IMA Journal of Numerical Analysis (2016) **36**, 1670–1684
doi:10.1093/imanum/drv060
Advance Access publication on November 12, 2015

A fast FFT-based discrete Legendre transform

Nicholas Hale
Department of Mathematical Sciences (Applied Mathematics), Stellenbosch University, Stellenbosch, 7600, South Africa
*Corresponding author: nickhale@sun.ac.za

and

Alex Townsend
Department of Mathematics, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139-4307, USA
ajt@mit.edu

[Received on 30 April 2015; revised on 29 September 2015]

An $\mathcal{O}(N(\log N)^2/\log\log N)$ algorithm for the computation of the discrete Legendre transform and its inverse is described. The algorithm combines a recently developed fast transform for converting between Legendre and Chebyshev coefficients with a Taylor series expansion for Chebyshev polynomials about equally spaced points in the frequency domain. Both components are based on the fast Fourier transform, and as an intermediate step we obtain an $\mathcal{O}(N\log N)$ algorithm for evaluating a degree-$(N-1)$ Chebyshev expansion on an N-point Legendre grid. Numerical results are given to demonstrate performance and accuracy.

Keywords: discrete Legendre transform; Legendre polynomials; Chebyshev polynomials; fast Fourier transform.

1. Introduction

Given N values $c_0,\dots,c_{N-1}$, the discrete Legendre transform (DLT) or finite Legendre transform calculates the discrete sums

$$f_k=\sum_{n=0}^{N-1} c_n P_n\left(x_k^{\text{leg}}\right),\quad 0\leqslant k\leqslant N-1, \tag{1.1}$$

where $P_n(x)$ denotes the degree-n Legendre polynomial and the Legendre nodes, $1>x_0^{\text{leg}}>\cdots>x_{N-1}^{\text{leg}}>-1$, are the roots of $P_N(x)$. The inverse DLT (IDLT), which computes $c_0,\dots,c_{N-1}$ given $f_0,\dots,f_{N-1}$, can be expressed as

$$c_n=\left(n+\frac{1}{2}\right)\sum_{k=0}^{N-1} w_k^{\text{leg}} f_k P_n\left(x_k^{\text{leg}}\right),\quad 0\leqslant n\leqslant N-1, \tag{1.2}$$

where $w_0^{\text{leg}},\dots,w_{N-1}^{\text{leg}}$ are the Gauss–Legendre quadrature weights. It is the goal of this paper to describe fast algorithms for computing the DLT and IDLT.

In a number of situations, the orthogonality of the Legendre polynomials with respect to the standard L^2-norm can make expansions in Legendre polynomials advantageous over the more commonly

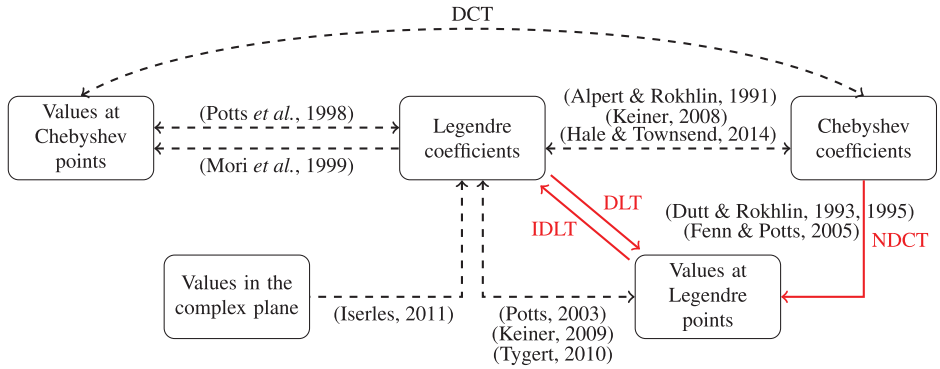

FIG. 1. Some existing fast algorithms related to the DLT and IDLT. Solid lines depict transforms discussed in this paper. The 'nonuniform discrete cosine transform' (NDCT) is described in Section 2, the DLT in Section 3 and the IDLT in Section 4. This table is not complete and some of the papers referenced belong to more than one of the connecting lines; for example, Keiner (2009) computes the DLT by combining a transformation from Legendre to Chebyshev coefficients with an NDCT. We have tried to include as many of the key papers as possible, without overcomplicating the diagram.

used Chebyshev expansions. However, because fast algorithms are not as readily available (or involve a precomputational cost that inhibits applications employing adaptive discretizations), Legendre expansions are typically considered less convenient in practice. By deriving fast algorithms for the DLT and IDLT, we take some steps towards removing this barrier and allowing Legendre expansions and adaptive discretizations at Legendre nodes to become a practical tool in computational science.

Our approach makes use of the fast Chebyshev–Legendre transform described in Hale & Townsend (2014), which is based on carefully exploiting an asymptotic expansion of Legendre polynomials and the fast Fourier transform (FFT). Using the FFT has advantages and disadvantages. On the one hand, fast and industrial-strength implementations of algorithms for computing the FFT are ubiquitous. On the other hand, the FFT is restricted to equally spaced samples and is therefore not immediately applicable to situations like computing the DLT. In this paper we overcome the equally spaced restriction by considering Legendre nodes as a perturbation of a Chebyshev grid and employing a truncated Taylor series expansion. A similar approach for the discrete Hankel transform is described in Townsend (2015a).

Examples of existing approaches that may be used or adapted to compute the DLT include butterfly schemes (O'Neil *et al.*, 2010), divide-and-conquer approaches (Keiner, 2008; Tygert, 2010) and the oversampled nonuniform discrete cosine transform (NDCT) (Potts, 2003; Fenn & Potts, 2005; Keiner *et al.*, 2009). The algorithm presented here has three main advantages: (1) it is simple (the core ingredients are just the FFT and Taylor approximations); (2) there is essentially no precomputational cost (unlike, for example, butterfly schemes in O'Neil *et al.*, 2010) and (3) there are no algorithmic parameters for the user to tweak (as they are all optimally determined by analysis). In particular, these final two points make the algorithm presented in this paper well suited to applications where N is not known in advance or when there are multiple accuracy goals. Fig. 1 shows a number of these approaches for computing the DLT and related transforms.

It is interesting to note that before the work of Glaser *et al.* (2007) there was no known stable method for computing the Legendre nodes in fewer than $\mathcal{O}(N^2)$ operations.[1] Therefore, prior to 2007, it seems

[1] The current state of the art for computing the Legendre nodes is due to Bogaert (2014), whose $\mathcal{O}(N)$ algorithm can rapidly compute millions of nodes to 16 digits of accuracy in under a second. For a discussion of the history of computing Gauss–Legendre quadrature nodes and weights, see Townsend (2015b).

likely that all DLT algorithms had a precomputational cost of at least $\mathcal{O}(N^2)$, although this is rarely discussed. It would be interesting to revisit these algorithms in the wake of these developments, but such a detailed review is beyond the scope of this article.

