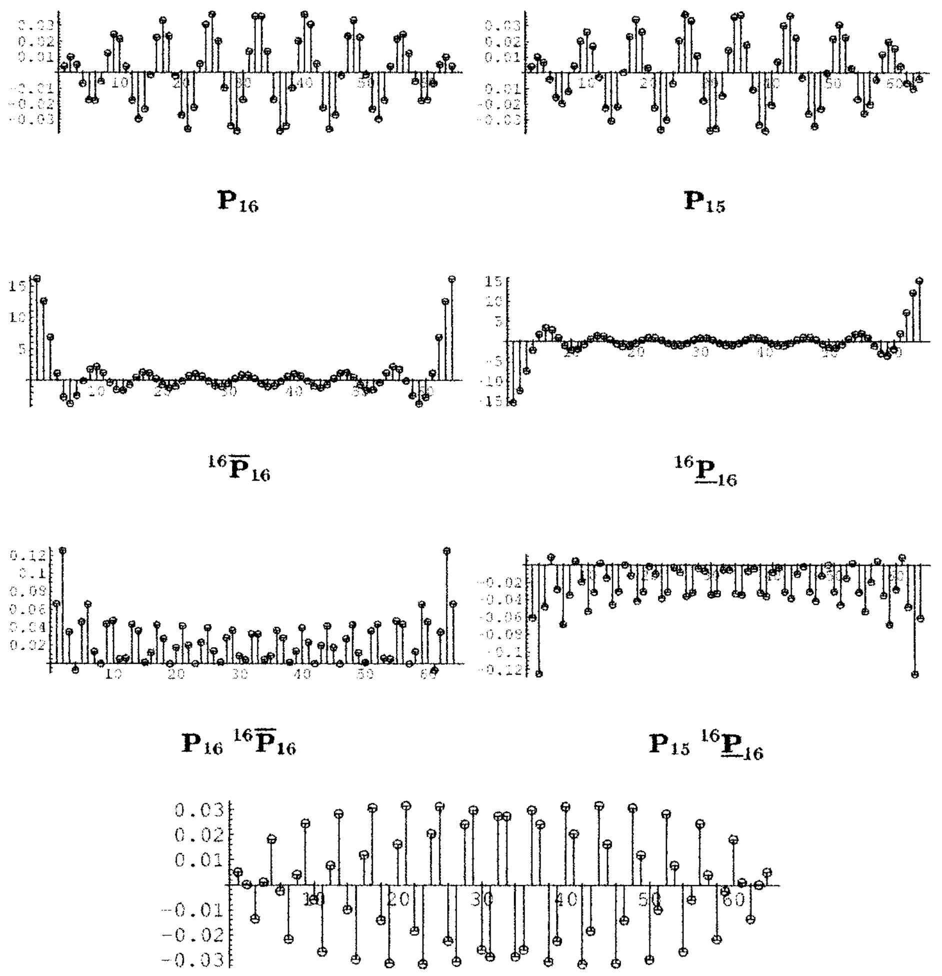

Figure 4. The vector of samples of the Legendre polynomial ${}^{0}P_{32}$ (bottom) decomposed as ${}^{0}\overline{\mathbf{P}}_{16}{}^{16}\overline{\mathbf{P}}_{16} + {}^{0}\overline{\mathbf{P}}_{15}\ {}^{16}\underline{\mathbf{P}}_{16}$, with products (and sums) computed componentwise. The Legendre polynomials have been scaled by the quadrature weights.

We begin by writing the DLT calculation in terms of the matrix valued sampled Legendre functions: given sampled data $\mathbf{f}$ to be transformed, the DLT is

$$\bigl\{\langle \mathbf{f}, \mathbf{P}_\ell\rangle \mid 0 \leqslant \ell < N\bigr\} = \bigl\{\bigl\langle \mathbf{F}, {}^{0}\mathbf{\Pi}_k\bigr\rangle \mid 0 \leqslant k < N\bigr\}. \tag{17}$$

Here $\mathbf{F} = (\mathbf{f}, \mathbf{0})$, $\mathbf{0}$ is the N-vector of zeroes, and the inner product of matrix-valued functions is performed by taking the regular matrix product and performing the multiplications of matrix entries as inner products. For example, given two vectors of samples, $\mathbf{s}_1, \mathbf{s}_0$:

$$\bigl\langle \mathbf{S}, {}^{r}\mathbf{\Pi}_n\bigr\rangle = \bigl\langle (\mathbf{s}_1, \mathbf{s}_0), {}^{r}\mathbf{\Pi}_n\bigr\rangle = \bigl(\bigl\langle \mathbf{s}_1, {}^{r}\overline{\mathbf{P}}_n\bigr\rangle + \bigl\langle \mathbf{s}_0, {}^{r}\underline{\mathbf{P}}_n\bigr\rangle, \bigl\langle \mathbf{s}_1, {}^{r}\overline{\mathbf{P}}_{n-1}\bigr\rangle + \bigl\langle \mathbf{s}_0, {}^{r}\underline{\mathbf{P}}_{n-1}\bigr\rangle\bigr).$$

In particular, with $\mathbf{S} = \mathbf{F} = (\mathbf{f}, \mathbf{0})$, and $r = 0$, the matrix inner product computes the pair of DLT coefficients: $\langle \mathbf{F}, {}^0\mathbf{\Pi}_n \rangle = (\langle \mathbf{f}, \mathbf{P}_n \rangle, \langle \mathbf{f}, \mathbf{P}_{n-1} \rangle)$.

We split the discrete Legendre transform (17) into the evaluation of the *low-degree* Legendre coefficients

$$\left\{ \left\langle \mathbf{F}, {}^0\mathbf{\Pi}_\ell \right\rangle \mid 1 \leqslant \ell < \frac{N}{2} \right\}$$

and the evaluation of the *high-degree* Legendre coefficients

$$\left\{ \left\langle \mathbf{F}, {}^0\mathbf{\Pi}_{N/2+\ell} \right\rangle \mid 1 \leqslant \ell < \frac{N}{2} \right\}.$$

In the previous section we saw that the high-degree projections accounted for the bulk of the computational cost. The Legendre decomposition (16) now provides for the reduction of high-degree inner products to low-degree:

$$\left\langle \mathbf{F}, {}^0\mathbf{\Pi}_{N/2+\ell} \right\rangle = \left\langle \mathbf{F}, {}^0\mathbf{\Pi}_{N/2} \, {}^{N/2}\mathbf{\Pi}_\ell \right\rangle = \left\langle \mathbf{F} \, {}^0\mathbf{\Pi}_{N/2}, {}^{N/2}\mathbf{\Pi}_\ell \right\rangle$$

with the last step easily verified from the definitions of the pointwise and inner products of the appropriately sized matrix-valued functions.

Applying this result and using $\mathcal{M}^0_{N/2}$ to denote the modulation operator in the last inner product: $\mathcal{M}^0_{N/2}\mathbf{F} = \mathbf{F} \, {}^0\mathbf{\Pi}_{N/2}$, the DLT computation may be rewritten as two sets of *low-degree* projections:

$$\begin{array}{rl} \textit{Low-degree coefficients} & \\ \textit{of } \mathbf{F} & \left\{ \left\langle \mathbf{F}, {}^0\mathbf{\Pi}_k \right\rangle \mid k = 1, \ldots, \frac{N}{2} - 1 \right\}, \\ \textit{Low-degree coefficients} & \\ \textit{of } \mathbf{F} \, {}^0\mathbf{\Pi}_{N/2} = \mathcal{M}^0_{N/2}\mathbf{F} & \left\{ \left\langle \mathcal{M}^0_{N/2}\mathbf{F}, {}^{N/2}\mathbf{\Pi}_k \right\rangle \mid k = 1, \ldots, \frac{N}{2} - 1 \right\}. \end{array} \tag{18}$$

Although we have now split the original problem into two collections of $N/2$ projections, we have yet to gain any speed-up; the individual projections still involve vectors of the original input size N. However, because of the reduced bandwidth of the Legendre functions involved we may use lemma 1 to make the desired reduction. This lemma shows that projection onto an N-sample trigonometric polynomial of degree $\leqslant N/2$ can be computed instead as a sum of length $N/2$ by smoothing and subsampling with the operator $\mathcal{L}^N_{N/2}$. Extending this operator to matrix-valued sampled functions by applying it to each matrix entry, we are in position to reduce the complexity of the DLT of size N by splitting into two sets of projections of size $N/2$:

Lemma 3. The N^2 cost of directly evaluating the DLT coefficients of a length N input vector $\mathbf{f}$:

$$\left\{ \langle \mathbf{f}, \mathbf{P}_\ell \rangle \mid 0 \leqslant \ell < N \right\} = \left\{ \left\langle \mathbf{F}, {}^0\mathbf{\Pi}_k \right\rangle \mid 0 \leqslant k < N \right\}, \quad \mathbf{F} = (\mathbf{f}, \mathbf{0}),$$

may be reduced (for big enough N) to less than $(3/4)N^2 + (27/4)N \log N$ by:

(i) forming modified half-sized input vectors $\mathbf{\Phi}_1 = \mathcal{L}^N_{N/2}\mathbf{F}$, $\mathbf{\Phi}_2 = \mathcal{L}^N_{N/2}\mathcal{M}^0{}_{N/2}\mathbf{F}$, and then

(ii) directly evaluating of the two sets of projections:

$$\left\{\left\langle \mathbf{\Phi}_1, {}^0\mathbf{\Pi}_k\right\rangle \mid k = 1, \ldots, \frac{N}{2} - 1\right\},$$
$$\left\{\left\langle \mathbf{\Phi}_2, {}^{N/2}\mathbf{\Pi}_k\right\rangle \mid k = 1, \ldots, \frac{N}{2} - 1\right\};$$

with half-length sampled Legendre functions ${}^{lN/2}\mathbf{\Pi}_k$ assembled from the $N/2$ samples ${}^{lN/2}\Pi_k(\cos \pi(2j+1)/N)$, $j = 0, \ldots, N/2 - 1$.

Proof. We have seen that the inner products in equation (18) compute the coefficients of the original DLT problem. By lemma 1, the value of any of these inner products is unchanged when we apply the lowpass and subsampling operator $\mathcal{L}^N_{N/2}$ to its factors because the Legendre function in each inner product has band-limit less than or equal to $N/2$. In fact, applying $\mathcal{L}^N_{N/2}$ to any of these N-sample Legendre functions has no effect other than to resample them on the regular grid of size $N/2$. This requires no computation, as we will simply prestore the appropriate function samples. Therefore, the DLT calculation is rewritten as two sets of inner products involving size $N/2$ vectors:

$$\left\{\langle \mathbf{f}, \mathbf{P}_\ell\rangle \mid 0 \leqslant \ell < N\right\} = \left\{\begin{array}{l} \left\{\left\langle \mathcal{L}\mathbf{F}, {}^0\mathbf{\Pi}_k\right\rangle = \left\langle \mathbf{\Phi}_1, {}^0\mathbf{\Pi}_k\right\rangle \mid k = 1, \ldots, \dfrac{N}{2} - 1\right\}, \\ \left\{\left\langle \mathcal{L}\mathcal{M}\mathbf{F}, {}^{N/2}\mathbf{\Pi}_k\right\rangle = \left\langle \mathbf{\Phi}_2, {}^{N/2}\mathbf{\Pi}_k\right\rangle \mid k = 1, \ldots, \dfrac{N}{2} - 1\right\}; \end{array}\right.$$

with the number of samples of the Legendre functions determined by the size of the inner product in which they appear (here $N/2$), and for notational simplicity we have written $\mathcal{L} = \mathcal{L}^N_{N/2}$ and $\mathcal{M} = \mathcal{M}^0_{N/2}$.

After forming the modified inputs $\mathbf{\Phi}_i$, the computation of the first group of $N/2$ inner products requires $(N/2)^2$ operations and the evaluation of the second group requires twice that, as each of its $N/2$ lagged Legendre functions must be projected onto the two $N/2$-vectors comprising $\mathbf{\Phi}_2 = \mathcal{L}\mathcal{M}\mathbf{F}$.

To this we must add the overhead of computing the modified inputs. Creating $\mathbf{\Phi}_1$ requires applying the lowpass operator to $\mathbf{f}$ at cost $(3/2)(N \log N + (N/2)\log(N/2))$. In building $\mathbf{\Phi}_2 = \mathcal{L}\mathcal{M}\mathbf{F}$, the modulation requires $2N$ operations, and then applying the lowpass operator twice (once for each of the two components of $\mathcal{M}\mathbf{F}$) requires an additional $2[(3/2)(N \log N + (N/2)\log(N/2))] = (9/2)N \log N - (3/2)N$ operations. Combining the costs of the modulation and the smoothing, we see that the total overhead is bounded by $(27/4)N \log N$. □

Remark. The above result implies that computing a sufficiently large Legendre transform by splitting into two half-sized sets of projections and directly evaluating them reduces the cost relative to drect evaluation of the original full-sized problem. One can even do a bit better than this by proceeding as in lemma 2 and performing the requisite inner products in the cosine transform domain, assuming that the cosine transforms of the lagged Legendre polynomials were prestored. The overhead is reduced to the cost of modulating the input and applying the DCT to the resulting vectors as well as to the original input vector, costing at most $3 \cdot (3/2)N \log N + 2N$. The total cost of all of the inner products with the prestored DCT's of Legendre functions costs no more than $1/2(N/2)(N/2 + 1)$ for the first group of inner products, and twice that for the second group. Thus a total of no more than $(3/8)N^2 + (9/2)N \log N + (11/4)N$ operations would be needed to compute in this manner. This is opposed to $(N/2)(N + 1) + (3/2)N \log N = (1/2)N^2 + (3/2)N \log N + (1/2)N$ for a complete semi-naive approach to the original full-sized problem. Hence we obtain an advantage for $N \geqslant 256$.

Of course, the real algorithmic advantage is obtained by performing this split recursively, subdividing the original problem into smaller and smaller subproblems. Here we briefly sketch the particulars; more details may be found in [11]. In order to see how the pieces fit together, we need a uniform description of the computational "unit" encountered at each division. This motivates the following definition.

