Accuracy Improvement of a Fast Algorithm for Discrete Legendre Polynomial Transforms

WA Gunarathna[1], HM Nasir[2]

[1]*Department of IT & Mathematics, Faculty of Engineering, General Sir John Kotelawala Defence University, Sri Lanka*

[2]*Department of Mathematics, Faculty of Science*
University of Peradeniya, Sri Lanka

[1] gunarathnawa@yahoo.com, [2] nasirhm11@yahoo.com

Abstract— Let $L=\{L_0,L_1,\ldots,L_{n-1}\}$ denote the collection of the first n Legendre polynomials, L_k, of degree k and let $X=\{x_0,x_1,\ldots,x_{n-1}\}\subset\mathbb{R}$ be a set of n sample points. The n-point discrete Legendre polynomial transform (DLT) of an input vector, $f=(f_0,f_1,\ldots,f_{n-1})\in\mathbb{R}^n$ is defined by the collection $\{\bar{L}_0,\bar{L}_1,\ldots,\bar{L}_{n-1}\}$, where $\bar{L}_j=\sum_{k=0}^{n-1}f_kL_j(x_k)$ for each $j=0,1,\ldots,n-1$. The DLT is a widely used numerical tool in a range of scientific and technical applications, including solving partial differential equations using spectral methods, molecular shape analysis , and statistical data analysis. A direct computation of the DLT (DDLT) holds a matrix-vector product of $\mathrm{P}f^{\mathrm{T}}$, where $\mathrm{P}=(L_i(x_j))_{i,j}$ for $i,j=0,1,\ldots,n-1$ and hence it requires a time complexity of $O(n^3)$. This cost gets prohibitively increased for large values of n and thus it is computationally undesirable for practical purpose. In our recent work, a fast algorithm of time complexity $O(n\log_2^2 n)$ (FDLT) has been formulated to compute the DLT efficiently. Although this algorithm is exact in hand computation, its accuracy is contaminated by lots of numerical errors when it is implemented in computer arithmetic. The purpose of this paper is to improve the accuracy of the FDLT algorithm. In the initial phase of the FDLT, it is required to compute the Vandermonde matrix vector multiplication efficiently given by $Z_0=Vf^{\mathrm{T}}$, where $V=(x_j{}^i)_{i,j}$ for $i,j=0,1,\ldots,n-1$. The available fast algorithm which is used to compute Z_0 is unstable in computer arithmetic. In this paper, instead of Vf^{T}, we apply the cosine transform, defined by Cf^{T}, where $C=(\cos(i\pi c_j/n))_{i,j}$ for $c_i\in[0,n]$ and $i,j=0,1,\ldots,n-1$. The Modified FDLT (MFDLT)has been implemented as a binary tree data structure in MATLAB and this program has been used to test its computational performances via numerical experiments. The numerical experiments demonstrate that the MFDLT permits high accuracy to be attained and that it is faster than the DDLT when $n>128$.

Keywords— Discrete Cosine Transform —DCT, Discrete Legendre Polynomial Transform—DLT, Fast Fourier Transform —FFT, Linear three term recurrence relation

I. INTRODUCTION

The Discrete Legendre Polynomial Transform, abbreviated to DLT, which is a type of polynomial transforms, is used in a wide range of scientific and technical applications, including solving partial differential equations using spectral methods, molecular shape analysis, and statistical data analysis.

The direct method in the context of the numerical computation of the DLT holds a matrix-vector product of $\mathrm{P}f^{\mathrm{T}}$, where $\mathrm{P}=(L_i(x_j))_{i,j}$ for $i,j=0,1,\ldots,n-1$, and it requires a time complexity of $O(n^3)$. This complexity is highly time-consuming for large values of n and thus it creates an undesirability for practical purposes.

In the DLT literature, several authors have made some attempts to bring down its complexity. In the subject of efficient computations of Spherical harmonic transforms , an exact algorithm was developed with time complexity of $O(n\log_2^2 n)$ to compute an n-point DLT at the natural Chebyshev nodes(Driscoll JR & Healy Jr DM.(1989; Driscoll JR & Healy DM.(1994)). Inda ((Inda et al (2001)) also proposed the efficient DLT parallel computations, while making parallel implementation of the same

algorithm. Independently, Potts et al (1998) proposed an approximate algorithm to compute efficiently a discrete orthogonal polynomial transform of size n at the Chebyshev points in $O(n\log_2{}^2 n)$ operations. Furthermore, the same authors have improved this algorithm to compute the discrete orthogonal polynomial transforms at arbitrary points ((Potts D et. Al (2003).