The paper is structured as follows. In the next section we develop an $\mathcal{O}(N \log N)$ algorithm for evaluating a Chebyshev series expansion at Legendre nodes (we refer to this as an NDCT). In Section 3 we combine this with the fast Chebyshev–Legendre transform from Hale & Townsend (2014) to obtain an $\mathcal{O}(N(\log N)^2/\log\log N)$ algorithm for computing the DLT. A similar algorithm for the IDLT is described in Section 4. Throughout we use the following notation: a column vector with entries $v_0, \ldots, v_{N-1}$ is denoted by $\underline{v}$, a row vector by $\underline{v}^{\mathrm{T}}$, a diagonal matrix with entries $v_0, \ldots, v_{N-1}$ by $D_{\underline{v}}$, and the $N \times N$ matrix with (j,k) entry $P_k(x_j)$ by $\mathbf{P}_N(\underline{x})$.

2. An NDCT

Before considering the DLT (1.1), we describe an $\mathcal{O}(N \log N)$ algorithm for evaluating a Chebyshev expansion at Legendre nodes,

$$f_k = \sum_{n=0}^{N-1} c_n T_n\left(x_k^{\mathrm{leg}}\right), \quad 0 \leqslant k \leqslant N-1. \tag{2.1}$$

This will become an important step in the DLT. Here, $T_n(x) = \cos(n \cos^{-1} x)$ is the degree-n Chebyshev polynomial of the first kind and the sums in (2.1) may be rewritten as

$$f_k = \sum_{n=0}^{N-1} c_n \cos\left(n\theta_k^{\mathrm{leg}}\right), \quad 0 \leqslant k \leqslant N-1, \tag{2.2}$$

where $x_k^{\mathrm{leg}} = \cos\left(\theta_k^{\mathrm{leg}}\right)$. If the Legendre nodes in (2.2) are replaced by the Chebyshev points of the first kind, i.e.,

$$x_k^{\mathrm{cheb_1}} = \cos\left(\theta_k^{\mathrm{cheb_1}}\right) = \cos\left(\frac{(k+1/2)\pi}{N}\right), \quad 0 \leqslant k \leqslant N-1, \tag{2.3}$$

then (2.1) can be computed in $\mathcal{O}(N \log N)$ operations via a diagonally scaled discrete cosine transform of type III (DCT-III) (Gentleman, 1972). Since (2.2) has a similar form to a DCT, but with nonequally spaced samples, θ_k^{leg}, we refer to (2.1) as an NDCT.

Our algorithm to compute the NDCT is summarized as follows: consider the transformed Legendre nodes, $\theta_0^{\mathrm{leg}}, \ldots, \theta_{N-1}^{\mathrm{leg}}$, as a perturbation of an equally spaced grid, $\theta_0^*, \ldots, \theta_{N-1}^*$, i.e.,

$$\theta_k^{\mathrm{leg}} = \theta_k^* + \delta\theta_k, \quad 0 \leqslant k \leqslant N-1,$$

and then approximate each $\cos\left(n\theta_k^{\mathrm{leg}}\right)$ term in (2.2) by a truncated Taylor series expansion about θ_k^*. If $|\delta\theta_k|$ is small (see Section 2.2), then only a few terms in the Taylor series expansion are required. Moreover, since the $\theta_0^*, \ldots, \theta_{N-1}^*$ are equally spaced, each term in the series can be computed in $\mathcal{O}(N \log N)$ operations via a DCT (see Section 2.3). The Legendre nodes, both $x_0^{\mathrm{leg}}, \ldots, x_{N-1}^{\mathrm{leg}}$ and $\theta_0^{\mathrm{leg}}, \ldots, \theta_{N-1}^{\mathrm{leg}}$, can be computed in $\mathcal{O}(N)$ operations using the algorithm described in Bogaert (2014).

2.1 *Computing an NDCT via a Taylor series expansion*

As in Hale & Townsend (2014) we find it convenient to work in the $\theta = \cos^{-1} x$ variable. There are three reasons for this.

(1) The quantities θ_k^{leg} can be computed more accurately than x_k^{leg} (Swarztrauber, 2003).

(2) The function $\cos^{-1} x$ is ill conditioned near $x = \pm 1$. In particular, since $(\mathrm{d}/\mathrm{d}x)\cos^{-1} x = (1-x^2)^{-1/2}$, $1 - (x_0^{\text{leg}})^2 = \mathcal{O}(N^{-2})$ and $1 - \left(x_{N-1}^{\text{leg}}\right)^2 = \mathcal{O}(N^{-2})$, one can expect rounding errors that grow like $\mathcal{O}(N)$ when evaluating $\cos^{-1}\left(x_0^{\text{leg}}\right)$ and $\cos^{-1}\left(x_{N-1}^{\text{leg}}\right)$. Furthermore, when evaluating $T_{N-1}\left(\cos^{-1}\left(x_0^{\text{leg}}\right)\right)$ and $T_{N-1}\left(\cos^{-1}\left(x_{N-1}^{\text{leg}}\right)\right)$, this rounding error can increase to $\mathcal{O}(N^2)$.

(3) The Taylor series of $T_n(\cos(\theta + \delta\theta)) = \cos(n(\theta + \delta\theta))$ about θ involves simple trigonometric terms and is more convenient to work with than the ultraspherical polynomials that arise in a Taylor series of $T_n(x + \delta x)$ about x.

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

$$\cos(n(\theta + \delta\theta)) = \sum_{\ell=0}^{\infty} \cos^{(\ell)}(n\theta)\frac{(\delta\theta)^\ell}{\ell!} = \sum_{\ell=0}^{\infty} (-1)^{\lfloor(\ell+1)/2\rfloor}\Phi_\ell(n\theta)\frac{(n\delta\theta)^\ell}{\ell!}, \tag{2.4}$$

where

$$\Phi_\ell(\theta) = \begin{cases} \cos(\theta), & \ell \text{ even}, \\ \sin(\theta), & \ell \text{ odd}. \end{cases}$$

If $|\delta\theta|$ is small then a good approximation to $T_n(\cos(\theta + \delta\theta))$ can be obtained by truncating this series after just a handful of terms. The following lemma determines the accuracy we can expect from taking the first L terms.

LEMMA 2.1 For any $L \geqslant 1$ and $n \geqslant 0$,

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

Proof. By the mean value form of the remainder we have

$$R_{L,n,\delta\theta} \leqslant \max_{\hat\theta\in[0,\pi]} \left|\cos^{(L)}(n\hat\theta)\right| \frac{|\delta\theta|^L}{L!} \leqslant \frac{(n|\delta\theta|)^L}{L!},$$

as required. □

For $0 \leqslant k \leqslant N-1$, we write $\theta_k^{\text{leg}} = \theta_k^* + \delta\theta_k$ and substitute the first L terms of (2.4) into (2.1) to obtain

$$f_{k,L,\delta\theta_k} = \sum_{n=0}^{N-1} c_n \left(\sum_{\ell=0}^{L-1} (-1)^{\lfloor (\ell+1)/2 \rfloor} \Phi_\ell(n\theta_k^*) \frac{(n\delta\theta_k)^\ell}{\ell!} \right) \tag{2.5}$$

$$= \sum_{\ell=0}^{L-1} (-1)^{\lfloor (\ell+1)/2 \rfloor} \frac{\delta\theta_k^\ell}{\ell!} \left(\sum_{n=0}^{N-1} (n^\ell c_n) \Phi_\ell(n\theta_k^*) \right), \tag{2.6}$$

where the second equality follows by interchanging the order of the summations.