Definition 1. For integers $N > 0$ a power of 2, and $R \geqslant 0$, and given input data $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_0)$, comprised of two row N-vectors, *the size N, lag R Legendre transform of* $\mathbf{S}$, $LT_N^R(\mathbf{S})$ is the vector of coefficients obtained by the inner product of matrix functions: $\langle \mathbf{S}, {}^R\mathbf{\Pi}_{N:1} \rangle$, using the $2N \times N$ matrix of lagged Legendre functions

$$
\begin{aligned}
{}^R\mathbf{\Pi}_{N:1} &= \left({}^R\mathbf{\Pi}_{N-1}\, {}^R\mathbf{\Pi}_{N-3} \quad \ldots \quad {}^R\mathbf{\Pi}_1 \right) && (19)\\
&= \begin{pmatrix} {}^R\overline{\mathbf{P}}_{N-1} & {}^R\overline{\mathbf{P}}_{N-2} & \cdots & {}^R\overline{\mathbf{P}}_0 \\ {}^R\underline{\mathbf{P}}_{N-1} & {}^R\underline{\mathbf{P}}_{N-2} & \cdots & {}^R\underline{\mathbf{P}}_0 \end{pmatrix} && (20)
\end{aligned}
$$

where ${}^R\overline{\mathbf{P}}_k$ and ${}^R\underline{\mathbf{P}}_k$ are appropriately sampled lagged Legendre functions defined in section 2.4:

$$
{}^R\overline{\mathbf{P}}_\ell = \begin{pmatrix} {}^R\overline{P}_\ell(\cos\theta_0) \\ \vdots \\ {}^R\overline{P}_\ell(\cos\theta_{N-1}) \end{pmatrix}, \qquad {}^r\underline{\mathbf{P}}_\ell = \begin{pmatrix} {}^R\underline{P}_\ell(\cos\theta_0) \\ \vdots \\ {}^R\underline{P}_\ell(\cos\theta_{N-1}) \end{pmatrix},
$$

with $\theta_k = \pi(2k + 1)/(2N)$. Note direct evaluation of LT_N^R costs no more than $2N^2$ operations

For example, the original Legendre polynomial transform of a data vector $\mathbf{f}$ of length N may be written using definition 1 as $LT_N^0(\mathbf{F})$ with $\mathbf{F} = (\mathbf{f}, \mathbf{0})$. Moreover, the two transforms obtained after splitting the Legendre polynomial transform also have the form described in this definition, as would the four transforms resulting from splitting these, etc.

The naive complexity of computing a Legendre transform of size M and lag L is at most $2M^2$ operations. In the special case of $L = 0$, section 2.2 presented a "semi-naive" approach which slightly improved upon this. Likewise, for general L there is a semi-naive approach offering a slight complexity advantage for the general transform in definition 1, as noted in the following lemma.

Lemma 4. Assuming that the cosine transforms $\mathcal{C}^L\overline{\mathbf{P}}_r$ and $\mathcal{C}^L\underline{\mathbf{P}}_r$ are prestored ($0 \leqslant r < M$), then the *LT* of size M and lag L, $LT^L_M(\mathbf{S})$, can be computed in at most $M^2 + 3M\log M + M$ operations by the semi-naive approach.

Proof. Using fast DCTs [26], at most $2 \cdot (\frac{3}{2}M\log M)$ operations are needed to compute the DCTs of the components of the input data, $\mathbf{s_0}$ and $\mathbf{s_1}$.Having done that, an additional $2 \cdot M(M+1)/2$ are needed to compute all of the pairs of inner products $\{\langle\mathcal{C}^L\overline{\mathbf{P}}_r, \mathcal{C}\mathbf{s_1}\rangle, \langle\mathcal{C}^L\underline{\mathbf{P}}_r, \mathcal{C}\mathbf{s_0}\rangle \mid r = 0, \ldots, M-1\}$ and to add together each of the pairs. □

Previously we showed a way to split the Legendre polynomial transform into two smaller transforms which enabled its evaluation with even lower complexity than semi-naive. This result may also be generalized to the Legendre transforms of definition 1. In the special case of the Legendre polynomial transform, lemma 3 demonstrated that the low- and high-degree projections of the original transform $LT^0_N(\mathbf{F})$ could be computed instead as $LT^0_{N/2}(\mathcal{L}\mathbf{F})$ and $LT^{N/2}_{N/2}(\mathcal{L}\mathcal{M}^0_{N/2}\mathbf{F})$, respectively. We saw that the total cost of direct evaluation of all the inner products in these two half-sized problems is half the cost of the direct evaluation of the original full sized problem. We also showed that the input vectors for the two smaller DLTs could be calculated efficiently from the data for the original full-sized problem. The latter point is critical, so as not to "eat all the profits" gained by the split.

To generalize this key step we construct a splitting operator Σ and characterize its complexity. Starting with the first split of the Legendre polynomial transform, let

$$\Sigma(\mathbf{F}) = \big(\mathcal{L}\mathcal{M}^0_{N/2}\mathbf{F}, \mathcal{L}\mathbf{F}\big) = \big(\mathcal{L}\mathcal{M}^0_{N/2}\mathbf{F}, \mathcal{L}\mathcal{M}^0_0\mathbf{F}\big). \tag{21}$$

The second equality follows since $\mathcal{M}^0_0\mathbf{F}$ is the modulation $\mathbf{F}^0\mathbf{\Pi}_0$ and using (12)

$${}^0\mathbf{\Pi}_0 = \begin{pmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{pmatrix},$$

where $\mathbf{1}$ and $\mathbf{0}$ are column vectors of 1's and of 0's, respectively. This is just the matrix form of the initial condition for the lagged Legendre functions.

With this notation the first divide and conquer step in the Legendre polynomial transform takes the form:

$$\begin{tikzcd}
 & \mathcal{M}^0_{N/2} \arrow[r, "\mathcal{L}"] & {[\Sigma\mathbf{F}]_1} \arrow[r] & LT^{N/2}_{N/2} \\
\mathbf{F} \arrow[ur] \arrow[dr] & & & \\
 & \mathcal{M}^0_0 \arrow[r, "\mathcal{L}"'] & {[\Sigma\mathbf{F}]_0} \arrow[r] & LT^0_{N/2}
\end{tikzcd}$$

with the Legendre transform now computed as two half-sized Legendre transforms of data modified by the splitting operator.

For a complete recursive subdivision we must likewise efficiently split the leaves of this tree, and so on. This requires a splitting operator appropriate for the lagged Legendre transforms, as described in the following lemma.

Lemma 5 (Splitting lemma). Let M be a positive integer divisible by two.

(i) A Legendre transform of size M can be computed as two Legendre transforms of size $M/2$. Specifically, for any lag L the operator LT^L_M applied to a given input can instead be computed as the separate operations of $LT^L_{M/2}$ and $LT^{L+M/2}_{M/2}$ on new inputs obtained from the original input.

(ii) Let $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_0)$ be the $M + M$ row vector of initial data for a LT of size M and lag L. Then at most $9M \log M + M$ operations are needed to transform this to the pair of half-sized inputs for the pair of LT's of size $M/2$ which together compute the original LT. The required inputs for the half-sized DLTs can be computed by the splitting operator Σ^L_M:

$$\Sigma^L_M \mathbf{S} = \left(\mathcal{L}\mathcal{M}^L_{1\cdot M/2}\mathbf{S}, \mathcal{L}\mathcal{M}^L_{0\cdot M/2}\mathbf{S}\right) = \left(\left[\Sigma^L_M\mathbf{S}\right]_1, \left[\Sigma^L_M\mathbf{S}\right]_0\right) \tag{22}$$

where the modulation operator is $\mathcal{M}^L_{n\cdot M/2}\mathbf{S} = \mathbf{S}\,{}^L\mathbf{\Pi}_{n\cdot M/2}$ and the lowpass operator is $\mathcal{L}\mathbf{S} = (\mathcal{L}^M_{M/2}\mathbf{s}_1, \mathcal{L}^M_{M/2}\mathbf{s}_0)$.

(iii) The operator LT^L_M has the decomposition

$$LT^L_M = \left(LT^{L+M/2}_{M/2} \oplus LT^L_{M/2}\right) \circ \Sigma^L_M; \tag{23}$$

that is,

$$LT^L_M(\mathbf{S}) = \left(LT^{L+M/2}_{M/2}\left(\left[\Sigma^L_M\mathbf{S}\right]_1\right), LT^L_{M/2}\left(\left[\Sigma^L_M\mathbf{S}\right]_0\right)\right).$$

If the component transforms are applied directly, the total complexity of evaluation by splitting is at most $M^2 + 9M \log M + M$.

Proof. The low-degree projections in the original size M problem, $\langle \mathbf{S}, {}^L\mathbf{\Pi}_k\rangle$, $0 \leqslant k < M/2$ only involve lagged Legendre polynomials of degree less than $M/2$. Consequently, according to lemma 1 these inner products may be computed as inner products of the lowpassed data with subsampled versions of the shifted Legendre polynomials, thereby reducing it to a *LT* of size $M/2$.

To reduce the set of high-order projections to a *LT* of half the size, we apply the recurrence formula (16) to obtain

$$\langle \mathbf{S}, {}^{L}\mathbf{\Pi}_{M/2+k}\rangle = \langle \mathbf{S}^{L}\mathbf{\Pi}_{M/2}, {}^{L+M/2}\mathbf{\Pi}_{k}\rangle$$

for k in the range $0 \leqslant k < M/2$. Since the Legendre functions ${}^{L+M/2}\mathbf{\Pi}_k$ have degree less than $M/2$ we can lowpass both sides of each inner product in the preceding and obtain the half-sized inner product

$$\langle \mathcal{L}\mathcal{M}^{L}_{M/2}\mathbf{S}, {}^{L+M/2}\mathbf{\Pi}_{k}\rangle. \tag{24}$$

This completes the proof of (i). The proof of (ii) follows immediately. The stated form of the splitting operator follows from the fact that ${}^{L}\Pi_0$ is the identity matrix (from the initial conditions of lagged Legendre recurrence). The complexity result follows from the fact that application of the modulation operator $\mathcal{M}^{L}_{M/2}$ requires at most $4M$ operations and application of $\mathcal{L}$ to the result costs an additional $2 \cdot \frac{3}{2}(M \log M + (M/2)\log(M/2)) = 3M(\log M + \frac{1}{2}[\log M - 1]) = \frac{9}{2}M \log M - \frac{3}{2}M$. In the low-degree problem, the modulation is trivial, and the lowpass operation costs the same (of course) as it does in the high-degree subproblem, leading to the total stated.

Finally, (iii) is simply a restatement of (i) and (ii) in the setting of matrix arithmetic. Adding the complexity of direct evaluation of two sized $M/2$ Legendre transforms to the complexity of splitting yields the final accounting. □

Using lemma 5 we can now describe the full algorithm in a succinct way. Starting with the original Legendre transform written as LT^{0}_{N}, part (iii) implies that this matrix factors as

$$LT^{0}_{N} = \left(LT^{N/2}_{N/2} \oplus LT^{0}_{N/2}\right) \circ \Sigma^{0}_{N}.$$

Now we apply lemma 5 to the half-sized *LT* matrices, $LT^{0}_{N/2}$ and $LT^{N/2}_{N/2}$ producing the factorization of LT^{0}_{N} as

$$\begin{aligned} LT^{0}_{N} &= \left\{\left[\left(LT^{3N/4}_{N/4} \oplus LT^{N/2}_{N/4}\right) \circ \Sigma^{N/2}_{N/2}\right] \oplus \left[\left(LT^{N/4}_{N/4} \oplus LT^{0}_{N/4}\right) \circ \Sigma^{0}_{N/2}\right]\right\} \circ \Sigma^{0}_{N} \\ &= \left(LT^{3N/4}_{N/4} \oplus LT^{N/2}_{N/4} \oplus LT^{N/4}_{N/4} \oplus LT^{0}_{N/4}\right) \circ \left(\Sigma^{N/2}_{N/2} \oplus \Sigma^{0}_{N/2}\right) \circ \Sigma^{0}_{N}. \end{aligned}$$

We may continue with this process of splitting Legendre transforms as often as desired. Using the notation we have introduced and lemma 5 we now have following.

Theorem 2. Let $N = 2^r$ and let $\mathbf{f}$ be any input vector of length N.

(i) The Legendre transform of $\mathbf{f}$ may be computed from $LT_N^0 \cdot (\mathbf{f}, \mathbf{0})$ via the factorization: $LT_N^0 = \mathcal{T}_t \mathcal{S}_{t-1} \cdots \mathcal{S}_0$, for any $t \leqslant r - 1$, where

$$\mathcal{S}_k = \bigoplus_{j=0}^{2^k-1} \Sigma_{N/2^k}^{jN/2^k} \tag{25}$$

with the 2^k splitting matrices $\Sigma_{N/2^k}^{jN/2^k}$ defined as in lemma 5, and

$$\mathcal{T}_t = \bigoplus_{j=0}^{2^t-1} LT_{N/2^t}^{jN/2^t}. \tag{26}$$

For a full decomposition, this reduces the size N transform LT_N^0 to $N/2$ 4×2 Legendre transforms

$$LT_2^{N-2} \oplus LT_2^{N-4} \oplus \cdots \oplus LT_2^0,$$

applied to data modified by the splitting operators.

(ii) The Legendre transform of $\mathbf{s}$ may be computed in $\mathrm{O}(N \log^2 N)$. Specifically, assuming that the computation is performed with the full r stage splitting described in (i) with precomputed and stored Legendre functions ${}^{lN/2^k}\mathbf{\Pi}_{N/2^k}$ used in the splitting operators, and precomputed and stored lagged Legendre functions occurring in the Legendre transforms LT_2^{2l}, then at most

$$8 \cdot \frac{N}{2} + \sum_{a=0}^{r-2} 2^a \cdot \left[9\frac{N}{2^a} \log \frac{N}{2^a} + \frac{N}{2^a} \right] = \frac{9}{2} N \log^2 N + N \log N - 10N$$

operations are needed to compute the Legendre transform of $\mathbf{f}$.

Proof. Part (i) follows directly from a recursive application of lemma 5. As for (ii), the first term in the complexity computation follows from the fact that $\mathcal{T}_{r-1}$ is block diagonal with $N/2$ (4×2) matrices LT_2^{2l} on the diagonal, each requiring 8 operations to multiply by a vector of length 4. The summation which is the second term is the complexity of the successive multiplications of the $\mathcal{S}_a$, each consisting of 2^a splitting operators of size $N/2^a$. The cost of each of these was obtained in lemma 5. Summing the complexities at each level gives (ii). □

The inductive nature of theorem 2 shows that the algorithm is simply given by the successive application of the splitting operators to an input vector resulting from previous stages. In practice, this splitting is applied as long as it offers computational advantage. At this point the calculation may be concluded by applying the shifted Legendre matrices according to the semi-naive algorithm (cf. lemma 4). The choices of how far to continue splitting before evaluating Legendre transforms gives rise to many variants of the basic algorithm, which can be optimized for particular situations. We will discuss this a bit more later on.