In our recent work (Gunarathna. W. A and Nasir. H. M(2013,), we have employed the following theorem, which was developed by Driscoll JR et al to formulate a fast algorithm (FDLT) with computational time complexity $O(n\log_2{}^2 n)$ to compute an n-point DLT efficiently at arbitrary sample points.

A. Theorem

Let $\{h_0(x), h_1(x), \ldots, h_{n\text{-}1}(x)\}$ be a collection of n functions of which the i^{th} function $h_i(x)$ is defined at $x = 0,1,\ldots,n\text{-}1$, where $n = 2^{\log_2 n}$ and satisfy the three-term recurrence relation:

$$h_{i+1}(x) = (a_i x + b_i)h_i(x) + c_i h_{i\text{-}1}(x) \qquad (1)$$

with the starting conditions $h_0(x) = 1, h_{\text{-}1}(x) = 0.$ Then the following collection:

$$\{\overline{F}(0), \overline{F}(1), \ldots, \overline{F}(n\text{-}1)\},$$

where $\overline{F}(i) = \Sigma_{j=0}^{n\text{-}1} f_i h_i(j)$ for all $i = 0,1,\ldots,n\text{-}1,$ can be computed in a time complexity of $O(n\log^2{}_2 n)$, where $f = (f_0, f_1, \ldots, f_{n\text{-}1})$ is a given fixed vector(Driscoll JR et al (1997)).

The numerical experiments confirmed that the FDLT permits high efficiency to be attained, whereas its accuracy is contaminated by lots of numerical errors. The approach of the present paper is to modify the FDLT algorithm, so that it can enable high accuracy to be attained. In the initial phase of the FDLT, it is required to compute efficiently the Vandermonde matrix vector multiplication given by $Z_0 = Vf^{\mathrm{T}}$, where $V = (x_j{}^i)_{i,j}$ for $i, j = 0,1,\ldots,n-1.$ The available fast algorithm which is used to compute Z_0 is unstable in computer arithmetic (Driscoll JR & Healy DM.(1994).; Driscoll JR et al (1997)). In this paper, instead of Vf^{T}, we try to apply the cosine transform, defined by Cf^{T}, where $C = (\cos(i\pi c_j / n))_{i,j}$ for $c_i \in [0, n]$ and $i, j = 0,1,\ldots, n-1$ to compute Z_0 (Tian, B. & Liu, Q. H. (1999)).

The organization of this paper is set to have the following structure: In Section II, we present preliminaries. Section III describes the formulation of our algorithm. Section IV is devoted to present the implementation of the algorithm and the numerical results. Section V makes comments on the numerical results. Finally, we give conclusions in Section VI.

II. PRELIMINARIES

In this section, we present the Legendre polynomials, some of their properties, discrete Legendre polynomial transforms, Cosine transforms, and some useful efficient computational techniques which are used in our work.

A. Legendre polynomials

Legendre polynomials, $L_k(x)$ are solutions of the Legendre differential equation

$$(1-x^2)\frac{d^2 p_n(x)}{dx^2} - 2x\frac{dp_n(x)}{dx} + n(n+1)p_n(x) = 0.$$

B. Orthogonality of Legendre polynomials

Let $L = \{L_0, L_1, \ldots, L_{n-1}\}$ denote the collection of the first n Legendre polynomials L_k of degree k. Then these polynomials are mutually orthogonal functions on the Hilbert space $L^2[-1,1]$ with respect to the following inner product defined by

$$\langle f, g\rangle = \int_{-1}^{1} f(x)g(x)w(x)dx,$$

where $w(x)$ is some related weight function (the weight function for the Legendre polynomials is 1) and the corresponding norm of the function f is given by

$$\|f\| = \sqrt{\langle f, f\rangle}$$

To put it another way,

$$\langle L_i, L_j\rangle = c\delta_{ij},$$

where

$$\delta_{ij} = \begin{cases} 1 & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}$$

is the Kronecker delta and c is a constant.

