FAST DISCRETE POLYNOMIAL TRANSFORMS WITH APPLICATIONS TO DATA ANALYSIS FOR DISTANCE TRANSITIVE GRAPHS

J. R. DRISCOLL[†¶], D. M. HEALY JR.[‡¶], AND D. N. ROCKMORE[§‖]

Abstract. Let $\mathcal{P} = \{P_0, \ldots, P_{n-1}\}$ denote a set of polynomials with complex coefficients. Let $\mathcal{Z} = \{z_0, \ldots, z_{n-1}\} \subset \mathbb{C}$ denote any set of **sample points**. For any $f = (f_0, \ldots, f_{n-1}) \in \mathbb{C}^n$ the **discrete polynomial transform** of f (with respect to $\mathcal{P}$ and $\mathcal{Z}$) is defined as the collection of sums, $\{\widehat{f}(P_0), \ldots, \widehat{f}(P_{n-1})\}$, where $\hat{f}(P_j) = \langle f, P_j \rangle = \sum_{i=0}^{n-1} f_i P_j(z_i) w(i)$ for some associated weight function w. These sorts of transforms find important applications in areas such as medical imaging and signal processing.

In this paper we present fast algorithms for computing discrete orthogonal polynomial transforms. For a system of N orthogonal polynomials of degree at most $N-1$ we give an $O(N \log^2 N)$ algorithm for computing a discrete polynomial transform at an arbitrary set of points instead of the N^2 operations required by direct evaluation. Our algorithm depends only on the fact that orthogonal polynomial sets satisfy a three-term recurrence and thus it may be applied to any such set of discrete sampled functions.

In particular, sampled orthogonal polynomials generate the vector space of functions on a distance transitive graph. As a direct application of our work we are able to give a fast algorithm for computing subspace decompositions of this vector space which respect the action of the symmetry group of such a graph. This has direct applications to treating computational bottlenecks in the spectral analysis of data on distance transitive graphs and we discuss this in some detail.

Key words. Fast Fourier Transform, FFT, discrete polynomial transform, orthogonal polynomials, three-term recurrence, distance transitive graph

AMS subject classifications. Primary 42C05, 42C10, 42-04, 33C90; Secondary 65T20, 62-04, 62-07, 05C99

1. Introduction.

The efficient decomposition of a function into a linear combination of orthogonal polynomials is a fundamental tool which plays an important role in a wide variety of computational problems. Applied science abounds with computations using such decompositions, along with the related computational techniques for calculation of correlation or projection of data onto a family of polynomials. To cite just a few examples, this sort of approach is used in spectral methods for solving differential equations [Bo, Te], data analysis [D], signal and image processing [OS], and the construction of Gauss quadrature schemes [Ga1]. In most cases, the choice of a particular family of polynomials is determined by some special property or underlying symmetry of the problem under investigation.

Perhaps the most familiar example is the representation of a discrete data sequence as a linear combination of phase polynomials. In this case the decomposition is known as the Discrete Fourier Transform (DFT) and is accomplished both efficiently and reliably through the use of the well-known Fast Fourier Transform algorithms (FFT) (cf. [ER] and the references therein). The DFT is a particularly simple orthogonal polynomial transform which corresponds to the projection of a data sequence $f = (f_0, \ldots, f_{N-1})$ onto the family of monomials $P_l(x) = m_l(x) = x^l$ evaluated at

[†]Driscoll Brewing, 81 Oakwood Drive, Murray Hill, NJ 07974, driscoll@cs.dartmouth.edu

[‡]Departments of Mathematics and Computer Science, Dartmouth College, Hanover, NH 03755, healy@cs.dartmouth.edu

[§]Departments of Mathematics and Computer Science, Dartmouth College, Hanover, NH 03755, rockmore@cs.dartmouth.edu

[¶]Partially supported by DARPA as administered by the AFOSR under contract AFOSR-90-0292.

[‖]Partially supported by NSF Mathematical Sciences Postdoctoral Fellowship.

the roots of unity $z_k = e^{2\pi ik/N}$, $k = 0, 1, ..., N-1$. Thus, the DFT is the collection of sums

$$\hat{f}(l) = \sum_{k=0}^{N-1} f_k P_l(z^k) = \sum_{k=0}^{N-1} f_k e^{2\pi ikl/N} \tag{1.1}$$

for the discrete frequencies $l = 0, 1, ...N-1$. The monomials form an orthogonal set whose properties account for the well-documented usefulness and algorithmic efficiency of the FFT algorithms. In particular, these algorithms allow the projections (1.1) to be computed in $O(N \log N)$ operations as opposed to the N^2 operations that a direct evaluation would require [ER]. (We assume a standard model in which a single complex multiplication and addition are defined as a single operation.)

In this paper we are concerned with the development of efficient algorithms for computing more general discrete polynomial transforms. Specifically, let $\mathcal{P} = \{P_0, \ldots, P_{n-1}\}$ denote a set of polynomials with complex coefficients. Let $\mathcal{Z} = \{z_0, \ldots, z_{n-1}\} \subset \mathbb{C}$ denote any set of **sample points**. If $f = (f_0, \ldots, f_{n-1})$ is any data vector (often thought of as a function with known values at the sample points), then the **discrete polynomial transform** of f (with respect to $\mathcal{P}$ and $\mathcal{Z}$) is defined as the collection of sums, $\{\widehat{f}(P_0), \ldots, \widehat{f}(P_{n-1})\}$, where

$$\hat{f}(P_j) = \langle f, P_j \rangle = \sum_{i=0}^{n-1} f_i P_j(z_i) w(i). \tag{1.2}$$

The function w is some associated weight function, often identically 1. Familiar examples of discrete polynomial transforms include the DFT (already mentioned) as well as the related Discrete Cosine Transform or DCT. In fact, both may be obtained as particular cases of **discrete monomial transforms** – i.e., discrete polynomial transforms in which $P_j = m_j$ is the monomial of degree j. Beyond such special cases we know of no prior general algorithm for computing discrete polynomial transforms which has complexity less than $O(n^2)$.

Inspection of equation (1.2) shows that direct computation of the discrete polynomial transform requires n^2 operations. For large n this cost quickly becomes prohibitive. The main result of this paper is an algorithm which computes general discrete orthogonal polynomial transforms in $O(n \log^2 n)$ operations. This relies primarily on the three-term recurrences satisfied by any orthogonal polynomial system and as such our algorithms also obtain for computing transforms over any set of spanning functions which satisfy such a recurrence. Related techniques have already found a number of applications attacking computational bottlenecks in problems in areas such as medical imaging, geophysics and matched filter design [DrH, MHR, HMR, HMMRT].

Our original motivation for studying these sorts of computations comes from problems which arise in performing spectral analysis of data on distance transitive graphs. This analysis is effectively the combinatorial analogue of the more familiar case of spectral analysis on continuous spaces like the circle or the 2-sphere. For instance, functions defined on distance transitive graphs admit a spectral decomposition which mirrors that of integrable functions on the 2-sphere. In particular, recall that the algebra of functions on the 2-sphere is generated by functions constant on circles of fixed distance from the north pole (circles of latitude), the so-called "zonal spherical functions" for the 2-sphere [He]. For each nonnegative integer m there is a uniquely defined (up to a constant) spherical function of degree m and the translates of this function under the action of $SO(3)$ (the symmetry group of the 2-sphere given by the group of rotations in 3-space) span a subspace of the vector space of functions on the 2-sphere which is invariant under the action of $SO(3)$. Similarly, distance transitive graphs have an associated symmetry group. After choice of a distinguished vertex, analogous to the choice of a "north pole" on the 2-sphere, the algebra of functions on a distance transitive graph is generated by analogously defined spherical functions (here distance on the graph is the usual shortest path distance between vertices). It turns out that these discrete functions are sampled orthogonal polynomials. Spectral analysis of data on a distance transitive graph, naturally viewed as a function on the graph, requires the expansion of the function in terms of a basis generated by the discrete spherical functions. The expansion may be reduced to the computation of discrete spherical transforms which are in fact discrete orthogonal polynomial transforms.

The spectral approach to data analysis, as described by Diaconis [D], is motivated by the observation that it is often appropriate and useful to view data as a function defined on an suitably chosen group, or more generally some homogeneous space of a group. Choice of a "natural" group in any given situation depends on various symmetries of the problem. The group theoretic setting of spectral analysis allows for the techniques of Fourier analysis to be applied. In particular, a data vector will have a natural decomposition into symmetry-invariant components which are calculated by computing the projections of the data vector into the various symmetry-invariant subspaces.

A familiar illustration of this approach comes from digital signal processing. Here, the standard analysis of stationary signals proceeds by decomposing the signal as a sum of sines and cosines with coefficients determined by usual abelian FFT. The sines and cosines of a given frequency determine subspaces of functions which are invariant under translation of the origin.

For a possibly less familiar example (due to Diaconis), consider the California Lottery game. Each player chooses a 6-element subset of $\{1, ..., 49\}$. Every such subset corresponds to a coset of $S_{49}/(S_6 \times S_{43})$ (here $S_6 \times S_{43}$ is identified with the subgroup of S_{49} that independently permutes the subsets $\{1, \ldots, 6\}$ and $\{7, \ldots, 49\}$). The vector space of functions defined on the cosets $S_{49}/(S_6 \times S_{43})$ is denoted as $M^{(43,6)}$. Each "run" of the game gives rise to a function $f \in M^{(43,6)}$ such that $f(x)$ is the number of people picking 6-set x.