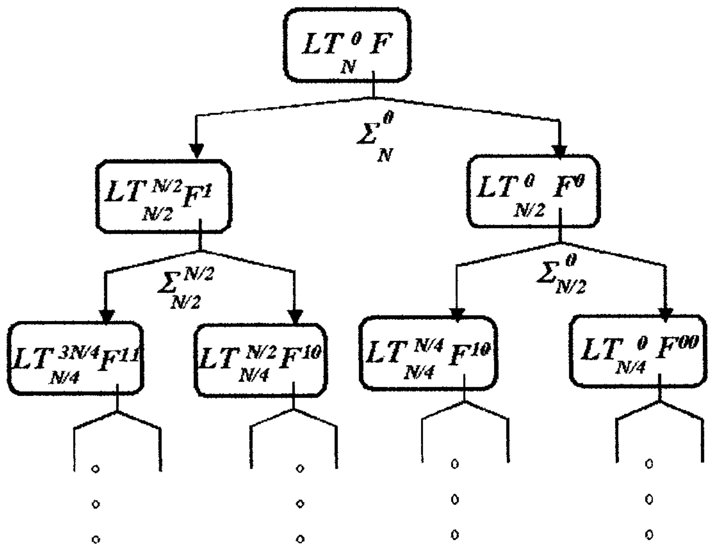

Figure 5. Schematic illustration of the computation of the Legendre transform by recursive splitting. At any level one can take the input vectors and evaluate the Legendre transforms, or elect to apply a splitting operator to produce input for the next level.

Several stages of the Legendre splitting are show schematically in figure 5.

In our discussion of Legendre transforms, we have primarily used the case of $m = 0$ for simplicity of illustration. The case of $m \neq 0$ is treated essentially identically. The approach we have used to date is to apply the splitting procedures already described in the case of order m associated Legendre transforms by using the recurrence

$$\begin{aligned} {}^m\Pi^m_{\ell+k}(\cos\theta)\sin^m\theta &= \left({}^m\Pi^m_\ell(\cos\theta)\quad {}^{\ell+m}\Pi^m_k(\cos\theta)\right)\sin^m\theta \\ &= {}^m\Pi^m_\ell(\cos\theta)\sin^m\theta\ {}^{\ell+m}\Pi^m_k(\cos\theta). \end{aligned}$$

We will see later that this particular choice can lead to numerical difficulties and that a new choice which we will introduce will be very helpful. However, up to now we have used this basic recurrence to permit a reduction of complexity for the computation of associated Legendre coefficients of high-degree. As before, we consider these to correspond to projections onto associated Legendre functions of degree above a splitting level S (say about half the bandwidth, $B/2$), which are reduced to projections onto low-degree lagged Legendre functions:

$$\left\langle \mathbf{F}, {}^m\boldsymbol{\Pi}^m_{S+\ell}\mathbf{sin}^m \right\rangle = \left\langle \mathbf{F}\mathbf{sin}^m, {}^m\boldsymbol{\Pi}^m_{S+\ell} \right\rangle = \left\langle \mathbf{F}^m\boldsymbol{\Pi}^m_S\mathbf{sin}^m, {}^{S+m}\boldsymbol{\Pi}^m_\ell \right\rangle.$$

Here we have introduced the notation $\mathbf{sin}^m$ for the vector obtained by sampling $\sin^m\theta$ on our sampling grid, and it is multiplied pointwise onto the components of the matrix-valued functions with which it appears.

These inner products involve reduced bandwidth trigonometric polynomials, and can be computed with reduced vectors after low pass filtering as before. One minor difference from the $m = 0$ case is that for the bandwidth B problem we need only to compute the inner products $\langle \mathbf{F}, {}^m\mathbf{\Pi}_k^m \mathbf{sin}^m \rangle$ for k in the reduced range of degrees $0 \leqslant k < B - m$. For simplicity we can, for example, extend to a problem with a full range of degrees $0 \leqslant k < B$ by simply defining the lagged Legendre functions to be 0 in the range $k > B - m$. The recursive splitting then proceeds just as in the $m = 0$ case.

We summarize the results reviewed in this section:

Theorem 3. For bandwidth B, a fixed power of 2, and for each m, $|m| \leqslant B$, the order m discrete Legendre transform of a data vector $\mathbf{f}$

$$\left\{ \langle \mathbf{f}, \mathbf{P}_\ell^m \rangle = \sum_{k=0}^{2B-1} [\mathbf{f}]_k P_\ell^m(\cos\theta_k) \mid \ell = |m|, |m+1|, \ldots, B-1 \right\}$$

may be computed in $\mathrm{O}(B \log^2 B)$ operations.

Remark. With a fast DLT for each order m, we can assemble an FFT for S^2 through a simple combination of DLTs and regular Abelian FFTs (see [5,11] for the simple proof). In this section, we sketched a fast algorithm which takes a direct DLT cast as a matrix/vector multiply, and rewrites it as a succession of applications of structured matrices corresponding to a matrix factorization of the DLT, whose structure permits an efficient computation. This gives the full spherical FFT as a matrix factorization as well. Inversion of this transform (i.e., the computation of sample data from Fourier coefficients) amounts to the transposition of the spherical FFT algorithm which in turn reorders the fast factorization and transposes the individual matrices. The resulting algorithm has a complexity of similar order as the fast forward transform, and hence is itself a fast inversion routine. Finally, a fast convolution algorithm for the sphere is obtained in analogy to the familiar result for the circle. There the Fourier transform of the convolution is the pointwise product of the Fourier transforms. For the sphere we simply replace pointwise product with a particular matrix product of Fourier coefficients of a given order. Thus a fast algorithm for convolution is given by first transforming (efficiently) both factors to the Fourier domain, computing the new coefficients by by a low complexity matrix product, and then computing an efficient inverse transform back to sample data. For details see [5,11].

3.2. *Speed and stability of previous implementations of the fast Legendre transform*

We have seen that the discrete Legendre transform (DLT) of length N may be performed in $\mathrm{O}(N \log^2 N)$ operations, with an algorithm that is exact, in exact arithmetic, for band-limited functions. Moreover, a careful error analysis of the effect of finite precision arithmetic on the algorithm and various computational experiments confirmed the stability of an order $m = 0$ DLT [5].

However, further computational experiments revealed that instabilities of growing severity occur as the order m of the associated Legendre transform increases [11]. A key to this is the behavior of the lagged Legendre functions appearing in the higher-order

Legendre decompositions, the core of our fast algorithm. In particular, if we split the Legendre transform at degree S, we use the decomposition:

$$\begin{pmatrix} P^m_{S+\ell}(\cos\theta) & P^m_{S+\ell-1}(\cos\theta) \\ * & * \end{pmatrix} = \begin{pmatrix} P^m_S(\cos\theta) & P^m_{S-1}(\cos\theta) \\ * & * \end{pmatrix} \times \begin{pmatrix} {}^S\overline{P}^m_\ell(\cos\theta) & {}^S\overline{P}^m_{\ell-1}(\cos\theta) \\ {}^S\underline{P}^m_\ell(\cos\theta) & {}^S\underline{P}^m_{\ell-1}(\cos\theta) \end{pmatrix}$$

to reduce the effective degree, and hence complexity, of a high-degree Legendre coefficient computation:

$$\langle f, P^m_{S+\ell}\rangle = \langle f P^m_S, {}^S\overline{P}^m_\ell\rangle + \langle f P^m_{S-1}, {}^S\underline{P}^m_\ell\rangle.$$

The decomposition used here is the obvious candidate, but its use causes severe stability problems for even moderate orders m. This is correlated with the behavior of the lagged Legendre functions which accompany its use. Specifically, the lag-S Legendre functions ${}^S\overline{P}^m_\ell(\cos\theta)$, ${}^S\underline{P}^m_\ell(\cos\theta)$ grow increasingly *large* with increasing ℓ when the argument θ approaches the ends of the domain interval, i.e. at $\theta = 0$, and at π, corresponding to the poles of the sphere. At these very same positions, the associated Legendre function $P^m_S(\cos\theta)$, $P^m_{S-1}(\cos\theta)$ become very *small*. See figure 6 for an example of this behavior of the factors in the Legendre decomposition for an order $m = 64$ problem.

While it has been evident for some time that this large dynamic range among the factors in the Legendre decomposition is related to the numerical problems, the particular mechanism was not explored in previous work as far as we know. In section 4 of this paper we examine this more closely and propose a new approach to this difficulty using a simple but effective modification of the Legendre decomposition.

In the past we have seen two methods proposed for dealing with these instabilities. First, one may apply stabilization procedures to certain portions of the original, basic algorithm at the cost of some runtime efficiency [18,22]. Alternatively, one may develop variants of the original algorithm that constrain the process of recursive splitting in order to avoid or minimize the deleterious influences of the dynamic range difficulties with certain of the lagged Legendre functions. We have pursued this course in [11]. While these variants have asymptotic complexity greater than the basic algorithm presented previously, implementations nevertheless yield numerically results at runtime speeds faster than both naive and seminaive algorithms for problem sizes of interest.

The most promising variants, the "simple split" and "hybrid" algorithms of [11], exploit the fact that certain lagged Legendre functions behave (numerically) better than others. The rate at which the maximum values of the lagged Legendre functions $|{}^r\overline{P}^m_l(x)|$, $|{}^r\underline{P}^m_l(x)|$ grow as $l \to \infty$ is slower for larger values of the lag r than for small. For example, we have $\max|{}^{768}\overline{P}^{512}_{10}(x)| \approx 3600$ while $\max|{}^{512}\overline{P}^{512}_{10}(x)| \approx 10^{11}$. Recalling the structure of recursive splitting for the fast Legendre transform from theorem 2 we find that the large lag Legendre functions occur in the splitting operators for the computations corresponding to the higher-degree subproblems. We find there are

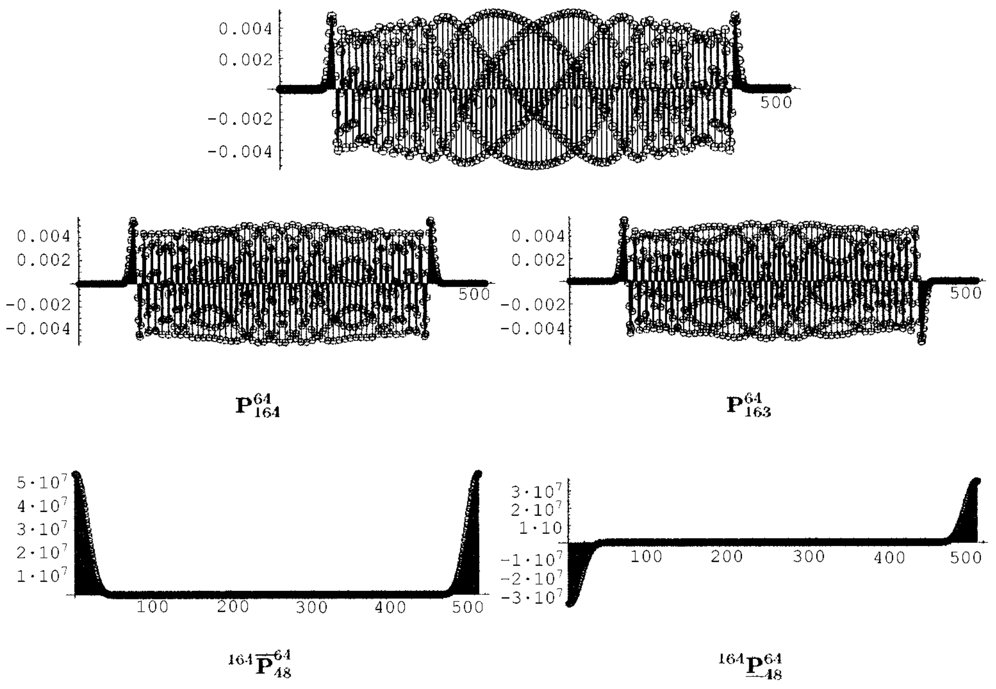

Figure 6. Order $m = 64$ Legendre function $\mathbf{P}^{64}_{212}$ (top) pictured with the factors in its decomposition as $\mathbf{P}^{64}_{164}\ {}^{164}\overline{\mathbf{P}}^{64}_{48} + \mathbf{P}^{64}_{163}\ {}^{164}\mathbf{P}^{64}_{-48}$. Note the large values of lagged Legendre functions ${}^{164}\overline{\mathbf{P}}^{64}_{48}$, ${}^{164}\mathbf{P}^{64}_{-48}$ near the endpoints of the interval.

portions of these computations which are numerically well behaved and others which are not. We observe these as constraints in the construction of the algorithm variants under discussion.

The idea behind the simple split algorithm is to immediately split the problem of size N into C many subproblems and apply a semi-naive approach on each subproblem. For the hybrid algorithm, one uses the semi-naive algorithm to compute, without involving any lagged Legendre polynomials, the lower-degree Legendre coefficients up to some previously determined bound, and then use the simple-split algorithm to compute the remaining, higher-degree coefficients.

We have run extensive numerical experiments on a wide variety of platforms, including a DEC Alpha 500/200, an HP Exemplar X-Class SPP2000/64, a SGI Origin 2000, and a 700 MHz Pentium 4 Linux workstation. To give some idea, results obtained on the Pentium 4 and plotted in figure 7 are qualitatively similar to those obtained on the other platforms. For an order $m = 0$ DLT, the hybrid and simple split algorithms are both significantly faster than the semi-naive by band-limit $bw = 512$, while offering numerically reliable results. See [11] for extensive details.

These are encouraging results, but for higher-order cases, the choice of algorithms is heavily constrained by the numerical problems and can suffer in runtime as a consequence. In the next section we show how to lift many of these constraints while retaining

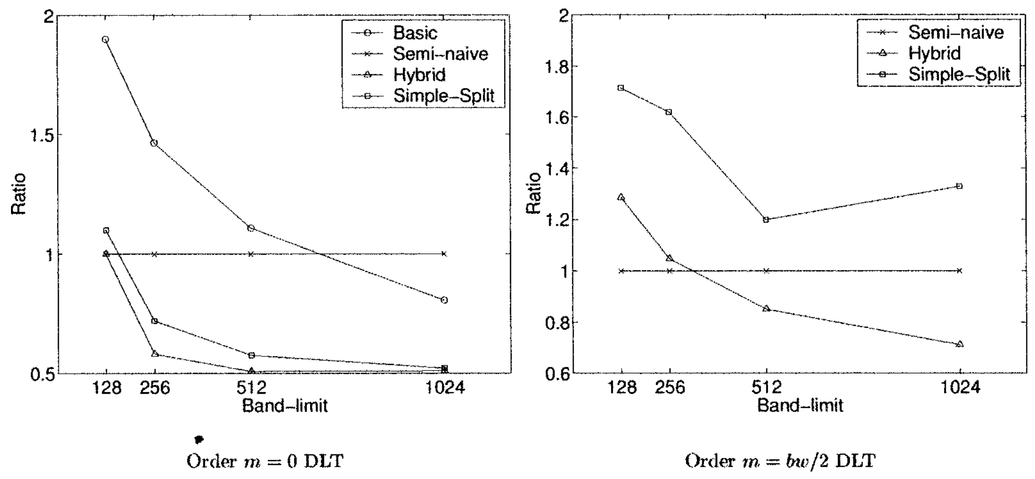

Figure 7. 700 MHz Pentium: DLT runtime ratios vs. semi-naive.

the same basic algorithmic structure through the use of a modified Legendre decomposition.