C. Three term recurrence relation

According to Farvard's theorem, the Legendre polynomials govern the following linear three term

recurrence relation (Chihara TS (1957); Gautschi W (1985).“Inda MA et al. (2001)):

$$L_{k+1}(x)=\frac{2k+1}{k+1}xL_k(x)-\frac{k}{k+1}L_{k-1}(x) \quad (2)$$

with stating conditions $L_0(x)=1$ and $L_{-1}(x)=0.$

D. Discrete Legendre Polynomial Transforms(DLT)

Let $L=\{L_0,L_1,\ldots,L_{n-1}\}$ denote the collection of the first n Legendre polynomials, L_k, of degree k and $X=\{x_0,x_1,\ldots,x_{n-1}\}\subset[-1,1]$ be a set of n sample points. The n-point discrete Legendre polynomial transform (DLT) of a real input vector,

$f=(f_0,f_1,\ \ldots,f_{n-1})$ is defined by the collection:

$$\{\overline{L}_0,\overline{L}_1,\ \ldots,\ \ \overline{L}_{n-1}\},$$

where

$$\overline{L}_j=\sum_{k=0}^{n-1}f_kL_j(x_k) \quad (3)$$

for all $j=0,1,\ldots,n-1.$

It can be easily seen that (3) has a matrix–vector product in the form of $\mathrm{P}f^{\mathrm{T}}$, where

$$\mathrm{P}=\begin{pmatrix}L_0(x_0) & L_1(x_0) & \cdots & L_{n-1}(x_0)\\ L_0(x_1) & L_1(x_1) & \cdots & L_{n-1}(x_1)\\ \vdots & \vdots & \cdots & \vdots\\ L_0(x_{n-1}) & L_1(x_{n-1}) & \cdots & L_{n-1}(x_{n-1})\end{pmatrix},f^{\mathrm{T}}=\begin{pmatrix}f_0\\ f_1\\ \vdots\\ f_{n-1}\end{pmatrix}.$$

The direct computation of this matrix –vector product holds a time complexity of $O(n^3)$ and hence it is not computationally desirable.

E. Discrete Fourier Transform(DFT)

The $n-$DFT of an input vector $(f_0,f_1,\ldots,f_{n-1})\in\mathbb{C}^{\mathrm{n}}$ is defined by the collection:

$$\{F_0,F_1,\ldots,F_{n-1}\},$$

where

$$F_k=\sum_{j=0}^{n-1}f_je^{-i2\pi jk/n}\ \text{for all}\ k=0,1,\ldots,n-1.$$

(Cooley JW. & Tukey JW (1965))

F. Inverse Discrete Fourier Transform (IDFT)

The n-point IDFT of the DFT is defined by the collection $\{f_0,f_1,\ldots,f_{n-1}\}$, where

$$f_j=\frac{1}{n}\sum_{k=0}^{n-}f_ke^{-i2\pi jk/n}\ \text{for all}\ j=0,1,\ldots,n-1.$$

The direct computation of either DFT or IDFT requires a time complexity of $O(n^2)$.

G. Fast computation of the DFT and IDFT

The n-DFT can be computed efficiently in a time complexity of $O(n\log_2 n)$ using the Fast Fourier Transform (FFT) (Cooley JW. & Tukey JW (1965); James S. Walker (1996), Heideman MT (1985)) and so can be the IDFT.

H. Cyclic Convolution

A sequence $a=(x_i)$ is called n-periodic if $a_{n+j}=a_j$ for all $j=0,1,2,\ldots.$

Let $a=(a_i),\ b=(b_i)$ be two n-periodic sequences of numbers. Then their cyclic convolution is defined by the sequence $c=(c_i)$, where

$$c_i=\sum_{m=0}^{n-1}a_mb_{i-m}$$

Notice that c_i is also denoted by $a*b_i$.

(James S. Walker(1996)).

I. Theorem: Fast Computation of Cyclic Convolution

Let $a=(a_i),b=(b_i)$ be two n-periodic sequences of numbers such that their n-point DFTs are given by $\{A_0,A_1,\ldots A_{n-1}\},\{B_0,B_1,\ldots,B_{n-1}\}$, respectively. Then the n-point DFT of the sequence $c=(a*b_i)$ is given by

$$\{A_0B_0,A_1B_1,\ldots,A_{n-1}B_{n-1}\}.$$

We ignore the proof of this theorem, but the interested readers are referred to James S. Walker (1996).

