A polynomial approach to orthogonal polynomial transforms

David K. Maslen

Max-Planck-Institut für Mathematik
Gottfried-Claren-Str. 26
53225 Bonn

Germany

MPI / 95-9

A POLYNOMIAL APPROACH TO ORTHOGONAL POLYNOMIAL TRANSFORMS

DAVID K. MASLEN

June 29, 1994

ABSTRACT. We present a new approach, based on polynomial arithmetic, to fast algorithms for the expansion of a function in orthogonal polynomial sequences. Such algorithms were recently constructed by Driscoll and Healy. Our treatment provides a clear formulation of the properties required by their algorithm, thus allowing variants of their algorithm to be developed in a more general context and under weakened conditions.

1. INTRODUCTION

Let $\{P_n\}$ be an orthogonal polynomial sequence, and let $x_0, \ldots, x_{N-1}$ be distinct real numbers. Then the orthogonal polynomial transform for this sequence is the map from function values, $f(x_0), \ldots, f(x_{N-1})$ of a polynomial of degree strictly less than N to the coefficients $a_0, \ldots, a_{N-1}$ of the expansion of f in the orthogonal polynomial sequence. Thus $a_0, \ldots, a_{N-1}$ are defined through the equations

$$f = \sum_{k=0}^{N-1} a_k P_k$$

where f is determined by the function values $f(x_0), \ldots, f(x_{N-1})$. Such transform occur frequently in applied sciences, particularly in spectral methods for the solution of partial differential equations [2].

A closely related computation is the calculation of the sums

$$\tilde{a}_k = \sum_{j=0}^{N-1} f(x_j) P_k(x_j)$$

for $0 \leq k < N$. This is the inverse transpose of the orthogonal polynomial transform and when $\{x_j\}$ are the roots of P_N it is equivalent to the orthogonal polynomial transform after taking scalar multiples of the $f(x_j)$ and the corresponding $\tilde{a}_k$. For more general sets of distinct points the polynomial transform may be calculated by means of an inverse transpose transform of twice the size. The main object of this paper is an algorithm, the Driscoll Healey algorithm, for the inverse transpose transform, though the orthogonal polynomial transform will feature in the derivations. We detail the relationship between the transform in section 2.

1991 *Mathematics Subject Classification.* 42C10.

The naive approach to these transforms via straightforward matrix multiplication takes N^2 scalar operations, where one operation counts for one multiplication plus one addition. There have been several attempts to find a more efficient algorithm, including [12] and [1] who derived approximate methods. In their seminal papers, Driscoll and Healey [5] together with Rockmore [6] proposed an exact $O(N(\log N)^2)$ algorithm for these transforms.

In the current paper we present a general derivation of their algorithm in the language of polynomial transforms. This clarifies the properties the algorithm depends on and removes some unnecessary hypotheses, in particular the hypothesis that an auxiliary sequence of polynomials they use has a constant coefficient recurrence relation.

For the purposes of this paper we ignore questions of stability and assume all operations are carried out in exact arithmetic. the stability of the algorithm depends strongly in the polynomial sequence and the points chosen to define the transform. The original algorithms of Driscoll and Healey have been implemented stably, with some modifications, in the important cases of Legendre and Gegenbauer polynomials. For a discussion and analysis of stabilization methods see [10].

An important example of an orthogonal polynomial transform occurs when the polynomial sequence is the sequence of Chebyshev polynomials. In this special case the transform of size N may be computed for any set of points in $O(N(\log N)^2)$ operations using classical fast Fourier transform techniques. When the points are the Chebyshev points, i.e. the roots of the N-th Chebyshev polynomial, these classical techniques give a transform of complexity $O(N \log N)$. Recently, Steidl and Tasche [13] gave an extremely efficient direct algorithm for this computation.

Loosely speaking, all the results in this paper are based on the idea of relating the polynomial transforms for two different sequences of orthogonal polynomials, showing that one may reduce the calculation of such a transform to the calculation of polynomial transforms at any other given orthogonal polynomial sequence. Hence an efficient algorithm for one class of orthogonal polynomial transforms yields algorithms for the polynomial transforms with respect to any other orthogonal polynomial sequence. The transforms for Chebyshev polynomials then provide a starting point for all other transforms.

In section 2 we introduce the properties of orthogonal polynomials, use them to relate orthogonal polynomial transforms at different sets of points, and to relate transforms and their inverse transposes. We introduce operators on the space of all polynomials and prove properties required for the development of our algorithms in an abstract setting. In section 3, we derive the basic algorithm and prove a complexity result relating the complexity of transforms with respect to two different sequences of orthogonal polynomials. In section 4 we give an example and an upper bound on the complexity of any inverse transpose transform. Then we generalize the properties we need for our algorithm in order to relate our results to those of [6]. Finally in section 5 we give an efficient algorithm for computing the precomputed data required for the polynomial transform algorithms.