4. Towards stable, fast high-order Legendre transforms

The numerical results of [11], briefly reviewed in the previous section, indicate numerical difficulties in our baseline fast Legendre transforms for the higher-order cases, and motivate some of the algorithmic variants we mentioned. Most of the difficulties come from instabilities of the lagged Legendre transforms for the higher-orders. Workarounds attempt to avoid the use of certain higher-order lagged transforms, and constrain the splitting of the original problem to do this. Often stability is purchased at a cost in increased complexity, particularly for large problems. Once again, see [11] for details.

The general situation is in marked contrast to the $m = 0$ case where theoretical bounds and experiment demonstrate numerical reliability of that algorithm. The schedule of splittings can be organized entirely on the grounds of speed without worrying about numerical problems. It would obviously be most desirable to have this freedom for the higher-order Legendre transforms; we would require the higher-order lagged Legendre functions to behave more like the $m = 0$ case in terms of numerical stability. In this section, we present a mechanism for obtaining this, through a new decomposition of the Legendre functions. This new decomposition dramatically improves the numerical properties of the basic fast DLT algorithms we have discussed. Our discussion is accompanied by experimental results.

We begin with an experimental description of the propagation of numerical errors arising from the use of the Legendre decomposition in its basic form. We find a somewhat surprising result that the decomposition itself does not cause the trouble until we introduce the lowpass operation in the fast algorithm. This suggests that simple modifications of the approach we have considered will be effective in the higher-order cases.

We continue by presenting a slight variation of the decomposition (12). This will prove to be an equally valid Legendre decomposition in the sense that the components still satisfy the basic Legendre recurrence, but with modified initial conditions. We show that we can take the same basic divide-and-conquer strategy presented in the last section and substitute the modified Legendre decomposition for (12). This makes an enormous difference in the numerical reliability at no extra cost in arithmetic operation. This will be demonstrated with some examples from large-size problems where we would otherwise be unable to apply the basic fast algorithm and obtain meaningful results.

4.1. Error mechanisms for high-order Legendre decompositions

The basic Legendre decomposition:

$$\left(P^m_{S+\ell}(\cos\theta),\, P^m_{S+\ell-1}(\cos\theta)\right) = \left(P^m_S(\cos\theta),\, P^m_{S-1}(\cos\theta)\right)\, {}^S\Pi^m_\ell(\cos\theta)$$

plays an essential role in the fast algorithm we have presented, allowing us to to rewrite projections of input data onto associated Legendre functions of higher degree, $S+\ell$, as projections onto the lower-degree lagged Legendre functions, ${}^S\Pi^m{}_\ell(\cos\theta)$. Thus, the behavior of this lagged Legendre function has direct bearing on the numerical stability of the algorithm. As previously noted, for m even slightly different from zero this function can take on extremely large values near the endpoints of its interval of definition, i.e. for θ near 0 or π. Furthermore, these values grow rapidly as the degree ℓ increases. Such functions certainly do occur when we split a large bandwidth problem, and so can pose a serious numerical difficulty. For illustration, note this behavior depicted in figure 6, in which an order $m = 64$ associated Legendre function of fairly high-degree, P^{64}_{212}, is split into a product of the associated Legendre functions at the splitting degree $S = 164$, and the lagged Legendre functions in ${}^{164}\Pi^{64}{}_{48}(\cos\theta)$. The graphs of these functions were generated by means of the Legendre recurrence (10) implemented in Mathematica routines with controllable precision and accuracy. The figure gives an accurate impression of the behavior of lagged Legendre functions near the domain boundary when the order m is nonzero. Note that the degree in this case is a moderate 48; the behavior only gets worse with increasing degree.

In contrast, when $m = 0$, explicit bounds as well as computational experiments show that the lagged Legendre functions arising in the Legendre decomposition are much better behaved. In [5], these bounds have been used to demonstrate good a priori stability for the computation of the Legendre polynomial transform using the Legendre decomposition which we have described in section 3. Computational experiments in [5,11] confirm this result. For $m \neq 0$, similar computational experiments reveal that our basic algorithm for the fast Legendre transform is numerically unreliable in its computation of the high-degree coefficients, and that this problem correlates with the growth of the lagged Legendre functions with increasing degree at the endpoints of the interval of definition. This is consistent also with analytic estimates on their growth which can be obtained by techniques similar to those discussed in [5] for Legendre polynomial case. Given this, it is tempting to dismiss the use of the basic splitting approach entirely in the high-order cases, on the grounds that its numerical difficulties are unavoidable consequences of imbalance in the size scale of the functions appearing in the Legendre decomposition, such as that illustrated in figure 6. After all, this decomposition attempts to express reasonably behaved Legendre functions near the endpoints of their domain in terms of products of very large and very small numbers. While such a decomposition may be true and valid in exact arithmetic, it becomes somewhat suspect in finite precision.

So it may be surprising that this last conclusion proves to be an oversimplification of the difficulties, and the dismissal of splitting algorithms for reasons which are based purely on misgivings about the Legendre recurrences appears to be premature. In fact, in figure 8 we illustrate that the Legendre decomposition can be computed reliably in fixed precision, and furthermore, it can be applied to compute inner products of input data reliably. Figure 8 derives from a Matlab implementation used to evaluate the finite precision matrix multiplications prescribed in the Legendre decomposition of $\mathbf{P}^m_{S+\ell}$, i.e., we compute

$$fl\big(\mathbf{P}^m_S\,{}^S\overline{\mathbf{P}}^m_\ell + \mathbf{P}^m_{S-1}\,{}^S\underline{\mathbf{P}}^m_\ell\big),$$

where fl denotes that the expression is evaluated in floating point double precision. The error

$$\mathbf{P}^m_{S+\ell} - fl\big(\mathbf{P}^m_S\,{}^S\overline{\mathbf{P}}^m_\ell + \mathbf{P}^m_{S-1}\,{}^S\underline{\mathbf{P}}^m_\ell\big)$$

is computed and displayed for each sample of this difference function, with no error bigger than 10^{-14}. These computational experiments experiments begin with precom-

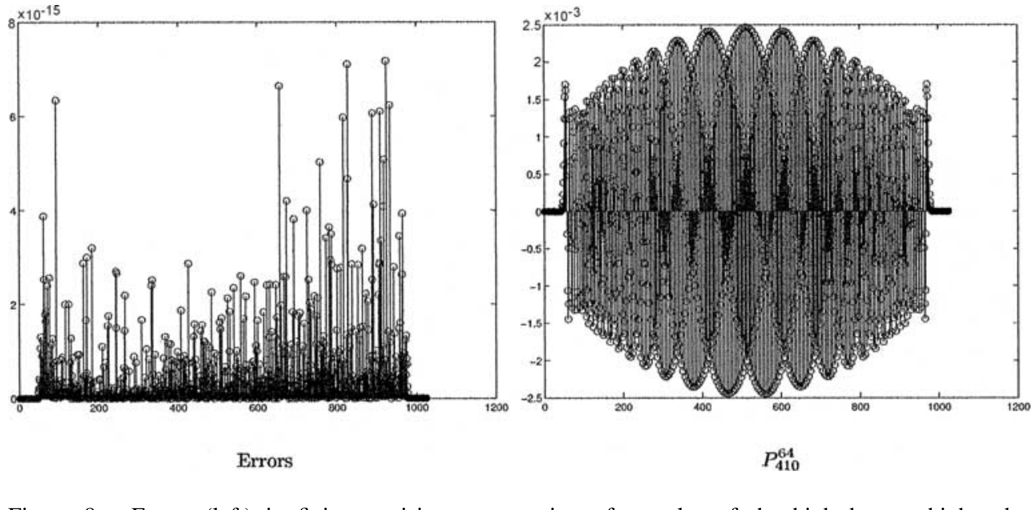

Figure 8. Errors (left) in finite precision computation of samples of the high-degree, high-order Legendre function P^{64}_{410} (right) from lower-degree Legendre functions and lagged Legendre functions: $\mathbf{P}^m_{S+\ell} - fl(\mathbf{P}^m_S\,{}^S\overline{\mathbf{P}}^m_\ell + \mathbf{P}^m_{S-1}\,{}^S\underline{\mathbf{P}}^m_\ell)$ for the case $m = 64$, $S = 288$, $\ell = 122$. The lagged Legendre functions in this decomposition have maximum values greater than 10^{10}. Despite wide dynamic range in the factors of the Legendre decomposition, finite precision multiplication of the factors in the decomposition of a high-degree associated Legendre function produces good agreement with that function.

putation (in Mathematica extended precision) of the appropriately sampled associated Legendre functions and lagged Legendre functions, which are then stored and ultimately input into Matlab (presumably) correct to the full double precision. All of the required multiplications and additions then are done in double precision in Matlab. These (and many other) experiments show that there is little difference between the high-degree Legendre function and the finite precision combination of lower-degree Legendre functions mandated by the decomposition.

We have applied this decomposition to the computation of Legendre coefficients. These can be computed either directly: $\langle \mathbf{f}, \mathbf{P}^m_{S+\ell}\rangle$ or by means of the splitting trick: $\langle \mathbf{f}\mathbf{P}^m_S, {}^S\overline{\mathbf{P}}^m_\ell\rangle + \langle \mathbf{f}\mathbf{P}^m_{S-1}, {}^S\underline{\mathbf{P}}^m_\ell\rangle$ with identical results in exact arithmetic. In fact, this is still quite a good approximation in finite precision arithmetic. Figure 9 shows an example of experimental computations done in Matlab demonstrating that the difference is quite small when the computations are done in double precision. We show the highest-degree coefficients, as these typically exhibit the largest errors. In this example, we find no coefficient has relative error bigger than 10^{-12}. Many similar numerical experiments strongly suggest that the inner products can be computed by the splitting without excessive numerical problems, provided the various prestored Legendre functions are correct.

However, in order to reap a speed-up from the reduced-degree inner products, we have seen in the previous section that we need to lowpass and subsample the modulated data before computing the reduced inner product. When the inner products are computed in finite precision with lowpassed and subsampled modulated data, i.e. as

$$fl\big(\big\langle \mathcal{L}\big(\mathbf{f}\mathbf{P}^m_S\big), {}^S\overline{\mathbf{P}}^m_\ell\big\rangle + \big\langle \mathcal{L}\big(\mathbf{f}\mathbf{P}^m_{S-1}\big), {}^S\underline{\mathbf{P}}^m_\ell\big\rangle$$

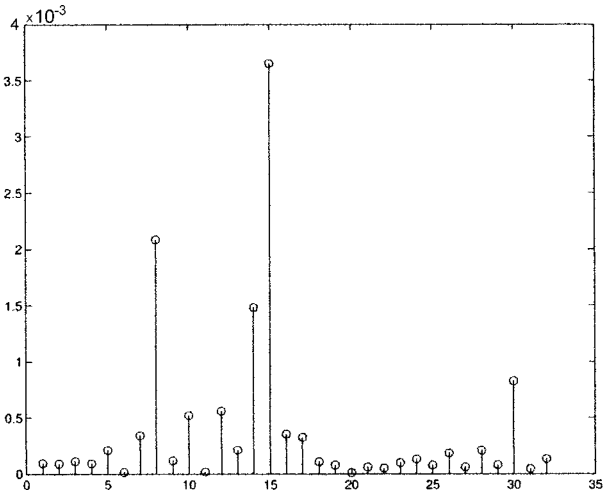

Figure 9. Despite wide dynamic range in Legendre decomposition, we obtain accurate calculation of DLT coefficients $\langle \mathbf{f}, \mathbf{P}^m_{S+\ell}\rangle$ as $fl(\langle \mathbf{f}\mathbf{P}^m_S, {}^S\overline{\mathbf{P}}^m_\ell\rangle + \langle \mathbf{f}\mathbf{P}^m_{S-1}, {}^S\underline{\mathbf{P}}^m_\ell\rangle)$. In this example, $m = 64$, $S = 288$, and shown here are the relative errors in the highest 32 Legendre coefficients of a random bandwidth 512 function, corresponding to degrees $S + \ell = 480, \ldots, 511$.

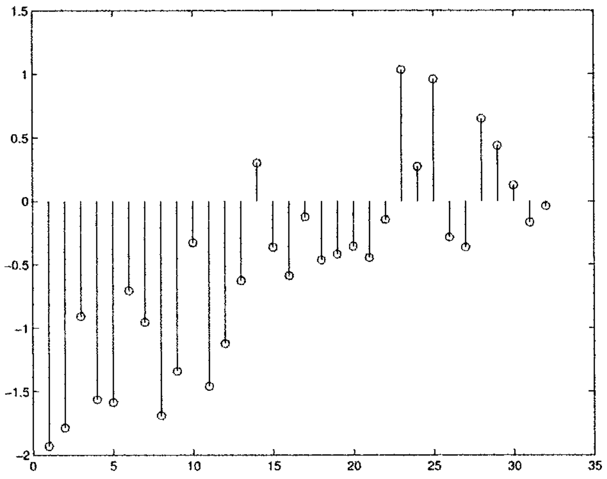

Figure 10. Finite precision computation of lowpass filter of modulated data $\mathcal{L}(\mathbf{f}\mathbf{P}_S^m)$ results in errors which ruin the finite precision computation of the coefficients in figure 9: $\langle(\mathbf{f}\mathbf{P}_S^m), {}^S\overline{\mathbf{P}}_\ell^m\rangle + \langle(\mathbf{f}\mathbf{P}_{S-1}^m), {}^S\underline{\mathbf{P}}_\ell^m\rangle$. Shown here are the $\log_{10}$ of the relative errors in the highest-degree coefficients.

things fall apart, as demonstrated in figure 10 which shows the errors in computing in this manner the same projections as done before in figure 9.

In fact, numerical experiments demonstrate that a small error is indeed made in computing the lowpassed modulated data, $\mathcal{L}(\mathbf{f}\mathbf{P}_S^m)$. Beyond this, the operation of lowpassing tends to blur the modulated data into the region where the lagged Legendre functions grow rapidly. Any errors here are magnified enormously when the result is projected onto the lagged Legendre functions, due to their large values at the endpoints. This can contribute significant errors in evaluating Legendre transforms by the methods we have described.