From this theorem, it can be quickly seen that

$$c=(a*b_i)=\mathrm{IDFT}(\{A_0B_0,A_1B_1,\ldots,A_{n-1}B_{n-1}\})$$

$$=\mathrm{IDFT}(\mathrm{DFT}(a).*\mathrm{DFT}(b)),$$

where '.*' operates the element-wise multiplication of $\mathrm{DFT}(a)$ and $\mathrm{DFT}(b)$.

Now this concludes that the cyclic convolution can be computed efficiently using only three FFTs, namely one IFFT and two FFTs.

J. Circulant Matrix

An $n\times n$ circulant matrix has the following form:

$$C := \begin{pmatrix} c_0 & c_1 & \cdots & c_{n-1} \\ c_{n-1} & c_0 & \cdots & c_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & \cdots & c_0 \end{pmatrix}.$$

K. Theorem: Fast: Circulant matrix-vector multiplication

The product $z = Cy$, of an $n \times n$ circulant matrix C and a column vector of size n can be performed in a time complexity of $O(n \log_2 n)$ (Tang Z.et al .(2004)).

It should be noted that the circulant matrix–vector product can be written as the cyclic convolution of the sequences $c = (c_0, c_1, \ldots, c_{n-1})$ and $y = (y_0, y_1, \ldots, y_{n-1})$ such that

$$z_i = y * c_i = \sum_{m=0}^{n-1} y_m c_{i-m}.$$

T Therefore, the circulant matrix–vector product can be computed efficiently using the technique of the fast computation of cyclic convolution.

L. Non-Uniform Discrete Cosine Transform

Let $\{x_0, x_1, \ldots, x_{n-1}\}$ be a set of non-uniform points in $[0, n]$. Then, the n- point Discrete Cosine Transform (DCT) of an input vector $c = (c_0, c_1, \ldots, c_{n-1})$ is defined by the collection $\{C_0, C_1, \ldots, C_{n-1}\}$, where

$$C_j = \sum_{k=0}^{n-1} c_k \cos\left(\frac{j\pi x_k}{n}\right) \text{ for all } j = 0, 1, \ldots, n-1..$$

If $x_k = k$, then $C_j = \sum_{k=0}^{n-1} c_k \cos\left(\frac{j\pi x_k}{n}\right)$,

This type of discrete cosine transform is called the uniform discrete cosine transform (UDCT)(James S. Walker (1996)).

M. Fast computation of uniform (UDCT).

$$C_j = \sum_{k=0}^{n-1} c_k \cos\left(\frac{j\pi k}{n}\right) = \mathrm{Re}.\left(\sum_{k=0}^{n-1} c_k e^{-i2\pi jk/2n}\right).$$

This is the DFT of the vector c with length $2n$ with n zeros padded and hence the UDCT can be computed efficiently using the FFT.

III. Formulation of accurate FDLT Algorithm (MFDLT))

Let x be a typical sample point in X
Then from (2), we have

$$L_{l+1}(x) = a_l L_l(x) + b_l L_{n-1}(x), \quad (3)$$

where $a_l = \frac{2l+1}{l+1}$, $b_l = -\frac{l}{l+1}$.

Let $Y = \{y_0, y_1, \ldots, y_{n-1}\}$ be a set of real numbers defined by the relation:

$$\cos\left(\frac{\pi}{n} y_j\right) = x_j \quad (4)$$

for all $j = 0, 1, \ldots, n-1$.
Then (4) gives

$$y_j = \frac{n}{\pi} \cos^{-1}(x_j) \text{ for } j = 0, 1, \ldots, n-1.$$

This is the novelty of our algorithm.
Let $Z_l = (Z_l(n-1), \ldots, Z_l(0), \ldots, Z_l(-n))^{\mathrm{T}}$ be a vector defined for each $l = 0, 1, \ldots, n-1$ by

$$Z_l(k) = \left\langle f, \cos\left(\frac{\pi k y}{n}\right) L_l \right\rangle = \sum_{j=0}^{n-1} f_j \cos\left(\frac{\pi k y_j}{n}\right) L_l \quad (5)$$

for all $k = -n, -n+1, \ldots, n-1$.
Now,

$$Z_{l+1}(k) = a_l \left\langle f, \cos\left(\frac{\pi k y}{n}\right) \cos\left(\frac{\pi y}{n}\right) L_l \right\rangle + b_l Z_{l-1}(k) \quad (6)$$

