Fast transform algorithm for legendre polynomial transforms

WA Gunarathna[1] and 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*

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

Abstract-- *Let* $P = \{p_0, p_1, \ldots, p_{n-1}\}$ *denote the collection of the first* n *Legendre polynomials* p_k *of degree* k *and let* $\mathrm{X} = \{x_0 . x_1, \ldots, x_{n-1}\}$ *be a set of* n *sample points. The Discrete Legendre polynomial transform (DLT) of an input vector,* $f = (f_0, f_1, \ldots, f_{n-1})$ *of size* n *is defined by* $\{L(0), L(1), \cdots, L(n-1)\}$*,where* $L(l) = \sum_{j=0}^{n-1} f_j p_l(x_j)$ *for each* $j = 0,1,\ldots,n-1$. *For the given collection of* n *Legendre polynomials, the straightforward method for computing DLT requires an* $O(n^3)$ *computational cost. This cost gets prohibitively increased for large values of* n *and hence it is too much for practical purpose. This paper describes an algorithm for fast computation of DLTs. The fast algorithm computes the DLT of any input vector of size* n *on a set of* n *arbitrary sample points in an* $O((n\log_2 n)^2)$ *cost of complexity. The numerical experiments carried out demonstrate that the fast algorithm is faster than the straightforward algorithm when* $n \geq 128$.

Keywords— **Fast Fourier Transform (FFT), Orthogonal polynomials, Legendre polynomial transforms, Linear three term recurrence relation.**

I.INTRODUCTION

Discrete Legendre polynomials (DLTs) have gained a great popularity among the scientific community due to its significant applications available in many areas such as weather forecasting, statistical data analysis, electrical, and telecommunicating engineering etc.

Legendre polynomials are solutions of the Legendre differential equation.

$$(1-x^2)\frac{d^2 p_n(x)}{dx^2} - 2x\frac{dp_n(x)}{dx} + n(n+1)p_n(x) = 0.(1.1)$$

Let $\mathcal{P}$= $\{p_0, p_1, \ldots, p_{n-1}\}$ denote the collection of the first n Legendre polynomials p_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

$$\langle f, g\rangle = \int_{-1}^{1} f(x)g(x)w(x)dx, \qquad (1.2)$$

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

$$\|f\| = \sqrt{\langle f, f\rangle} \qquad (1.3)$$

To put it another way,

$$\langle p_i, p_j\rangle = c\delta_{ij}, \qquad (1.4)$$

where

$$\delta_{ij} = \begin{cases} 1 & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \qquad (1.5)$$

Is the Kronecker delta and c is a constant.

Let $\mathrm{X} = \{x_0 . x_1, \ldots, x_{n-1}\} \subset [-1,1]$ be any set of n sample points. The Discrete Legendre polynomial

transform (DLT) of an input vector, $f = (f_0, f_1, \ldots, f_{n-1})$ of size n is defined by the collection:

$$\{L(0), L(1), \ldots, L(n-1)\},$$

where

$$L(l) = \sum_{j=0}^{n-1} f_j p_l(x_j) \qquad (1.6)$$

for each $l = 0,1,\ldots,n-1$.

It is not difficult to see that the whole computation involved in (1.6) can be represented in the following matrix-vector multiplication:

$$\begin{pmatrix} L(0) \\ \hat{L}(1) \\ \vdots \\ \hat{L}(n-1) \end{pmatrix} =$$

$$\begin{pmatrix} p_0(x_0) & p_1(x_0) & \cdots & p_{n-1}(x_0) \\ p_0(x_1) & p_1(x_1) & \cdots & p_{n-1}(x_1) \\ \vdots & \vdots & \cdots & \vdots \\ p_0(x_{n-1}) & p_1(x_{n-1}) & \cdots & p_{n-1}(x_{n-1}) \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{n-1} \end{pmatrix} \tag{1.7}$$

In can be shown that the total number of operations required to perform straightforwardly the matrix-vector product in (1.8) is

$$T(n) = n^3 + n^2 - n = O(n^3). \blacksquare \tag{1.8}$$

This polynomial complexity quickly becomes heavy for large values of n and thus it causes severe computational obstacles in many applications. Some tread has been devoted to reduce this computational complexity. In the context of computing the Fourier series on the 2-sphere, some authors (Driscoll JR & Healy Jr DM.(1989) ; Driscoll JR & Healy DM.(1994)) used an $O\left((n\log_2 n)^2\right)$ algorithm of to compute the Legendre polynomials transforms at the natural Chebyshev points. Independently, the paper (Potts, D , Steidl, G & Tasche M(1998)) describes a fast algorithm to compute a discrete orthogonal polynomial transform of size n at the Chebyshev points in $O\left((n\log_2 n)^2\right)$ operations. Furthermore, this algorithm has been improved to compute a discrete polynomial of size n at arbitrary points which lies in the closed interval [-1,1] in the same computational complexity (Potts D (2003)

In this paper, we make use the following theorem appears in Driscoll JR et al (1997) to design an algorithm to compute the Legendre polynomials transform on any set of sample points in [-1 1] in $O(n\log_2^2 n)$ operations.

A. Theorem 1.1. Let n be a positive integer power of 2.

If $\{h_l(x)\}_{l,x=0}^{n-1}$ be a collection of functions that satisfy the three-term recurrence relation

$$h_{l+1}(x) = (a_l x + b_l)h_l(x) + c_l h_{l-1}(x), \tag{1.9}$$

with the initial conditions: $h_0(x) = 1, h_{-1}(x) = 0$,

then for any input vector $f = (f_0, f_1, \cdots, f_{n-1})$, we can compute the collection:

$$\left\{\sum_{j=0}^{n-1} f_j g_l(j)\right\}_{l=0,\cdots,n-1}, \tag{1.10}$$

in $O\left((n\log_2 n)^2\right)$ operations.

The organization structure of this paper is as follows: in Section 2, we present standard structured matrices and efficient computational methods for their matrix-vector multiplications. Section 3 describes the fast computation of the Legendre polynomial transforms. Section 4 presents the numerical results of the numerical experiments. Finally, we give conclusions in Section 5.

II. EFFICIENT COMPUTATION OF STRUCTURED MATRIX - VECTOR MULTIPLICATIONS.

In this section, we present some standard structured matrix-vector multiplications and their efficient computational methods without proof.

A. Fourier matrix

Definition: An $n \times n$ Fourier matrix, denoted by F_n, is defined to be the square matrix given by

$$F_n = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & w & \cdots & w^{n-1} \\ \vdots & \vdots & \vdots & \\ 1 & w^{n-1} & \cdots & w^{(n-1)(n-1)} \end{pmatrix} \tag{2.1}$$

, where $w = e^{-2\pi i/n}$.

It can be seen that the product of the Fourier matrix F_n with an arbitrary input column vector of size n yields the standard discrete Fourier transform (DFT) , can be performed more efficiently by the well known fast Fourier transform (FFT)(Cooley JW. & Tukey JW (1965)).

Note that $F_n(\overline{F}_n) = (\overline{F}_n)F_n = I$ and thus $F_n^{-1} = \overline{F}_n$, where $\overline{F}_n$ is the conjugate matrix of F_n .

The related matrix vector product of this inverse with an arbitrary input vector is the standard inverse Discrete Fourier transform (DFT). We summarize the material found from (Cooley JW. & Tukey JW (1965, Tang Z. Duraiswami R. & Gumerov NA.(2004).) in the following theorem without proof:

1) Theorem 2.1: The DFT and IDFT each can be done in $O(n\log n)$ operations.

Note that the product of an $n \times n$ Fourier matrix and an $n \times 1$ vector can be done using at most $\frac{3n}{2}\log n$ operations. ■

B. Circulant Matrix

Definition: An $n \times n$ square matrix C has the following form is called a circulant matrix.

$$C := \begin{pmatrix} c_0 & c_1 & \cdots & c_{n-1} \\ c_{n-1} & c_0 & \cdots & c_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & \cdots & c_0 \end{pmatrix} \tag{2.2}$$

,where $c_0, c_1, \cdots . c_{n-1}$ are complex numbers.