A spectral analysis approach to analyzing such a data vector is to decompose the vector into symmetry-invariant components, where here a natural choice of symmetry group is the symmetric group S_{49}. Standard analysis from the representation theory of the symmetric group shows that $M^{(43,6)}$ has a unique finest decomposition into seven S_{49}-invariant components, $M^{(43,6)} = S^{(49)} \oplus S^{(48,1)} \oplus \ldots \oplus S^{(43,6)}$. This decomposition has a natural data analytic interpretation. The invariant subspace $S^{(49)}$ measures the constant contribution. The other invariant subspaces $S^{(49-j,j)}$ naturally measure the "pure" contribution of the popularity of the various j-sets (that is the number of people including a given j-set in their 6-set). Computation of the projections onto these subspaces can be reduced to computing the relevant spherical transforms which in this case turn out to be certain discrete Hahn polynomial transforms. The methods of this paper allow these transforms to be computed efficiently.

The California Lottery example is in fact an example of data on a distance transitive graph. More generally, the k-sets of an n-set comprise a distance transitive graph by joining any two k-sets which differ by only a single element. This graph possesses certain Hahn polynomials as its spherical functions (cf. [St3], Section

II.3). Other examples include the n-gon graph, with dihedral group symmetry as well as the n-dimensional hypercube with hyperoctahedral group of symmetries. In the former case the spherical functions are obtained from the Chebyshev polynomials, $T_n(x) = \cos(n \arccos(x))$ [Bi] and in the latter, the Krawtchouk polynomials give the spherical functions (cf. [St3], Section II.2).

As we shall see in Section 3, in general, the problem of finding an FFT for distance transitive graphs may be reduced to that of the efficient computation of the projection onto the spherical functions for the graph, which are an orthogonal family of special functions on the graph. In many important examples, (cf. [St1, St3] and the many references therein) these spherical functions are actually sampled orthogonal polynomials, and the spherical transform amounts to projection onto these polynomials in a weighted ℓ^2 space.

The organization of the paper is as follows: Section two discusses fast orthogonal polynomial transforms, beginning with previously known results for monomial transforms and concluding with our main computational result which is an efficient discrete orthogonal polynomial transform. This material is elementary, and relies on nothing more than the recurrence relations satisfied by the polynomials in question. Section three treats our main application of interest: fast algorithms for projection onto spherical functions on distance transitive graphs. We include here the necessary group theoretic background and notation, and give explicit examples of the algorithm for spherical functions on several graphs of interest. The fast spherical transform algorithm may be modified in order to provide a fast inverse transform, and from this we also obtain a fast convolution algorithm for functions on distance transitive graphs. Section four discusses the connection of these results to the computation of isotypic projections required for spectral analysis. We close in Section five with some final remarks.

We would like to thank the referees for their suggestions following a careful reading of the manuscript. We also thank Peter Kostelec and Doug Warner for their help with the final typesetting.

2. Fast Polynomial Transforms. The goal of this section is to produce algorithms for fast evaluation of polynomial transforms with an eye to their eventual application to the efficient computation of spherical transforms. The general algorithm proceeds in two steps. The initial phase is an efficient projection onto the monomials. From here we are able to use the three-term recurrence to obtain a divide and conquer approach for relevant fast polynomial transforms. In general, our approach is to formulate the initial problem as a particular matrix-vector multiplication and then present the fast algorithm as a particular matrix factorization.

We proceed by first recalling the fast monomial transform. This is obtained by writing it as the transpose of multiple-point polynomial evaluation and then formulating a well-known efficient algorithm for the latter process (cf. [BM], Chapter 4) as a structured matrix factorization. We explain the full algorithm next and then close this section with an example.

2.1. Fast monomial transforms. The simplest polynomial transform problem we could consider is the projection of a vector $f = (f_0, \ldots, f_{n-1})$ onto the family of monomials evaluated at the finite point set, $\{z_0, ..., z_{n-1}\}$,

$$\hat{f}(k) = \langle f, z^k \rangle = \sum_{\ell=0}^{n-1} f_\ell (z_\ell)^k \qquad (k = 0, ..., n-1). \tag{2.1}$$

Note that viewed as a matrix-vector multiplication, this is the evaluation of multiplication of the suitably defined Vandermonde matrix times the vector of samples. That is,

$$< f, z^k >= (V \cdot f)_k,$$

where

$$V = V(z_0, ..., z_{n-1}) = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ z_0 & z_1 & \cdots & z_{n-1} \\ . & & & . \\ . & & & . \\ . & & & . \\ {z_0}^{n-1} & {z_1}^{n-1} & \cdots & {z_{n-1}}^{n-1} \end{pmatrix}. \tag{2.2}$$

The familiar example of the abelian DFT is obtained by taking the evaluation points to be the n^{th} roots of unity in $\mathbb{C}$, $z_\ell = \exp(\frac{2\pi i \ell}{n})$. This projection may be obtained by the familiar FFT divide and conquer strategy in $O(n \log n)$ operations as opposed to the obvious $O(n^2)$. General references include [BM, ER, N, TAL].

Notice that the abelian FFT gives rise to our first efficient spherical transform, corresponding to the n-gon graph. This is a fast discrete Chebyshev transform with samples at the Chebyshev points, $\cos \frac{2\pi \ell}{n}$. This can be obtained by applying the usual FFT to a real-valued data sequence.

Our fast monomial transform is based on the formulation of the FFT which considers the *transpose* of projection and develops the algorithm as polynomial evaluation at the roots of unity,

$$\hat{f}(k) = (V^t \cdot f)_k = \sum_{\ell=0}^{n-1} f_\ell (z_k)^\ell$$

for $z_k = \exp(2\pi i k/n)$. This version of the FFT is achieved by efficient recursive application of the division algorithm (cf. [BM]).

An advantage of this perspective is that it allows an easy generalization to the direct evaluation of polynomials at n real points. We now review a well-known $O(n \log^2 n)$ algorithm for polynomial evaluation which we may formulate as a factorization of the matrix V^t into block diagonal matrices with Toeplitz blocks of geometrically decreasing size. It is this structure which permits the fast computation of the matrix-vector product. Consequently, we obtain a corresponding factorization of the transpose, V and hence, an algorithm for projection which also requires $O(n \log^2 n)$ operations. For ease of exposition, we assume n is a power of 2.

Lemma 2.1. *Let $n = 2^k$ and let V be the Vandermonde matrix for the set of complex points $z_0, z_1, ..., z_{n-1}$, as defined in (2.2). The matrix-vector product $V^t \cdot f$, for $f \in \mathbb{C}^n$ (corresponding to the evaluation of the polynomial $f_0 + f_1 z + \cdots + f_{n-1} z^{n-1}$ at the points $z_0, z_1, ..., z_{n-1}$) may be accomplished in $O(n \log^2 n)$ operations. Likewise, the product $V \cdot f$ for $f \in \mathbb{C}^n$ (corresponding to projection of a sampled function f onto the monomials sampled at $z_0, z_1, ..., z_{n-1}$) may be accomplished in $O(n \log^2 n)$ operations.*

Proof. Let $\phi(z) = \sum_{i=0}^{n-1} f_i z^i$, $n = 2^k$, with $k \geq 0$. We may evaluate ϕ at any of the z_j, $j = 0, 1, \ldots, n-1$ by the division algorithm because $\phi(z_j) = \phi(z) \mod (z - z_j)$. The division may be done in $O(n \log n)$ for a given z_j, but to proceed this way for each of the z_j separately is prohibitively expensive.

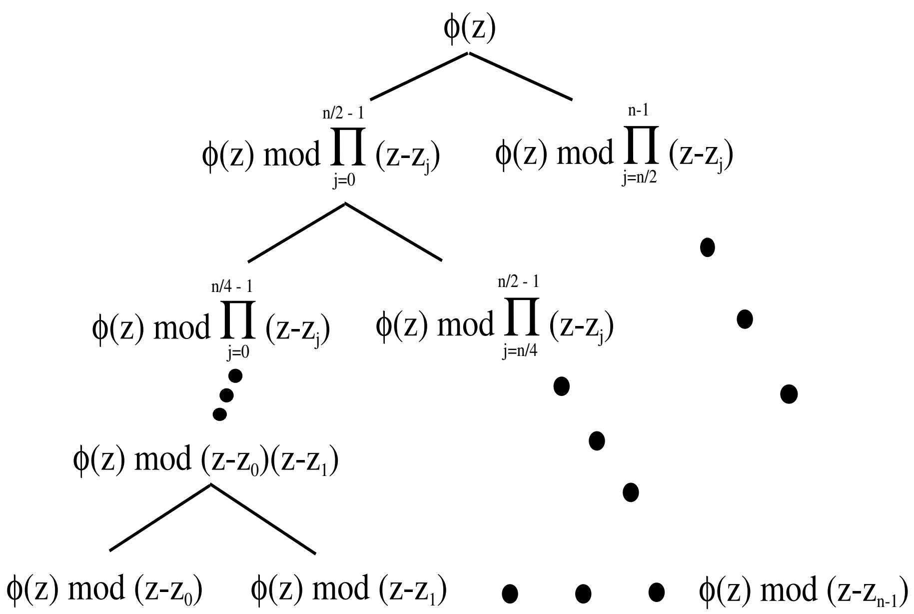

FIG. 2.1. *Tree for evaluating a polynomial $\phi(z)$ at $n = 2^k$ points. Note that it has $k = \log n$ levels.*

Instead we use a familiar divide and conquer strategy, simultaneously reducing the original polynomial modulo each linear factor $(z - z_j)$ in $k = \log n$ stages, as shown in Figure 2.1. Notice that fast polynomial arithmetic algorithms allow for the various moduli to be precomputed in $O(n \log^2 n)$ operations (cf. [BM], Section 4.3).