COROLLARY 2.2 If f_k is as in (2.1) and $f_{k,L,\delta\theta_k}$ as in (2.5), then, for any $L \geqslant 1$,

$$|f_k - f_{k,L,\delta\theta_k}| \leqslant \sum_{n=0}^{N-1} |c_n| R_{L,n,\delta\theta_k} \leqslant \frac{(N|\delta\theta_k|)^L}{L!} \|\underline{c}\|_1, \quad 0 \leqslant k \leqslant N-1.$$

Proof. This follows immediately from applying Lemma 2.1 to each of the terms in (2.1). □

Hence, if we approximate $\theta_0^{\text{leg}}, \ldots, \theta_{N-1}^{\text{leg}}$ by points $\theta_0^*, \ldots, \theta_{N-1}^*$ such that

- $\max_{0 \leqslant k \leqslant N-1} |\theta_k^{\text{leg}} - \theta_k^*|$ is sufficiently small (see Section 2.2), and
- $\theta_0^*, \ldots, \theta_{N-1}^*$ is equally spaced,

then the number of terms, L, required to achieve an accuracy of $|f_k - f_{k,L,\delta\theta_k}| \leqslant \varepsilon \|\underline{c}\|_1$ is small. Moreover, the inner sums of (2.6) can be computed via discrete cosine and sine transforms (see Section 2.3), which can be computed in $\mathcal{O}(N \log N)$ via an FFT. This gives us the foundations of a fast algorithm.

2.2 *Legendre nodes as a perturbation of a Chebyshev-like grid*

The Chebyshev points of the first kind, $x_0^{\text{cheb}_1}, \ldots, x_{N-1}^{\text{cheb}_1}$, are an approximation to $x_0^{\text{leg}}, \ldots, x_{N-1}^{\text{leg}}$, which satisfy the requirements in the two bullet points above. In particular, letting $\theta_k^{\text{cheb}_1} = \cos^{-1}\left(x_k^{\text{cheb}_1}\right)$, one can readily show that (Olver *et al.*, 2010, (18.16.3))

$$|\theta_k^{\text{leg}} - \theta_k^{\text{cheb}_1}| \leqslant \frac{\pi}{2(N+1)}, \quad 0 \leqslant k \leqslant N-1.$$

However, this bound is a little weak. A better bound is given in Lemma 2.3.

Another possibility is to consider the leading-order term of the asymptotic expansion of Legendre polynomials of large degree (Stieltjes, 1890):

$$P_N(\cos\theta) \sim \sqrt{\frac{4}{\pi}} \frac{\Gamma(N+1)}{\Gamma(N+3/2)} \frac{\cos((N+1/2)\theta - \pi/4)}{(2\sin\theta)^{1/2}}, \quad N \to \infty.$$

The zeros of this leading term are

$$\theta_k^{\text{cheb}_*} = \frac{(k+3/4)\pi}{N+1/2}, \quad 0 \leqslant k \leqslant N-1,$$

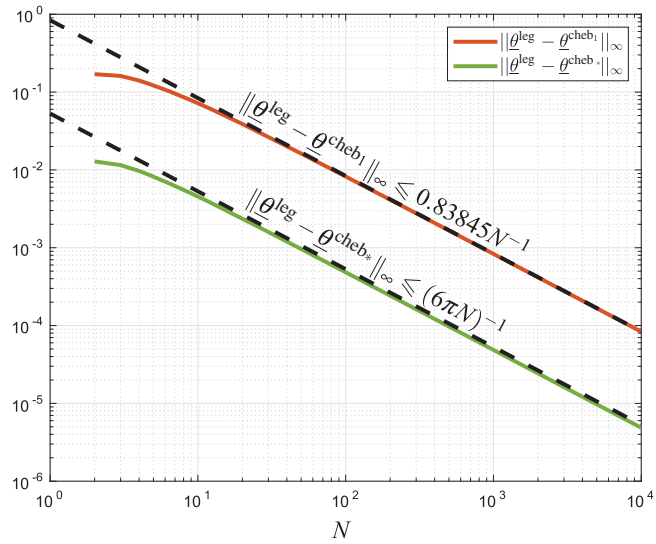

FIG. 2. The maximum distance between corresponding entries of $\underline{\theta}^{\text{leg}}$ and $\underline{\theta}^{\text{cheb}_1}$ (upper solid line) and $\underline{\theta}^{\text{leg}}$ and $\underline{\theta}^{\text{cheb}_*}$ (lower solid line) as a function of N. The dashed lines depict the bounds derived in Lemma 2.3, which appear to be tight.

which are also equally spaced and provide us with an approximation to Legendre nodes. In fact, multiplying the numerator and denominator by a factor of 2 and comparing with (2.3), we see that these points correspond precisely to the odd terms (i.e., when the index k is odd) in the $(2N+1)$-point Chebyshev grid of the first kind.

The following lemma shows how closely $\theta_k^{\text{cheb}_*}$ and $\theta_k^{\text{cheb}_1}$ approximate θ_k^{leg}, and Fig. 2 demonstrates that the derived bounds are tight.

LEMMA 2.3 Let $\theta_0^{\text{leg}}, \ldots, \theta_{N-1}^{\text{leg}}$ denote the N roots of $P_N(\cos\theta)$. Then,

$$\max_{0\leqslant k\leqslant N-1} \left|\theta_k^{\text{leg}} - \theta_k^{\text{cheb}_*}\right| \leqslant \frac{1}{3\pi(2N+1)} \leqslant \frac{1}{6\pi N}, \tag{2.7}$$

and

$$\max_{0\leqslant k\leqslant N-1} \left|\theta_k^{\text{leg}} - \theta_k^{\text{cheb}_1}\right| \leqslant \frac{3\pi^2+2}{6\pi(2N+1)} \leqslant \frac{0.83845}{N}. \tag{2.8}$$

Proof. We first consider $|\theta_k^{\text{leg}} - \theta_k^{\text{cheb}_*}|$ and restrict our attention to $0 \leqslant k \leqslant \lfloor N/2\rfloor - 1$. From Szegö (1936, Theorem 6.3.2) we know that $\theta_k^{\text{cheb}_*} < \theta_k^{\text{leg}}$. Moreover, by Brass *et al.* (1996, Theorem 3.1 & eqn. (3.2)) and $\cot(x) < 1/x$, we have

$$\theta_k^{\text{leg}} - \theta_k^{\text{cheb}_*} \leqslant \frac{1}{2(2N+1)^2}\cot\theta_k^{\text{cheb}_*} \leqslant \frac{4N+2}{2(2N+1)^2(4k+3)\pi} \leqslant \frac{1}{3(2N+1)\pi}, \quad 0\leqslant k\leqslant \lfloor N/2\rfloor - 1.$$

By symmetry of the θ_k^{leg} and $\theta_k^{\text{cheb}_*}$ the same bound holds for $k > \lfloor N/2\rfloor - 1$ when we take the modulus.