We denote the circulant matrix C of order n by $C[c_0, c_1, \cdots, c_{n-1}]$.

The circulant matrix processes constant element along the diagonals from top left to bottom right and in fact, it can be easily seen that the circulant matrix can be spanned by the entities in the first row.

1) Theorem 2.2 Circulant matrix-vector multiplication. The product of a circulant matrix of order n and a column vector of size n can be performed in $O(n\log_2 n)$i operations (Tang Z. Duraiswami R. & Gumerov NA.(2004)).

It should be noted that the circulant matrix-vector product Cy can be written as the cyclic convolution of the sequences,

$$c = (c_0, c_1, \cdots, c_{n-1}) \text{ and } (y_0, y_1, \cdots, y_{n-1}).$$

To put it another way,

$$z_j = c * y_j = \sum_{m=0}^{n-1} y_m c_{j-m} \tag{2.3}$$

for all $j = 0,1,\cdots, n-1$

Now by the cyclic convolution theorem, we get:

$$\mathrm{DFT}(c * y_j) = (\mathrm{DFT}(c_0, c_1, \cdots, c_{n-1})\mathrm{DFT}(y_0, y_1, \cdots, y_{n-1}))$$
$$(c * y_j) = \mathrm{IDFT}(\mathrm{DFT}(c_0, c_1, \cdots, c_{n-1})\mathrm{DFT}(y_0, y_1, \cdots, y_{n-1})),$$

This concludes that the circulant matrix- vector multiplication can be performed using only three FFTs (one IFFT and two FFTs) and one element –wise vector product. Since each FFT needs $\frac{3n}{2}\log n$ arithmetic operations, the number of operations required performing the circulant matrix -, vector multiplication is

$$T(n) = 3 \times \frac{3n}{2}\log n + n = \frac{9n}{2}\log n + n \tag{2.4}$$

That is , $T(n)$ **is** $O(n\log_2 n)$ ■

C.Toeplitz matrix

A Toeplitz matrix of order n is defined to be

$$T = \begin{pmatrix} t_0 & t_1 & \cdots & t_{n-1} \\ t_{-1} & t_0 & \cdots & t_{n-2} \\ \vdots & \vdots & \cdots & \vdots \\ t_{-n+1} & t_{-n+2} & \cdots & t_0 \end{pmatrix}. \tag{2.5}$$

We will denote the Toeplitz matrix of order n by $T[t_{-n+1}, \cdots, t_0, \cdots, t_{n-1}]$,

where $t_{-n+1}, \dots, t_0, \dots t_{n-1}$ are complex numbers.

A Toepltz matrix can be spanned by its first column and its first row. Besides that, the entities of the Toeplitz matrix are constant along the sub diagonals parallel to the main diagonal.

1) Theorem 2.4:Toeplitz matrix -vector multiplication
   The product of a Toeplitz matrix of order n and a column vector of size n can be performed in $O(n\log n)$ arithmetic operations (Tang Z. Duraiswami R. & Gumerov NA.(2004)).

Proof

Let T be a Toeplitz matrix of order n and y be a column vector of size$n$.

Define the circulant matrix C of order 2n as follows:

$$C=\begin{pmatrix} T & S \\ S & T \end{pmatrix} \tag{2.6}$$

, where S is a square matrix of order n given by

$$S=\begin{pmatrix} 0 & t_{-n+1} & \cdots & t_{-1} \\ t_{n-1} & 0 & \cdots & t_{-2} \\ \vdots & \vdots & \ddots & \vdots \\ t_1 & t_2 & \cdots & 0 \end{pmatrix} \tag{2.7}$$

Also let $z=\binom{y}{O}$ be a column vector of size $2n$, where O denotes an $n\times 1$ zero matrix.

Now we get

$$Cz=\begin{pmatrix} T & S \\ S & T \end{pmatrix}\binom{y}{O}=\binom{Ty}{Sy} \tag{2.8}$$

Now this concludes that the Toeplitz matrix- vector multiplication cam be done in $O(n\log n)$ arithmetic operations.■

It should be noticed that only three FFTs of order $2n$ each and one point-wise multiplication of sequence of length are needed to perform the Toeplitz matrix-vector multiplication and that the total number of arithmetic operations is equal to: $3\left(\frac{3(2n)}{2}\log 2n\right)+2n=9n\log 2n+2n$

$$=9n\log n+11n$$

$$=O(n\log n)\,.\blacksquare \tag{2.9}$$

D Vandermonde matrix

Definition *:* A Vandermonde matrix of order n is defined by the following marix.

$$V=\begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ z_0 & z_1 & \cdots & z_{n-2} & z_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ z_0^{n-2} & z_1^{n-2} & \cdots & z_{n-2}^{n-2} & z_{n-1}^{n-2} \\ z_0^{n-1} & z_1^{n-1} & \cdots & z_{n-2}^{n-2} & z_{n-1}^{(n-1)} \end{pmatrix}. \tag{2.10}$$

The Vandermonde matrix of the complex number $z_0,z_1,\cdots,z_{n-1}$ is denoted by

$$V[z_0 \quad z_1 \quad \cdots \quad z_{n-1}].$$

The material found from Driscoll JR et al (1997) and Moore SS et al (1993)), we summarize in the following theorem without derivation.

1) *Theorem 2.5:*Vandermonde matrix-vector multiplication
   The product of a Vandermonde matrix of order n (as defined in 2.5) and an arbitrary input vector of size n can be performed in $O(n\log^2 n)$ operations.

III. LEGENDRE POLYNOMIAL TRANSFORM

The key objective of this section is make use of Theorem 1.1 to design a fast algorithm to compute Legendre polynomial transforms on a set of arbitrary sample points.

A. *Corollary 3.1.*
   Let $\{p_0(x),p_1(x),\cdots,p_{n-1}(x)\}$ be a collection of Legendre polynomials and let

$\{x_0,x_1,\cdots,x_{n-1}\}$ be a set of n arbitrary sample points in $[-1,1]$.

Then the n-point the Legendre polynomial transform of an input vector $f=(f_0,f_1,\cdots,f_{n-1})$, defined by

$$\left\{\sum_{j=0}^{n-1} f_j p_l(x_j)\right\}_{l=0,1,\cdots,n-1} \tag{3.1}$$

can be computed in $O(n\log^2 n)$ operations.

Proof

In this proof we assume that a single operation holds the combination of one multiplication and one addition.

Let $$L(l)=\sum_{j=0}^{n-1} f_j p_l(x_j) \text{ for } l=0,1,\cdots,n-1. \tag{3.2}$$

Then we must calculate $L(0),L(1),\cdots,L(n-1)$.

Let $\{p_0(x),p_1(x),\cdots,p_{n-1}(x)\}$ be a collection of Legendre polynomials.

Then according to Farvard's theorem, these orthogonal polynomials satisfy the following linear three term recurrence relation (Chihara TS (1957), Inda MA et al. .(2001)):

$$\begin{cases} P_{l+1}(x)=\dfrac{2l+1}{l+1}xP_{n-1}(x)-\dfrac{l}{l+1}P_{n-1}(x) \\ P_0(x)=1 \\ P_{-1}(x)=0 \end{cases} \tag{3.3}$$

Let $x_j(j=0,1,2,...,n-1)$ denote a typical sample point.

Then by substituting $x=x_j$ into (3.3), we get,

$$P_{l+1}(x_j)=\frac{2l+1}{l+1}x_jP_{l-1}(x_j)-\frac{l}{l+1}P_{l-1}(x_j) \tag{3.4}$$

Let $Z_1^{(n)}=\begin{pmatrix} Z_1^{(n)}(0) \\ Z_1^{(n)}(1) \\ \vdots \\ Z_1^{(n)}(n-1)\end{pmatrix}$ be a vector whose kth element,

$Z_l^{(n)}(k)$ is defined by the following sum for each $l=0,\cdots,n-1$.

$$Z_l^{(n)}(k)=\left\langle f,(\lambda x_j)^k p_l(x_j)\right\rangle=\sum_{j=0}^{n-1}f_j(\lambda x_j)^k p_k(x_j) \tag{3.5}$$

for each $k=0,1,\ldots,n-1$.