4.2. Modified Legendre decompositions and stable splitting algorithms

We have seen that the standard associated Legendre recurrence leads to difficulties in our fast algorithm for even moderate orders m as the degree of the computation becomes large. This appears to be related to the rapid growth with degree of the values of the lagged Legendre functions near the endpoints of their domain interval. The large values in this region contribute to large errors when multiplied against the part of the modulated data which blurs into this region by the lowpass operation. The inner products are corrupted beyond recovery.

The bad behavior of the lagged Legendre functions near the endpoints is not shared by the associated Legendre functions despite the fact that they satisfy similar recurrence. A critical difference lies in the initial conditions used to start that recurrence. The associated Legendre recurrence begins with initial condition $P_m^m(\cos\theta) = \sin^m\theta$; the higher-degree associated Legendre functions look like a product of this factor with the trigonometric polynomials in $\cos\theta$ which are generated by subsequent steps of the recurrence. We can view this $\sin^m\theta$ as a weighting factor applied to the minimal lag Legendre functions, i.e. $P^m_{m+l}(\cos\theta) = {}^m\overline{P}^m_l(\cos\theta)\sin^m\theta$, imposing a high-order zero at the endpoints of the domain interval and zeroing out what would otherwise be very large values there with only the lagged Legendre function.

This disparity causes real difficulties when we use the conventional decomposition of order $m \neq 0$ associated Legendre functions in the basic splitting step of the fast Legendre transform algorithm described in section 3. For the associated Legendre functions of high-degree (above some splitting degree $S > m$), the Legendre decomposition looks like:

$$\begin{aligned}\begin{pmatrix} P^m_{S+\ell}(\cos\theta) & P^m_{S+\ell-1}(\cos\theta) \\ * & * \end{pmatrix} &= {}^m\Pi^m_{S-m+\ell}(\cos\theta)\sin^m\theta \\ &= \left({}^m\Pi^m_{S-m}(\cos\theta)\ {}^S\Pi^m_\ell(\cos\theta)\right)\sin^m\theta \\ &= {}^m\Pi^m_{S-m}(\cos\theta)\sin^m\theta\ {}^S\Pi^m_\ell(\cos\theta) \\ &= \begin{pmatrix} P^m_S(\cos\theta) & P^m_{S-1}(\cos\theta) \\ * & * \end{pmatrix} {}^S\Pi^m_\ell(\cos\theta).\end{aligned}$$

We see that none of the powers of the sine weighting function are allocated to the lagged Legendre functions. When these are used in inner products, we can expect their large values to contribute errors, as discussed above.

Based on these observations, we describe and test a modification of the associated Legendre decomposition for the cases $m \neq 0$, in which the sine power weight is allocated more equitably among the various factors in the Legendre decomposition. That is, we replace the decomposition above with one of the form:

$$\begin{aligned}{}^m\Pi^m_{S-m+\ell}(\cos\theta)\sin^m\theta &= \left({}^m\Pi^m_{S-m}(\cos\theta)\quad {}^S\Pi^m_\ell(\cos\theta)\right)\sin^m\theta \\ &= {}^m\Pi^m_{S-m}(\cos\theta)\sin^{\mu_1}\theta\ {}^S\Pi^m_\ell(\cos\theta)\sin^{\mu_2}\theta\end{aligned}$$

where $\mu_1 + \mu_2 = m$.

A sample decomposition of this form is illustrated in figure 11. It is evident that these new decompositions can be chosen so as to moderate the imbalance of behavior between the factors in the basic Legendre decomposition, which was illustrated in figure 6. The idea here is that this will result in a decreased sensitivity to lowpass filter errors in computing the reduced inner products and which we saw illustrated in figure 10. As a matter of fact, with this simple modification the crippling numerical instability of our baseline algorithm is reduced enormously. Indeed when we perform a similar numerical experiment with the new decompositions we see an encouraging reduction in error, as shown in many examples in the next subsection.

This positive seeming development does indeed translate into significant improvements in the numerical reliability of higher-order Legendre transforms computed by means of the splitting trick and hence, in our fast algorithms for a wide range of problem sizes. We briefly describe the straightforward incorporation of the new decompositions into our algorithm.

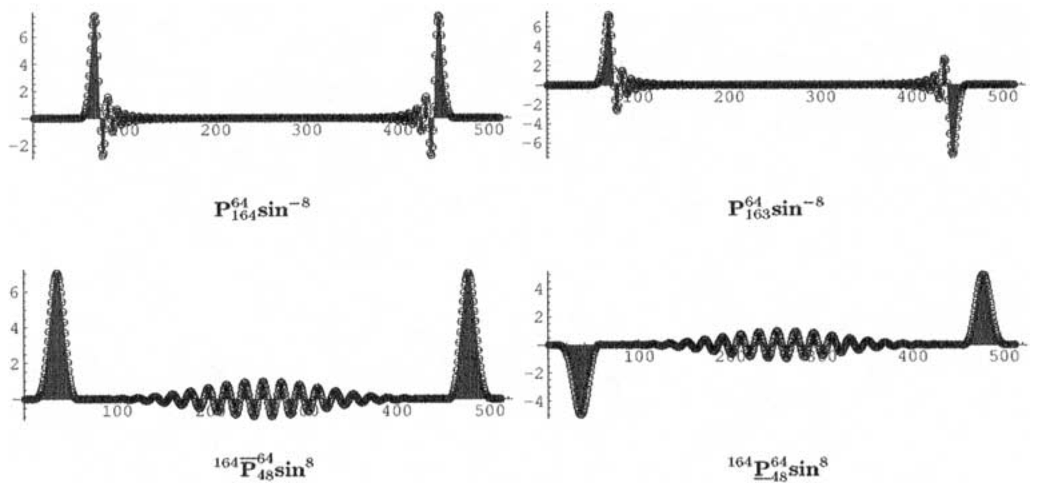

Figure 11. Factors in a modified decomposition (definition 2 with $\mu_2 = 8$) of Legendre function $\mathbf{P}^{64}_{212}$. Compare to its baseline decomposition (corresponding to $\mu_2 = 0$), presented in figure 6. Note the significant reduction in the imbalance between the factors in this modified decomposition.

We begin by extending our existing notation for the lagged Legendre matrix valued functions to include the possibility of including a weighting factor given by an appropriate power of sine.

Definition 2. For fixed order m we define the sine power-weighted Legendre functions ${}^r\mathbf{\Pi}^{m,\mu}_n$ as the matrix-valued function obtained by sampling ${}^r\Pi^m_n(\cos\theta)\sin^\mu\theta$ on our regular sampling grid, i.e.

$$ {}^r\mathbf{\Pi}^{m,\mu}_n = {}^r\mathbf{\Pi}^m_n\mathbf{sin}^\mu = \begin{pmatrix} {}^r\overline{\mathbf{P}}^m_n\mathbf{sin}^\mu & {}^r\overline{\mathbf{P}}^m_{n-1}\mathbf{sin}^\mu \\ {}^r\underline{\mathbf{P}}^m_n\mathbf{sin}^\mu & {}^r\underline{\mathbf{P}}^m_{n-1}\mathbf{sin}^\mu \end{pmatrix} $$

with our usual pointwise product convention

$$ {}^r\overline{\mathbf{P}}_\ell\mathbf{sin}^\mu = \begin{pmatrix} {}^r\overline{P}_\ell(\cos\theta_0)\sin^\mu\theta_0 \\ \vdots \\ {}^r\overline{P}_\ell(\cos\theta_{N-1})\sin^\mu\theta_{N-1} \end{pmatrix}, \qquad {}^r\underline{\mathbf{P}}_\ell\mathbf{sin}^\mu = \begin{pmatrix} {}^r\underline{P}_\ell(\cos\theta_0)\sin^\mu\theta_0 \\ \vdots \\ {}^r\underline{P}_\ell(\cos\theta_{N-1})\sin^\mu\theta_{N-1} \end{pmatrix}. $$

These functions clearly satisfy generalized decomposition:

$$ {}^r\mathbf{\Pi}^{m,\mu}_{n+k} = {}^r\mathbf{\Pi}^{m,\mu_1}_n\, {}^{r+n}\mathbf{\Pi}^{m,\mu_2}_k \tag{27} $$

for any non-negative μ_i adding to μ.

This extends our previous notation in the sense that ${}^{l}\mathbf{\Pi}_k^{m,0} = {}^{l}\mathbf{\Pi}_k^m$ and the sampled associated Legendre functions appear in this notation as (the first row of) the lag-m weighted Legendre functions:

$${}^{m}\mathbf{\Pi}_k^{m,m} = \begin{pmatrix} \mathbf{P}_{k+m}^m & \mathbf{P}_{k+m-1}^m \\ {}^{m}\underline{\mathbf{P}}_k^m \mathbf{sin}^m & {}^{m}\underline{\mathbf{P}}_{k-1}^m \mathbf{sin}^m \end{pmatrix}.$$

We now apply this sort of decomposition instead of the usual one with our usual splitting approach to fast Legendre transforms. Consider the computation of a bandwidth B order m associated Legendre transform of a vector $\mathbf{f}$ by splitting the problem at some degree S. We take $S \approx B/2 + m/2$, this being the degree which would divide the original problem into two equal sized subproblems with $(B-m)/2$ projections each. This allows us to decompose the Legendre projections into two sets of projections of roughly the same size:

$$\begin{aligned}
&\bigl\{\langle \mathbf{f}, \mathbf{P}_\ell^m\rangle \mid m \leqslant \ell < B\bigr\} \\
&\quad = \bigl\{\langle \mathbf{f}, \mathbf{P}_{m+k}^m\rangle \mid 0 \leqslant k < B-m\bigr\} \\
&\quad = \bigl\{\langle \mathbf{F}, {}^{m}\mathbf{\Pi}_k^{m,m}\rangle \mid 0 \leqslant k < B-m\bigr\}, \qquad \mathbf{F} = (\mathbf{f}, \mathbf{0}), \\
&\quad = \begin{cases} \textit{Low-degree Legendre coefficients} & \bigl\{\langle \mathbf{F}, {}^{m}\mathbf{\Pi}_\ell^{m,m}\rangle \mid 0 \leqslant \ell < S-m\bigr\} \\ \textit{High-degree Legendre coefficients} & \bigl\{\langle \mathbf{F}, {}^{m}\mathbf{\Pi}_{S-m+\ell}^{m,m}\rangle \mid 0 \leqslant \ell < B-S\bigr\}. \end{cases}
\end{aligned}$$

We now rewrite this as two low-degree Legendre transforms by means of the modified Legendre recurrence in definition 2:

$$\begin{cases} \textit{Low-degree coefficients of } \mathbf{F} & \bigl\{\langle \mathbf{F}, {}^{m}\mathbf{\Pi}_\ell^{m,m}\rangle \mid 0 \leqslant \ell < S-m\bigr\}, \\ \textit{Low-degree coefficients of } \mathbf{F}\,{}^{m}\mathbf{\Pi}_{S-m}^{m,m-m_1} & \bigl\{\langle \mathbf{F}\,{}^{m}\mathbf{\Pi}_{S-m}^{m,m-m_1}, {}^{S}\mathbf{\Pi}_\ell^{m,m_1}\rangle \mid 0 \leqslant \ell < B-S\bigr\}. \end{cases}$$

As usual, the point of this is to obtain a speed-up by computing high-degree associated Legendre coefficients as low-degree projections onto lagged Legendre functions, but the parameters of splitting degree S and sine power m_1 must be taken appropriately. The choice requires the balancing of numerical reliability against computational efficiency. On the one hand, m_1 needs to be large enough to compress the dynamic range of the lagged Legendre functions ${}^{S}\mathbf{\Pi}_\ell^{m,m_1}$ which are used in computing the high-degree inner products:

$$\bigl\langle {}^{m}\mathcal{M}_{S-m}^{m,m-m_1}\mathbf{F}, {}^{S}\mathbf{\Pi}_\ell^{m,m_1}\bigr\rangle, \qquad \ell = 0, \ldots, B-S-1, \tag{28}$$

where we have introduced the generalized notation ${}^{m}\mathcal{M}_{S-m}^{m,m-m_1}\mathbf{F}$ for $\mathbf{F}\,{}^{m}\mathbf{\Pi}_{S-m}^{m,m-m_1}$ in order to emphasize the role of modulation.

At the same time, in order to derive benefit from the split, we need to be able to compute all of the inner products in (28) with a reduced complexity concomitant with the reduced bandwidths. Our usual approach is to halve the bandwidth and the complexity, so we would like the highest bandwidth Legendre function computed, ${}^{S}\mathbf{\Pi}_{B-S-1}^{m,m_1}$, to have half the original bandwidth $B/2$. The bandwidth of this function is $m_1 + B - S - 1$, so $m_1 \leqslant S - B/2$. Therefore, increasing m_1 drives up S, the splitting degree.

In practice this will constrain m_1, as other constraints on S prevent us from taking it to be too large. For example, we need to keep to keep the subproblems roughly equal with respect to the number of projections in order to derive benefit from divide and conquer, with each accounting for half the total projections, or $(B-m)/2 \approx S-m \approx B-S$, or $S \approx B/2 + m/2$. In fact, there are even tighter constraints on S to be met. We will soon show how to choose these various parameters. First we demonstrate our motivation for investigating these decompositions.

4.3. Stabilized Legendre transforms through modified Legendre decomposition

We have conducted numerous experiments to determine the numerical accuracy of fast computation of Legendre transforms by means of the modified Legendre decompositions (27) and the modulation concepts (28) introduced in the previous subsection. We find that choosing S and m_1 appropriately enables us to compute the high-degree Legendre coefficients with halved complexity by using a lowpassed version of modulated data and a subsampled version of the degree k lagged Legendre function: $\langle \mathcal{L}({}^m\mathcal{M}_{S-m}^{m,m-m_1}\mathbf{F}), {}^S\mathbf{\Pi}_k^{m,m_1}\rangle$. This produces the desired reduction in complexity for computing the high-degree Legendre transform coefficients if the bandwidths of the various Legendre functions are within certain bounds. If this can done while choosing m_1 sufficiently large, the computation of high-degree high-order Legendre transform coefficients will also be numerically reliable. We now demonstrate this for several problem sizes of large bandwidth B, and various orders m.

Prototyping libraries were constructed in the Mathematica 4.1 and Matlab 6.1 environments for the purpose of implementing and testing the Legendre transform via the splitting approach based on various choices of the decomposition for order $m \neq 0$ Legendre functions. While the resulting variety of fast DLT algorithms are certainly all equivalent in exact arithmetic, the results of our tests show that the numerical reliability of higher-order transforms in finite precision computation is strongly dependent on the particular choice of decomposition.