Each downward edge in the tree in Figure 2.1 represents the reduction of a polynomial $p(x)$ modulo another polynomial of form $m_S(z) = \Pi_{z_j \in S}(z - z_j)$, corresponding to a certain subset S of the evaluation points $z_0, ..., z_{n-1}$. To move down this edge of the tree, we need an algorithm to efficiently compute the remainder $r_S(x)$, and incidentally $q(x)$, in the division algorithm representation

$$p(x) = q(x)m_S(x) + r_S(x).$$

The input p, is a remainder from a previous stage, and has degree $d-1$, where d is a power of 2. The precomputed modulus, m_S has degree $d/2$. Therefore, r_S has degree $d/2 - 1$.

The key point is that in this tree, r_S is equivalent mod m_S not only to its immediate ancestor p, but also to every ancestor of p, all the way back to the original polynomial ϕ. Indeed, p was itself obtained as a remainder modulo $m_{\tilde{S}}$ from **its** ancestor P, and Figure 2.1 shows that always $S \subset \tilde{S}$, so $m_S \mid m_{\tilde{S}}$. Therefore, $m_S | (r_S - P)$. So, upon reaching the leaves of the tree, we have actually computed $\phi(z)$ modulo the linear factors $(z - z_j)$ as desired.

To see how to compute the basic reduction steps efficiently, we write the division algorithm representation $r = p - qm$ in matrix form. It is natural to split this equation into a high order and a low order part, due to the vanishing of the higher order coefficients of r, corresponding to powers $d/2, ..., d-1$. The low order equation involving the non-zero coefficients of r looks like

$$\begin{pmatrix} r_{\frac{d}{2}-1} \\ \vdots \\ \vdots \\ \vdots \\ r_0 \end{pmatrix} = \begin{pmatrix} p_{\frac{d}{2}-1} \\ \vdots \\ \vdots \\ \vdots \\ p_0 \end{pmatrix} - \begin{pmatrix} m_0 & m_1 & \cdots & \cdots & m_{\frac{d}{2}-1} \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & m_1 \\ 0 & \cdots & \cdots & 0 & m_0 \end{pmatrix} \begin{pmatrix} q_{\frac{d}{2}-1} \\ \vdots \\ \vdots \\ \vdots \\ q_0 \end{pmatrix}. \tag{2.3}$$

The upper triangular Toeplitz matrix in (2.3) is comprised of the lower order coefficients of the polynomial m; for future reference we call this matrix M.

Now, the higher order terms of r are zero, so the high order equation reduces to:

$$\begin{pmatrix} p_{d-1} \\ \vdots \\ \vdots \\ \vdots \\ p_{\frac{d}{2}} \end{pmatrix} = \begin{pmatrix} m_{\frac{d}{2}} & 0 & \cdots & \cdots & 0 \\ m_{\frac{d}{2}-1} & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ m_1 & \cdots & \cdots & m_{\frac{d}{2}-1} & m_{\frac{d}{2}} \end{pmatrix} \begin{pmatrix} q_{\frac{d}{2}-1} \\ \vdots \\ \vdots \\ \vdots \\ q_0 \end{pmatrix}. \tag{2.4}$$

Since $m_{d/2} = 1$, the lower triangular Toeplitz matrix in (2.4) is invertible. Its inverse, G, is also lower triangular and Toeplitz, and may be computed in $O(d \log d)$ operations by a Newton iteration and then prestored (cf. [BM], Chapter 4 as well as Remark 3 following the proof). Insert the result into equation (2.3). This gives

$$\begin{aligned} \begin{pmatrix} r_{\frac{d}{2}-1} \\ \vdots \\ r_0 \end{pmatrix} &= \begin{pmatrix} p_{\frac{d}{2}-1} \\ \vdots \\ p_0 \end{pmatrix} - MG \begin{pmatrix} p_{d-1} \\ \vdots \\ p_{\frac{d}{2}} \end{pmatrix} \\ &= \left(\begin{array}{c|c} -MG & I_{d/2} \end{array} \right) \begin{pmatrix} p_{d-1} \\ \vdots \\ p_{\frac{d}{2}} \\ \hline p_{\frac{d}{2}-1} \\ \vdots \\ p_0 \end{pmatrix}. \end{aligned} \tag{2.5}$$

Here $I_{d/2}$ is the $d/2 \times d/2$ identity matrix.

We must here briefly recall that standard techniques using the FFT allow Toeplitz matrices of dimension b to be multiplied by an arbitrary vector of length b in at most $O(b \log b)$ operations. This is done by framing the computation as the convolution of two sequences. More specifically, the Toeplitz matrix of order b is extended to a circulant matrix of order $2b$. A zero-padded version of the original data vector is then multiplied by this matrix in order to obtain the appropriate product. The new matrix-vector multiplication is precisely the circular convolution of two sequences of length $2b$ and as such is performed efficiently by computing the FFT's of each of the sequences, performing the pointwise multiplications of the resulting sequences and finally computing an inverse Fourier transform (requiring one more FFT) of this sequence. Thus, in total three FFT's are required as well as one pointwise multiplication of a sequence of length $2b$. If $2b = 2^r$ then an FFT of length $2b$ requires at most $\frac{3}{2} \cdot 2b \cdot r = 3br$ operations (cf. [BM], p. 84). Consequently, the multiplication of a Toeplitz matrix of order $b = 2^{r-1}$ by an arbitrary vector requires at most $3 \cdot 3br + 2b$ operations or $O(b \log b)$ operations.

As M and G are both Toeplitz, the above discussion shows that the product

$$M \cdot G \cdot \begin{pmatrix} p_{d-1} \\ \vdots \\ p_{\frac{d}{2}} \end{pmatrix}$$

can be computed in at most $2(9 \cdot \frac{d}{2} r + 2 \cdot \frac{d}{2}) = 9dr + d$ operations where $d = 2^r$. This is effected by first performing the multiplication

$$G \cdot \begin{pmatrix} p_{d-1} \\ \vdots \\ p_{\frac{d}{2}} \end{pmatrix}$$

and then multiplying this result by M. This means then that the multiplication in (2.5) may be accomplished in $9dr + d + \frac{d}{2} = 9dr + \frac{3d}{2}$ operations. The additional term of $\frac{d}{2}$ comes from the multiplication of the identity sub-block against the low order coefficients. Note that this is the cost of a single reduction in a single stage of the algorithm.

Looking back at the tree (see Fgure 2.1), we see that the first stage of the algorithm consists of two reductions from order $n = 2^k$ to order $n/2$, by two polynomial divisions. Consequently, if we let $T(n)$ denote the number operations required to compute the order n problem, then we obtain the following recurrence,

$$T(n) = 2T(\frac{n}{2}) + 2 \cdot (9nk + \frac{3}{2}n) = 18n \log n + 3n. \tag{2.6}$$

Iteration of the recurrence (2.6) yields

$$T(n) \leq 18n \log^2 n + 3n \log n \tag{2.7}$$

showing that the entire computation may be performed in $O(n \log^2 n)$ operations.

This sequence of reductions can be encoded as a structured matrix factorization of V^t. Let $M_{l,i}$ denote the upper triangular Toeplitz matrix associated (in the sense of (2.3)) to the polynomial which gives the i^{th} modulus at level l of the tree in Figure 2.1. Thus, $M_{1,0}$ is associated with $(z - z_0) \cdots (z - z_{n/2-1})$, $M_{1,1}$ is associated with $(z - z_{n/2}) \cdots (z - z_{n-1})$, $M_{2,0}$ is associated with $(z - z_0) \cdots (z - z_{n/4-1})$ and in general $M_{l,i}$ is associated with the product $(z - z_{i(n/2^l)}) \cdots (z - z_{(i+1)(n/2^l)-1})$.

Similarly, let $G_{l,i}$ denote **inverse** of the lower triangular Toeplitz matrix associated (in the sense of (2.4)) to the polynomial which gives the i^{th} modulus at level l of the tree in Figure 2.1. As discussed above $G_{l,i}$ is itself lower triangular and Toeplitz.

Define the matrices $R_{l,i}$ as in (2.5) by

$$R_{l,i} = \begin{array}{cc} \left(\begin{array}{c|c} -M_{l,i} G_{l,i} & I_{2^{k-l}} \end{array} \right) & \begin{array}{c} \uparrow \\ 2^{k-l} \\ \downarrow \end{array} \\ \begin{array}{cc} \leftarrow 2^{k-l} \rightarrow & \leftarrow 2^{k-l} \rightarrow \end{array} & \end{array} \tag{2.8}$$

for $1 \leq l \leq k = \log n$ and $0 \leq i < 2^l$. Then the previous discussion shows that V^t has a factorization into $k = \log n$ factors as the matrix product

$$\begin{pmatrix} \begin{pmatrix} R_{k,0} \\ - - - \\ R_{k,1} \end{pmatrix} & & & \bigcirc \\ & \begin{pmatrix} R_{k,2} \\ - - - \\ R_{k,3} \end{pmatrix} & & \\ & & \ddots & \\ \bigcirc & & & \begin{pmatrix} R_{k,n-2} \\ - - - \\ R_{k,n-1} \end{pmatrix} \end{pmatrix} \cdots$$

$$\cdots \begin{pmatrix} \begin{pmatrix} & R_{2,0} & \\ - & - & - \\ & R_{2,1} & \end{pmatrix} & \bigcirc \\ \bigcirc & \begin{pmatrix} & R_{2,2} & \\ - & - & - \\ & R_{2,3} & \end{pmatrix} \end{pmatrix} \begin{pmatrix} & R_{1,0} & \\ - & - & - \\ & R_{1,1} & \end{pmatrix}$$

By transposition, a similarly structured factorization of V is then also obtained. The reversal of order obviously does not change the complexity of the sequence of multiplications; each matrix is still block diagonal, with the blocks themselves comprised of products of triangular Toeplitz sub-blocks as before. $\Box$

Remarks. 1. As previously mentioned, Lemma 2.1 is a restatement of what is now a classical result of the complexity for polynomial evaluation. For variations on this algorithm as well as pointers to the more recent literature, we refer the reader to the survey article of Pan [P] and the extensive bibliography contained therein.