Also, we have the following trigonometric identity:

$$\cos\left(\frac{\pi k y}{n}\right) \cos\left(\frac{\pi y}{n}\right) = \frac{1}{2}\left(\cos\left((k+1)\frac{\pi y}{n}\right) + \cos\left((k-1)\frac{\pi y}{n}\right)\right) \quad (7)$$

The combined result of (5) and (6) yields the following recurrence:

$$Z_{l+1}(k) = u_l Z_l(k+1) + v_l Z_l(k-1) + w_l Z_{l-1}(k), \quad (8)$$

where $u_l = v_l = a_l / 2$ and $w_l = b_l$.
Now, we employ the idea presented in Driscoll (1997) to compute Z_l for $l = 0, 1, 2, \ldots, n-1$.
Our aim is to compute $Z_l(0)$ for all $l = 0, 1, \ldots, n-1$.
We will compute $Z_l(0)$ in three phases:

1) Initial phase : Computation of Z_0.

Equation (5) computes Z_0, where

$$Z_0(k) = \sum_{j=0}^{n-1} f_j \cos\left(\frac{\pi k y_j}{n}\right).$$

Now, we have $\overline{L}_0 = Z_0(0)$.

2) Intermediate phase: Computation of Z_1.
We get from (7)

$$Z_l(k) = u_0 Z_0(k+1) + v_0 Z_0(k-1)$$

for all $k = 0,1,\ldots,n-1$. From the equation, we can calculate Z_1 with an extra $3n$ operations and we then have $L_1 = Z_1(0)$.

3) Final phase: Computation of Z_l for $l = 1,2,\ldots,n-1$.

(8) can be rewritten in the following matrix form :

$$Z_{l+1} = A(u_l, v_l)Z_l + w_l I Z_{l-1}, \quad (9)$$

where

$$A(u_l, v_l) = \begin{pmatrix} u_l & 0 & v_l & 0 & \ldots & 0 & 0 \\ 0 & u_l & 0 & v_l & \ldots & 0 & 0 \\ 0 & 0 & u_l & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ldots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & u_l & 0 & v_l \\ v_l & 0 & 0 & \ldots & 0 & u_l & 0 \end{pmatrix}$$

is a $2n \times (2n+2)$ rectangle matrix and I denotes a $2n \times 2n$ identity matrix.

Let W_l be a vector defined by

$$W_0 = Z_0$$
$$W_1 = Z_1$$
$$W_{l+1} = C(u_l, v_l)Z_l + w_l I Z_{l-1}, \quad (10)$$

where $C(u_l, v_l)$ is a $2n \times 2n$ circulant matrix defined by

$$C(u_l, v_l) = \begin{pmatrix} 0 & v_l & 0 & \ldots & 0 & 0 & u_l \\ u_l & 0 & v_l & \cdots & 0 & 0 & 0 \\ 0 & u_l & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & u_l & 0 & v_l \\ v_l & 0 & 0 & \cdots & 0 & u_l & 0 \end{pmatrix}.$$

Then it is not difficult to see that for $s = 1,2,\ldots,$

$$\begin{pmatrix} W_l \\ W_{l+1} \end{pmatrix} = \Omega(l)\Omega(l-1)\ldots\Omega(l-s)\begin{pmatrix} Z_{l-s-1} \\ Z_{l-s} \end{pmatrix} \quad (11)$$

where $\Omega(j)$ is a $4n \times 4n$ matrix defined by

$$\Omega(j) = \begin{pmatrix} O & I \\ w_j I & C(u_j, v_j) \end{pmatrix} \text{ for } j = 1,2,\ldots,$$

and O and I denote the $2n \times 2n$ zero matrix and identity matrix, respectively.

Also, from (9) and (11), it can be easily shown that $W_{l+s-1}(j) = Z_{l+s-1}(j)$ and $W_{l+s}(j) = Z_{l+s}(j)$ for all

$j = -(n-s-1),\ldots,0,\ldots,(n-s-1).$

Let $R(p,q) = \Omega(q)\Omega(q-1)\cdots\Omega(p+1)$, where p and q are non-negative integers such that $q = p + s$.