In all experiments presented in this section, Matlab was used exclusively to implement the fixed precision computations of the splitting algorithm described previously. These algorithms draw upon prestored data structures comprised of the appropriate sampled Legendre functions. These latter were synthesized using the recurrence relations (27) in Mathematica, taking advantage of its extended precision capability. This was done so that the prestored data structures would be accurate to the limits of double precision used in the actual computations. These data structures were next imported into Matlab as files of double precision data floating point numbers, presumed correct to that precision. In a like fashion, the sampling weights were computed in Mathematica and imported into Matlab for use in the finite precision computations.

We also made use of Mathematica to create the test data for the various routines, again with the aim of providing inputs to Matlab accurate at double precision. We generated these test signals together with their Legendre transform coefficients as follows. For fixed bandwidth B and order m we used Mathematica to generate a list of $B-m$

(pseudo) random Gaussian coefficients $\{r_k \mid m \leqslant k < B\} = \mathbf{r}$, computed and stored in double precision. These were used as coefficients in the synthesis formula:

$$\mathbf{f} = \sum_{k=m}^{B-1} r_k \mathbf{P_k}^m \tag{29}$$

where $\mathbf{P}_k^m$ is the associated Legendre function, correctly sampled (and normalized), and the computation of the functions and of the sum was done in extended precision. The resulting input vector $\mathbf{f}$ was imported as input for the Matlab Legendre transform routines, along with the corresponding coefficients r_k for use as the "ground truth" values of the Legendre transform coefficients. Both $\mathbf{f}$ and $\mathbf{r}$ are presumed correct and consistent to double precision.

The input data $\mathbf{f}$ and prestored data structures of Legendre functions were next used in finite precision implementations of the various operations of the splitting algorithms described in the last subsection. All of these operations: i.e. modulation by Legendre functions, DCT-based low pass operations, and inner products against lagged Legendre functions, were evaluated using Matlab double precision implementations. Taken together, these routines implemented the splitting algorithms for the order m and bandwidth B Legendre transforms. The resulting computed Legendre transform coefficients were then compared to the ground truth coefficients, and the results measured by computing the base 10 logarithm of the relative error for each coefficient.

These Matlab implementations were used to perform many numerical experiments; some illustrative examples are presented in the several figures which follow. Each of these depicts the results of computing DLTs with one or two levels of splitting and all demonstrate the extremely useful effects of employing the modified Legendre decompositions. As we have a family of these modified Legendre decompositions parameterized by the sine power m_1, we present results for various values of this parameter to demonstrate the variation in numerical stability.

For each given DLT implementation, its performance is indicated by presenting the relative errors incurred when it is used to compute the Legendre transform coefficients for a randomized test input, described previously. We generally display the results of this error computation for the highest-degree Legendre transform coefficients occuring in the problem of interest, as these incur the most error in the fixed precision implementations.

Each figure presents results for a given bandwidth, order, and choice of splitting strategy. With these parameters fixed, the variation in performance with m_1 is indicated by including several subfigures indexed by the value of m_1 used in the DLT whose performance it indicates.

In figure 12 we show the results for a single split of a bandwidth $B = 512$ order $m = 64$ Legendre transform as we vary the Legendre decomposition indexed by m_1. In this particular example we compute the Legendre coefficients in the high-degree range using a single split at degree $S = 270$, resulting in the computation of the $242 = 512 - 270$ Legendre transform coefficients by the inner products

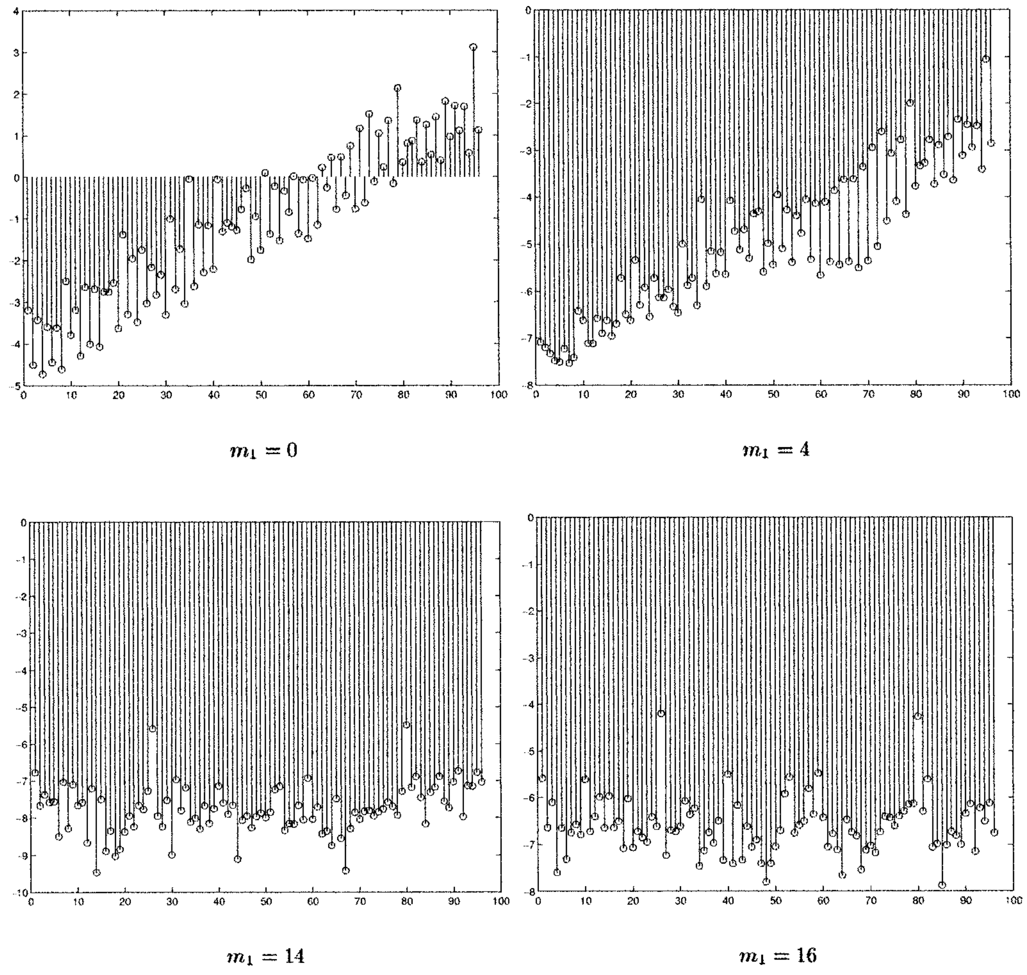

Figure 12. Errors in high-degree coefficients of $m = 64$ DLT, computed using a single split at $S = 270$ and several different Legendre decompositions. Shown here are $\log_{10}$ relative errors in the coefficients of degree 416–511.

$$\left\{\left\langle \mathcal{L}\left(\mathbf{F}^{64}\mathbf{\Pi}_{206}^{64,64-m_1}\right), {}^{270}\mathbf{\Pi}_{\ell}^{64,m_1}\right\rangle \mid 0 \leqslant \ell < 242\right\}.$$

Each one of these involves projection onto a function of bandwidth no more than 256 provided $m_1 \leqslant 14$. This justifies our use of the lowpass operator $\mathcal{L}$ to reduce the bandwidth (hence complexity) of the inner product computations from 512 to 256. The point to notice is that increasing m_1 within its allowable range has the effect of going from a situation in which the high-degree computations of Legendre transform coefficients are useless to one in which we have between 6–8 digits of precision, i.e. the relative error is of the order of 10^{-6} or less.

In figures 13–15 we give more examples of bandwidth $B = 512$ transforms for order $m = 128$ and $m = 256$, contrasting the relative errors incurred with the baseline

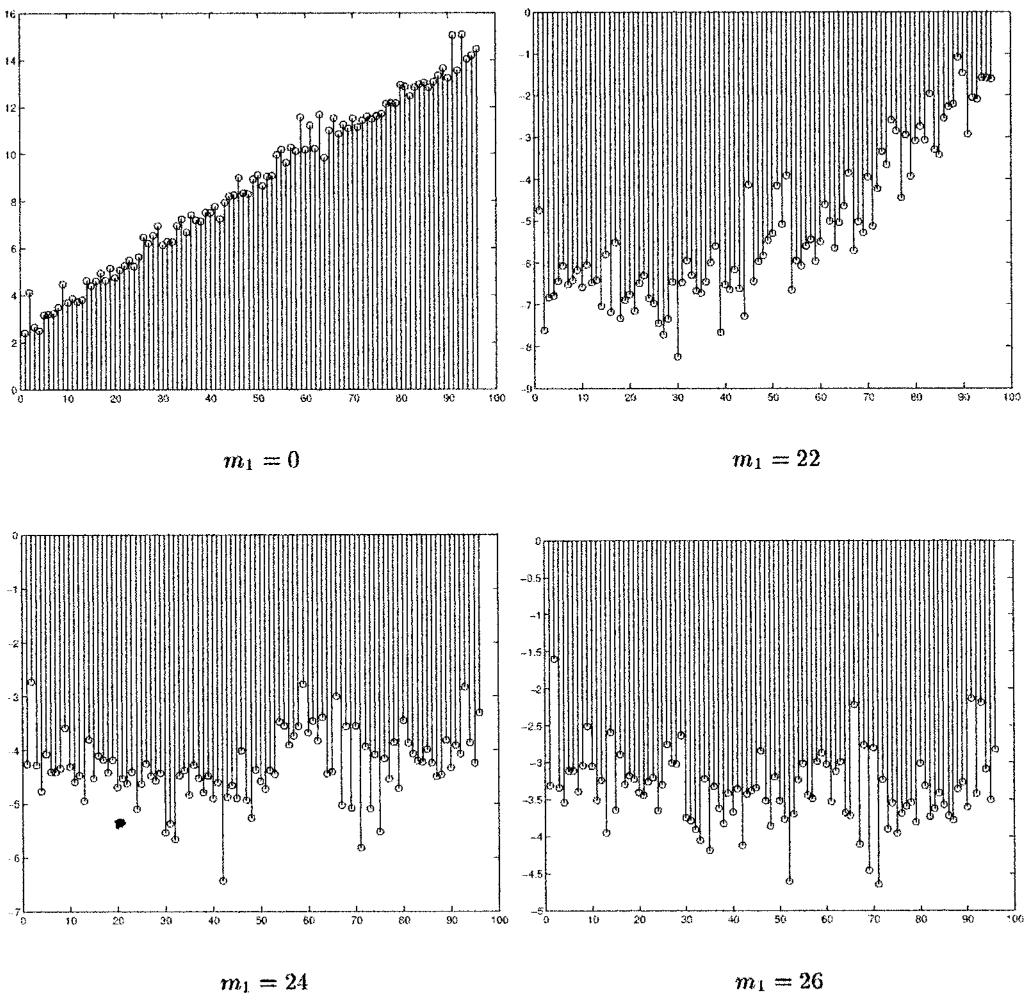

Figure 13. Similar to previous figure but for DLTs with $m = 128$. Split at $S = 300$.

Legendre decomposition to those obtained with the best Legendre decomposition. Once again, the new approach makes an enormous difference.

In figure 16 we present a bandwidth $B = 1024$, order $m = 64$ example, showing high-degree coefficients computed by two levels of splitting. The situation is similar to the previous examples, except that we now have to determine the allocation of sine powers for two levels of splitting. As figure 16 indicates, this can be accomplished by a search over the two parameters within a (small) range of candidates.

These experiments are indicative of the sort of results we have obtained. Thus, we conclude that the modified Legendre decompositions may be used to dramatically improve the numerical reliability of the fast computation of the high-degree Legendre coefficients. In the next subsection we give a few more details on how the various parameters for the split are chosen.

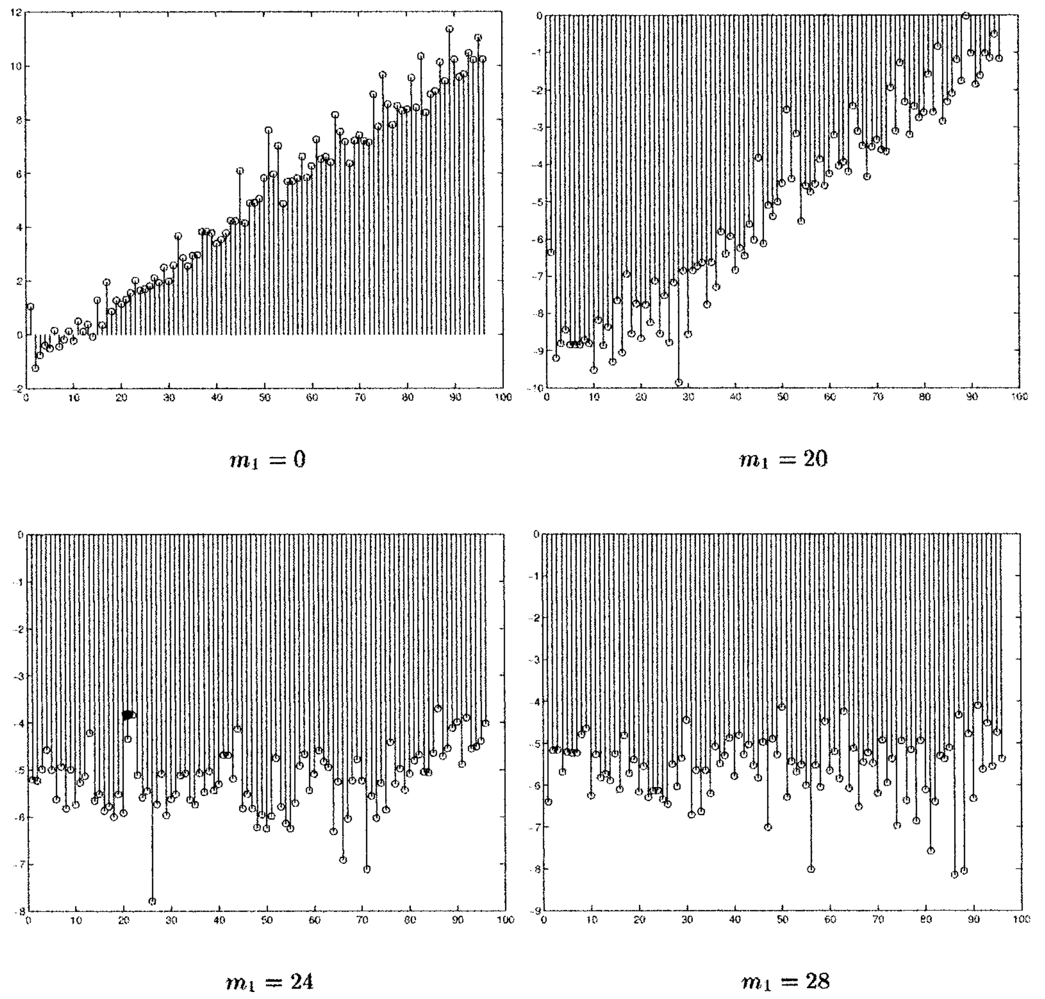

Figure 14. Similar to previous figure. DLTs with $m = 128$, but now split at $S = 320$.