TABLE 1 *The terms multiplying* $\|\underline{c}\|_1$ *in* (2.9) *and* (2.10) *for* $1 \leqslant L \leqslant 18$. *The shaded boxes show the smallest value of* L *for which these fall below single and double precision*

L	1	2	3	4	5	6
$0.83845^L/L!$	8.4×10^{-1}	3.5×10^{-1}	9.8×10^{-2}	2.1×10^{-2}	3.5×10^{-3}	4.8×10^{-4}
$((6\pi)^L L!)^{-1}$	5.3×10^{-2}	1.4×10^{-3}	2.5×10^{-5}	3.3×10^{-7}	3.5×10^{-9}	3.1×10^{-11}
L	7	8	9	10	11	12
$0.83845^L/L!$	5.8×10^{-5}	6.1×10^{-6}	5.6×10^{-7}	4.7×10^{-8}	3.6×10^{-9}	2.5×10^{-10}
$((6\pi)^L L!)^{-1}$	2.4×10^{-13}	1.6×10^{-15}	9.2×10^{-18}	4.9×10^{-20}	2.3×10^{-22}	1.0×10^{-24}
L	13	14	15	16	17	18
$0.83845^L/L!$	1.6×10^{-11}	9.7×10^{-13}	5.4×10^{-14}	2.9×10^{-15}	1.4×10^{-16}	6.6×10^{-18}
$((6\pi)^L L!)^{-1}$	4.2×10^{-27}	1.6×10^{-29}	5.7×10^{-32}	1.9×10^{-34}	5.9×10^{-37}	1.7×10^{-39}

The bound on $|\theta_k^{\text{leg}} - \theta_k^{\text{cheb}_1}|$ follows from the triangle inequality. That is, for $0 \leqslant k \leqslant N-1$, we have

$$\left|\theta_k^{\text{leg}} - \theta_k^{\text{cheb}_1}\right| \leqslant \left|\theta_k^{\text{leg}} - \theta_k^{\text{cheb}_*}\right| + \left|\theta_k^{\text{cheb}_1} - \theta_k^{\text{cheb}_*}\right| \leqslant \frac{1}{3\pi(2N+1)} + \frac{\pi(2k-N+1)}{2N(2N+1)}.$$

Since $k \leqslant N-1$, we obtain

$$\left|\theta_k^{\text{leg}} - \theta_k^{\text{cheb}_1}\right| \leqslant \frac{1}{3\pi(2N+1)} + \frac{\pi(N-1)}{4N(N+1/2)} \leqslant \frac{3\pi^2+2}{6\pi(2N+1)} \leqslant \frac{0.83845}{N}, \quad 0 \leqslant k \leqslant N-1.$$

□

COROLLARY 2.4 Choosing $\theta_k^* = \theta_k^{\text{cheb}_1}$ or $\theta_k^* = \theta_k^{\text{cheb}_*}$ for $0 \leqslant k \leqslant N-1$ in (2.6), we have

$$\max_{0\leqslant k\leqslant N-1} |f_k - f_{k,L,\delta\theta_k^{\text{cheb}_1}}| \leqslant \frac{(0.83845)^L}{L!}\|\underline{c}\|_1 \tag{2.9}$$

or

$$\max_{0\leqslant k\leqslant N-1} |f_k - f_{k,L,\delta\theta_k^{\text{cheb}_*}}| \leqslant \frac{1}{(6\pi)^L L!}\|\underline{c}\|_1, \tag{2.10}$$

respectively.

Proof. The proof follows immediately from combining Corollary 2.2 and Lemma 2.3. □

We note that in (2.9) and (2.10) the terms multiplying $\|\underline{c}\|_1$ are independent of N, and so the number of terms required in (2.6) to obtain a given precision is bounded independently of N. Table 1 shows these terms for $1 \leqslant L \leqslant 18$. Double precision is obtained for $L \gtrsim 9$ terms when $\theta_k^* = \theta_k^{\text{cheb}_*}$ and for $L \gtrsim 17$ terms when $\theta_k^* = \theta_k^{\text{cheb}_1}$.

2.3 *Computing the discrete cosine and sine transforms*

To complete our fast algorithm for the NDCT, we require a fast way to compute the sums contained inside the parentheses of (2.6). In particular, writing (2.6) in vector form, we have

$$\underline{f}_{k,L} = \sum_{\substack{\ell=0 \\ \text{even}}}^{L-1} \frac{(-1)^{\lfloor(\ell+1)/2\rfloor}}{\ell!} D^{\ell}_{\underline{\delta\theta}} \mathrm{DCT}(\underline{c}^{[\ell]}) + \sum_{\substack{\ell=1 \\ \text{odd}}}^{L-1} \frac{(-1)^{\lfloor(\ell+1)/2\rfloor}}{\ell!} D^{\ell}_{\underline{\delta\theta}} \mathrm{DST}(\underline{c}^{[\ell]}), \tag{2.11}$$

where $[\underline{c}^{[\ell]}]_n = n^\ell c_n$, $n = 0, \ldots, N-1$. When $\underline{\theta}^* = \underline{\theta}^{\mathrm{cheb}_1}$ the DCT and DST take the form

$$\sum_{n=0}^{N-1} n^\ell c_n \cos\left(\frac{n(k+1/2)\pi}{N}\right), \quad \sum_{n=0}^{N-1} n^\ell c_n \sin\left(\frac{n(k+1/2)\pi}{N}\right), \quad 0 \leqslant k \leqslant N-1,$$

which are readily related to the DCT-III and DST-III, respectively. When $\underline{\theta}^* = \underline{\theta}^{\mathrm{cheb}_*}$, they become

$$\sum_{n=0}^{N-1} n^\ell c_n \cos\left(\frac{n((2k+1)+1/2)\pi}{2N+1}\right), \quad \sum_{n=0}^{N-1} n^\ell c_n \sin\left(\frac{n((2k+1)+1/2)\pi}{2N+1}\right), \quad 0 \leqslant k \leqslant N-1,$$

which are equivalent to the odd terms of a DCT-III and DST-III of length $2N+1$. Hence, both may be evaluated in $\mathcal{O}(N \log N)$ operations.

REMARK 2.5 Although, for an accuracy of double precision, the $\underline{\theta}^{\mathrm{cheb}_*}$ points require around half the number of terms in the Taylor series compared with $\underline{\theta}^{\mathrm{cheb}_1}$ (see Table 1), we see from above that each term will be roughly twice as expensive to evaluate. Hence, we expect little difference between the two approaches when there is a 16-digit accuracy goal.

REMARK 2.6 The numerical results in this paper are computed in MATLAB. Although MATLAB uses FFTW for FFT computations (Frigo & Johnson, 2005), it does not allow access to FFTW's DCT and DST routines (`REDFT01` and `RODFT01`, respectively).[2] Instead, we use the DCT and DST codes implemented in Chebfun (Driscoll *et al.*, 2014), which compute the DCT and DST via a complex-valued FFT of double the length. Based on our experiments, we believe this to be a factor of 2–4 slower than calling `REDFT01` and `RODFT01` directly.