2. Our proof treats only the case of n equal to a power of 2, but may be extended to the general case in a straightforward manner with the same asymptotic result.

3. Notice that the above algorithm requires $O(n \log n)$ storage. To see this, recall that the matrix-vector multiplications involving the matrices $M_{l,i}$ and $G_{l,i}$ are effected by extending these matrices to the appropriate circulant matrices of twice the size and then performing the subsequent matrix-vector multiplications as circular convolutions using the FFT's of the associated sequences. The matrices $M_{l,i}$ and $G_{l,i}$ are of dimension 2^{k-l} (where $n = 2^k$) and thus are extended to circulants of size 2^{k-l+1}. We need only store the DFT of a single row of this circulant, so in total we require $2n \log n$ storage to keep the necessary data from the all of the $M_{l,i}$ and $G_{l,i}$.

To generate this initial data structure we require $O(n \log^2 n)$ operations. For this, we first note that to generate the necessary DFT's from all of the $M_{l,i}$ and $G_{l,i}$ we require at most $O(n \log^2 n)$ operations, assuming we have constructed the $M_{l,i}$ and $G_{l,i}$. The $M_{l,i}$ and the $G_{l,i}^{-1}$ are obtained from the polynomial coefficients of the various supermoduli in the division tree of Figure 2.1. These may be generated recursively from the bottom of the tree up, using efficient polynomial multiplication routines which require $O(m \log m)$ operations to multiply two polynomials of degree m.(cf. [BM], p. 86). Thus, at most $O(n \log^2 n)$ operations are needed to generate all

$M_{l,i}$ and the $G_{l,i}^{-1}$ (cf. [BM], p. 100). Finally, to invert any particular $G_{l,i}^{-1}$ in order to obtain $G_{l,i}$ an additional $O(l\,2^l)$ is needed (cf. [BM], p. 96), so that in total, we require $O(n\log^2 n)$ operations to precompute the necessary data structure.

4. Notice that one direct result of an efficient monomial transform is that we can obtain an FFT at nonuniformly spaced frequencies. This amounts to evaluating the polynomial above at n non-uniformly spaced points on the unit circle, and can be accomplished in $O(n\log^2 n)$ operations. An application of this to fast scanning for MRI is discussed in [MHR] as well as issues of stability of the fast algorithm.

Nonuniform FFT's also immediately provide an $O(n\log^2 n)$ Chebyshev polynomial transform on the uniform grid $\{\frac{k}{n}-1|k=0,...,2n-1\}$ in $[-1,1]$ by applying the non-uniform Fourier transform to a real data sequence f at the points $\exp(i\theta_k)$ with $\cos(\theta_k)=\frac{k}{n}$. This turns out to be useful, and we will explore it in more detail later.

Lemma 2.1 has many applications. We record some here for later use.

COROLLARY 2.2. *Each of the following three computations can be obtained in $O(n\log^2 n)$ operations:*

(1) The l^2 projections of a discrete function onto the monomials sampled at the points $x_0, x_1, ..., x_{n-1}$ in $\mathbb{R}$,

$$\sum_{k=0}^{n-1} f_k {x_k}^l, \quad l=0,...,n-1.$$

(2) The l^2 projections of a discrete function onto the Chebyshev polynomials $T_n(x)$ sampled uniformly at the points

$$\left\{u_k = 2\frac{k}{n}-1\middle|k=0,...,n-1\right\}\subset[-1,1],$$

$$\sum_{k=0}^{n-1} f_k T_l(u_k), \quad l=0,...,n-1.$$

(3) The l^2 projections of a discrete function onto the shifted Chebyshev polynomials $T_n^*(x)=T_n(2x-1)$, on the regular grid

$$\left\{v_k=\frac{k}{n}\middle|k=0,...,n-1\right\}\subset[0,1].$$

Proof. (1) Immediate application of the lemma.

(2) Take $\theta_k=\arccos(u_k)$ in $[0,\pi]$, $k=0,\ldots,n-1$. Define points z_j, $j=0,\ldots,2n-1$ in the unit circle by:

$$z_j=\begin{cases} e^{i\theta_j} & \text{if } 0\le j<n-1\\ e^{i(\theta_{j-n}-\pi)} & \text{if } n\le j<2n.\end{cases}$$

Then

$$\begin{aligned}\sum_{k=0}^{n-1} f_k T_l(u_k) &= \sum_{k=0}^{n-1} f_k\cos(l\theta_k)\\ &=\sum_{k=0}^{n-1}\frac{1}{2}f_k(z_k^l+\overline{z_k^l})\end{aligned}$$

$$= \sum_{j=0}^{2n-1} F(z_j) z_j^l,$$

where $F(z_0) = f_0$; $F(z_k) = \frac{1}{2} f_k$, $k = 1, \ldots, n-1$; $F(z_n) = 0$; and $F(\overline{z}) = F(z)$. The result follows by applying Lemma 2.1 to this last expression.

Alternatively, one may apply Lemma 2.1 separately to the real and imaginary parts of f, evaluating at the points z_j, $j = 0, \ldots, n-1$, and taking real part.

(3) Follows from (2) by a change of variables. □

For reasons of numerical stability, the projections described in the last two parts of the corollary provide a useful alternative to projection onto monomials on uniform grids. Even though the Chebyshev polynomials are not discretely orthogonal on the uniform grid, they still are much better conditioned than the monomials [Ga2, Hi]. It should also be noted that certain modifications of the resulting algorithm for projection onto the Chebyshev polynomials are required for stable computation. These modifications do not affect the efficiency of the algorithm in any appreciable way (cf. [MHR]).

2.2. Three-term recurrence relations and fast projection. We wish to extend the results of Section 2.1 to obtain an algorithm for the fast projection onto functions other than the monomials or the Chebyshev polynomials. In particular, we are interested in doing this for the spherical functions for distance transitive graphs. These functions satisfy three-term recurrence relations which permit us to make an efficient change of basis from monomials or Chebyshev polynomials. The following theorem demonstrates this in a case of interest for the current paper. It is evident that the argument can be applied in more general situations.

THEOREM 2.3. *Let $n = 2^k$ and let $\Phi_i(x)$, $i = 0, \ldots, n-1$ comprise a family of functions defined at the positive integers $x = 0, 1, \ldots, n-1$ and satisfying a three-term recurrence there:*

$$\Phi_{l+1}(x) = (a_l x + b_l)\Phi_l(x) + c_l \Phi_{l-1}(x),$$

with initial conditions $\Phi_0 = 1$, $\Phi_{-1} = 0$. Then the projections of a data vector $f = (f_0, \ldots, f_{n-1})$ defined by

$$\hat{f}(l) = \sum_{j=0}^{n-1} f_j \Phi_l(j) w_j = < f, \Phi_l >$$

where w is a weight function, can be computed for all $l < n$ in $O(n \log^2 n)$ operations.

Proof. Without loss of generality we may assume $w_i = 1$ for each i. (In the more general case the weights could be absorbed immediately into f.) By Lemma 2.1 we can effect the projection onto the monomials of degree less than n, sampled at the points $x = 0, 1, \ldots, n-1$ in $O(n \log^2 n)$ operations. We now use the three-term recurrence to transform these into the desired projections onto the Φ_l.

Define the sequence Z_l for each $l = 0, 1, \ldots, n-1$ by

$$Z_l(k) = < f, x^k \Phi_l > = \sum_{j=0}^{n-1} f_j j^k \Phi_l(j), \tag{2.9}$$

for $k = 0, 1, \ldots, n-1$. Our goal is to obtain the values $Z_l(0) = < f, x^0 \Phi_l >$. However, what we may compute efficiently from the initial data are the values $Z_0(k) = < f, x^k >$.

In terms of the Z_l, the recurrence,

$$\Phi_{l+1}(x) = (a_l x + b_l)\Phi_l(x) + c_l \Phi_{l-1}(x), \tag{2.10}$$

translates into

$$\begin{aligned} Z_{l+1}(k) &= a_l < f, x^{k+1}\Phi_l > + b_l < f, x^k \Phi_l > + c_l < f, x^k \Phi_{l-1} > \\ &= a_l Z_l(k+1) + b_l Z_l(k) + c_l Z_{l-1}(k). \end{aligned} \tag{2.11}$$

Observe that the weights in (2.11) do not depend on the k index. That is, the sequence Z_{l+1} is obtained by adding scalar multiples of the sequences Z_l, Z_{l-1}, and a shifted version of Z_l.

According to Lemma 2.1, and specifically, equation (2.7) we can compute the sequence Z_0 in $18n\log^2 n + 3n\log n$ operations. Setting $Z_{-1} = 0$, the recurrence (2.11) gives Z_1 in at most $2n$ additional operations. In particular this gives the value $Z_1(0) = \widehat{f}(1)$. Proceeding in this direct fashion one could successively build the sequences Z_l and the obtain the values $Z_l(0)$. This of course yields no savings, requiring n operations of length $2n$, so $O(n^2)$ in total.*

Instead, following [DrH] we are able to use a divide and conquer approach to solve the problem more efficiently. To explain this, it is instructive to view the computation graphically. For this, consider the $n \times n$ grid below, the l-axis in the horizontal direction and the k axis in the vertical direction. We can consider the function Z defined on the grid with values $Z(l,k) = Z_l(k)$. Using the recurrence (2.11), one sees immediately that the computation of $Z_l(k)$ (for $k < n-l$) only requires the prior computation of $Z_i(j)$ for (i,j) in the triangle defined by the vertices $(l,k), (0,k), (0, l+k)$.