Acknowledgements. Thanks to Persi Diaconis for involving me with this problem and his encouragement and advice. Thanks to Dennis Healey and Dan Rockmore for many conversations and explanations. Finally thanks to the Harvard math department and the Max-Planck-Institut für Mathematik for their crucial support.

2. Background

Let $\mathcal{P}$ denote the space of all polynomials with real coefficients and $\mathcal{P}_N$ denote the space of all such polynomials of degree strictly less than N. The definition of orthogonal polynomial sequence we shall use is that of [3], p. 7. Hence $\{P_n\}$ is an orthogonal polynomial sequence provided each P_n is a polynomial of degree n, $\mathcal{L}[P_m P_n] = 0$ if $m \neq n$, and $\mathcal{L}[P_n^2] \neq 0$, where $\mathcal{L}$ is a linear functional on the space of polynomials called the moment functional. We restrict ourselves to real polynomials and the case where the moment functional is positive definite. In this situation the moment functional is given by Stieltjes integration with respect to a nondecreasing integrator.

There are many different ways of representing polynomials, each useful for different operations. For multiplication the most convenient representation is the point value representation. If $x_0, \ldots, x_{N-1}$ is a sequence of N distinct points on the real line and f is a polynomial of degree strictly less than N, then the sequence of function values, $f(x_0), \ldots f(x_{n-1})$, is called the point value representation of f at the points $\{x_j\}$. The coefficient representation of f with respect to a sequence of orthogonal polynomials, $\{P_n\}$, is the sequence of coefficients $a_0, \ldots, a_{N-1}$, where

$$f = \sum_{k=0}^{N-1} a_k P_k$$

Evidently the orthogonal polynomial transform is the map from the point value representation of f to its coefficient representation.

This gives us a way to relate polynomial transforms at different sets of points. Suppose $\{T_n\}$ is an orthogonal polynomial sequence, and we have algorithms for calculating the polynomial transform for $\{T_n\}$ at the points $x_0, \ldots, x_{N-1}$ and the inverse transform at the points $y_0, \ldots, y_{N-1}$. Then composing these two maps gives us a map from the point value representation of f at the x_j to the point value representation of f at the y_j. Thus we may relate the polynomial transforms for a second set of polynomials, $\{P_n\}$, at the x_j, to the transform for $\{P_n\}$ at the points y_j. This is only a useful technique when we do have algorithms for the $\{T_n\}$ transforms. Fortunately when the T_n are Chebyshev polynomials we may use classical techniques to compute the transforms in $O(N(\log N)^2)$. The paper [6] contains an exposition of the technique and [10] [11] develop techniques for the stabilization of these algorithms.

One of the important properties of orthogonal polynomials we will use is the Gauss quadrature rule. For a proof of the following lemma see [3].

Lemma 2.1 (Gauss Quadrature). *Assume $\{P_n\}$ is an orthogonal polynomial sequence for the positive definite moment functional, $\mathcal{L}$, and $x_0^N, \ldots, x_{N-1}^N$ are the roots of P_N. Then there are numbers, $w_0^N, \ldots, w_{N-1}^N > 0$ such that for any polynomial, f, of degree at most $2N-1$, we have*

$$\mathcal{L}[f] = \sum_{j=0}^{N-1} w_j^N f(x_j^N)$$

The numbers w_j^N are unique and are called the Gaussian weights for the sequence $\{P_n\}$.

The Gaussian quadrature lemma immediately gives the connection between an orthogonal polynomial transform for $\{P_n\}$ at the roots of P_N, and the inverse transpose transform. In the notation of the introduction,

$$\begin{aligned} a_k &= \frac{1}{\mathcal{L}[P_n^2]}\mathcal{L}[f.P_k] \\ &= \frac{1}{\mathcal{L}[P_n^2]}\sum_{j=0}^{N-1} w_j^N f(x_j^N)P_k(x_j^N) \end{aligned}$$

When the points $x_0, \ldots, x_{N-1}$ are not the roots of P_N we need a different technique to relate the polynomial transform to its inverse transpose. In this case we use the transforms for the Chebyshev polynomials to evaluate the polynomial, f, at $2N$ distinct points, $y_0, \ldots y_{2N-1}$. Then we find real numbers $w_0, \ldots, w_{2N-1}$ such that

$$\sum_{j=0}^{2N-1} w_j P_k(y_j) = \mathcal{L}[P_0]\delta_{0,k}$$

for $0 \leq k < 2N$. Clearly the functional $\sum_{j=0}^{2N-1} w_j f(y_j)$ agrees with $\mathcal{L}$ on $\mathcal{P}_{2N}$, so in the notation of the introduction,

$$a_k = \sum_{j=0}^{2N-1} w_j f(y_j)P_k(y_j)$$

for $0 \leq k < 2N$. By this technique we may relate an orthogonal polynomial transform to an inverse transpose transform of twice the size at an arbitrary set of $2N$ distinct points.