REMARK 2.7 This section described a Taylor-based NDCT, which is closely related to the Taylor-based NFFT described by Kunis (2008). Alternative approaches based on oversampling and low-pass filters are described in Dutt & Rokhlin (1993) and Potts (2003). Although for general point distributions the oversampled NFFT and NDCT are considered faster than the Taylor-based approaches (Ware, 1998) we choose to use a Taylor-based NDCT because (a) it is simple and (b) Lemma 2.3 provides a precise bound on the number of terms required in the expansion for a given accuracy (see Table 1). Since the required number of terms is small, we expect the Taylor-based method to be competitive in this particular case.

[2] The MATLAB signal processing toolbox has a `dct()` function, but this is computed via an FFT of double the length.

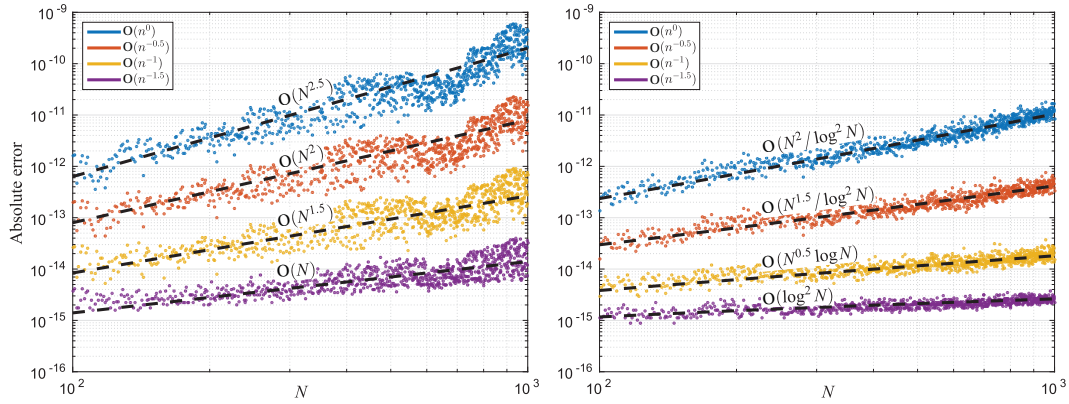

FIG. 3. Errors in computing the NDCT of vectors with various rates of decay using the direct method (left) and the new FFT-based approach with $\underline{\theta}^* = \underline{\theta}^{\text{cheb}_1}$ (right). In both cases a vector of length N is created using `randn(N,1)` in MATLAB and then scaled so that the nth entry is $\mathcal{O}(n^0)$, $\mathcal{O}(n^{-0.5})$, $\mathcal{O}(n^{-1})$ and $\mathcal{O}(n^{-1.5})$, giving the four curves in each panel. The dashed lines depict heuristic observations of the error growth in each case. The log factors may seem somewhat arbitrary, but the lines give a much better fit to the data when these are included.

2.4 *Direct approach*

To test the fast algorithm above, we construct $\mathbf{T}_N\left(\underline{x}^{\text{leg}}\right)$ via the three-term recurrence relation:

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n \geqslant 1.$$

Although this approach requires $\mathcal{O}(N^2)$ operations to compute $\mathbf{T}_N\left(\underline{x}^{\text{leg}}\right)\underline{c}$, the memory cost can easily be reduced to $\mathcal{O}(N)$ by computing the matrix–vector product as a recurrence. Hereinafter, we refer to this as the *direct approach*.

We note that one could instead compute the NDCT using the closed-form expression of $T_n(\cos\theta)$ to construct $\mathbf{T}_N\left(\underline{x}^{\text{leg}}\right) = \cos\left(\underline{\theta}^{\text{leg}}[0,1,\ldots,N-1]\right)$ and then evaluate the matrix–vector product $\underline{f} = \mathbf{T}_N\left(\underline{x}^{\text{leg}}\right)\underline{c}$. However, such a closed form is not available when we come to compute $\mathbf{P}_N\left(\underline{x}^{\text{leg}}\right)$ in the next section.

2.5 *Numerical results for the NDCT*

All the numerical results were performed on a 3.40 GHz Intel Core i7-2600 PC with MATLAB 2015a in Ubuntu Linux. The vector of transformed nodes $\underline{\theta}^{\text{leg}}$ is computed with the `legpts()` command in Chebfun (Driscoll *et al.*, 2014, v5.2), which uses Bogaert's algorithm from Bogaert (2014). As discussed in Remark 2.6, DCTs and DSTs are computed using the `chebfun.dct()` and `chebfun.dst()` routines. MATLAB codes for reproducing all of the results contained within this paper are available online (Hale & Townsend, 2015).

As a first test we take randomly distributed vectors $\underline{c}$ with various rates of decay and compare the accuracy of both the direct and FFT-based approaches against an extended precision computation (Fig. 3).[3] The results for $\underline{\theta}^{\text{cheb}_1}$ and $\underline{\theta}^{\text{cheb}_*}$ in the FFT-based approach are virtually indistinguishable so we show only the former. We find that the FFT-based approaches are more accurate than the direct

[3] In particular, the vector corresponding to, say, $N = 50$ with $\mathcal{O}(n^{-0.5})$ decay can be reproduced exactly by the MATLAB code `rng(0,'twister'); c = rand(50,1)./sqrt(1:50).';`.

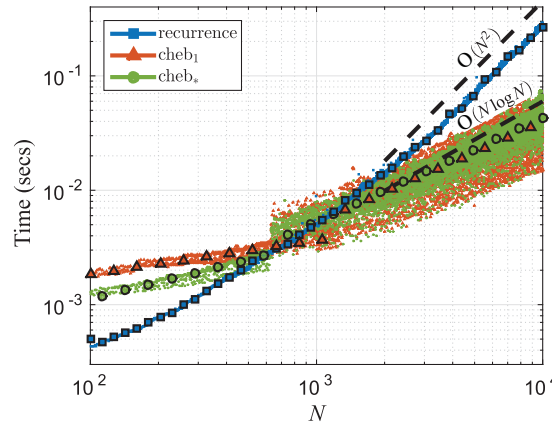

FIG. 4. Time to compute an NDCT of various lengths using the direct approach (squares) and FFT-based approaches from Section 2 with $\underline{x}^{\text{cheb}_1}$ (triangles) and $\underline{x}^{\text{cheb}_*}$ (circles). For larger values of N the two FFT-based approaches require a comparable time, and the crossover with the direct approach occurs at around $N = 1,000$. We believe the jump in the $\underline{x}^{\text{cheb}_*}$ time (circles) at around $N = 650$ is caused by a switch in algorithmic details of the FFT routine.

approach, despite our algorithm involving many significant approximations. In many applications the Chebyshev expansion in (2.1) is derived by an approximation of a smooth function. In this setting, if the function is Hölder continuous with $\alpha > 1/2$, we observe that the FFT-based algorithm has essentially no error growth with N.