Our goal is to compute the values $Z_l(0)$ for $0 \le l \le n-1$. As discussed above, initial computation of the first two columns, $\{Z_j(k) \mid j = 0,1 \text{ and } 0 \le k \le n-1\}$ can be obtained in $18n\log^2 n + 3n\log n + 2n$ operations. In particular, the values $Z_0(0)$ and $Z_1(0)$ are obtained.

To compute the remaining $Z_l(0)$ we wish to rewrite the recurrence (2.11) as a matrix equation. For any complex numbers α, β, γ define a $2n \times 2n$ matrix $T_n(\alpha,\beta,\gamma)$ by

$$T_n(\alpha,\beta,\gamma) = \begin{pmatrix} 0 & I \\ \gamma I & \beta I + \alpha N \end{pmatrix}$$

where I denotes the $n \times n$ identity matrix and N the $n \times n$ nilpotent matrix with ones on the superdiagonal and zeros elsewhere. Note that $T_n(\alpha,\beta,\gamma)$ is a block matrix composed of four $n \times n$ Toeplitz blocks. With this notation, the recurrence (2.11) may be rewritten as

$$\begin{pmatrix} Z_l \\ Z_{l+1} \end{pmatrix} = T_n(a_l, b_l, c_l) \cdot \begin{pmatrix} Z_{l-1} \\ Z_l \end{pmatrix}. \tag{2.12}$$

Iteration of the recurrence is then realized as a product of such matrices and so for any m,

$$\begin{pmatrix} Z_l \\ Z_{l+1} \end{pmatrix} = R_n(l-m-1, l) \cdot \begin{pmatrix} Z_{l-m-1} \\ Z_{l-m} \end{pmatrix}. \tag{2.13}$$

*Strictly speaking, the recurrence can be applied to the initial sequence Z_0 to obtain the correct values of $Z_l(k)$ for $k < n-l$. For example, to get $Z_1(n-1)$ with equation (13) requires the value of $Z_0(n)$, which we do not have. This "edge effect" propagates as l increases, but does not affect the values of the sequences that we actually need for the algorithm.

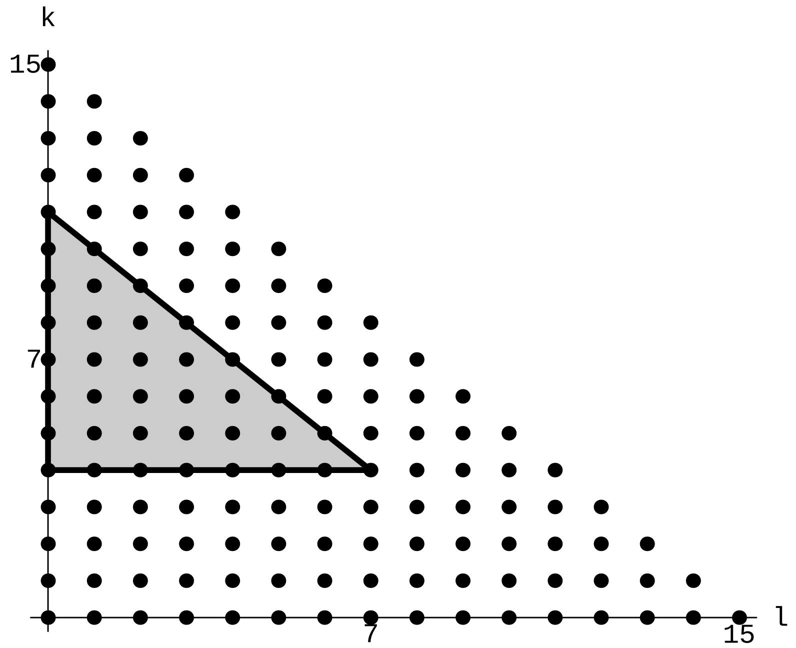

FIG. 2.2. *Computation of $Z_6(4)$ depends only on the computation of $Z_i(j)$ for (i,j) in the shaded triangle.*

The product

$$R_n(l-m-1,l) = T_n(\alpha_l,\beta_l,\gamma_l)\cdots T_n(\alpha_{l-m-1},\beta_{l-m-1},\gamma_{l-m-1})$$

is still a block matrix made up of four $n\times n$ Toeplitz blocks. Consequently, the values of Z_{l-1} and Z_l can be computed from those of Z_{l-m-1} and Z_{l-m} by a single matrix-vector multiplication using a $2n\times 2n$ matrix with $n\times n$ Toeplitz blocks.

In particular,

$$\begin{pmatrix} Z_{\frac{n}{2}} \\ Z_{\frac{n}{2}+1} \end{pmatrix} = R_n(0,\frac{n}{2})\cdot\begin{pmatrix} Z_0 \\ Z_1 \end{pmatrix}. \tag{2.14}$$

Recalling the discussion within the proof of Lemma 2.1, multiplication of an $n\times n$ Toeplitz matrix by a vector may be performed by standard FFT techniques using at most $9n(1+\log n)+2n$ operations. Four such matrix-vector multiplications are required to compute (2.14) so that an additional $36n(1+\log n)+8n$ operations are required to compute $Z_{\frac{n}{2}}$ and $Z_{\frac{n}{2}+1}$, and in particular, $Z_{\frac{n}{2}}(0)$ and $Z_{\frac{n}{2}+1}(0)$ from our initial data of Z_0 and Z_1.

The point of this is to decompose the problem into two half-sized subproblems; we shall compute the Z_l for $l>\frac{n}{2}+1$ by applying equation (15) to $Z_{\frac{n}{2}}$ and $Z_{\frac{n}{2}+1}$. As we indicated previously in Figure 2.2, the the values $Z_l(0)$ for $l>\frac{n}{2}+1$ only depend on the initial half segments of the sequences $Z_{\frac{n}{2}}$, and $Z_{\frac{n}{2}+1}$. Similarly, the the values $Z_l(0)$ for $l<\frac{n}{2}$ may be computed by applying the recurrence to the initial half segments of the sequences Z_0, and Z_1. This is re-exhibited in Figure 2.3 in which the diagonal lines display the dependence of the desired output $Z_l(0)$ on the various "subtriangles" in the grid for a problem of size $n=16$.

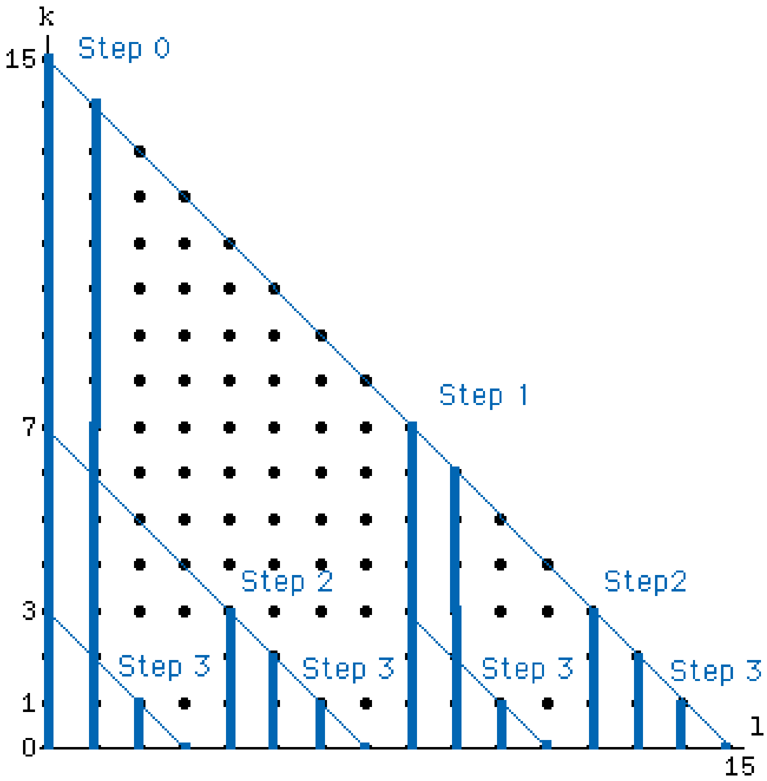

FIG. 2.3. *Computation of the $Z_l(0)$ for $l < n$ by a cascade of convolutions of of decreasing size. The relevant ranges of Z are highlighted, and the step in which they are calculated is indicated.*

Consequently, we see that to continue to obtain the remaining $Z_l(0)$, we need only keep $Z_j(k)$ for $0 \leq k < n/2$ and $j = 0, 1, \frac{n}{2}$, and $\frac{n}{2}+1$. Thus, step two proceeds by throwing away half of each of the sequence $Z_0, Z_1, Z_{\frac{n}{2}}$, and $Z_{\frac{n}{2}+1}$ and then computing $Z_{\frac{n}{4}}(k), Z_{\frac{n}{4}+1}(k), \quad (0 \leq k < n/2)$ from the truncated sequences Z_0 and Z_1 and similarly, $Z_{\frac{3n}{4}}(k), Z_{\frac{3n}{4}+1}(k), \quad (0 \leq k < n/2)$ from the truncated sequences $Z_{\frac{n}{2}}$ and $Z_{\frac{n}{2}+1}$.

At the end of step two, we own the first halves of the sequences $Z_0, Z_1; Z_{\frac{n}{4}}, Z_{\frac{n}{4}+1}$; $Z_{\frac{n}{2}}, Z_{\frac{n}{2}+1}; Z_{\frac{3n}{4}}, Z_{\frac{3n}{4}+1}$. Again, we throw away the latter halves of each (half) sequence and continue by performing four multiplications by Toeplitz matrices of size $\frac{n}{4}$, and so on.