Then (11) gives the following equation

$$\begin{pmatrix} W_{n/2} \\ W_{n/2+1} \end{pmatrix} = R\left(0, \frac{n}{2}\right)\begin{pmatrix} Z_0 \\ Z_1 \end{pmatrix} \quad (12)$$

The (12) yields that

$W_{n/2}(j) = Z_{n/2}(j)$ and that $W_{n/2+1}(j) = Z_{n/2+1}(j)$ for $j = -(n/2-1),\ldots,0,\ldots,(n/2+1).$

This means that we can correctly calculate $Z_{n/2}(j)$ and $Z_{n/2+1}(j)$ from $Z_0(j)$ and $Z_1(j)$ for

$$j = -(n/2-1),\ldots,0,\ldots,(n/2+1).$$

In particular, we then have $Z_{n/2}(0)$ and $Z_{n/2+1}(0)$.

In this case, since $R(0, n/2)$ is comprised of four $2n \times 2n$ circulant matrices, the product on the right of (12) can be computed efficiently by operating four fast circulant matrix vector multiplications. To save number of operations for the remainder of the computation, we wish to exploit the Divide and Conquer algorithm: we first split the entire problem into two sub problems of size n/2 each. Also, It is worthwhile to note that the outer upper and lower quarters of W_0 and W_1 do not affect the computation of $Z_l(0)$ for $l < n/2$ and those of $Z_{n/2}$ and $Z_{n/2+1}$ do not affect the computation of $Z_l(0)$ for $l > n/2+1$. Therefore, we proceed the second stage by throwing away those quarters from $Z_0, Z_1, Z_{n/2},$ and $Z_{n/2+1}$. Now from (11), we calculate $W_{n/4}, W_{n/4+1}, W_{3n/4},$ and $W_{3n/4+1}$ as follows:

$$\begin{pmatrix} W_{n/4} \\ W_{n/4+1} \end{pmatrix} = R\left(0, \frac{n}{4}\right)\begin{pmatrix} Z_0 \\ Z_1 \end{pmatrix} \quad (13)$$

$$\begin{pmatrix} W_{3n/4} \\ W_{3n/4+1} \end{pmatrix} = R\left(\frac{n}{2}, \frac{3n}{4}\right)\begin{pmatrix} Z_0 \\ Z_1 \end{pmatrix} \quad (14)$$

Then, (13) and (14) give $Z_{n/4}(0), Z_{n/4+1}(0), Z_{3n/4}(0),$ and $Z_{3n/4+1}(0)$. At the end of the second stage, we have only two identical sub problems of size $n/2$ and then the third stage is conducted by again splitting each of them into two sub problems of size $n/4$. This means that we have four identical sub problems of size $n/4$ each. Now applying the same method employed in the second stage, we calculate $Z_{n/8+i}(0), Z_{3n/8+i}, Z_{5n/8+i},$ and $Z_{7n/8+i}$ for $i = 0,1$, from (11). Now, keeping on running the same

procedure up to the problem size of 4, we can accomplish the entire computation. The reader may find further details regarding this and the efficient computation of the storage of this algorithm in (Driscoll JR(1997); Gunarathna. W. A and Nasir. H. M(2013)).

III. IMPLEMENTATION AND NUMERICAL RESULTS

MATLAB implementation of our algorithm (MFDLT)has been written in MATLAB Binary Tree Data Structure. We also used the fft () function in MATLAB to compute DFT. This section shows numerical results of the numerical experiments, which have been carried out, in order to test the performances of the algorithm. All the computations were performed on a personal computer with Intel(R) Pentium 2.1 GHz processor, with 2.00 GB RAM, 64 bit Windows 7 operating system using MATLAB version 12 codes.

A. Numerical Example Experiments

1) Compute the DLT defined by

$$\overline{L}_j = \sum_{k=0}^{n-1} f_k L_j(x_k)$$

for $j = 0,1,\ldots,n-1$. In this experiment, the samples $X = \{x_0, x_1, \ldots, x_{n-1}\}$ are randomly chosen from the interval $[0,1]$ and so are the input vectors $f = (f_0, f_1, \ldots, f_{n-1})$ from [0,1].