Here we have introduced a scalar factor λ to control overflow (underflow).

Then we see that:

$$\begin{cases} Z_0^{(n)}(k)=\left\langle f,(\lambda x_j)^k\right\rangle=\sum_{j=0}^{n-1}f_j(\lambda x_j)^k \\ Z_l^{(n)}(0)=\left\langle f,p_l(x_j)\right\rangle=\sum_{j=0}^{n-1}f_jp_k(x_j)\end{cases} \tag{3.6}$$

Now all the terms in the sequence $\left\{\sum_{j=0}^{n-1}f_jp_l(x_j)\right\}_{l=0,1,\cdots,n-1}$ can be calculated by calculating all the terms in the sequence $\{Z_l^{(n)}(0)\}_{l=0,\cdots,n-1}$.

Now, from (3.4), we get:

$$Z_{l+1}^{(n)}(k)=\left\langle f,x_j^k p_{l+1}(x_j)\right\rangle$$

$$=\left\langle f,(\lambda x_j)^k\left[\frac{2l+1}{l+1}x_jp_l(x_j)-\frac{l}{l+1_l}p_{l-1}(x_j)\right]\right\rangle$$

$$=\frac{2l+1}{\lambda(l+1)}\left\langle f,(\lambda x_j)^{k+1}p_l(x_j)\right\rangle-\frac{l}{l+1}\left\langle f,(\lambda x_j)^k p_{l-1}(x_j)\right\rangle$$

Hence we get the following recurrence:

$$Z_{l+1}^{(n)}(k)=\frac{2l+1}{\lambda(l+1)}Z_l^{(n)}(k+1)-\frac{l}{l+1}Z_{l-1}^{(n)}(k) \tag{3.7}$$

Computation of the terms in the sequence $\{Z_l^{(n)}(0)\}_{l=0,\cdots,n-1}$ .can be done under three stages as explained below:

Computational Stage 1: Computation of $Z_0^{(n)}$.

$$Z_0^{(n)}=\begin{pmatrix} Z_0^{(n)}(0) \\ Z_0^{(n)}(1) \\ \vdots \\ Z_0^{(n)}(n-1)\end{pmatrix}=\begin{pmatrix}\sum_{j=0}^{n-1}f_j(\lambda x_j)^0 \\ \sum_{j=0}^{n-1}f_j(\lambda x_j) \\ \vdots \\ \sum_{j=0}^{n-1}f_j(\lambda x_j)^{n-1}\end{pmatrix}$$

$$=\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \lambda x_0 & \lambda x_1 & \cdots & \lambda x_{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ (\lambda x_0)^{n-1} & (\lambda x_1)^{n-1} & \cdots & (\lambda x_{n-1})^{n-1}\end{pmatrix}\begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{n-1}\end{pmatrix}$$

This is a Vandermonde matrix-vector multiplication of order n and hence $Z_0^{(n)}$ can be computed more efficiently in $O\left(n(\log n)^2\right)$ operations using Theorem 2.5. In particular, we get

$$L(0)=Z_0^{(n)}(0) \tag{3.8}$$

Computational stage 2: Computation of $Z_1^{(n)}$

It can be seen from (3.7) that

$$Z_1^{(n)}(k) = \left(\frac{1}{\lambda}\right) Z_0^{(n)}(k+1) \qquad (3.9)$$

Now,

$$Z_1^{(n)} = \begin{pmatrix} Z_1^{(n)}(0) \\ Z_1^{(n)}(1) \\ \vdots \\ Z_1^{(n)}(n-1) \end{pmatrix} = \left(\frac{1}{\lambda}\right)\begin{pmatrix} Z_0^{(n)}(1) \\ Z_0^{(n)}(2) \\ \vdots \\ Z_0^{(n)}(n) \end{pmatrix} \qquad (3.10)$$

From computational stage 1, we have $Z_0{}^{(n)}(k)$ for all $k = 0,1,\cdots,n-1$.Therefore, now $Z_1^{(n)}(k)$ for each $k = 0,1,\cdots,n-2$ can be done at most n additional operations. In particular, we get

$$L(1) = Z_1^{(n)}(0) \qquad (3.11)$$

Computational stage 3: Computation o $Z_l^{(n)}$ for $l = 2,3,\cdots,n-1$.

We first make the following notation:

The vector obtained by leaving *the lower half* of the vector

$Z_1^{(2m)} = \begin{pmatrix} Z_1^{(2m)}(0) \\ Z_1^{(2m)}(1) \\ \vdots \\ Z_1^{(2m)}(2m-1) \end{pmatrix}$ is denoted by $Z_1^{(m)} = \begin{pmatrix} Z_1^{(m)}(0) \\ Z_1^{(m)}(1) \\ \vdots \\ Z_1^{(m)}(m-1) \end{pmatrix}$

and

$R^{(2m)}(,)$, which is defined by (3.16), is a square matrix of order $4m$ for $m = 2,4,8,\cdots,\frac{n}{2}$.

From (3.7) we have

$$Z_{l+1}^{(n)}(k) = \frac{2l+1}{\lambda(l+1)} Z_l^{(n)}(k+1) - \frac{l}{l+1} Z_{l-1}^{(n)}(k)$$

We can further write $Z_{l+1}^{(n)}$ in the following matrix form:

$$\begin{pmatrix} Z_{l+1}^{(n)}(0) \\ Z_{l+1}^{(n)}(1) \\ \vdots \\ Z_{l+1}^{(n)}(n-1) \end{pmatrix} = \frac{2l+1}{\lambda(l+1)}\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}\begin{pmatrix} Z_l^{(n)}(1) \\ Z_l^{(n)}(2) \\ \vdots \\ Z_l^{(n)}(n) \end{pmatrix}$$

$$-\frac{l}{l+1}\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}\begin{pmatrix} Z_{l-1}^{(n)}(0) \\ Z_{l-1}^{(n)}(1) \\ \vdots \\ Z_{l-1}^{(n)}(n-1) \end{pmatrix} \qquad (3.12)$$

Now define a new sequence $Z'^{(n)}_l \; (l = 0,1,\cdots,n-1)$ such that:

$$Z''_0{}^{(n)} = Z_0{}^{(n)}$$

$$Z''_1{}^{(n)} = Z_1{}^{(n)}$$

$$\begin{pmatrix} Z'^{(n)}_{l+1}(0) \\ Z'^{(n)}_{l+1}(1) \\ \vdots \\ Z'^{(n)}_{l+1}(n-1) \end{pmatrix} = \frac{2l+1}{\lambda(l+1)}\begin{pmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{pmatrix}\begin{pmatrix} Z_l^{(n)}(0) \\ Z_l^{(n)}(1) \\ \vdots \\ \\ Z_l^{(n)}(n-1) \end{pmatrix}$$

$$-\frac{l}{l+1}\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}\begin{pmatrix} Z_{l-1}^{(n)}(0) \\ Z_{l-1}^{(n)}(1) \\ \vdots \\ Z_{l-1}^{(n)}(n-1) \end{pmatrix} ,(3.13)$$

where $l = 1,2,\cdots,n-2.$

In short form, we can write

$$Z'^{(n)}_{l+1} = \frac{2l+1}{\lambda(l+1)} N Z_l^{(n)} - \frac{l}{l+1} I Z_{l-1}^{(n)}, \qquad (3.14)$$

where

$I = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$, the $n \times n$ identity matrix, and (5.11)

$N = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{pmatrix}$, the $n \times n$ nilpotent matrix

containing ones on the super diagonal and zeros elsewhere.