All of this is illustrated again by Figure 2.3 for a problem of size $n = 16$ in which we have indicated which values in the grid we have obtained after each step in the algorithm. Thus it shows that step 0 results in the sequences Z_0 and Z_1. After step 1, we have obtained additionally the sequences Z_8 and Z_9 which we then truncate in half while also cutting in half the sequences Z_0 and Z_1. From this subset of data we can then compute one quarter of the sequences Z_4, Z_5, Z_{12}, and Z_{13} and, after truncating each of the previous data sequences in half, a quarter of the sequences Z_0, Z_1, Z_8 and Z_9 as well. Finally, in the last step we obtain the remaining pieces, one eighth of the sequences $Z_2, Z_3, Z_6, Z_7, Z_{10}, Z_{11}, Z_{14}$ and Z_{15}.

The complexity of the algorithm follows easily from an argument similar to that of Lemma 2.1. The process of "throwing away" is just a standard projection, so even if we include it in our estimate it requires at most an additional $4n$ operations at the first step. Having cut the vectors $Z_0, Z_1, Z_{n/2}$ and $Z_{n/2+1}$ in half, we now have two identical subproblems half the size of the original problem in which we computed $Z_{n/2}$ and $Z_{n/2+1}$ from Z_0 and Z_1. Thus, if we let $T(n)$ denote the number of operations needed to compute the elements $Z_j(0)$ from the initial data of Z_0 and Z_1, then we obtain the recurrence

$$(2.15)\quad T(n) = 36n(1+\log n) + 8n + 4n + 2T(\frac{n}{2}) = 48n + 36n\log n + 2T(\frac{n}{2}).$$

Iterating (2.15) yields,

$$(2.16)\qquad T(n) \leq 48n\log n + 36n\log^2 n.$$

Finally, throwing in the operations needed to initially compute Z_0 and Z_1, we see that at most $48n\log n + 36n\log^2 n + 9n(1+\log n) + 2n = O(n\log^2 n)$ operations are needed to compute the projections. □

Remarks. 1. As regards the storage requirements of the precomputed data structure, it is clear that all that is needed are the various Toeplitz matrices in the array:

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

In fact, each should be suitably augmented to circulant matrices in order to effect an efficient matrix-vector multiplication. Analysis similar to that used in Remark 3 of Lemma 2.1 shows that this will require $O(n\log n)$ storage.

2. The Toeplitz matrices in the array above may be generated in $O(n\log^2 n)$ operations. The idea here is to build the larger R matrices at the top of this array from the smaller matrices lower down which will have already been computed.

We start by filling in the bottom level of the array, building all the matrices of the form $R_4(2j, 2j+2)$ Notice that we actually only need every other one of these for the lowest level of our data structure, but the rest are required for building the next level. These matrices may be combined pairwise, as detailed below, in order to obtain the matrices at the next level. Explicitly, we combine $R_4(4j, 4j+2)$. with $R_4(4j+2, 4j+4)$ to obtain $R_8(4j, 4j+4)$. Again, only half of these results are needed to fill out the second level of the data structure, and the rest are required for building the third level. Continuing in this fashion we end up with all the matrices we need, up to $R_n(0, \frac{n}{2})$.

The basic step is: given matrices $R_p(j, j+m)$ and $R_p(j+m, j+2m)$, determine $R_{2p}(j, j+2m)$ efficiently. To see this, it is helpful to note that each of the four Toeplitz blocks of one of these matrices, say $R_p(j, j+m)$, may be written as a polynomial expression in the powers of the nilpotent matrix N, of degree no more than m. The blocks are completely determined by the coefficients of these polynomials, and multiplication of a pair of R matrices corresponds to 2 by 2 matrix multiplication using polynomial arithmetic on the entries. Thus, the entries of $R_{2p}(j, j+2m)$ may be computed from those of $R_p(j, j+m)$ and $R_p(j+m, j+2m)$, using fast polynomial arithmetic for polynomials of degree no more than m. Therefore this may be done in $O(m\log m)$ operations.

The complexity of obtaining the entire array is now determined as in several similar calculations we have done earlier; at the l'th level (starting at the bottom) of $\log n$ levels, we have $\frac{n}{2^l}$ matrices to compute at $O(l\ 2^l)$ complexity each. This leads to the given complexity of $O(n \log^2 n)$ for the entire array.

2.3. Some practical considerations and an example. The approach of Section 2.2, while theoretically interesting, is in fact numerically rather suspect. Part of the problem comes from the step of first projecting the data vector onto the monomials. These functions, while linearly independent in exact arithmetic, are so close to being dependent as to be nearly useless in practice. See, for example, [Ga2] for a discussion of the condition number of expansions of polynomial functions in the monomial basis and other bases. Numerical experiments confirm that when the algorithm presented above is implemented in floating point arithmetic it can produce very unreliable answers for problems of modest size.

To treat this potential problem, we now prove a slight generalization of the recurrence technique of the last section that permits us to replace the monomials with other polynomial bases that satisfy simple constant coefficient recurrence relations. In particular, we have in mind the shifted Chebyshev polynomials. They satisfy the recurrence

$$T^*_{n+1} = (4x-2)T^*_n(x) - T^*_{n-1}(x). \tag{2.17}$$

Such a recurrence can be run in either the forward direction or the backward direction, in which case we obtain

$$T^*_{n-1}(x) = (4x-2)T^*_n(x) - T^*_{n+1}(x).$$

Consequently, we see that running the recurrence backwards from 0, allows the definition of $T^*_{-k}(x) = T^*_k(x)$ for all values of k. Notice that in general $T^*_k(x)$ is a polynomial of degree $|k|$. Equality for negative and positive indices follows from the fact that the recurrence is the same in either direction.

More generally, any constant coefficient recurrence can be run in either direction, producing polynomials of degree $|k|$ for index k, assuming initial conditions that dictate polynomials of degree 0 and 1 for indices 0 and 1 respectively. For example, if a system satisfies the recurrence

$$p_{k+1}(x) = (\alpha x + \beta)p_k(x) + \gamma p_{k-1}(x),$$

then we obtain the "backwards recurrence"

$$p_{k-1}(x) = -\frac{1}{\gamma}(\alpha x + \beta)p_k(x) + \frac{1}{\gamma}p_{k+1}(x).$$

This simple observation allows us to couple our algorithm with polynomials that satisfy such recurrences.

THEOREM 2.4. *Suppose that the polynomial families* $\{p_k(x) | -n \le k < n\}$ *and* $\{\Phi_l(x) | l = 0, ..., n-1\}$ *satisfy three-term recurrences*

$$p_{k+1}(x) = (\alpha x + \beta)p_k(x) + \gamma p_{k-1}(x) \tag{2.18}$$

$$\Phi_{l+1}(x) = (a_l x + b_l)\Phi_l(x) + c_l\Phi_{l-1}(x), \tag{2.19}$$

with $deg(p_k) = |k|$ and $\Phi_0 = 1$, and set $\Phi_{-1} = 0$. Suppose that the projections $< f, p_k >$ are known, $-n \leq k < n$. From this the projections $< f, \Phi_l >$, $l = 0, ..., n-1$ can be computed in $O(n \log^2 n)$ operations.

Proof. Again, we may assume without loss of generality that the weight function is identically one. Define the sequence Z_l for each $l = 0, 1, ..., n-1$ by

$$Z_l(k) = < f, p_k \Phi_l >, \tag{2.20}$$

for $k = -n, \ldots, 0, 1, ..., n-1$. Our goal is to obtain the values $Z_l(0) = < f, p_0 \Phi_l >$. Instead, we have the values $Z_0(k) = < f, p_k \Phi_0 >$. We hope to proceed by convolution, as in Theorem 2.3, and push up from Z_0 to the higher sequences Z_l.

The recurrence (2.19) for the Φ_l shows that for $l > 0$,

$$\begin{aligned} Z_{l+1}(k) &= a_l < f, x p_k \Phi_l > + b_l < f, p_k \Phi_l > + c_l < f, p_k \Phi_{l-1} > \\ &= a_l < f, x p_k \Phi_l > + b_l Z_l(k) + c_l Z_{l-1}(k). \end{aligned}$$

Now use the recurrence (2.18) for the p_k to see that

$$\begin{aligned} < f, x p_k \Phi_l > &= \frac{1}{\alpha} < f, p_{k+1} \Phi_l > - \frac{\beta}{\alpha} < f, p_k \Phi_l > - \frac{\gamma}{\alpha} < f, p_{k-1} \Phi_l > \\ &= \frac{1}{\alpha} Z_l(k+1) - \frac{\beta}{\alpha} Z_l(k) - \frac{\gamma}{\alpha} Z_l(k-1). \end{aligned}$$

Therefore

$$\begin{aligned} Z_{l+1}(k) &= \frac{a_l}{\alpha} \left[Z_l(k+1) + (\frac{\alpha}{a_l} b_l - \beta) Z_l(k) - \gamma Z_l(k-1) \right] + c_l Z_{l-1}(k) \quad (2.21) \\ &= u_l Z_l(k+1) + v_l Z_l(k) + w_l Z_l(k-1) + c_l Z_{l-1}(k) \quad (2.22) \end{aligned}$$

Observe that the weights in the expression (2.22) are independent of k. As in the case of Theorem 2.3, the sequence Z_{l+1} is obtained from the sequences Z_l and Z_{l-1} by convolving each with a fixed mask and then adding the resulting vectors. However, there is a difference. To describe this, it is again, instructive to view the computation graphically. Following Theorem 2.3 we we consider a function Z defined on a grid described by l and k axes such that $Z(l, k) = Z_l(k)$. From this point of view the recurrence (2.22) indicates that a given value $Z_{l+1}(k)$ only depends on knowing any two adjacent vertical lines of data contained within the triangle determined by the vertices $(l, k), (0, k+l)$ and $(0, k-l)$. Figure 2.4 displays an example.