Further, we consider the brute force computation of the initial phase

$$Z_0(k) = \sum_{j=0}^{n-1} f_j \cos\left(\frac{\pi k y_j}{n}\right) \text{ for } k = 0,1,\ldots,n-1.$$

2) We compute the DLT defined by

$$\overline{L}_j = \sum_{k=0}^{n-1} f_k L_j(x_k) \text{ for } j = 0,1,\ldots,n-1.$$

In this experiment, the samples $X = \{x_0, x_1, \ldots, x_{n-1}\}$ are uniformly distributed in the interval, $[0,1]$, where $x_i = \cos(i\pi/n)$ for each $i = 0,1,\ldots,n-1$ and the input vectors $f = (f_0, f_1, \ldots, f_{n-1})$ are randomly chosen from the interval $[0,1]$. We compute the initial phase given below using the fast cosine (uniform) transform.

$$Z_0(k) = \sum_{j=0}^{n-1} f_j \cos\left(\frac{\pi k y_j}{n}\right) \text{ for } k = 0,1,\ldots,n-1.$$

In both experiments, the accuracy of the algorithm is tested by computing relative errors (RE) with respect to the following formulae:

$$RE_\infty = \frac{\max\limits_{0\le i\le n-1} \left| z_i - z_i^* \right|}{\max\limits_{0\le i\le n-1} \left| z_i^* \right|} \quad (15)$$

$$RE_2 = \sqrt{\frac{\sum\limits_{i=0}^{n-1} \left| z_i - z_i^* \right|^2}{\sum\limits_{i=0}^{n-1} \left| z_i^* \right|^2}} \quad (16)$$

In both (15) and (16), z and z^* stand for the result computed by our algorithm and the corresponding result computed by the direct computation of the recurrence given by Equation (8), respectively. Equation (15) computes the relative errors with respect to the maximum norm or infinity norm, while Equation (16) computes the relative errors with respect to the p-norm with $p = 2$. MATLAB rand () function has been used to generate random sample points x and random input vectors f. To compute MATLAB execution times (in seconds) for the MFDLT and DLT, MATLAB cpu time function has been operated.

B. Numerical Results

Table 1. Numerical results of Experiment 1

n	Errors		Time(sec.)	
	RE_∞	RE_2	MFDLT	DLT
128	2.422 E-15	9.668 E-15	0.7956	0.3432
256	4.097 E-15	1.606 E-14	5.3508	3.0109
512	1.247E-13	6.805 E-13	95.9250	82.7117
1024	4.751 E-13	2.996 E-12	2060.4	1997.0

Table 2. Numerical results of Experiment 2

n	Errors		Time(sec.)	
	RE_∞	RE_2	MFDLT	DLT
128	1.644 E-14	8.470 E-14	0.4524	0.3120
256	5.331 E-14	3.247 E-13	1.8096	3.3072
512	2.354 E-13	2.108 E-12	8.8764	94.942
1024	1.699 E-12	2.21 1E-11	52.3383	2122.3

IV. DISCUSSION

In accordance with the experiment results, it can be observed from Tables 1-2 that the relative errors RE_∞ and RE_2 are very small. Therefore, it is confirmed that the MFDLT permits high accuracy to be attained. Table I demonstrates that the MATLAB execution times elapsed for the MFDLT are not less than those of the DLT. This is due to the fact that in Experiment 1, the costly direct computation of the NUDCT was used; however this can be avoided by accelerating the NUDCT. Table 2 shows that the MFDLT is faster than the DLT when $n > 128$ (about 40 times faster at $n = 1024$). This is because we have used the uniform fast (exact) fast cosine transform for computing the initial phase. Furthermore, these experiments display that initial phase is more costly than the other phases of the MFDLT algorithm.