4.4. Some details for fast divide-and-conquer with modified Legendre decompositions

It is quite simple to use the modified decompositions to obtain fast Legendre transforms by divide-and-conquer in essentially the same way as devised previously. This results in significant improvements in stability for a wide range of problem sizes without increased computational cost or structural complexity of the algorithm. Here we make a few necessary points on what must be done and illustrate with computational examples.

The fast algorithm takes the computation of a bandwidth B and order m associated Legendre transform of a vector $\mathbf{f}$ and splits it at some degree S, allowing us to write the set of Legendre projections as two smaller sets of projections:

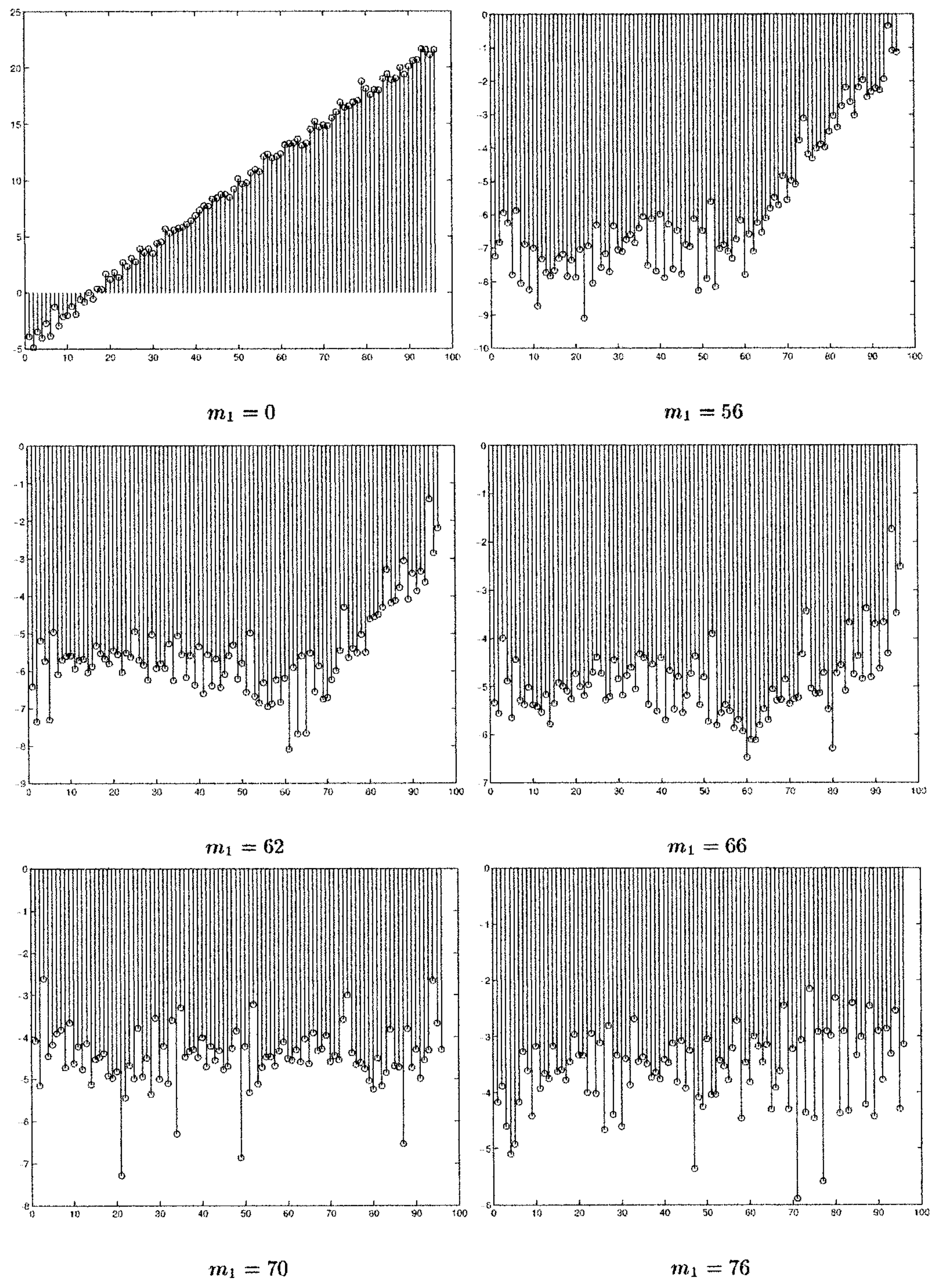

Figure 15. Similar to previous figure but for DLTs with $m = 256$. Split at $S = 380$.

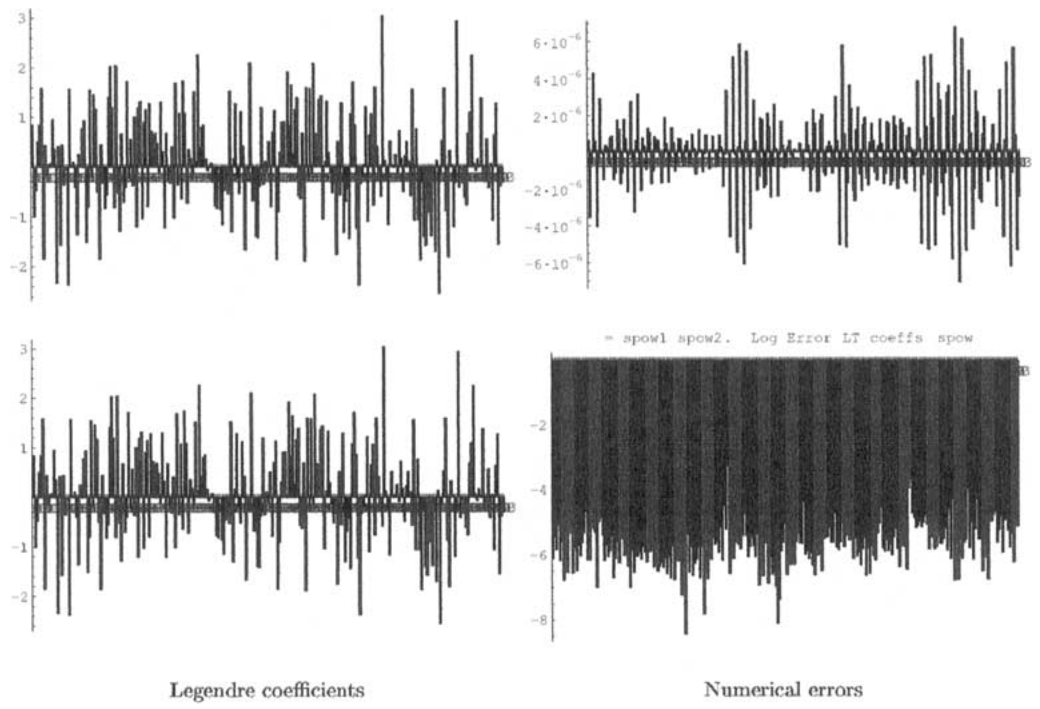

Figure 16. Double split computation of $m = 64$ DLT coefficients, computed with a split at degree 530 and then at 770, with corresponding sine powers of 52 and 8. Results shown for degrees 770–1023. Left: Legendre coefficients in this range. Computed (top) and actual (bottom). Right: finite precision effects in computing coefficients. Errors (top) and $\log_{10}$ of relative errors (bottom) in computing the coefficients.

$$
\begin{aligned}
&\left\{\langle \mathbf{f}, \mathbf{P}_\ell^m\rangle \mid m \leqslant \ell < B\right\}\\
&\quad = \left\{\left\langle \mathbf{F}, {}^m\mathbf{\Pi}_k^{m,m}\right\rangle \mid 0 \leqslant k < B - m\right\}\\
&\quad = \begin{cases} \textit{Low-degree Legendre coefficients} & \left\{\left\langle \mathbf{F}, {}^m\mathbf{\Pi}_\ell^{m,m}\right\rangle \mid 0 \leqslant \ell < S - m\right\}, \qquad \text{(L)}\\ \textit{High-degree Legendre coefficients} & \left\{\left\langle \mathbf{F}, {}^m\mathbf{\Pi}_{S-m+\ell}^{m,m}\right\rangle \mid 0 \leqslant \ell < B - S\right\} \qquad \text{(H)} \end{cases}
\end{aligned}
$$

where $\mathbf{F} = (\mathbf{f}, 0)$.

Balancing the number of projections in the two subproblems created by the split requires $S - m = (B - m)/2$, i.e. S should be close to $B/2 + m/2$. On the other hand, notice that S is the bandwidth of the highest-degree projection in the low-degree subproblem, which suggests that S should be $B/2$, as we desire to reduce both subproblems to projections onto functions of bandwidth no more than $B/2$. We will see that in practice, S must be chosen somewhere in between to permit a balance between the need to reduce the bandwidth and hence complexity of the subproblems, and the need to modify the Legendre decomposition to obtain greater numerical reliability.

Following our general approach from section 3, the next step is to use the modified Legendre recurrence in definition 2 to reduce the high-degree subproblem to low-degree projections:

$$\begin{array}{rlr} \textit{Low-degree coefficients} & & \\ \textit{of } \mathbf{F} & \left\{\left\langle \mathbf{F}, {}^{m}\mathbf{\Pi}_{\ell}^{m,m} \right\rangle \mid 0 \leqslant \ell < S - m\right\}, & \text{(L)} \\ \textit{Low-degree coefficients} & & \\ \textit{of } \mathbf{F}\,{}^{m}\mathbf{\Pi}_{\ell}^{m,m-m_H} & \left\{\left\langle \mathbf{F}\,{}^{m}\mathbf{\Pi}_{S-m}^{m,m-m_H}, {}^{S}\mathbf{\Pi}_{\ell}^{m,m_H} \right\rangle \mid 0 \leqslant \ell < B - S\right\}. & \text{(H)} \end{array}$$

We now have two parameters to specify: S and m_H. These must be chosen jointly to enable the stable computation of high-degree Legendre transform coefficients as low-degree projections, thus providing both computational speedup and numerical reliability. This, in fact, requires a trade-off of the need for numerical reliability against computational efficiency. On the one hand, m_H needs to be large enough to compress the dynamic range of the lagged Legendre functions ${}^{S}\mathbf{\Pi}_{\ell}^{m,m_H}$ in subproblem (H). On the other hand, it is also necessary to obtain reduced complexity by reducing the highest bandwidth present in that subproblem. To halve the bandwidth and hence the complexity of the projections in (H), the highest bandwidth Legendre function, ${}^{S}\mathbf{\Pi}_{B-S-1}^{m,m_H}$, must have no more than half the original bandwidth $B/2$. The bandwidth of this function is $B - S - 1 + m_H$ so that requires $m_H \leqslant S - B/2 + 1$.

What constraint is there on S? Consideration of the low-degree subproblem (L) shows that $S - B/2$ can not be overly large. Specifically, the maximum bandwidth in (L) is encountered in its last projection, onto ${}^{m}\mathbf{\Pi}_{S-m-1}^{m,m}$. This function has bandwidth $(S - m - 1) + m = S - 1$. For $S > B/2$ this is too high to permit computation without aliasing errors after lowpassing and subsampling to bandwidth $B/2$. Yet we have already seen that $S - B/2 + 1 \geqslant m_H$, and m_H must often be larger than 0 in order to permit stabilization of the high-degree subproblem.

We address this by applying the modified Legendre decomposition to the low-degree subproblem, an option always available but not previously exercised. The simplest form of this creates an equivalent set of low-degree projections by modulating the data with some of the powers of the sine, which permits corresponding demodulation of the Legendre functions, reducing their bandwidth. Specifically we have the following breakdown of the original problem:

$$\begin{cases} \left\{\left\langle \mathbf{F}\,{}^{m}\mathbf{\Pi}_{0}^{m,m_L}, {}^{m}\mathbf{\Pi}_{\ell}^{m,m-m_L} \right\rangle \mid 0 \leqslant \ell < S - m\right\}, & \text{(L)} \\ \left\{\left\langle \mathbf{F}\,{}^{m}\mathbf{\Pi}_{S-m}^{m,m-m_H}, {}^{S}\mathbf{\Pi}_{\ell}^{m,m_H} \right\rangle \mid 0 \leqslant \ell < B - S\right\}. & \text{(H)} \end{cases}$$

The band-limits of these two subproblems must each be less than or equal to $B/2$ to obtain the usual reduction in complexity through lowpassing without introducing aliasing error. That is

$$\begin{aligned} &\text{Bandlimit of } \mathbf{L}\text{:} \quad S - m + (m - m_L) < \frac{B}{2}, \\ &\text{Bandlimit of } \mathbf{H}\text{:} \quad B - S + m_H < \frac{B}{2}. \end{aligned}$$

This implies that $0 \leqslant m_H \leqslant S - B/2 + 1$ and $S - (B/2) - 1 \leqslant m_L \leqslant m$.

Moving now to stability requirements, we have seen previously that for large problems m_H should be strictly bigger than 0. Increasing m_H from 0 initially leads to a net reduction in dynamic range among the factors of the Legendre decomposition and a corresponding increase in accuracy of the high-degree Legendre transform coefficients in (H). This holds up to a point, beyond which performance decays.

On the other hand, increasing m_L increases the dynamic range of the lagged Legendre functions in subproblem (L) and is only done to squeeze that problem within the desired $B/2$ bandlimit. It should be as small as possible therefore.

This situation leads to one simple and reasonable choice of parameters. By experiment (or through the use of bounds) a good value m_H is determined, as small as is practical. Then simply take $m_L = m_H = \mu$, which makes $S = B/2 + \mu$. This permits the lowpass operation without aliasing in both subproblems, and results in the desired split of the original $B \times B$ associated Legendre transform into two $B/2 \times B/2$ transforms:

$$\left\{\left\langle \mathcal{L}\left(\mathbf{F}^m \boldsymbol{\Pi}_0^{m,\mu}\right), {}^m\boldsymbol{\Pi}_\ell^{m,m-\mu}\right\rangle \mid 0 \leqslant \ell < \frac{B}{2} + \mu - m\right\},$$

$$\left\{\left\langle \mathcal{L}\left(\mathbf{F}^m \boldsymbol{\Pi}_{B/2+\mu-m}^{m,m-\mu}\right), {}^{B/2+\mu}\boldsymbol{\Pi}_\ell^{m,\mu}\right\rangle \mid 0 \leqslant \ell < \frac{B}{2} - \mu\right\}.$$

We have verified experimentally the efficacy of this procedure. This is illustrated for some problem sizes of interest in figures 17 and 18. The computational complexity of this is as before, and this splitting procedure can be iterated to whatever depth is required for fast algorithms.