Now, by (3.12) and (3.13) , we have

$$Z'^{(n)}_l(k)=Z^{(n)}_l(k) \text{ for } k=0,1,\cdots,n-2 \text{ and}$$

$$\begin{pmatrix} Z'^{(n)}_l \\ Z'^{(n)}_{l+1} \end{pmatrix}=\begin{pmatrix} 0 & I \\ -\frac{l}{l+1}I & \frac{2l+1}{\lambda\,(l+1)}N \end{pmatrix}\begin{pmatrix} Z_{l-1}{}^{(n)} \\ Z_l{}^{(n)} \end{pmatrix} \quad (3.14)$$

for $k=0,1,\cdots n-2$

Then we can establish the following matrix equations on any positive ineger p.

$$\begin{pmatrix} Z'_{l-1}{}^{(n)} \\ Z'_{l}{}^{(n)} \end{pmatrix}=\begin{pmatrix} 0 & I \\ -\frac{l-1}{l}I & \frac{2l-1}{\lambda l}N \end{pmatrix}\begin{pmatrix} Z_{l-2}{}^{(n)} \\ Z_{l-1}{}^{(n)} \end{pmatrix}$$

$$=\begin{pmatrix} 0 & I \\ -\frac{l-1}{l}I & \frac{2l-1}{\lambda l}N \end{pmatrix}\begin{pmatrix} Z'_{l-2}{}^{(n)} \\ Z'_{l-1}{}^{(n)} \end{pmatrix}$$

$$\begin{pmatrix} Z'_{l-2}{}^{(n)} \\ Z'_{l-1}{}^{(n)} \end{pmatrix}=\begin{pmatrix} 0 & I \\ -\frac{l-2}{l-1}I & \frac{2l-3}{\lambda(l-1)}N \end{pmatrix}\begin{pmatrix} Z_{l-3}{}^{(n)} \\ Z_{l-2}{}^{(n)} \end{pmatrix}$$

$$=\begin{pmatrix} 0 & I \\ -\frac{l-2}{l-1}I & \frac{2l-3}{\lambda(l-1)}N \end{pmatrix}\begin{pmatrix} Z'_{l-3}{}^{(n)} \\ Z'_{l-2}{}^{(n)} \end{pmatrix}$$

$$\vdots$$

$$\begin{pmatrix} Z'_{l-p}{}^{(n)} \\ Z'_{l-p+1}{}^{(n)} \end{pmatrix}=\begin{pmatrix} 0 & I \\ -\frac{l-p}{l-p+1}I & \frac{2l-2p+1}{\lambda(l-p+1)}N \end{pmatrix}\begin{pmatrix} Z_{l-p-1}{}^{(n)} \\ Z_{l-p}{}^{(n)} \end{pmatrix}$$

From these matrix equations, we may get the following:

$$\begin{pmatrix} Z'_{l}{}^{(n)} \\ Z'_{l+1}{}^{(n)} \end{pmatrix}=\begin{pmatrix} 0 & I \\ -\frac{l}{l+1}I & \frac{2l+1}{\lambda(l+1)}N \end{pmatrix}\begin{pmatrix} 0 & I \\ -\frac{l-1}{l}I & \frac{2l-1}{\lambda l}N \end{pmatrix} \cdots \begin{pmatrix} 0 & I \\ -\frac{l-p}{l-p+1}]I & \frac{2l-2p+1}{\lambda(l-p+1)}N \end{pmatrix}\begin{pmatrix} Z_{l-p-1}{}^{(n)} \\ Z_{l-p}{}^{(n)} \end{pmatrix}$$

$$\begin{pmatrix} Z'_{l}{}^{(n)} \\ Z'_{l+1}{}^{(n)} \end{pmatrix}=R^{(2n)}(l-p-1,l)\begin{pmatrix} Z_{l-p-1}{}^{(n)} \\ Z_{l-p}{}^{(n)} \end{pmatrix} \quad (3.15)$$

, where

$$R^{(2n)}(l-p-1,l)$$

$$=\begin{pmatrix} 0 & I \\ -\frac{l}{l+1}I & \frac{2l+1}{\lambda(l+1)}N \end{pmatrix}\begin{pmatrix} 0 & I \\ -\frac{l-1}{l}I & \frac{2l-1}{\lambda l}N \end{pmatrix}$$

$$\cdots\begin{pmatrix} 0 & I \\ -\frac{l-p}{l-p+1}]I & \frac{2l-2p+1}{\lambda(l-p+1)}N \end{pmatrix} \quad (3.16)$$

We can use the fact that $Z^{(n)}_l(k)=Z'^{(n)}_l(k)$ for $k=0,1,\cdots,n-2$ to find $Z^{(n)}_l$.

The matrix $R^{(2n)}(l-p-1,l)$ is comprised of $p+1$ factor matrices for which each factor matrix contains four $n\times n$ Toeplitz matrices. Since the product of two Toeplitz matrices is again a Toeplitz matrix, the matrix $R^{(2n)}(l-p-1,l)$ can be reduced to the form

$R^{(2n)}(l-p-1,l)=\begin{pmatrix} T_1{}^{(n)} & T_2{}^{(n)} \\ T_3{}^{(n)} & T_4{}^{(n)} \end{pmatrix}$, where

$T_i{}^{(n)}$ is an $n\times n$ Toeplitz block for $i=1,2,3,4$.

Then we see that:

$$\begin{pmatrix} Z_l{}^{(n)} \\ Z_{l+1}{}^{(n)} \end{pmatrix}=\begin{pmatrix} T_1{}^{(n)} & T_2{}^{(n)} \\ T_3{}^{(n)} & T_4{}^{(n)} \end{pmatrix}\begin{pmatrix} Z_{l-p-1}{}^{(n)} \\ Z_{l-p}{}^{(n)} \end{pmatrix}$$

$$=\begin{pmatrix} T_1{}^{(n)}Z_{l-p-1}{}^{(n)}+T_2{}^{(n)}Z_{l-p}{}^{(n)} \\ T_3{}^{(n)}Z_{l-p-1}{}^{(n)}+T_4{}^{(n)}Z_{l-p}{}^{(n)} \end{pmatrix} \quad 3.17)$$

This means that we can obtain the vectors $Z_l{}^{(n)}$ and $Z_{l+1}{}^{(n)}$ from the vectors $Z_{l-p-1}{}^{(n)}$ and $Z_{l-p}{}^{(n)}$ using four Toeplitz matrix vector-multiplications.

In the case where $l=n/2$ and $p+1=n/2$, we get from (3.12)

$$\begin{pmatrix} Z'_{\frac{n}{2}}{}^{(n)} \\ Z'_{\frac{n}{2}+1}(n) \end{pmatrix}=R^{(n)}\left(0,\frac{n}{2}\right)\begin{pmatrix} Z_0{}^{(n)} \\ Z_1{}^{(n)} \end{pmatrix} \quad (3.18)$$

Now applying four Toeplitz matrix-vector multiplications into (3.18,) we calculate:

$$\begin{cases} L\left(\frac{n}{2}\right) = Z_{\frac{n}{2}}^{(n)}(0) = Z'^{(n)}_{\frac{n}{2}}(0) \\ L\left(\frac{n}{2}+1\right) = Z_{\frac{n}{2}+1}^{(n)}(0) = Z'^{(n)}_{\frac{n}{2}+1}(0) \end{cases} \tag{3.19}$$

Now the entire problem of size n is split into two sub problems of size $n/2$ each.

Then $\left\{Z'^{(m)}_l \mid l = \frac{n}{2}+1, \cdots, n-1 \text{ and } m = \frac{n}{4}, \frac{n}{8}, \cdots, 2\right\}$

and $\left\{Z'^{(m)}_l \mid l = 2, \cdots, \frac{n}{2}-1 \text{ and } m = \frac{n}{4}, \frac{n}{8}, \cdots, 2\right\}$ are computed by applying (3.15) to $Z^{(n)}_{\frac{n}{2}}$ and $Z^{(n)}_{\frac{n}{2}+1}$ and $Z^{(n)}_0$ and $Z^{(n)}_1$, respectively.