V. CONCLUSION

In this paper, we have proposed a more accurate algorithm for the recently developed fast algorithm for discrete Legendre polynomial transforms. The proposed algorithm permits high accuracy to be attained. The initial phase of the recently developed algorithm was computed by the fast Vandermonde matrix vector product. Part of the Vandermonde matrix vector-product takes advantage of efficient computation of coefficients of the monomials in the form $\prod_{x_j \in S \subset X}(x - x_j)$, where X is a related sample given by $X = \{x_0, x_1, \ldots, x_{n-1}\}$. These coefficients are notoriously sensitive (Moore SS(1993)) to make the Vandermonde matix-vector product unstable in a reasonable computer arithmetic. However, instead of that, the proposed algorithm has used the discrete cosine transform, which is more stable in computer arithmetic than monomials. In addition to that, this modification does not bring down the efficiency of the previously developed algorithm whenever the discrete cosine transform is computed efficiently. Some approximate but stable algorithms with reasonable time complexity, for example $O(n \log n)$, for the efficient computation of the non-uniform discrete cosine transforms are also available in the literature (Tian, B., & Liu, Q. H. (1999)).

REFERENCES

Chihara TS.(1957). "On co-recursive orthogonal Polynomials," Proceedings of the American Mathematical Society, vol. 8, 899-905pp.

Cooley JW. & Tukey JW (1965). "An algorithm for the machine calculation of complex Fourier series," Mathematics of computation, vol. 19, 297- 301pp.

Driscoll JR & Healy Jr DM.(1989). "Asymptotically fast algorithms for spherical and related transforms," In Foundations of Computer Science, 30th Annual Symposium , 344-349pp, IEEE.

Driscoll JR & Healy DM.(1994). "Computing Fourier transforms and convolutions on the 2- sphere," Advances in applied mathematics, vol.15, 202-250 pp.

Driscoll JR , Healy Jr, DM. & Rockmore DN (1997). "Fast discrete polynomial transform with applications to data analysis for distance transitive graphs," SIAM Journal on Computing, vol.26, 1066-1099 pp.

Gautschi W (1985)."Orthogonal polynomials—constructive theory and applications,".Journal of Computational and Applied Mathematics, vol. 12, 61-76 pp.

Gunarathna. W. A and Nasir. H. M,'' Fast Transform Algorithm for Legendre Polynomial Transforms'', Proceedings of the KDU International Research Symposium on Sri Lanka as a Hub in Asia: The Way Forward, General Sir John Kotelawala Defence University, Sri Lanka, 22nd- 23rd August, 2013, ISBN 978- 955-0301-07-2, pp 248-263.

Heideman MT., Johnson D. H. & Burrus CS (1985). "Gauss and the history of the fast Fourier transform,'' Archive for history of exact sciences, vol.34, 265-277pp.

Inda MA, Bisseling RH & Maslen DK.(2001). "On the efficient parallel computation of Legendre transforms,'' SIAM Journal on Scientific Computing, no.23, 271-303pp.

James S. Walker.,Fast fourier transforms:CRC PressI Llc.,1996.

Moore SS. Healy Jr DM. & Rockmore DN.(1993)."Symmetry stabilization for fast discrete monomial

transforms and polynomial evaluation," Linear algebra and its applications, vol. 192, 249-299 pp.

Potts, D , Steidl, G & Tasche M(1998). "Fast algorithms for discrete polynomial transforms,", Mathematics of Computation of the American Mathematical Society, vol. 67, 1577-1590 pp.

Potts D (2003). ''Fast algorithms for discrete polynomial transforms on arbitrary grids," Linear algebra and its applications, vol. 366, 353-370 pp.

Tang Z. Duraiswami R. & Gumerov NA.(2004). "Fast algorithms to compute matrix-vector products for Pascal matrices".

Tian, B., & Liu, Q. H. (1999, July). Nonuniform fast cosine transform and the Chebyshev PSTD algorithm. In Antennas and Propagation Society International Symposium, 1999. IEEE (Vol. 4, pp. 2184-2187). IEEE.

AUTHOR BIOGRAPHY

[1]Nasir HM is a senior lecturer at Mathematics Department of the University of Peradeniya, Sri Lanka. He received Master of Engineering (M.Eng) and Ph.D in computational Mathematics, both from the University of Electro Communications, Tokyo, JAPAN in 1999 and 2003, respectively. His research interests include Numerical solutions of partial differential equations, and multi complex analysis.

[2]WA Gunarathna is a Mathematics lecturer at General Sir John Kotelawala Defence University, Ratmalana, Sri Lanka. His research interests include design of efficient numerical algorithms for polynomial transforms and numerical solutions of parabolic partial differential equations. He is currently reading for an M.Phill in Mathematics at the Postgraduate Institute of Science, University of Peradeniya, Sri Lanka.