The preceding argument justifies restricting our attention to the following special case. We assume from now on that $\{T_n\}$ and $\{P_n\}$ are orthogonal polynomial sequences, that $x_0^N, \ldots, x_{N-1}^N$ are the roots of T_N, and we shall restrict our attention to finding an algorithm for the inverse transpose transform for $\{P_n\}$ at the points, x_j^N, given that we have algorithms for the polynomial transform for the $\{T_n\}$ at the x_j^N and its inverse.

To fully relate our stated problem to polynomial transforms we need a method for finding numbers $w_0, \ldots, w_{N-1}$ satisfying

$$\sum_{j=0}^{N-1} w_j P_k(x_j^N) = \mathcal{L}[P_0]\delta_{0,k} \tag{1}$$

for $0 \leq k < N$. One method would be to solve the linear equations (1) directly though this could be time consuming. The following theorem gives a formula for these numbers in many situations.

Theorem 2.2. *Assume φ, ψ are infinitely supported Borel measures supported on a compact interval, and $\{P_n\}$, $\{T_n\}$ are real orthonormal polynomial sequences with respect to φ and ψ respectively. Assume φ is absolutely continuous with respect to*

ψ. Let $\mathcal{T}_N$ denote the projection from $L^1(\psi)$ onto $\mathcal{P}_N$ given by truncation of the formal expansion in the $\{T_n\}$, so

$$\mathcal{T}_N(h) = \sum_{k=0}^{N-1} \left[\int h.T_k d\psi \right] .T_k$$

for h in $L^1(\psi)$. Let $x_0^N, \ldots, x_{N-1}^N$ be the roots of T_N, let $u_0^N, \ldots, u_{N-1}^N$ be the corresponding Gaussian weights and let $w_0, \ldots, w_{N-1}$ be defined by (1), where $\mathcal{L}[P_0] = \int P_0 d\varphi$. Then

$$w_j = u_j^N . \left[\mathcal{T}_N(\frac{d\varphi}{d\psi}) \right] (x_j^N)$$

where $\frac{d\varphi}{d\psi}$ is the Radon Nikodym derivative.

Proof.

$$\begin{aligned}
&\sum_{j=0}^{N-1} u_j^N . \left[\mathcal{T}_N(\frac{d\varphi}{d\psi}) \right] (x_j^N) . P_k(x_j^N) \\
&= \int \mathcal{T}_N(\frac{d\varphi}{d\psi}) . P_k d\psi \\
&= \int \sum_{l=0}^{N-1} \left(\int \frac{d\varphi}{d\psi} . T_l d\psi \right) T_l . P_k d\psi \\
&= \int \sum_{l=0}^{N-1} \left(\int T_l . P_k d\psi \right) T_l . \frac{d\varphi}{d\psi} d\psi \\
&= \int P_k d\varphi = \mathcal{L}[P_0] \delta_{0,k}
\end{aligned}$$

for $0 \leq k < N$. □

Recurrence relations and truncation operators. Any orthogonal polynomial sequence satisfies a three term recurrence relation

$$P_{n+1} = (A_n x + B_n) . P_n + C_n . P_{n-1}$$

where $A_n \neq 0$ and $C_n \neq 0$. A similar property which may be obtained by iterating the recurrence is the Clebsch Gordan property, which is that

$$P_n . Q \in \text{span}\{P_{n-m}, \ldots, P_{n+m}\}$$

when $\deg Q \leq m$. Iterating the recurrence relation also shows that there are polynomials $Q_{l,m}$, $R_{l,m}$ with $\deg Q_{l,m} = m$ and $\deg R_{l,m} \leq m - 1$ such that

$$P_{l+m} = Q_{l,m} . P_l + R_{l,m} . P_{l-1}$$

These recurrence related properties together with the Gaussian quadrature formula form the basis of the algorithm.

To formulate our algorithm in its most general form we introduce the following operators, called truncation operators. These depend on the auxiliary orthogonal

polynomial sequence, $\{T_N\}$. For any polynomial, $f = \sum_{k\geq 0} b_k T_k$, define $\mathcal{T}_N f$ to be the projection onto $\mathcal{P}_N$ given by

$$\mathcal{T}_N f = \sum_{k=0}^{N-1} b_k T_k$$

Let $s = \frac{1}{\mathcal{M}(T_0)}\mathcal{M}$ where $\mathcal{M}$ is the moment functional for $\{T_n\}$. The important properties of $\mathcal{T}_N$ and s are as follows.