Because in this case, the recurrence (2.22) both "reaches" down and up in k, slight modifications of the proof of Theorem 2.3 are required. As the complexity counts are very similar, we will only point out the major changes and leave the details to the interested reader.

In analogy with Theorem 2.3, our goal is to express the full computation as a divide and conquer algorithm. Figure 2.5 indicates how this can be accomplished. Starting with the full data of Z_0 and Z_1, we will construct $Z_{\frac{n}{2}}$ and $Z_{\frac{n}{2}+1}$. The values Z_j for $j < \frac{n}{2} - 1$ only depend on one half the sequences Z_0 and Z_1, and similarly for $Z_{\frac{n}{2}+j}$ and $Z_{\frac{n}{2}}$ and $Z_{\frac{n}{2}+1}$. Thus, by keeping only the relevant values of these two pairs of sequences we will have divided the original computation into two computations of one half the size. Continuing in this fashion we ultimately obtain all values $Z_l(0)$. We

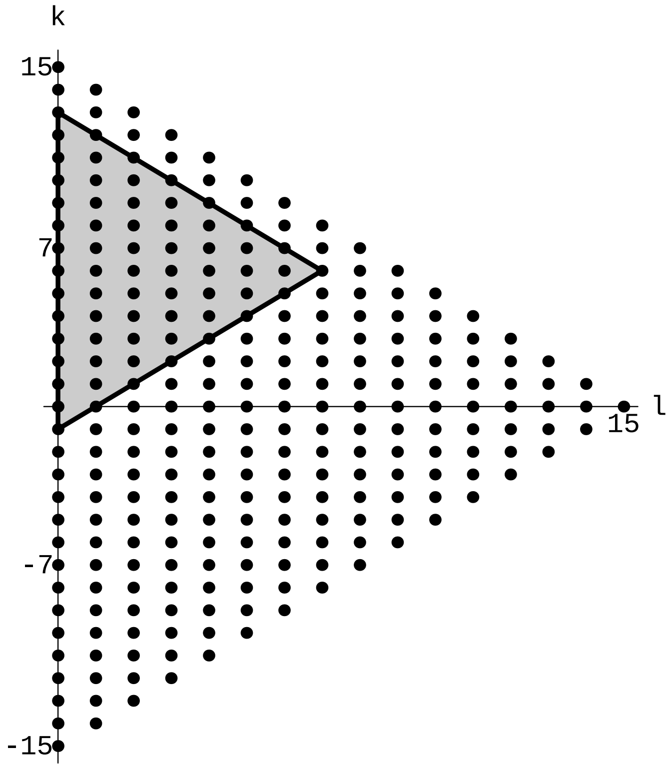

FIG. 2.4. *Computation of $Z_7(6)$ depends only on the computation of $Z_i(j)$ for (i, j) in the shaded triangle.*

need only show that the "divide" portion of the algorithm can be performed efficiently - that is, in $O(j \log j)$ operations for a problem of size j.

Again, we need the initial data of Z_0 and Z_1. We assume that Z_0 is given. By definition,

$$
\begin{aligned}
Z_1(k) &= \langle f, p_k \Phi_1 \rangle \\
&= \langle f, (a_0 x + b_0) p_k \rangle \\
&= a_0 \langle f, x p_k \rangle + b_0 Z_0(k).
\end{aligned}
$$

Notice that by using the recurrence (2.18) for p_{k+1} we may build the the first summand out of at most three shifted copies of Z_0. Thus, as a first step, in total, at most an additional $3n$ operations are needed to compute Z_1 from Z_0.

To compute the remaining Z_l, we wish to rewrite the recurrence (2.22) as a matrix equation. We can do this - up to "edge-effects" - as we will now explain. For any complex numbers w, y, z, let $C_n(w, y, z)$ denote the $2n \times 2n$ circulant matrix

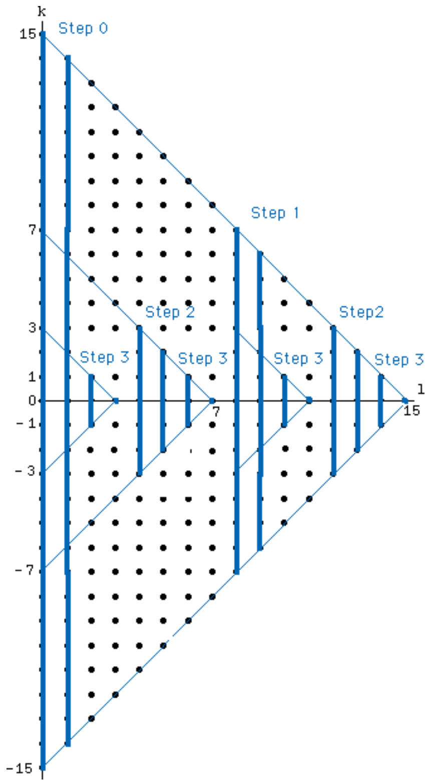

FIG. 2.5. *Computation of the $Z_l(0)$ for $l < 16$ by a cascade of convolutions of of decreasing size. The relevant ranges of Z are highlighted, and the step in which they are calculated is indicated.*

determined by setting the second row equal to $w, y, z, 0, \ldots, 0$,

$$C_n(w, y, z) = \begin{pmatrix} y & z & 0 & \cdots & 0 & 0 & w \\ w & y & z & \cdots & 0 & 0 & 0 \\ 0 & w & y & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & w & y & z \\ z & 0 & 0 & \cdots & 0 & w & y \end{pmatrix}. \tag{2.23}$$

Using the convention that

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

consider the vectors Z'_l and Z'_{l+1} defined by the expression

$$\begin{pmatrix} Z'_l \\ Z'_{l+1} \end{pmatrix} = \begin{pmatrix} 0 & I_{2n} \\ c_l I_{2n} & C_n(w_l, v_l, u_l) \end{pmatrix} \cdot \begin{pmatrix} Z_{l-1} \\ Z_l \end{pmatrix}. \tag{2.24}$$

Notice that $Z'_l = Z_l$ but that Z'_{l+1} differs from Z_{l+1} by at worst the entries $Z_{l+1}(n-1)$ and $Z_{l+1}(-n)$. This is precisely the aforementioned "edge-effect". If we make the definition

$$A_n(l) = \begin{pmatrix} 0 & I_{2n} \\ c_l I_{2n} & C_n(w_l, v_l, u_l) \end{pmatrix} \tag{2.25}$$

then we can make the following more general statement: a product of matrices of the form of $A_n(l+r)\cdots A_n(l)$ will still be composed of four $2n \times 2n$ circulant blocks and the edge-effects incurred when by multiplying this product by the vector $\binom{Z_l}{Z_{l+1}}$ will still only affect (at most) the $r-1$ outermost values of Z_{l+r-1} and Z_{l+r}. More precisely, a simple inductive argument yields the following claim.

Claim. *Let $0 \le r < n$. Define Z'_{l+r-1} and Z'_{l+r} by*

$$\begin{pmatrix} Z'_{l+r-1} \\ Z'_{l+r} \end{pmatrix} = A_n(l+r)\cdots A_n(l) \cdot \begin{pmatrix} Z_{l-1} \\ Z_l \end{pmatrix}.$$

Then, $Z'_i(j) = Z_i(j)$ for $j = -(n-r-2), \ldots, n-r-2$ and $i = l+r-1, l+r$.

The import of the Claim is that if we compute the product $R \cdot \begin{pmatrix} Z_0 \\ Z_1 \end{pmatrix}$ where

$$R = A_n(0)\cdots A_n(\frac{n}{2}-1) \tag{2.26}$$

then we will correctly compute the values $Z_{\frac{n}{2+i}}(j)$ for $i = 0, 1$ and $-(\frac{n}{2}-1) \le j \le \frac{n}{2}-1$. Notice that this is precisely the data we need in order to effect the "divide" portion of this algorithm (cf. Figure 2.4 and Figure 2.5). The matrix R is composed of four $2n \times 2n$ circulant blocks and thus, the matrix-vector multiplication (2.26) requires at most $64n(1+\log n)+16n$ operations (cf. the discussion with proof of Theorem 2.3). We continue by throwing away the upper and lower quarters of the vectors $Z_0, Z_1, Z'_{\frac{n}{2}}$ and $Z'_{\frac{n}{2}+1}$, forming two new subproblems of size $\frac{n}{2}$ and repeating the procedure. The analysis now follows that of the proof of Theorem 2.3. $\square$

Remark. Notice that if initially, only the projections onto the p_k for positive k were given, then the projection onto the p_k for negative k could obtained efficiently by using Theorem 2.3 applied to the backwards recurrence.

In particular, the shifted Chebyshev polynomials satisfy all of our requirements: they have a constant coefficient recurrence relation, fast projection is possible by Corollary 2.2, $T_k = T_{-k}$, and they possess relatively salutary numerical properties, even on a uniform grid. We have applied them in the case of the Hahn polynomial transforms with a great improvement in numerical accuracy over the method of Theorem 2.3:

EXAMPLE 1. *Fast Hahn Polynomial Transform:* To illustrate Theorem 2.4 we proceed with an example and discuss its application to the specific case of the **Hahn polynomials**, a well-known discrete orthogonal polynomial family. This is in fact the relevant family of orthogonal polynomials for the California Lottery example discussed in Section 1, and provide the spherical functions for the k-sets of n-set graph. We begin by summarizing the relevant properties of the Hahn polynomials. We follow Stanton's notation [St1] wherein is also contained a good bibliography for further sources. For general facts about orthogonal polynomials, Chihara's book [C] provides a friendly introduction to the subject.