Now in the case where $l = n/4, p+1 = n/4$ and in the case where $l = 3n/4, p+1 = n/4$, we get from (3.12)

$$\begin{cases} \begin{pmatrix} Z'^{(n/2)}_{\frac{n}{4}} \\ Z'^{(n/2)}_{\frac{n}{4}+1} \end{pmatrix} = R^{(n/2)}\left(0, \frac{n}{4}\right) \begin{pmatrix} Z_0^{(n/2)} \\ Z_1^{(n/2)} \end{pmatrix} \\ \begin{pmatrix} Z'^{(n/2)}_{\frac{3n}{4}} \\ Z'^{(n/2)}_{\frac{3n}{4}+1} \end{pmatrix} = R^{(n/2)}\left(\frac{n}{2}, \frac{3n}{4}\right) \begin{pmatrix} Z^{(n/2)}_{\frac{n}{2}} \\ Z^{(n/2)}_{\frac{n}{2}+1} \end{pmatrix} \end{cases} \tag{3.20}$$

Now we get from (3.20),

$$\begin{cases} L\left(\frac{n}{4}\right) = Z^{(n/2)}_{\frac{n}{4}}(0) = Z'^{(n/2)}_{\frac{n}{2}}(0) \\ L\left(\frac{n}{4}+1\right) = Z^{(n/2)}_{\frac{n}{4}+1}(0) = Z'^{(n/2)}_{\frac{n}{2}}(0) \\ L\left(\frac{3n}{4}\right) = Z^{(n/2)}_{\frac{3n}{4}}(0) = Z'^{(n/2)}_{\frac{3n}{4}}(0) \\ L\left(\frac{3n}{4}+1\right) = Z^{(n/2)}_{\frac{3n}{4}+1}(0) = Z'^{(n/2)}_{\frac{3n}{4}+1}(0) \end{cases} \quad 3.18)$$

At this level we have only the two sub problem of size $n/2$ and the next stage is done by again splitting each of those sub problems into sub problems of $n/4$ each. That is, in this level there are four sub problems of size $n/4$ each.

Now putting $l = n/8, 3n/8, 5n/8, 7n/8$ and $p+1 = n/8$ into (3.12), we get the following:

$$\begin{cases} \begin{pmatrix} Z'^{(n/4)}_{\frac{n}{8}} \\ Z'^{(n/4)}_{\frac{n}{8}+1} \end{pmatrix} = R^{(n/4)}\left(0, \frac{n}{8}\right) \begin{pmatrix} Z_0^{(n/4)} \\ Z_1^{(n/4)} \end{pmatrix} \\ \begin{pmatrix} Z'^{(n/4)}_{\frac{3n}{8}} \\ Z'^{(n/4)}_{\frac{3n}{8}+1} \end{pmatrix} = R^{(n/4)}\left(\frac{n}{4}, \frac{3n}{8}\right) \begin{pmatrix} Z^{(n/4)}_{\frac{n}{4}} \\ Z^{(n/4)}_{\frac{n}{4}+1} \end{pmatrix} \\ \begin{pmatrix} Z'^{(n/4)}_{\frac{5n}{8}} \\ Z'^{(n/4)}_{\frac{5n}{8}+1} \end{pmatrix} = R^{(n/4)}\left(\frac{n}{2}, \frac{5n}{8}\right) \begin{pmatrix} Z^{(n/4)}_{\frac{n}{2}} \\ Z^{(n/4)}_{\frac{n}{2}+1} \end{pmatrix} \\ \begin{pmatrix} Z'^{(n/4)}_{\frac{7n}{8}} \\ Z'^{(n/4)}_{\frac{7n}{8}+1} \end{pmatrix} = R^{(n/4)}\left(\frac{3n}{4}, \frac{7n}{8}\right) \begin{pmatrix} Z^{(n/4)}_{\frac{3n}{4}} \\ Z^{(n/4)}_{\frac{3n}{4}+1} \end{pmatrix} \end{cases} \tag{3.21}$$

Now from (3.21), we calculate:

$$\begin{cases} L\left(\frac{n}{8}\right) = Z^{(n/4)}_{\frac{n}{8}}(0) = Z'^{(n/4)}_{\frac{n}{8}}(0) \\ L\left(\frac{n}{8}+1\right) = Z^{(n/4)}_{\frac{n}{8}+1}(0) = Z'^{(n/4)}_{\frac{n}{8}+1}(0) \\ L\left(\frac{3n}{8}\right) = Z^{(n/4)}_{\frac{3n}{8}}(0) = Z'^{(n/2)}_{\frac{3n}{8}}(0) \\ L\left(\frac{3n}{8}+1\right) = Z^{(n/4)}_{\frac{3n}{8}+1}(0) = Z'^{(n/4)}_{\frac{3n}{8}+1}(0) \\ L\left(\frac{5n}{8}\right) = Z^{(n/4)}_{\frac{5n}{8}}(0) = Z'^{(n/4)}_{\frac{5n}{8}+1}(0) \\ L\left(\frac{5n}{8}+1\right) = Z^{(n/4)}_{\frac{5n}{8}+1}(0) = Z'^{(n/4)}_{\frac{5n}{8}+1}(0) \\ L\left(\frac{7n}{8}\right) = Z^{(n/4)}_{\frac{7n}{8}}(0) = Z'^{(n/4)}_{\frac{7n}{8}}(0) \\ L\left(\frac{7n}{8}+1\right) = Z^{(n/4)}_{\frac{7n}{8}+1}(0) = Z'^{(n/4)}_{\frac{7n}{8}+1}(0) \end{cases} \tag{3.22}$$

Continuing the similar manner we can obtain the rest of the vales of $L(l) = Z_l^{(m)}(0)$. ■

1). Computational complexity of the algorithm:

Let $T(n)$ denote the number of operations required to compute $\{Z_l^{(n)}(0)\}_{l=2,\cdots,n-1,}$ from $Z_0^{(n)}$ and $Z_1^{(n)}$.

At the first , having left out the lower halves of the vectors $Z_0^{(n)}, Z_1^{(n)}, Z_{\frac{n}{2}}^{(n)}$, and $Z_{\frac{n}{2}+1}^{(n)}$, we possess two sub problems of size $n/2$ each. Now each of these sub problems requires a total of $T\left(\frac{n}{2}\right)$ operations.

To compute $Z_{\frac{n}{2}}^{(n)}$ and $Z_{\frac{n}{2}+1}^{(n)}$ from $Z_0^{(n)}$ and $Z_1^{(n)}$ from (3.18,) we have to perform four Toeplitz matrix-vector multiplications of size n and thus it requires

$4(9n\log n + 11n)$ operations owing to (2.9).

$$T(n) = 36n\log n + 44n + 2T\left(\frac{n}{2}\right)$$

$$= 36n\log n + 44n + 2T\left(\frac{n}{2}\right) \text{ for } n \geq 2 \quad (3.23)$$

Now iterating on (3.23) shows that:

$$T(2) = 2T(1) + 36\times 2\times 1 + 44\times 2$$

$$T(4) = 2T(2) + 36\times 4\times 2 + 44\times 4$$

$$T(8) = 2T(4) + 36\times 8\times 3 + 44\times 8$$

$\vdots$

$$T\left(\frac{n}{8}\right) = 2T\left(\frac{n}{16}\right) + 36\times\frac{n}{8}\times(k-3) + 44\times\frac{n}{8}$$

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + 36\times\frac{n}{4}\times(k-2) + 44\times\frac{n}{4}$$

$$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + 36\times\frac{n}{2}\times(k-1) + 44\times\frac{n}{2}$$

And we see the following derivation:

$$T(n) =$$
$$2\left(2\left(2\left(T\left(\frac{n}{8}\right) + 18\frac{n}{4}(k-2) + 44.\frac{n}{4}\right) + 36\frac{n}{2}(k-1) + 44.\frac{n}{2}\right)\right)$$
$$+36n\log n + 44n$$

$$= 2^3 T\left(\frac{n}{2^3}\right) + 36n[(k-2) + (k-1) + k] + 44n\times 2$$

$$= 2^3\, T\left(\frac{n}{2^3}\right) + 36n[(k-2) + (k-1) + k] + 44n \times 3$$

Continuing the same manner, we ultimately obtain:

$$T(n) = 2^k T(1) + 36n(1+2+3+\cdots+k) + 44n\times k$$

$$= 36n(1+2+3+\cdots+k) + 44n\times k \text{ since } T(1) = 0\,.$$

$$= 36n\frac{k(k+1)}{2} + 44n\times k$$

$$\leq 36n\left(\frac{k^2+k^2}{2}\right) + 44nk$$

$$= 36nk^2 + 44nk \text{ since } k \geq 1\,.$$

Since $k = \log n$, we get,

$T(n) \leq 36n\log^2 n + 44n\log n$ whenever $n > 1$.