Lemma 2.3. (i) $\mathcal{T}_N^2 = \mathcal{T}_N$
(ii) $\mathrm{Im}\mathcal{T}_N = \mathcal{P}_N$
(iii) $\mathcal{T}_1 = s$
(iv) *If* $M \leq N$ *then* $\mathcal{T}_M \mathcal{T}_N = \mathcal{T}_M$.
(v) *If* $\deg Q \leq m \leq N$ *then* $\mathcal{T}_{N-m}(f.Q) = \mathcal{T}_{N-m}\left[(\mathcal{T}_N f).Q\right]$.

Proof. (i), (ii), (iii) and (iv) are trivial. For (v) assume that $f = \sum_{k\geq 0} b_k T_k$, and that $\deg Q \leq m$. By the Clebsch Gordan property we know that $T_k.Q$ is in the linear span of $T_{k-m}, \ldots, T_{k+m}$, and hence that $\mathcal{T}_{N-m}(T_k.Q) = 0$ for $k \geq N$. Therefore

$$\begin{aligned} \mathcal{T}_{N-m}(f.Q) &= \mathcal{T}_{N-m} \sum_{k=0}^{N-1} b_k T_k.Q \\ &= \mathcal{T}_{N-m}\left[(\mathcal{T}_N f).Q\right] \end{aligned}$$

□

We shall also use operators s_N and $\mathcal{S}_N$ defined as follows. Let

$$s_N f = \frac{1}{\mathcal{M}(T_0)} \sum_{j=0}^{N-1} u_j^N f(x_j^N)$$

where u_j^N are the Gaussian weights for $\{T_n\}$, and let $\mathcal{S}_N f$ denote the remainder of f modulo T_N, so $\mathcal{S}_N f$ is the unique polynomial of degree strictly less that N which agrees with f at the points $x_0^N, \ldots, x_{N-1}^N$. Here are the properties of s_N and $\mathcal{S}_N$ we will use.

Lemma 2.4. (i) $\mathcal{S}_N^2 = \mathcal{S}_N$
(ii) $\mathrm{Im}\mathcal{S}_N = \mathcal{P}_N$
(iii) $\mathcal{S}_N(f.g) = \mathcal{S}_N\left((\mathcal{S}_N f).(\mathcal{S}_N g)\right)$
(iv) *If* $\deg f \leq N + m$ *then* $\mathcal{T}_{N-m} f = \mathcal{T}_{N-m}\mathcal{S}_N f$
(v) *If* $\deg f < 2N$ *then* $s_N f = sf$.
(vi) $s_N = s_N \mathcal{S}_N$

Proof. (i), (ii), (iii) and (vi) are easy, and (v) follows from the Gauss quadrature formula. To prove (iv) assume $\deg f \leq N + m$. By long division, there is a polynomial, p, of degree at most m such that

$$f = \mathcal{S}_N f + T_N.p$$

By the Clebsch Gordan property we know $T_N.p$ is in the span of $T_{N-m}, \ldots, T_{N+m}$, and hence that $\mathcal{T}_{N-m}(T_N.p) = 0$. Therefore $\mathcal{T}_{N-m} f = \mathcal{T}_{N-m}\mathcal{S}_N f$. □

We may phrase the problem of finding the inverse transpose transform of a polynomial, f of degree strictly less than N in terms of the operators s_N and s. It is equivalent to finding the numbers

$$\mathcal{M}(T_0).s_N(f.P_l) = \mathcal{M}(T_0).s(f.P_l)$$

for $0 \leq l < N$.

3. The basic algorithm

We have reduced the computation of orthogonal polynomial transforms and their inverse transposes to the following problem. Given a linear operator, s, on $\mathcal{P}$, an orthogonal polynomial sequence $\{P_n\}$ and a polynomial, f, of degree strictly less than N, compute the numbers $s(f.P_l)$ for $0 \leq l < N$.

First we shall assume only that we know how to apply operators, $\mathcal{T}_n$, for $0 \leq n < N$ such that the $\mathcal{T}_N$ and s satisfy the properties of lemma 2.3. We give an algorithm that only requires polynomial multiplication, addition, and application of the operators $\mathcal{T}_n$. When the $\mathcal{T}_n$ are defined as in section 2 using an auxiliary polynomial sequence, $\{T_n\}$ we relate the complexity of the algorithm to the complexity of the orthogonal polynomial transform for $\{T_n\}$ assuming that the polynomial, f is given in point value representation at the Gaussian points of $\{T_n\}$. In this situation we also use the operators, $\mathcal{S}_n$, to limit the degrees of the polynomials to which we apply $\mathcal{T}_n$.