Our second test investigates the time taken to compute a real-valued NDCT of length N. Figure 4 compares the direct approach (squares) and the FFT-based approach with θ^{cheb_1} (triangles) and θ^{cheb_*} (circles), in the range $100 \leqslant N \leqslant 10{,}000$. The variation of computational times between consecutive values of N is typical of FFT-based algorithms, where the theoretical cost of the FFT as well as the precise algorithm employed by FFTW depends on the prime factorization of N. The triangles and circles lie on a piecewise linear least squares fit to the computational timings, to suggest some kind of 'average' time for nearby values of N. The variation also makes it difficult to say when the FFT-based algorithm becomes more efficient than the direct approach, but it occurs around $N = 1,000$.

3. The discrete Legendre transform

Our procedure for computing the DLT in (1.1) splits naturally into two stages. The first deals with the nonuniform frequency present in $P_n(\cos\theta)$ and converts the transform to one involving $T_n(\cos\theta)$ with uniform frequency in θ. The second stage deals with the nonuniform grid $\cos\left(\theta_0^{\text{leg}}\right), \ldots, \cos\left(\theta_{N-1}^{\text{leg}}\right)$ and is the NDCT from the previous section.

Denote by M the $N \times N$ upper-triangular matrix given by

$$M_{kn} = \begin{cases} \dfrac{1}{\pi}\dfrac{\Gamma(n/2+1/2)}{\Gamma(n/2+1)}, & 0 = k \leqslant n \leqslant N-1, \quad n \text{ even}, \\ \dfrac{2}{\pi}\dfrac{\Gamma((n-k)/2+1/2)}{\Gamma((n+k)/2+1)}, & 0 < k \leqslant n \leqslant N-1, \quad k+n \text{ even}, \\ 0 & \text{otherwise}. \end{cases}$$

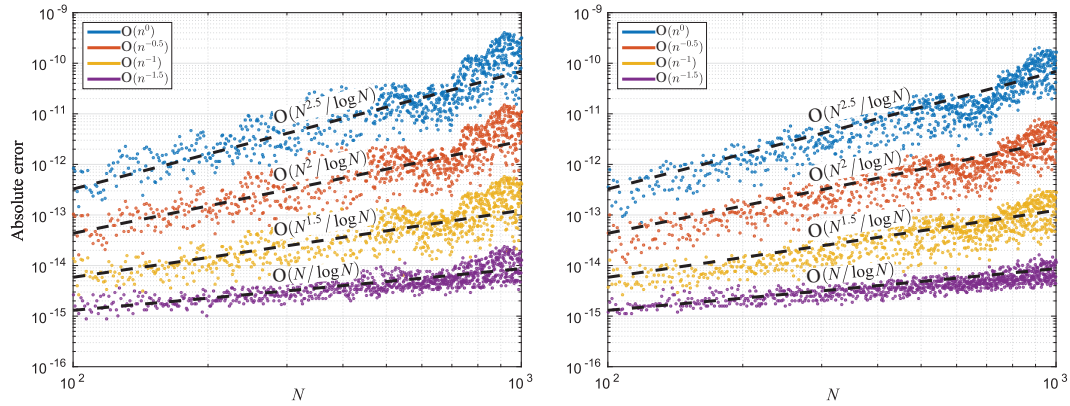

FIG. 5. Errors in computing the DLT of vectors with various rates of decay using the direct approach (left) and the FFT-based approach (right). In both cases, a vector of length N is created using `randn(N,1)` in MATLAB and then scaled so that the nth entry is $\mathcal{O}(n^0)$, $\mathcal{O}(n^{-0.5})$, $\mathcal{O}(n^{-1})$ and $\mathcal{O}(n^{-1.5})$, giving the four curves in each panel. The dashed lines depict heuristic observations of the error growth in each case.

If $\underline{c}$ is the vector of Legendre coefficients in (1.1) and $\hat{\underline{c}} = M\underline{c}$, then we have (Alpert & Rokhlin, 1991)

$$f_k = \sum_{n=0}^{N-1} c_n P_n\left(x_k^{\text{leg}}\right) = \sum_{n=0}^{N-1} \hat{c}_n T_n\left(x_k^{\text{leg}}\right), \quad 0 \leqslant k \leqslant N-1. \tag{3.1}$$

Now, the summation on the right-hand side takes the form of the NDCT (2.1) and we may use the algorithm described in Section 2.3 to compute $\underline{f}$ in $\mathcal{O}(N \log N)$ operations.

To compute $\hat{\underline{c}}$ directly, i.e., via the matrix–vector product $M\underline{c}$, requires $\mathcal{O}(N^2)$ operations. Instead, we note that the transform $\hat{\underline{c}} = M\underline{c}$ can be computed in $\mathcal{O}(N(\log N)^2/\log\log N)$ operations using the FFT-based algorithm described in Hale & Townsend (2014) or by the fast multipole-based approach in Alpert & Rokhlin (1991). In this paper, we use the algorithm in Hale & Townsend (2014) because it has no precomputational cost and so allows for adaptive discretizations.

Another convenient property of the transforms in this paper is that there is an asymptotically optimal selection of algorithmic parameters for any working tolerance. In the NDCT, the number of terms in the Taylor series expansion can be precisely determined based on the working tolerance. We would like this to also hold for the $\mathcal{O}(N(\log N)^2/\log\log N)$ Chebyshev–Legendre transform described in Hale & Townsend (2014). We remark that though the paper considered only a working tolerance of machine precision throughout, the algorithmic parameters were derived and determined in terms of $\varepsilon > 0$. We slightly modified our implementation to exploit arbitrary working tolerances.

3.1 *Numerical results for the discrete Legendre transform*

First, we take random vectors $\underline{c}$ with normally distributed entries and then scale them to have algebraic rates of decay $\mathcal{O}(n^0)$, $\mathcal{O}(n^{-0.5})$, $\mathcal{O}(n^{-1})$ and $\mathcal{O}(n^{-1.5})$, as in Section 2.5. The accuracy of both the direct and FFT-based approaches is compared against an extended precision computation. The results are shown in Fig. 5 and one can see that the errors for the direct and FFT-based approach are comparable. In most practical applications the Legendre coefficients are associated with a smooth function, so they decay at least algebraically.

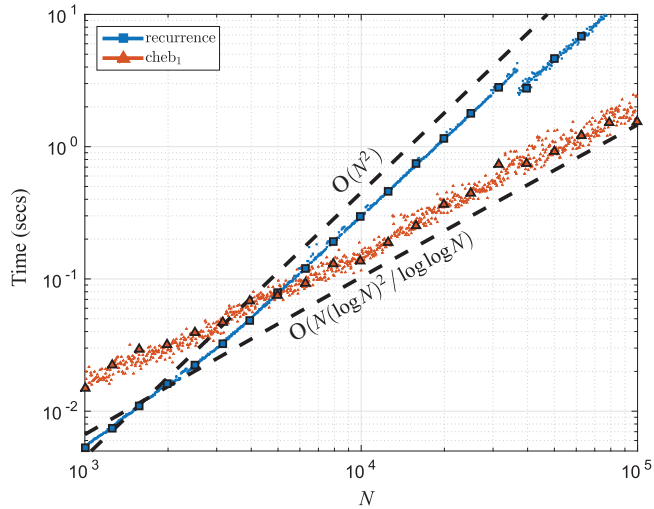

FIG. 6. Time taken to compute the DLT of length N using the direct approach (squares) and FFT-based approach (triangles). Here, we choose 1,000 logarithmically spaced values of N in the range 10^3–10^5. The larger squares and rectangles are every 50th one of these. We omit the results for the FFT-based approach using the points $\underline{x}^{\text{cheb}_*}$ as they are indistinguishable from those of $\underline{x}^{\text{cheb}_1}$.