According to Theorem 2.5, to compute $Z_0^{(n)}$ we require at most $18n\log^2 n + 3n\log n$ operations (Driscoll JR (1997) et al.,). Also the computation of $Z_1^{(n)}$ from (3.9) requires at most $2n$ operations.

Therefore, to compute both $Z_0^{(n)}$ and $Z_1^{(n)}$, we require at most $18n\log^2 n + 3n\log n + 2n$ operations.

Therefore, the entire problem of order n requires at most

$$44n\log n + 36n\log^2 n + 18n\log^2 n + 3n\log n + 2n$$

$$= 2n + 47n\log n + 54n\log^2 n \text{ operations}$$

Now we see that

$$2n + 47n\log n + 54n\log^2 n \leq 103n\log^2 n$$

whenever $n > 1$.

Hence, the computational cost of the algorithm is $O((n\log_2 n))^2$. ■

2) Properties of the matrix $R = R^{(2n)}(l-p-1,l)$

1. The matrix R can be reduced to a $2n\times 2n$ matrix containing four Toeplitz blocks of order n each. Note that the multiplication of two Toeplitz matrices yields a Toeplitz matrix and so does the sum of two Toeplitz matrices. Hence, by block -wise multiplication of matrices in (3.16), we can easily see that the matrix R reduces to $2n\times 2n$ matrix containing four Toeplitz blocks of order n each.

2. Given two matrices of the form $R_1 = R^{(m)}(j,j+s), R_2 = R^{(m)}(j+s.j+2s)$ generate efficiently a matrix of the form $R_3 = R^{(2m)}(j,j+2s)$.

Let N be the Nilpotent matrix of order n given by

$$N = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix},$$

containing ones on the supper diagonal and zeros elsewhere.

Then we see that $N^n = 0$ for $n = 1,2,\cdots$.

Consider an upper triangular Toeplitz matrix of order n given by

$$T^{(n)} = \begin{pmatrix} a_0 & a_1 & \cdots & a_{n-1} \\ 0 & a_0 & \cdots & a_{n-2} \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & a_0 \end{pmatrix}.$$

We now see that

$$T^{(n)} = a_0\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} + a_1\begin{pmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} + \cdots$$

$$+ a_{n-1}\begin{pmatrix} 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

$$= a_0 I + a_1 N + \cdots + a_{n-1}N^{n-1} \qquad (3.23)$$

This shows that an upper triangular Toeplitz matrix of order n can be written as an $(\mathrm{n}\text{-}1)^{\mathrm{th}}$ degree polynomial of the Nilpotent matrix N, whose coefficients are the entities (entities in the first row) of the Toeplitz matrix.

Now each of four Toeplitz matrices in $R_1 = R^{(m)}(j,j+s)$ may be written as a polynomial of the Nilpotent matrix N whose degree at most s .This is because of R_1 has a matrix product with s factor matrices. To see it,

$$R_1 = \begin{pmatrix} 0 & I \\ -\dfrac{j+s}{j+s+1}I & \dfrac{2j+2s+1}{j+s+1}N \end{pmatrix}\begin{pmatrix} 0 & I \\ -\dfrac{j+s-1}{j+s}I & \dfrac{2j+2s-1}{j+s}N \end{pmatrix}$$

$$\cdots\begin{pmatrix} 0 & I \\ \dfrac{j+1}{j+2}I & \dfrac{2j+3}{j+2}N \end{pmatrix}$$

$$= \begin{pmatrix} a_{10} + a_{11}N + \cdots + a_{1s-2}N^{s-2} & b_{10} + b_{11}N + \cdots + b_{1s-1}N^{s-1} \\ c_{10} + c_{11}N + \cdots c_{1s-1}N^{s-1} & d_{10} + d_{11}N + \cdots + d_{1s}N^{s} \end{pmatrix}$$

Then we get

$$R_1 = \begin{pmatrix} \sum_{k=0}^{s-2} a_{1k}N^k & \sum_{k=0}^{s-1} b_{1k}N^k \\ \sum_{k=0}^{s-1} c_{1k}N^k & \sum_{k=0}^{s} d_{1k}N^k \end{pmatrix} \qquad (3.24)$$

In another ward, R_1 is a polynomial matrix of order 2.

Similarly, we can write R_1 and R_2 as polynomial matrices of order 2 as follows:

$$R_2 = \begin{pmatrix} 0 & I \\ -\dfrac{j+2s}{j+2s+1}I & \dfrac{2j+4s+1}{j+2s+1}N \end{pmatrix}\begin{pmatrix} 0 & I \\ -\dfrac{j+2s-1}{j+2s}I & \dfrac{2j+4s-1}{j+2s}N \end{pmatrix}$$

$$\cdots\begin{pmatrix} 0 & I \\ -\dfrac{j+s+1}{j+s+2}I & \dfrac{2j+2s+3}{j+s+2}N \end{pmatrix}$$

$$=\begin{pmatrix} a_{20}+a_{21}N+\cdots+a_{2s-2}N^{s-2} & b_{20}+b_{21}N+\cdots+b_{2s-1}N^{s-1} \\ c_{20}+c_{21}N+\cdots c_{2s-1}N^{s-1} & d_{20}+d_{21}N+\cdots+d_{2s}N^{s}\end{pmatrix}$$

and hence we get

$$R_2=\begin{pmatrix} \sum\limits_{k=0}^{s-2}a_{2k}N^k & \sum\limits_{k=0}^{s-1}b_{2k}N^k \\ \sum\limits_{k=0}^{s-1}c_{2k}N^k & \sum\limits_{k=0}^{s}d_{2k}N^k \end{pmatrix} \qquad (3.25)$$

$$R_3=\begin{pmatrix} 0 & I \\ -\frac{j+2s}{j+2s+1}I & \frac{2j+4s+1}{j+2s+1}N \end{pmatrix}\begin{pmatrix} 0 & I \\ -\frac{j+2s-1}{j+2s}I & \frac{2j+4s-1}{j+2s}N \end{pmatrix}$$

$$\cdots\begin{pmatrix} 0 & I \\ -\frac{j+s+1}{j+s+2}I & \frac{2j+2s+3}{j+s+2}N \end{pmatrix}$$

$$\begin{pmatrix} 0 & I \\ -\frac{j+s}{j+s+1}I & \frac{2j+2s+1}{j+s+1}N \end{pmatrix}\begin{pmatrix} 0 & I \\ -\frac{j+s-1}{j+s}I & \frac{2j+2s-1}{j+s}N \end{pmatrix}$$
$$\cdots\begin{pmatrix} 0 & I \\ \frac{j+1}{j+2}I & \frac{2j+3}{j+2}N \end{pmatrix}$$

$$=\begin{pmatrix} \sum\limits_{k=0}^{s-2}a_{2k}N^k & \sum\limits_{k=0}^{s-1}b_{2k}N^k \\ \sum\limits_{k=0}^{s-1}c_{2k}N^k & \sum\limits_{k=0}^{s}d_{2k}N^k \end{pmatrix}\begin{pmatrix} \sum\limits_{k=0}^{s-2}a_{1k}N^k & \sum\limits_{k=0}^{s-1}b_{1k}N^k \\ \sum\limits_{k=0}^{s-1}c_{1k}N^k & \sum\limits_{k=0}^{s}d_{1k}N^k \end{pmatrix}$$

Then we have

$$R_3=R_2R_1 \qquad (3.26)$$

This means that we can obtain the polynomial matrix R_3 of order 2 as a product of the two polynomial matrices R_1 and R_2 of order 2.