Assume $\mathcal{T}_n$, $0 \leq n < N$, and s satisfy the properties of lemma 2.3, that f is a polynomial and $\{P_l\}$ is an orthogonal polynomial sequence. The algorithm proceeds by computing polynomials $Z_l^K = \mathcal{T}_K(f.P_l)$ for various values of l and K; when f has degree strictly less than N we have $f = Z_0^N$, whereas $s(f.P_l) = Z_l^1$ for $0 \leq l < N$. Using property *iv.* of lemma 2.3 together with the recurrence relation

$$P_{l+m} = Q_{l,m}.P_l + R_{l,m}.P_{l-1}$$

one immediately obtains recurrences between the Z_l^K.

$$Z_{l+m}^{K-m} = \mathcal{T}_{K-m}\left[Z_l^K.Q_{l,m} + Z_{l-1}^K.R_{l,m}\right] \tag{2}$$

$$Z_{l+m}^{K-m-1} = \mathcal{T}_{K-m-1}\left[Z_l^K.Q_{l,m} + Z_{l-1}^K.R_{l,m}\right] \tag{3}$$

At this point it is convenient, though not strictly necessary, to require that N is a power of 2

We start by using the formulas $Z_0^N = \mathcal{T}_N f$ and $Z_1^N = \mathcal{T}_N(f.P_1)$ to find Z_0^N and Z_1^N; this is stage 0 of the algorithm. At stage k of the algorithm, with $k \geq 1$, we find the $Z_l^{N/2^k}$, $Z_{l-1}^{N/2^k}$ for $l = p(N/2^k) + 1$, $0 \leq p < 2^k$. When $p = 2q + 1$ is odd we use the recurrence (2) with $K = N/2^{k-1}$, $l = q(N/2^{k-1}) + 1$, $m = N/2^k$ and the recurrence (3) with $K = N/2^{k-1}$, $l = q(N/2^{k-1}) + 1$, $m = N/2^{k-1} - 1$ to find $Z_l^{N/2^k}$, $Z_{l-1}^{N/2^k}$ from the data at stage $k-1$. When p is even we simply use the relations $Z_l^{N/2^k} = \mathcal{T}_{N/2^k} Z_l^{N/2^{k-1}}$ and $Z_{l-1}^{N/2^k} = \mathcal{T}_{N/2^k} Z_{l-1}^{N/2^{k-1}}$. Once we have found these polynomials we no longer need the data form the previous stages. The algorithm terminates immediately after stage $k = \log_2 N - 1$, at which point the numbers Z_l^1 for $0 \leq l < N$ have all been computed.

Now we assume the operators $\mathcal{T}_n$, s and $\mathcal{S}_n$ are defined as in section 2 using an auxiliary polynomial sequence, $\{T_n\}$. In this situation we may use the operators,

$\mathcal{S}_n$, to reduce the complexity of the multiplication stages. By property iii. of lemma 2.4, we see that

$$\begin{aligned} Z_{l+m}^{K-m} &= \mathcal{T}_{K-m}\left[\mathcal{S}_K\left(Z_l^K.Q_{l,m}\right) + \mathcal{S}_K\left(Z_{l-1}^K.R_{l,m}\right)\right] \\ Z_{l+m}^{K-m-1} &= \mathcal{T}_{K-m-1}\left[\mathcal{S}_K\left(Z_l^K.Q_{l,m}\right) + \mathcal{S}_K\left(Z_{l-1}^K.R_{l,m}\right)\right] \end{aligned}$$

and we may use these recurrences in place of (2) and (3) in the algorithm. The advantage of this is that multiplication of a degree $K-1$ polynomial, Z, by a degree m polynomial, Q, takes at least $m+K$ scalar multiplications, whereas the computation $\mathcal{S}_K(Z.Q)$ takes only K multiplications in the appropriate point value representation.

In practice the truncation operators, $\mathcal{T}_n$, are best applied in the coefficient representation with respect to $\{T_n\}$ where they require no arithmetic operations. On the other hand, multiplications are best performed in point value representation. We now give the algorithm with the transformations between these representations explicitly included.