While we achieve a similar error to the direct approach, our FFT-based approach has a lower complexity and is more computationally efficient for large N. Figure 6 shows a comparison of the computational times for the direct and FFT-based approach. When comparing with Fig. 4, we see that the computational time for the DLT is dominated by the leg2cheb transformation. For an accuracy goal of double precision, the crossover between the direct and FFT-based approach is approximately $N = 5{,}000$.

4. The inverse discrete Legendre transform

We now turn to the IDLT, which can be seen as taking values of a function on an N-point Legendre grid and returning the corresponding Legendre coefficients of the degree-$(N-1)$ polynomial interpolant. The basis of the algorithm we propose is analogous to the forward transform and has the same $\mathcal{O}(N(\log N)^2/\log\log N)$ complexity. Unlike most NFFT algorithms which resort to an iterative approach for computing the inverse transform (Dutt & Rokhlin, 1993; Anderson & Dahleh, 1996), here we take advantage of the orthogonality of Legendre polynomials, and in particular the exactness of Gauss–Legendre quadrature, to derive a direct method for the IDLT.

It is convenient to work with matrix notation rather than summations, and we denote the $N \times N$ matrix with (j,k) entries $P_k\left(x_j^{\text{leg}}\right)$ as $\mathbf{P}_N\left(\underline{x}^{\text{leg}}\right)$ and the $N \times N$ matrix with (j,k) entry $T_k\left(x_j^{\text{leg}}\right)$ as $\mathbf{T}_N\left(\underline{x}^{\text{leg}}\right)$. Using the orthogonality of Legendre polynomials and the exactness of Gauss–Legendre quadrature, the IDLT (1.2) can then be conveniently written as

$$\underline{c} = \mathbf{P}_N^{-1}\left(\underline{x}^{\text{leg}}\right)\underline{f} = D_{\underline{s}}\mathbf{P}_N^{\mathrm{T}}\left(\underline{x}^{\text{leg}}\right)D_{\underline{w}^{\text{leg}}}\underline{f}, \tag{4.1}$$

where $\underline{s} = \left(\|P_0\|_2^{-2}, \ldots, \|P_{N-1}\|_2^{-2}\right)^{\mathrm{T}}$ and $\underline{w}^{\text{leg}}$ is the vector of Gauss–Legendre quadrature weights. From (3.1) we know that, for any vector $\underline{c}$, $\mathbf{P}_N\left(\underline{x}^{\text{leg}}\right)\underline{c} = \mathbf{T}_N\left(\underline{x}^{\text{leg}}\right)\hat{\underline{c}} = \mathbf{T}_N\left(\underline{x}^{\text{leg}}\right)M\underline{c}$, and hence we have

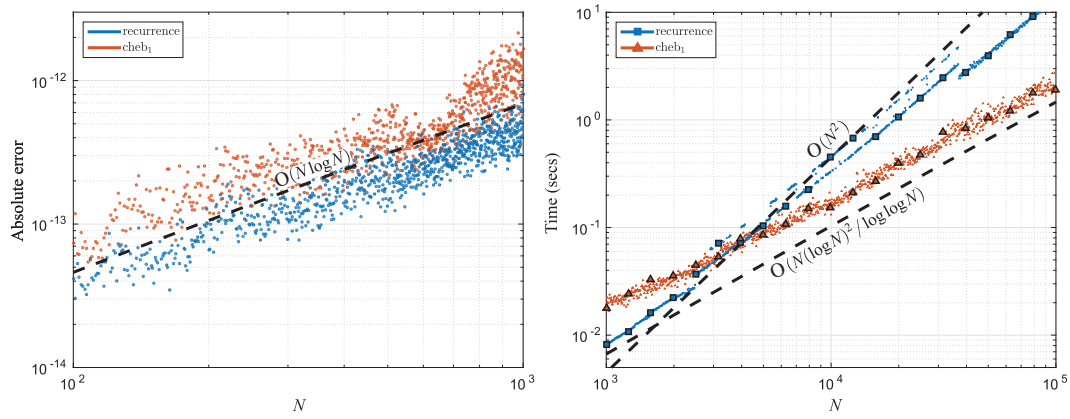

FIG. 7. Left: errors in computing the IDLT of vectors using the direct method and the FFT-based approach. A vector of length N is created using `randn(N,1)` in MATLAB with the dashed line showing the observed growth of the maximum absolute error. Right: time taken to compute the IDLT of length N using the direct approach (squares) and FFT-based approach (triangles).

the matrix decomposition $\mathbf{P}_N\left(\underline{x}^{\text{leg}}\right) = \mathbf{T}_N\left(\underline{x}^{\text{leg}}\right) M$. Substituting this into (4.1) we obtain

$$\underline{c} = D_{\underline{s}} M^{\mathrm{T}} \mathbf{T}_N^{\mathrm{T}}\left(\underline{x}^{\text{leg}}\right) D_{\underline{w}} \underline{f}.$$

Thus, $\underline{c}$ can also be computed in $\mathcal{O}(N(\log N)^2/\log\log N)$ operations for the following reasons.

(1) $D_{\underline{w}}$ and $D_{\underline{s}}$ are diagonal matrices, and so can be readily be applied to a vector in $\mathcal{O}(N)$ operations.

(2) $\mathbf{T}_N^{\mathrm{T}}\left(\underline{x}^{\text{leg}}\right)$ is precisely the transpose of the NDCT in Section 2. Since the transpose of the DCTs and DSTs in Section 2.3 can themselves be expressed in terms of DCTs and DSTs (of type II), $\mathbf{T}_N^{\mathrm{T}}\left(\underline{x}^{\text{leg}}\right)$ can be approximated in $\mathcal{O}(N\log N)$ operations by taking the transpose of (2.11).

(3) The matrix–vector product with M^{T} is closely related to the Chebyshev–Legendre transform and can be computed in $\mathcal{O}(N(\log N)^2/\log\log N)$ operations by a slight modification of the algorithm in Hale & Townsend (2014). (Alternatively, one could use the ideas in Alpert & Rokhlin, 1991.)

4.1 *Numerical results for the inverse discrete Legendre transform*

For the inverse transform, the direct approach requires $\mathcal{O}(N^2)$ operations to compute the IDLT of length N, rather than the naive $\mathcal{O}(N^3)$ standard inversion algorithm. This is because the direct approach we have implemented exploits the discrete orthogonality relation that holds for Legendre polynomials (see (4.1)). The matrix–vector product with $\mathbf{P}_N^{\mathrm{T}}\left(\underline{x}^{\text{leg}}\right)$ in (4.1) is computed by using the three-term recurrence relations satisfied by Legendre polynomials, except now along rows instead of columns.