Here we can employ the fast polynomial multiplication technique to efficiently compute R_3 .We use this property to compute the whole storage of the algorithm

B. Computation of the storage of the algorithm

Now it can be easily seen that we need to compute all the matrices in the following array to fulfill the storage requirement for Algorithm the algorithm.

$$\begin{array}{ccccccc}
 & & & R_n\left(0.\frac{n}{2}\right) & & & \\
 & & R_{\frac{n}{2}}\left(0,\frac{n}{4}\right) & & R_{\frac{n}{2}}\left(\frac{n}{2},\frac{3n}{4}\right) & & \\
 R_{\frac{n}{4}}\left(0.\frac{n}{8}\right) & R_{\frac{n}{4}}\left(\frac{n}{4},\frac{3n}{8}\right) & & & R_{\frac{n}{4}}\left(\frac{n}{2},\frac{5n}{8}\right) & & R_{\frac{n}{4}}\left(\frac{3n}{4},\frac{7n}{8}\right) \\
 & & & \vdots & & & \\
 R_4(0,2) & R_4(4,6) & \cdots & & \cdots & & R_4(n-4,n-2)
\end{array}$$

Figure 3.1:Pre-computed storage required for the algorithm.

Using property 2 of 4.2 Properties of the matrix $R=R^{(2n)}(l-p-1,l)$, we may generate element in the above array.

1) Method of computation of the storage

Step 0: First compute the set $\left\{\begin{pmatrix} 0 & I \\ c_jI & a_jN+b_jI \end{pmatrix}: \text{for } j=1,2,\cdots,n-2\right\}$.

Step 1: Compute the set

$\left\{R^{(4)}(2j,2j+2) \mid \text{for } j=0,1,,\cdots,\frac{n}{2}-2\right\}$, where

$$R^{(4)}(2j,2j+2)$$

$$=\begin{pmatrix} 0 & I \\ -\frac{2j+2}{2j+3}I & \frac{4j+5}{2j+3}N \end{pmatrix}\begin{pmatrix} 0 & I \\ -\frac{2j+1}{2j+2}I & \frac{4j+3}{2j+2}N \end{pmatrix}$$

Here $R^{(4)}(2j,2j+2)$ may be computed more efficiently using the method of polynomial multination.

Then:

a. The set $\left\{R^{(4)}(2j,2j+2) \mid \text{for } j=0,2,4,\cdots,\frac{n}{2}-2\right\}$ gives all the elements required to be inserted in the bottom (first) level of the array.
b. The set $\left\{R^{(4)}(2j,2j+2) \mid \text{for } j=1,3,5,\cdots,\frac{n}{2}-3\right\}$ is required to compute the elements required to be inserted in the second level of the array.

Step 2: Compute the set

$\left\{R^{(8)}(4j,4j+4) \mid \text{for } j=0,1,,\cdots,\frac{n}{4}-2\right\}$, where

$$R^{(8)}(4j,4j+4)=R^{(4)}(4j+2,4j+4)R^{(4)}(4j,4j+2)$$

$$\text{for } j=0,1,2,\cdots,\frac{n}{4}-2.$$

This can be calculated more efficiently using the fast polynomial multiplication method.

Then:

a. The set $\left\{R^{(8)}(4j,4j+4) \mid \text{for } j=0,2,4,\cdots,\frac{n}{4}-2\right\}$ gives all the elements required to be inserted in the second level of the array.

b. The set $\left\{R^{(8)}(4j,4j+4) \mid \text{for } j=1,3,5,\cdots,\frac{n}{4}-3\right\}$ is required to compute the elements in the third level of the array.

And employ the same procedure until we get $R^{(2n)}\left(0,\frac{n}{2}\right)$.

3) Computational complexity of the storage

All the matrices in the tree of Figure 3.71 can be computed in $O\left(n(\log_2 n)^2\right)$ operations.

Poof

To compute $R^{(n)}\left(0,\frac{n}{2}\right)$ from $R^{(n/2)}\left(0,\frac{n}{4}\right)$ and $R^{(n/2)}\left(\frac{n}{2},\frac{3n}{4}\right)$, we have to perform eight polynomial multiplications with two polynomials. The degree of each polynomial is $\left(\frac{n}{4}-1\right)$ at most. Therefore, it requires $8\left[\frac{3n}{4}\log_2\left(\frac{n}{4}\right)+\frac{5n}{4}\right]+2n$ operations at most. Here the additional term $2n$ comes from the addition of polynomials.

Let $T(n)$ denote the number of operations, at most, required to compute the whole storage.

Then, we have:

$$T(n)=2T\left(\frac{n}{2}\right)+8\left[\frac{3n}{4}\log_2\left(\frac{n}{4}\right)+\frac{5n}{4}\right]+2n=6nk.$$

Iteration on n shows that

$$T(n)=6n\left[k^2-(1+2+\cdots+k)\right]$$

$$=6n\left[k^2-\frac{k}{2}(k+1)\right]$$

$$=3nk^2-3nk$$

$$\leq 6nk^2 \text{ since } k\geq 1.$$

Since $k=\log_2 n$, we get,

$$T(n)\leq 6n(\log_2 n)^2$$

whenever $n>1$.

This shows the computational complexity of the storage is $O\left(n(\log_2 n)^2\right)$. ■

Algorithm 3.1 : Pre-computation of the data structure of Legendre polynomial transform

---

INPUT: n is a power of 2

OUTPUT: $R^{(8)}(0,2),\cdots,R^{(8)}(n-4,n-2),\cdots,R^{(2n)}\left(0,\frac{n}{2}\right)$

COMPLEXITY : $O\left(n\log_2^2 n\right)$

STAGES:

```
0. Inserting data to the leaf nodes of the intial tree.
   for $j=1$ to $n-2$ do
       READ $\begin{pmatrix} 0 & I \\ c_j I & a_j N + b_j I \end{pmatrix}$
   endfor

1. Inserting the remining data to the node of the intial tree
   for $i=\log_2 n-1$ to 1 do
       for $j=1$ to $2^i-1$ do
           $R_1 \leftarrow$ Initialnode$(i+1,2j-1)$: polynomial matrix of order 2
           $R_2 \leftarrow$ Initialnode$(i+1,2j)$: polynomial matrix of order 2
```

Obtain the four polynomials $p, q, r,$ and s by applying the fast polynomial multiplication on the entities of R_1 and R_2

```
           $p \leftarrow R_2(1,1)R_1(1,1)+R_2(1,2)R_1(2,1)$
           $q \leftarrow R_2(1,1)R_1(1,2)+R_2(1,2)R_1(2,2)$
           $r \leftarrow R_2(2,1)R_1(1,1)+R_2(2,2)R_1(2,1)$
           $s \leftarrow R_2(2,1)R_1(1,2)+R_2(2,2)R_1(2,2)$
           $R_3 \leftarrow \begin{pmatrix} p & q \\ r & s \end{pmatrix}$
           Initialnode$(i,j) \leftarrow R_3$
       endfor
   endfor
```

Algorithm 3.2: Fast computation of Legendre polynomial transforms

---

INPUT:$f=(f_0,\cdots,f_{n-1})$: Vector with n is a power of 2.

OUTPUT: $F=(F(0),\cdots,F(n-1))$:Legendre polynomial transform of F.