Assume f is a polynomial of degree strictly less than N given in point value representation at $x_0^N, \ldots x_{N-1}^N$, the roots of T_n. At stage 0 we apply a polynomial transform to f to get $Z_0^N = f$ in the coefficient representation for $\{T_n\}$. From now on all the polynomials Z_l^K will be given in this representation. To find Z_1^N we use the Clebsch Gordan relation for $P_1.T_k$ to relate the coefficient representation of Z_1^N to the coefficient representation of Z_0^N in $3N-2$ scalar multiplications and $2N-2$ scalar additions. At stage k we use 2^k inverse polynomial transforms of size $N/2^{k-1}$ to find $Z_l^{N/2^{k-1}}$, $Z_{l-1}^{N/2^{k-1}}$ in point value form at the points $x_j^{N/2^{k-1}}$ for $l = q(N/2^{k-1}) + 1$, $0 \leq q < 2^{k-1}$. The computations of the form $\mathcal{S}_{N/2^{k-1}}(Z.Q)$, $\mathcal{S}_{N/2^{k-1}}(Z.R)$ then take a total of $4N$ scalar multiplications in point value representation. Next we apply 2^k polynomial transforms of size $N/2^k$ and truncate the resulting sequences to obtain the $Z_l^{N/2^k}$ required for the next stage of computation. Counting all these operations gives the following theorem.

Theorem 3.1. *Assume c_N is a bound for the number of operations required to compute an orthogonal polynomial transform or inverse transform of a degree N polynomial with respect to the orthogonal polynomial sequence $\{T_n\}$ at the roots of T_N. Let $\{P_n\}$ be an orthogonal polynomial sequence. Then the inverse transpose transform of a degree N polynomial with respect to the sequence $\{P_n\}$ at the roots of T_N may be computed in less than*

$$c_N + 3\left(\sum_{k=1}^{\log_2 N-2} 2^k c_{\frac{N}{2^k}}\right) + 4N\log_2 N + (\frac{1}{2}c_2 - 1) - 2$$

operations.

4. Examples and generalizations

Assume the orthogonal polynomials, T_n, are the Chebyshev polynomials and the corresponding moment functional has total mass 1. Then $x_j^N = \cos\frac{(2j+1)\pi}{2N}$ and $u_j^N = \frac{1}{N}$. Steidl and Tasche [13] give an algorithm for the inverse orthogonal polynomial transform with respect to the Chebyshev polynomials in no more than $c_N = \frac{3}{2}N\log_2 N - N + 1$ scalar operations when N is a power of two (we count

one operations as a multiplication plus an addition and in this case the number of additions dominates). It is elementary to transpose their algorithm and obtain an algorithm for the forward orthogonal transform in the same number of operations. Thus we get an immediate corollary to theorem 3.1.

Corollary 4.1. *Assume $\{P_n\}$ is an orthogonal polynomial sequence. Then the inverse transpose transform of a degree N polynomial with respect to the sequence $\{P_n\}$ at the points $\cos\frac{(2j+1)\pi}{2N}$, $0 \le j < N$ may be computed in less than*

$$\frac{9}{4}N(\log_2 N)^2 + \frac{1}{4}N\log_2 N + \frac{9}{2}N - 7$$

scalar operations.

The basic algorithm also works, with minor modifications, in the following general situation. We are given operators $\mathcal{T}_M^K$ for $1 \le M \le K \le N$ such that

(i) $(\mathcal{T}_M^K)^2 = \mathcal{T}_M^K$
(ii) $\mathrm{Im}\mathcal{T}_M^K = \mathcal{P}_M$
(iii) $\mathcal{T}_1^N = s$
(iv) If $M \le K \le N$ then $\mathcal{T}_M^K \mathcal{T}_K^N = \mathcal{T}_M^N$.
(v) If $\deg Q \le m \le K \le N$ then $\mathcal{T}_{K-m}^N(f.Q) = \mathcal{T}_{N-m}^K\left[(\mathcal{T}_K^N f).Q\right]$.

The new algorithm uses the quantities $Z_l^K = \mathcal{T}_K^N(f.P_l)$, and the recurrences in this context are

$$\begin{aligned} Z_{l+m}^{K-m} &= \mathcal{T}_{K-m}^K\left[Z_l^K.Q_{l,m} + Z_{l-1}^K.R_{l,m}\right] \\ Z_{l+m}^{K-m-1} &= \mathcal{T}_{K-m-1}^K\left[Z_l^K.Q_{l,m} + Z_{l-1}^K.R_{l,m}\right] \end{aligned}$$

We may use this generalization to recover the original Driscoll Healey algorithm [5] as follows. For any power of two, N, define operators $\mathcal{C}_N$ that map a polynomial, f onto a finite sequence, $\mathcal{C}_N f(0), \ldots, \mathcal{C}_N f(N-1)$, of length N, where

$$\mathcal{C}_N f(k) = \sum_{j=0}^{N-1} f(\cos\frac{j\pi}{N})\cos\frac{jk\pi}{N}$$

Let $\hat{\mathcal{T}}_n$ denote the operator that truncates a sequence to length K, define $\mathcal{T}_M^K = \mathcal{C}_M^{-1}\hat{\mathcal{T}}_M\mathcal{C}_K$, and let $sf = \sum_{j=0}^{N-1} f(\cos\frac{j\pi}{N})$.