5. Summary and future directions

We have presented a divide-and-conquer algorithm for the efficient and exact computation of the discrete Legendre transform of arbitrary order. This algorithm is an extension of those presented previously in [5,11], and takes a step towards addressing constraints which can arise in the application of these earlier approaches. The fast discrete Legendre transforms enable the construction of fast forward and inverse Fourier transforms of a band-limited function on the 2-sphere. Taken together, these latter algorithms provide a fast algorithm to compute the exact convolution of two such functions. These algorithms have a wide range of potential applicability in a great range of scientific disciplines.

The approach presented in this paper provides several new parameters which can be adjusted in the process of constructing fast Legendre transforms. This provides a new approach to dealing with the numerical difficulties in the high-order Legendre transforms. This method differs from previous ones that we know of by modifying the Legendre decomposition itself. We presented experimental results for one and two levels of splitting

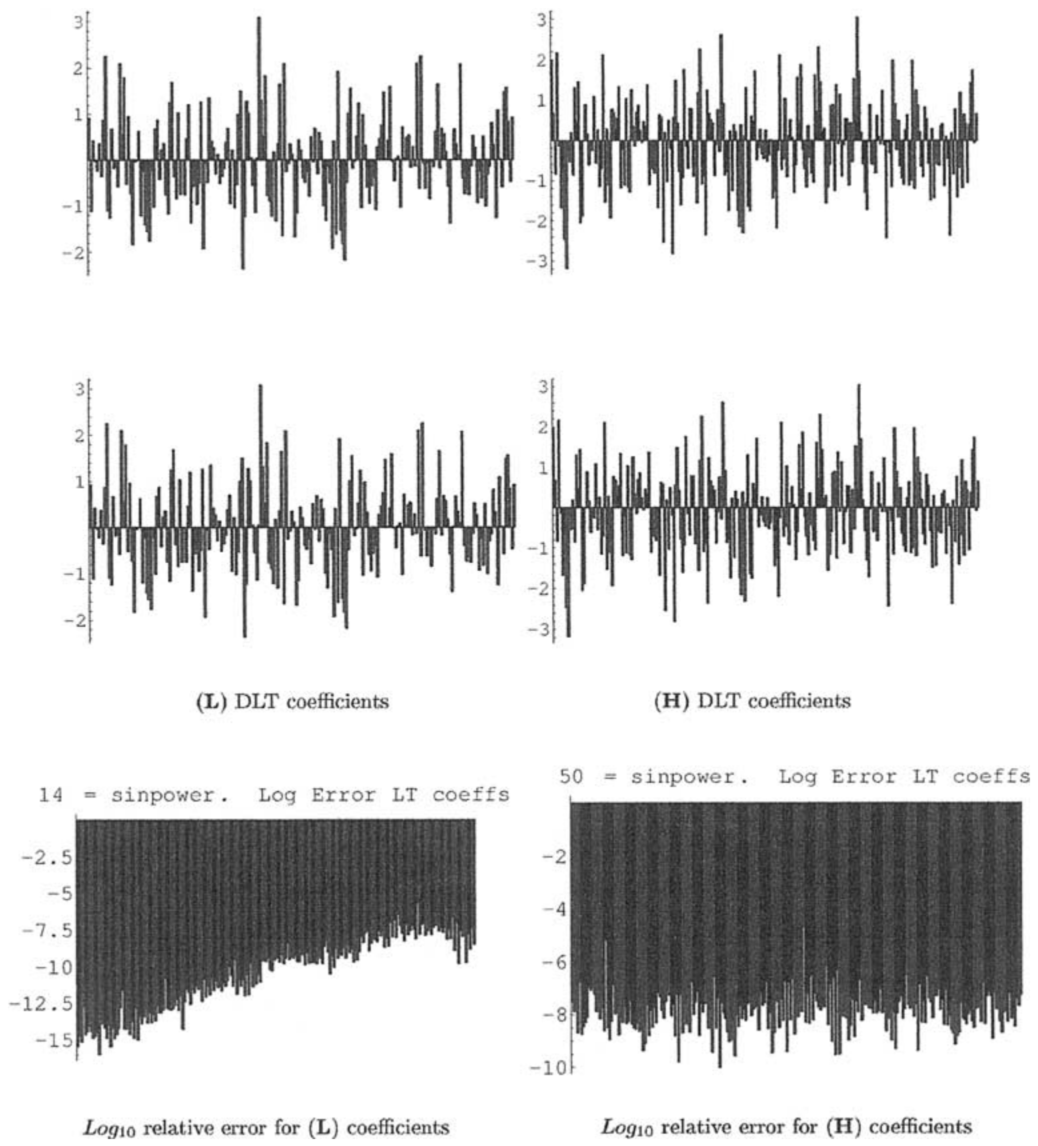

Figure 17. Computation of DLT by splitting a $BW = 512$, $m = 64$, DLT at degree $S = 270$ into (L) (left column) and (H) (right) subproblems. Here $\mu = 14$. Top row: computed coefficients. Middle row: actual coefficients. Bottom row: $\log_{10}$ of relative error for each coefficient in the range.

that suggest our new approach will make a significant improvement in the performance of our algorithms.

This approach has its limits as well, although these are much less constraining than our previous approach. We are currently considering further decompositions of the Legendre functions which decouple the behavior at the poles from that in the middle of the interval.

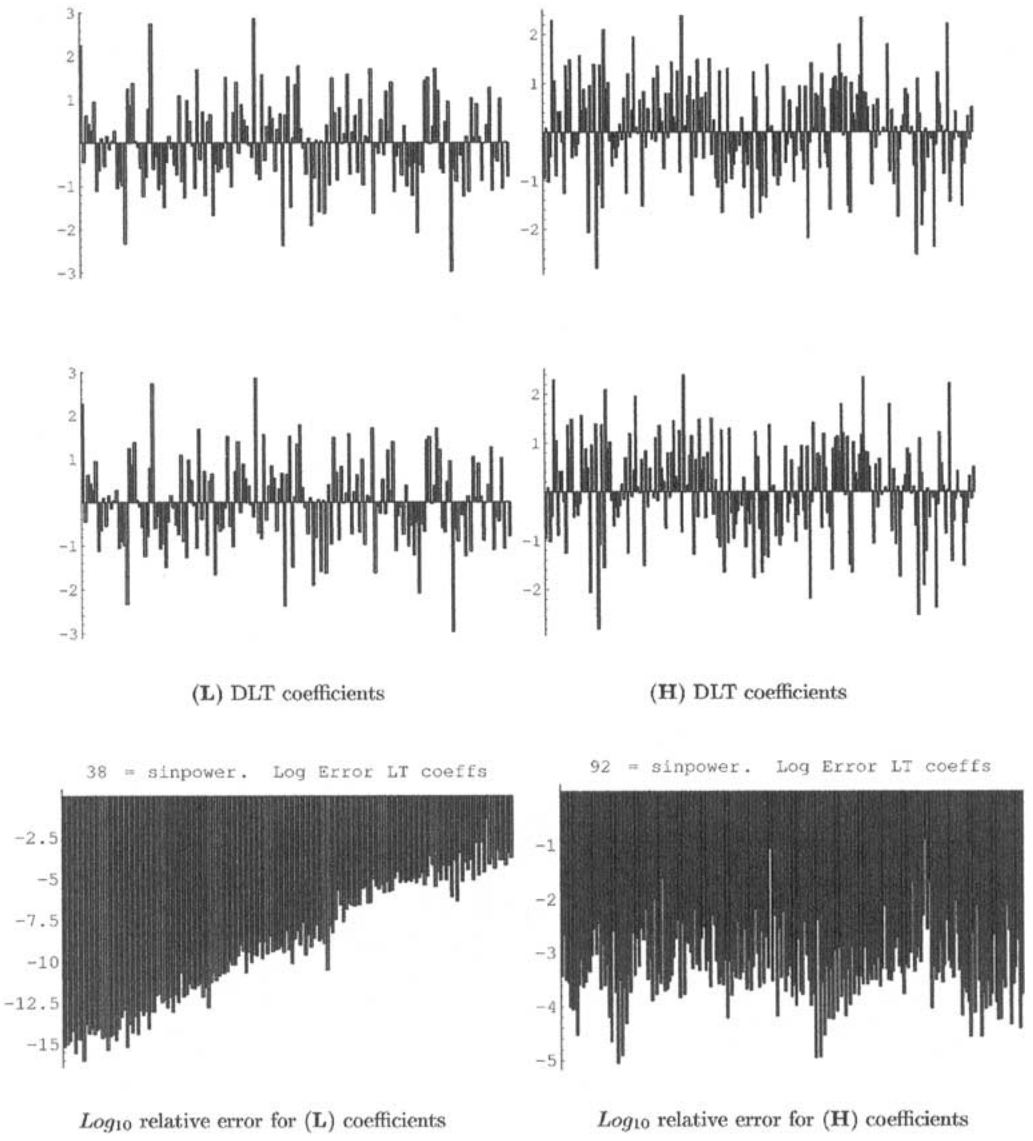

Figure 18. Computation of DLT by splitting into (L) (left column) and (H) (right) subproblems for $BW = 512$, $m = 128$. Top row: computed coefficients, middle row: actual coefficients. Bottom row: $\log_{10}$ of relative error. Here the splitting degree is 296 and $m_L = 38$, $m_H = 36$.

References

[1] G. Chirikjian and A. Kyatkin, *Engineering Applications of Noncommutative Harmonic Analysis: With Emphasis on Rotation and Motion Groups* (Lewis Pub., 2001).

[2] J.W. Cooley and J.W. Tukey, An algorithm for machine calculation of complex Fourier series, Math. Comp. 19 (1965) 297–301.

[3] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms* (MIT Press, Cambridge, MA, 1990).

[4] G.A. Dilts, Computation of spherical harmonic expansion coefficients via FFTs, J. Comput. Phys. 57(3) (1985) 439–453.

[5] J.R. Driscoll and D. Healy, Computing Fourier transforms and convolutions on the 2-sphere (extended abstract), in: *Proc. of the 34th IEEE FOCS*, 1989, pp. 344–349; Adv. in Appl. Math. 15 (1994) 202–250.

[6] J.R. Driscoll, D. Healy and D. Rockmore, Fast discrete polynomial transforms with applications to data analysis for distance transitive graphs, SIAM J. Comput. 26(4) (1997) 1066–1099.

[7] D.F. Elliott and K.R. Rao, *Fast Transforms: Algorithms, Analyses, and Applications* (Academic Press, New York, 1982).

[8] D. Healy, H. Hendriks and P. Kim, Spherical deconvolution with application to geometric quality assurance, Technical Report, Department of Mathematics and Computer Science, Dartmouth College (1993).

[9] D. Healy and P. Kim, An empirical Bayes approach to directional data and efficient computation on the sphere, Ann. Statist. 24(1) (1996) 232–254.

[10] D. Healy, S. Moore and D. Rockmore, Efficiency and stability issues in the numerical convolution of Fourier transforms and convolutions on the 2-sphere, Technical Report PCS-TR94-222, Department of Computer Science, Dartmouth College (1994).

[11] D. Healy, D. Rockmore, P. Kostelec and S. Moore, FFTs for the 2-sphere – improvements and variations, J. Fourier Anal. Appl., to appear.

[12] M.T. Heideman, D.H. Johnson and C.S. Burrus, Gauss and the history of the fast Fourier transform, Arch. Hist. Exact Sci. 34(3) (1985) 265–277.

[13] R. Jakob-Chien and B.K. Alpert, A fast spherical filter with uniform resolution, J. Comput. Phys. 136 (1997) 580–584.

[14] D. Maslen, Efficient computation of Fourier transforms on compact groups, J. Fourier Anal. Appl. 4(1) (1998) 19–52.

[15] D. Maslen, Fast transforms and sampling for compact groups, Ph.D. thesis, Department of Mathematics, Harvard University (1993).

[16] D. Maslen and D. Rockmore, Generalized FFTs, in: *Groups and Computation II*, eds. L. Finkelstein and W. Kantor, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 28 (Amer. Math. Soc., Providence, RI, 1997) pp. 183–237.

[17] M.P. Mohlenkamp, A fast transform for spherical harmonics, J. Fourier Anal. Appl. 5(2/3) (1999) 159–184.

[18] S. Moore, Efficient stabilization methods for fast polynomial transforms, Ph.D. thesis, Department of Mathematics and Computer Science, Dartmouth College (1994).

[19] S. Moore, D. Healy and D. Rockmore, Symmetry stabilization for polynomial evaluation and interpolation, Linear Algebra Appl. 192 (1993) 249–299.

[20] S. Moore, D. Healy, D. Rockmore and P. Kostelec, SpharmonicKit is a freely available collection of C programs for doing Legendre and scalar spherical transforms, Developed at Dartmouth College, available at `www.cs.dartmouth.edu/~geelong/sphere/`.

[21] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing* (Prentice-Hall, Englewood Cliffs, NJ, 1989).

[22] D. Potts, G. Steidl and M. Tasche, Fast and stable algorithms for discrete spherical Fourier transforms, Linear Algebra Appl. 275/276 (1998) 433–450.

[23] D. Rockmore, Some applications of generalized FFTs (appendix with D. Healy), in: *Groups and Computation II*, eds. L. Finkelstein and W. Kantor, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 28 (Amer. Math. Soc., Providence, RI, 1997) pp. 329–369.

[24] SPHEREPACK is a freely available collection of FORTRAN programs that facilitates computer modeling of geophysical processes developed at NCAR by J. Adams and P. Swarztrauber.

[25] W.F. Spotz and P.N. Swarztrauber, A performance comparison of associated Legendre projections, J. Comput. Phys. 168(2) (2001) 339–355.

[26] G. Steidl and M. Tasche, A polynomial approach to fast algorithms for discrete Fourier-cosine and Fourier-sine transforms, Math. Comp. 56 (1991) 281–296.

[27] P.N. Swarztrauber and W.F. Spotz, Generalized discrete spherical harmonic transforms, J. Comput. Phys. 159(2) (2000) 213–230.

[28] C. Temperton, On scalar and vector transform methods for global spectral models, Mon. Wea. Rev. 119 (1991) 1303–1307.

[29] C. Van Loan, *Computational Framework for the Fast Fourier Transform* (SIAM, Philadelphia, PA, 1992).

[30] N.J. Vilenkin, *Special Functions and the Theory of Group Representations* (Amer. Math. Soc., Providence, RI, 1968).

[31] N. Yarvin and V. Rokhlin, A generalized one-dimensional fast multipole method with applications to filtering of spherical harmonics, J. Comput. Phys. 147(2) (1998) 5594–5609.