COMPLEXITY: $O(n\log_2^2 n)$

STAGES:

```
0. Compute the vector $Z_0^{(n)}$
       $Z_0^{(n)} \leftarrow Vf^T$ ; compute by performing the fast Vandermond matrix-vector multiplication

1. for $k=0$ to $n-1$ do
       $Z_1^{(n)}(k) \leftarrow \frac{a_0}{\lambda_0} Z_0^{(n)}(k+1) + \frac{b_0}{\lambda_0} Z_0^{(n)}(k)$
       $Z_1^{(n)} \leftarrow (Z_1^{(n)}(0),\cdots,Z_1^{(n)}(n-1))$
   endfor
   newrootnode $\leftarrow \begin{pmatrix} Z_0^{(n)} \\ Z_1^{(n)} \end{pmatrix}$ ; inserting data to the root node of the new tree.

2. Inserting data to the remaining nodes of the new tree
   for $i=1$ to $\log_2 n-1$ do
   for $j=1$ to $2^{i-1}$
       $m \leftarrow \frac{n}{2^i}$
       $a \leftarrow$ newnode$(i-1,j)$
       $b \leftarrow$ Transpose of $(a(1),\cdots,a(m))$
       $c \leftarrow$ Transpose of $(a(2m+1),\cdots,a(3m))$
       $d \leftarrow \begin{pmatrix} b \\ c \end{pmatrix}$
       $R \leftarrow$ Initialnode$(i,j)$
       Obtain the following vectors $u$ and $v$ by applying the fast polynomial multiplication:
           $R(1,1) \leftarrow$ the first element in the first row of the polynomial matrix $R$
           $R(1,2) \leftarrow$ the second element in the first row of the polynomial matrix $R$
           $R(2,1) \leftarrow$ the first element in the second row of the polynomial matrix $R$
           $R(2,2) \leftarrow$ the second element in the second row of the polynomial matrix $R$
       $u \leftarrow R(1,1)\begin{pmatrix} a(1) \\ \vdots \\ a(2m) \end{pmatrix} + R(1,2)\begin{pmatrix} a(2m+1) \\ \vdots \\ a(4m) \end{pmatrix}$
       $v \leftarrow R(2,1)\begin{pmatrix} a(1) \\ \vdots \\ a(2m) \end{pmatrix} + R(2,2)\begin{pmatrix} a(2m+1) \\ \vdots \\ a(4m) \end{pmatrix}$
       $x \leftarrow$ Transpose of $(u(1),\cdots,u(m))$
       $y \leftarrow$ Transpose of $(v(1),\cdots,v(m))$
```

```
                $e \leftarrow \begin{pmatrix} x \\ y \end{pmatrix}$

        newnode$(i, 2j-1) \leftarrow d$

        newnode$(i, 2j) \leftarrow e$

            $E \leftarrow$ newnode ( $\log_2 n - 1, j$ )
            $F($evenindex$) \leftarrow E(1)$
            $F($odd index$) \leftarrow E(5)$
            Return
                $F \leftarrow (F(0), F(1), \cdots, F(n-1))$

      endfor

  endfor
```

IV. NUMERICAL EXPERIMENTS AND RESULTS

In this section, we present numerical results of numerical experiments carried out via numerical examples, in order to test the efficiency of the fast algorithm for the Legendre polynomial transforms. All the computations were carried out 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. The fast and direct algorithms were implemented using tree data structure array in MATLAB. We denote the CPU time taken by the straightforward algorithm by t(SA) and the corresponding time taken by the fast algorithm by t(FA). The CPU times mentioned each of the following tables are given in seconds and both array length(n) and CPU times (in milliseconds) are scaled by base 2 logarithm in each of the following graphs.

A Example 4.1.

In this example, we take $f = \left(1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{n}\right)$ as the given input vector and

$$X = \left\{ x_j = \cos \frac{j\pi}{n} : j = 0, 1, \cdots, n-1 \right\}$$

as the set of sample points. We compute the CPU time elapsed to calculate the sum $L(l) = \sum_{j=0}^{n-1} f_j p_l(x_j)$ for each $l = 0, 1, \cdots, n-1$.

Array length (n)	t(SA)	t(Fast)
16	0.05	0.39
32	0.08	0.44
64	0.23	0.49
128	0.76	0.73
256	4.41	1.51
512	30.82	4.27
1024	247.62	14.59
2048	1920.60	54.52
4096	*	213.92
8192	*	865.48

Table 4.1: CPU times taken by each algorithm for Example 4.1

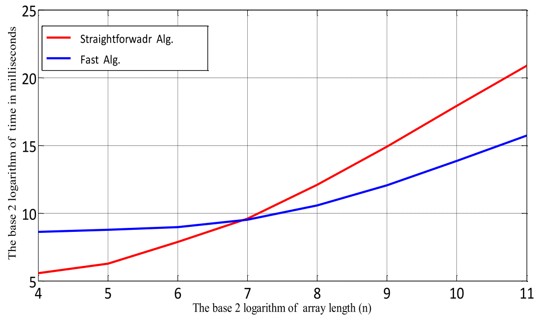

Fig 4.1. Graph of CPU time comparison for Example 4.1

B Example 4.2.

In this example, we take $f = (1, 2, 3, \ldots, n)$ as the given input vector and

$$X = \left\{ x_j = 2\frac{j}{n} - 1 : j = 0, 1, \cdots, n-1 \right\}$$

as the set of sample points. We compute the CPU time elapsed to calculate the sum

$$L(l) = \sum_{j=0}^{n-1} f_j p_l(x_j) \quad \text{for each } l = 0, 1, \cdots, n-1.$$

Array length(n)	t(SA)	t(FA)
16	0.06	0.40
32	0.09	0.45
64	0.16	0.50
128	0.81	0.78
256	4.60	1.54

512	31.39	4.30
1024	245.81	14.94
2048	1891.50	56.34
4096	*	222.78
8192	*	908.36

Table 4.2. CPU times taken by each algorithm for Example 4.2

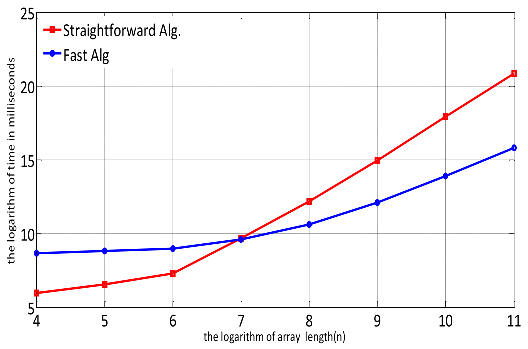

Fig 4.2. Graph of CPU time comparison for Example 4. 2

V. CONCLUSION

This paper presents a fast algorithm to compute Legendre polynomial transforms on set of arbitrary points. The heart of the design of the algorithm is based on a linear three-term recurrence satisfied by the Legendre polynomials transforms and a theorem presented in Driscoll JR et.al. (1997. We introduce a scalar factor λ to control overflow/underflow such that :

$$Z_l^{(n)}(k)=\left\langle f,(\lambda x_j)^k\, p_l(x_j)\right\rangle=\sum_{j=0}^{n-1} f_j(\lambda x_j)^k\, p_k(x_j).$$

For our computational purposes, we used

$\lambda=\|X\|$, the Euclidean norm of X,

where $X=\{x_0,x_1,\cdots,x_{n-1}\}$ is the set of sample points. This λ may necessarily affect sample points to control overflow and underflow for a considerable size of array length. However, the problem which we now have is that how to find a suitable value for λ that fits with a given problem. Formulating a criterion to find an appropriate λ that fits with a given Legendre polynomial transform will be future interests of our research work.

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.

Heideman MT, Johnson DH & 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.

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”.

BIOGRAPHY OF AUTHORS

[1]WA Gunarathna is a Mathematics lecturer (probationary) at General Sir John Kotelawala Defence University, Ratmalana, Sri Lanka. His research interests include design of efficient numerical algorithms for polynomial transforms and numeric solutions of diffrential equations. He is currently reading for an MPhill in Mathematics at the Postgraduate Institute of Science (PGIS), University of Peradeniya.

[2]Nasir HM is a senior lecturer of Mathematics at the Department of Mathematics, University of Perdeniya. He received Master of Engineering (MEng) and PhD 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.