Lemma 4.2. *The operators $\mathcal{T}_M^K$, s just defined satisfy properties* (i)-(v).

Proof. Properties (i), (ii), (iii), and (iv) are easy. For (v) note that if f and Q are polynomials and $\deg Q \le m$ then the Clebsch Gordan property of Chebyshev polynomials shows that $\mathcal{C}_K(f.Q)(k)$ is a linear combination of $\mathcal{C}_K(f)(k-m), \ldots, \mathcal{C}_K(f)(k+m)$. The coefficients of the combination depend on Q but do not depend on f or K. Hence $\mathcal{C}_K((\mathcal{T}_K^N f).Q)$ is a linear combination of $\hat{\mathcal{T}}_K\mathcal{C}_N f(k-m), \ldots, \hat{\mathcal{T}}_K\mathcal{C}_N(k+m)$ and the coefficients of this linear combination are the same as those relating $\mathcal{C}_N(f.Q)(k)$ to $\mathcal{C}f(k-m), \ldots, \mathcal{C}_K f(k+m)$. If $k < K-m$ then $\hat{\mathcal{T}}_K\mathcal{C}_N f(l) = \mathcal{C}_N f(l)$ for $k-m \le k \le k+m$. Therefore

$$\hat{\mathcal{T}}_{K-m}\mathcal{C}_N(f.Q) = \hat{\mathcal{T}}_{K-m}\mathcal{C}_K\left[(\mathcal{T}_K^N f).Q\right]$$

Applying $\mathcal{C}_{K-m}$ to both sides gives the desired result. □

5. Precomputation

The algorithms described above assume we have found polynomials $Q_{l,m}$, $R_{l,m}$ such that

$$P_{l+m} = Q_{l,m}.P_l + R_{l,m}.P_{l-1} \tag{4}$$

and that these are given in point value representation. For the algorithm described in section 3 this amounts to finding the values $Q_{l,m}(x_j^K)$, $R_{l,m}(x_j^K)$ for $l = pN/2^k$, $m = N/2^{k+1}$ or $m = N/2^{k+1} - 1$, $K = N/2^k$ and $0 \leq k \leq \log_2 N$, $0 \leq j < K$, where x_j^K are the roots of the auxiliary polynomial T_K. The polynomials $Q_{l,m}$, $R_{l,m}$ are not uniquely defined for all values of l and m but we may still use recurrences to find polynomials with the right properties.

Lemma 5.1. *Assume we have defined polynomials $Q_{l+k,m-k}$, $R_{l+k,m-k}$, $Q_{l,k}$, $R_{l,k}$, $Q_{l,k-1}$, $R_{l,k-1}$ with $k \leq m$, so that $P_{r+t} = Q_{r,t}.P_r + R_{r,t}.P_{r-1}$ for the corresponding values of r and t. If we now define*

$$\begin{aligned} Q_{l,m} &= Q_{l+k,m-k}Q_{l,k} + R_{l+k,m-k}Q_{l,k-1} \\ R_{l,m} &= Q_{l+k,m-k}R_{l,k} + R_{l+k,m-k}R_{l,k-1} \end{aligned} \tag{5}$$

then $Q_{l,m}$, $R_{l,m}$ satisfy (4).

One could use the relations (5) with $k = 1$, together with the initial values $Q_{l,0} = 1$, $R_{l,0} = 0$, $Q_{l,1} = A_l x + B_l$, $R_{l,1} = C_l$, to find $Q_{l,m}$, $R_{l,m}$ for any l, m. However we only need these polynomials for particular values of l and m so this would be unnecessarily expensive in both time and memory. Fortunately the recursion (5) restricts to a smaller set of l, m values; by substituting $l, 2m, m$ or $l, 2m-1, m$ for l, m, k in (5) we obtain

$$\begin{aligned} Q_{l,2m} &= Q_{l+m,m}Q_{l,m} + R_{l+m,m}Q_{l,m-1} \\ R_{l,2m} &= Q_{l+m,m}R_{l,m} + R_{l+m,m}R_{l,m-1} \\ Q_{l,2m-1} &= Q_{l+m,m-1}Q_{l,m} + R_{l+m,m-1}Q_{l,m-1} \\ R_{l,2m-1} &= Q_{l+m,m-1}R_{l,m} + R_{l+m,m-1}R_{l,m-1} \end{aligned} \tag{6}$$