For our first numerical experiment we take randomly generated vectors with normally distributed entries and compare the accuracy of both the direct and FFT-based approach against an extended precision computation (see Fig. 7, left). Unlike in Sections 2.5 and 3.1 we do not consider decay in these vectors, as they will typically correspond to function values in space where there is no reason to expect decay. We find that the errors incurred by our FFT-based IDLT are consistent with the direct approach.

In Fig. 7 (right) we show the execution time for computing the IDLT with the direct approach (squares) and FFT-based approach (triangles). As with the DLT, our IDLT algorithm is more computationally efficient than the direct approach when $N \geqslant 5{,}000$.

5. Conclusion

We have presented fast and simple algorithms for the discrete Legendre transform and its inverse, which rely on an NDCT and the Chebyshev–Legendre transform. Both components are based on the FFT and have no precomputational cost. For an N-point transform both algorithms have a complexity of $\mathcal{O}(N(\log N)^2/\log\log N)$ operations and are faster than the direct approach when $N \geqslant 5{,}000$. MATLAB code to compute the transformations (and all the results contained in this paper) are available online (Hale & Townsend, 2015). The codes are also accessible in Chebfun (Driscoll *et al.*, 2014) via chebfun.dlt() and chebfun.idlt(), respectively. As part of the analysis of the algorithm we derived a bound on the distance between associated points in Chebyshev and Legendre grids of the same size (see Lemma 2.3), which we believe to be tighter than any similar bounds appearing in the literature.

Acknowledgements

We thank André Weideman and Anthony Austin for reading a draft version of this manuscript, and the anonymous referees for their useful suggestions.

Funding

This material is based upon work supported by the National Science Foundation under Grant No. DMS 1522577 (Townsend) and a Subcommittee B postdoctoral fellowship from Stellenbosch University (N.H.).

References

Alpert, B. K. & Rokhlin, V. (1991) A fast algorithm for the evaluation of Legendre expansions. *SIAM J. Sci. Statist. Comput.*, **12**, 158–179.

Anderson, C. & Dahleh, M. D. (1996) Rapid computation of the discrete Fourier transform. *SIAM J. Sci. Comput.*, **17**, 913–919.

Bogaert, I. (2014) Iteration-free computation of Gauss–Legendre quadrature nodes and weights. *SIAM J. Sci. Comput.*, **36**, A1008–A1026.

Brass, H., Fischer, J.-W. & Petras, K. (1996) The Gaussian quadrature method. *Abh. Braunschweig. Wiss. Ges.*, **47**, 115–150.

Driscoll, T. A., Hale, N. & Trefethen, L. N. (2014) *Chebfun Guide*. Oxford, UK: Pafnuty Publications.

Dutt, A. & Rokhlin, V. (1993) Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Comput.*, **14**, 1368–1393.

Dutt, A. & Rokhlin, V. (1995) Fast Fourier transforms for nonequispaced data. II. *Appl. Comput. Harmon. Anal.*, **2**, 85–100.

Fenn, M. & Potts, D. (2005) Fast summation based on fast trigonometric transforms at non-equispaced nodes. *Numer. Linear Algebra Appl.*, **12**, 161–169.

Frigo, M. & Johnson, S. G. (2005) The design and implementation of FFTW3. *Proceedings of the IEEE*, **93**, 216–231. Special issue on 'Program Generation, Optimization, and Platform Adaptation'.

Gentleman, W. M. (1972) Implementing Clenshaw–Curtis quadrature II: computing the cosine transformation. *Commun. ACM*, **15**, 343–346.

Glaser, A., Liu, X. & Rokhlin, V. (2007) A fast algorithm for the calculation of the roots of special functions. *SIAM J. Sci. Comput.*, **29**, 1420–1438.

Hale, N. & Townsend, A. (2014) A fast, simple, and stable Chebyshev–Legendre transform using an asymptotic formula. *SIAM J. Sci. Comput.*, **36**, A148–A167.

HALE, N. & TOWNSEND, A. (2015) http://www.github.com/nickhale/dlt/. Accessed 30 April 2015. MATLAB code to reproduce the results in this paper.

ISERLES, A. (2011) A fast and simple algorithm for the computation of Legendre coefficients. *Numer. Math.*, **117**, 529–553.

KEINER, J. (2008) Gegenbauer polynomials and semiseparable matrices. *Electron. Trans. Numer. Anal.*, **30**, 26–53.

KEINER, J. (2009) Computing with expansions in Gegenbauer polynomials. *SIAM J. Sci. Comput.*, **31**, 2151–2171.

KEINER, J., KUNIS, S. & POTTS, D. (2009) Using NFFT 3 — a software library for various nonequispaced fast Fourier transforms. *ACM Trans. Math. Software*, **36**, Art. 19, 30.

KUNIS, S. (2008) Nonequispaced fast Fourier transforms without oversampling. *PAMM*, **8**, 10977–10978.

MORI, A., SUDA, R. & SUGIHARA, M. (1999) An improvement on Orszag's fast algorithm for Legendre polynomial transform. *Trans. Inform. Process. Soc. Japan*, **40**, 3612–3615.

OLVER, F. W. J., LOZIER, D. W., BOISVERT, R. F. & CLARK, C. W. (eds) (2010) *NIST Handbook of Mathematical Functions*. New York, NY: Cambridge University Press. Print companion to **?**.

O'NEIL, M., WOOLFE, F. & ROKHLIN, V. (2010) An algorithm for the rapid evaluation of special function transforms. *Appl. Comput. Harmonic Anal.*, **28**, 203–226.

POTTS, D. (2003) Fast algorithms for discrete polynomial transforms on arbitrary grids. *Linear Algebra Appl.*, **366**, 353–370. Special issue on structured matrices: analysis, algorithms and applications (Cortona, 2000).

POTTS, D., STEIDL, G. & TASCHE, M. (1998) Fast algorithms for discrete polynomial transforms. *Math. Comput.*, **67**, 1577–1590.

STIELTJES, T.-J. (1890) Sur les polynômes de Legendre. *Ann. Fac. Sci. Toulouse Sci. Math. Sci. Phys.*, **4**, G1–G17.

SWARZTRAUBER, P. N. (2003) On computing the points and weights for Gauss–Legendre quadrature. *SIAM J. Sci. Comput.*, **24**, 945–954.

SZEGÖ, G. (1936) Inequalities for the zeros of Legendre polynomials and related functions. *Trans. Amer. Math. Soc.*, **39**, 1–17.

TOWNSEND, A. (2015a) A fast analysis-based discrete Hankel transform using asymptotic expansions. *SINUM*, **53**, 1897–1917.

TOWNSEND, A. (2015b) The race for high order Gauss–Legendre quadrature. *SIAM News*, March.

TYGERT, M. (2010) Recurrence relations and fast algorithms. *Appl. Comput. Harmon. Anal.*, **28**, 121–128.

WARE, A. F. (1998) Fast approximate Fourier transforms for irregularly spaced data. *SIAM Rev.*, **40**, 838–856.