We use these recurrences to compute all the required $Q_{l,m}$, $R_{l,m}$ in $\log_2 N + 1$ stages. At stage 0 we are given the initial data $Q_{l,0}$, $R_{l,0}$, $Q_{l,1}$, $R_{l,1}$. At stage $k \geq 1$ we compute the values of $Q_{l,m}$, $R_{l,m}$ at the points $\{x_j^{2^k}\}$ for $l = p2^k + 1$ and $m = 2^k, 2^k - 1$, $0 \leq p < N/2^k$. To obtain the data at stage $k+1$ from the data at stage k, first find the values of $Q_{l,m}$, $R_{l,m}$ for stage k at the points $\{x_j^{2^{k+1}}\}$ using orthogonal polynomial transforms of size 2^k and inverse transforms of size 2^{k+1} with respect to the $\{T_n\}$. Next use the recurrence (6) with $m = 2^k$, $l = p2^{k+1} + 1$, $0 \leq p < N/2^{k+1}$ to find the $Q_{l,m}$, $R_{l,m}$ of stage $k+1$ at the points $\{x_j^{2^{k+1}}\}$. Stage 0 requires no computation as the results for $m = 0$ are part of the initial data. At the end of this we have in fact calculated twice as much data as we need; the results required are those for which p is even. We get the following complexity estimate for the precomputation.

Theorem 5.2. *Let all the notation be as in theorem* 3.1. *Then the precomputed data for the algorithm of that theorem may be computed in less than*

$$4c_N + 6N\left(\sum_{k=0}^{\log_2 N-1} \frac{c_{2^k}}{2^k}\right) + 8N\log_2 N$$

scalar operations.

When we use the Chebyshev polynomials as our auxiliary sequence we obtain

Corollary 5.3. *Let all the notation be as in theorem* 3.1 *and assume the sequence* $\{T_n\}$ *is the Chebyshev polynomials. Then the precomputed data may be computed in less than*

$$\frac{9}{2}N(\log_2 N)^2 + \frac{7}{2}N\log_2 N + 8N + 4$$

scalar operations.

REFERENCES

1. B. Alpert and V. Rokhlin, *A fast algorithm for the evaluation of Legendre transforms*, SIAM -J -Sci. -Statist. -Comput., **12**, No. 1, pp. 158-179 (1991).
2. G.L. Browning, J.J. Hack and P.N. Swarztrauber, *A comparison of three numerical methods for solving differential equations on the sphere*, Monthly Weather Review, **117**, pp. 1058-75, May (1989).
3. T.S. Chihara, *An Introduction to Orthogonal Polynomials*, Gordon and Breach, New York (1978).
4. P. Diaconis and D. Rockmore, *Efficient computation of the Fourier transform on finite groups*, Journal of the A.M.S. Vol. 3, No. 2, April (1990).
5. J.R. Driscoll and D.M. Healy Jr., *Computing Fourier Transforms and Convolutions on the 2-Sphere*, Report, Dept. of Math. and Com. Sci., Dartmouth College, NH (1992).
6. J.R. Driscoll, D.M. Healy Jr., and D. Rockmore, *Fast Spherical Transforms on Distance Transitive Graphs*, Report, Dept. of Math. and Com. Sci., Dartmouth College, NH (1992).
7. D.K. Maslen, *Fast Transforms and Sampling for Compact Groups*, Ph.D. Thesis, Harvard University, MA (1993).
8. D.K. Maslen, *Sampling of Functions and Sections for Compact Groups*, Preprint, Max-Planck-Institut-für-Mathematik, Bonn, Germany (1994).
9. A. Ghizetti and A. Ossicini, *Quadrature Formulae*, Birkenhäuser Verlag, Basel (1970).
10. S.S.B. Moore, *Efficient Stabilization Methods for Fast Polynomial Transforms*, Ph.D. Thesis, Dartmouth College, NH (1994).
11. S.S.B. Moore, D.M. Healey Jr. and D.N. Rockmore, *Symmetry Stabilization for Fast Discrete Monomial Transforms and Polynomial Evaluation*, Linear algebra and its applications, **192**, pp. 249-299 (1993).
12. S.A. Orszag, Fast eigenfunction tranforms, in Science and Computers, ed. G.C. Rota, Academic Press, NY, pp. 23-30 (1986).
13. G. Steidl and M. Tasche, *A polynomial approach to fast algorithms for discrete Fourier-cosine and Fourier-sine transforms*, Mathematics of Computation, **56**, No. 193, pp. 281-296, January (1991).

MAX-PLANCK-INSTITUT FÜR MATHEMATIK, 53225 BONN, GERMANY
E-mail address: maslen@mpim-bonn.mpg.de