The Hahn polynomials

$$Q_j(x;\alpha,\beta,N)=\sum_{i=0}^{j}\frac{(-j)_i\,(1+\alpha+\beta+j)_i\,(-x)_i}{i!\,(1+\alpha)_i\,(-N)_i}$$

are defined on the finite set $x=0,1,...,N$ for $j=0,1,...,N$. They are orthogonal with respect to the hypergeometric distribution

$$W(j;\alpha,\beta,N)=\binom{-1-\alpha}{-\beta+\mathrm{N}}\binom{-1-j}{\beta},$$

which is positive in the cases we consider, $\alpha,\beta<-N$.

For the purpose of our calculation we scale things so that all the action takes place on the uniform grid in $[0,1]$, $\left\{k/(N+1)\middle|k=0,...,N\right\}$. For fixed α, β, N, define

$$Q_n^*(x)=Q_n(\,(N+1)x;\alpha,\beta,N),$$

for x in the grid. Then we have the three-term recurrence,

$$Q_{n+1}^*(x)=\left(\frac{b_n+d_n-(N+1)x}{b_n}\right)Q_n^*(x)-\frac{d_n}{b_n}Q_{n-1}^*(x) \tag{2.27}$$

derived from that for Hahn polynomials, with

$$b_n=\frac{(n+\alpha+\beta+1)(n+\alpha+1)(N-n)}{(2n+\alpha+\beta+1)(2n+\alpha+\beta+2)}$$

and

$$d_n=\frac{n(n+\beta)(n+\alpha+\beta+N+1)}{(2n+\alpha+\beta)(2n+\alpha+\beta+1)}.$$

The calculation begins with projections of the data vector f onto the shifted Chebyshev polynomials $T_n^*(x)$. The projections are taken as l^2 inner products on the uniform grid in $[0,1]$, weighted by the hypergeometric distribution W on the grid. For the balance of this example we will use this weighted l^2 inner product.

As described in Corollary 2.2, these projections may all be done in $O(N\log^2 N)$ time. We now think of these as projections onto the functions $T_n^*Q_0^*$, and employ the techniques described in Theorem 2.4 for efficiently changing this information into the desired projections onto the Q_n^*.

We use the coefficients of the recurrence (2.17) to construct the convolution masks described in Theorem 2.4. In practice, we also normalize the recurrences by the l^2 norms of the Hahn polynomials. Define

$$Z_l(k)=\frac{1}{||Q_l^*||}<f,T_k^*Q_l^*>,\qquad -N\le k<N,$$

with $T_k^*=T_{-k}^*$. Then

$$Z_l(k)=A_{l-1}Z_{l-1}(k)+B_{l-1}\left\{Z_{l-1}(k-1)+Z_{l-1}(k+1)\right\}+C_{l-1}Z_{l-2}(k),$$

with

$$A_l=\frac{b_l+d_l-\frac{N+1}{2}}{b_l}\frac{||Q_l^*||}{||Q_{l+1}^*||}$$

One-step convolution coefficient masks for producing a sequence Z_l from lower index sequences Z_{l_0} and Z_{l_0+1} may be generated by an appropriate recursion. In the derivation that follows, M_j, N_j denote the one-step convolution masks for obtaining Z_{j+1} from Z_j and Z_{j-1}; in the present case $M_j = \{..., 0, B_j, A_j, B_j, 0, ...\}$ and $N_j = \{..., 0, 0, C_j, 0, 0, ...\}$. Then

$$\begin{array}{rcccc} Z_{l+1} & = & M_l * Z_l & + & N_l * Z_{l-1} \\ & = & M_l * (M_{l-1} * Z_{l-1} + N_{l-1} * Z_{l-2}) & + & N_l * Z_{l-1} \\ & = & (M_l * M_{l-1} + N_l) * Z_{l-1} & + & (M_l * N_{l-1}) * Z_{l-2} \\ & = & M_l\,(2) * Z_{l-1} & + & N_l\,(2) * Z_{l-2} \end{array}$$

Likewise we can continue all the way down to Z_{l_0} and Z_{l_0+1} :

$$Z_{l+1} = M_l\,(l-l_0) * Z_{l_0+1} + N_l\,(l-l_0) * Z_{l_0}$$

with multi-step masks $M_l\,(j)$ and $N_l\,(j)$ defined recursively by

$$\begin{aligned} M_l\,(j+1) &= M_l\,(j) * M_{l-j} + N_l\,(j) \\ N_l\,(j+1) &= M_l\,(j) * N_{l-j} \end{aligned}$$

with initial conditions

$$M_l\,(1) = M_l, \qquad\qquad N_l\,(1) = N_l.$$

Using this, we generate and prestore the various multi-step masks as described in Theorem 2.4, and run the tree of convolutions. Figure 2.6 shows the resulting steps of a small calculation, starting with Z_0 and Z_1, and then filling out the various other sequences in the tree.

3. Fast Spherical Transforms for Distance Transitive Graphs. We now employ the various results of the last section to obtain results leading up to a Fast Fourier transform for distance transitive graphs. As mentioned, these results are of interest in several problems of data analysis, such as the California Lottery example mentioned in the Section 1.

Briefly, the setting is as follows. Let G be a finite group acting as isometries on a finite graph X with distance function $d(\cdot,\cdot)$ given by the usual measure of shortest path. Recall that X is **distance transitive** (for G) if given any two pairs of points (x,y), $(x',y'), \in X$ such that $d(x,y) = d(x',y')$, there exists $s \in G$ such that $(sx, sy) = (x', y')$. Let $L^2(X)$ denote the vector space of complex-valued functions on X. Then $L^2(X)$ affords a linear representation of G by left translation. In this case, $L^2(X)$ may be decomposed into irreducible subspaces

$$L^2(X) = V_0 \oplus \cdots \oplus V_N$$

where N is the maximum distance between two points in X.

Fix a basepoint $x_0 \in X$ and let H be the stabilizer of x_0 in G. Then X is in a natural 1:1 correspondence with G/H and $L^2(G/H)$ is the vector space of right H-invariant functions on G. If Ω_k denotes the sphere of radius k about x_0, then the algebra of functions constant on each Ω_k is isomorphic to the algebra of H-biinvariant functions on G, denoted $L^2(H\backslash G/H)$.

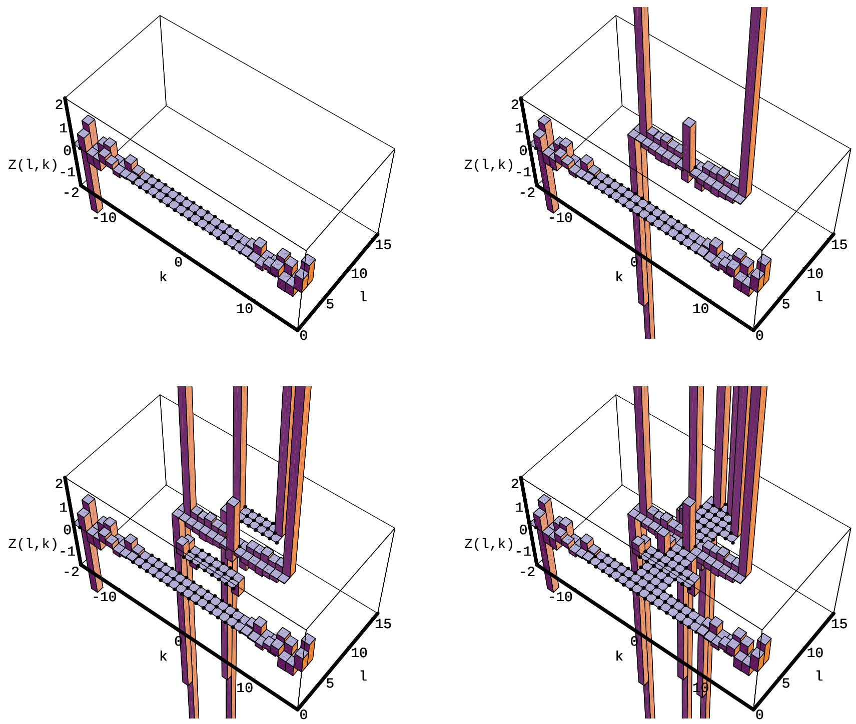

FIG. 2.6. *Stages in the computation of the Z_l for the Hahn polynomials. Z_0 is projection onto the Chebyshev polynomials. The desired transform values are the $Z_l(0)$. This example is the transform of Q_8^*.*

The fact that $L^2(X)$ is multiplicity-free is equivalent to the existence in each V_k of a unique function ϕ_k, constant on each Ω_k and normalized by $\phi_k(x_0) = 1$. Classically, the function ϕ_k is called the k^{th} **spherical function** on X. (cf. [He] and the references therein, as well as the remarks at the close of Section 3.1.)

Let $x_j \in \Omega_j$. Then in analogy with the classical case (see e.g. [He], Chapter 4) we define for any function f constant on each Ω_k the **spherical transform** of f at ϕ_i to be the sum

$$\widehat{f}(\phi_i) = \sum_{j=0}^{N} f(x_j)\phi_i(x_j)\,|\Omega_j|.$$

The **discrete spherical transform** of f (DST) is the collection of transforms $\{\widehat{f}(\phi_i)\}_i$.

Direct computation of the DST requires $O(N^2)$ operations. For large N this cost quickly becomes prohibitive. In this section we give an algorithm which computes the DST for spherical functions from distance transitive graphs in $O(N \log^2 N)$ operations. By the same techniques we may also invert the transform in the same number of operations and consequently we obtain an $O(N \log^2 N)$ for convolution of two H-biinvariant functions.

Section 4 will show that the problem of finding an FFT for distance transitive graphs may be reduced to that of finding an efficient projection onto the spherical functions, an orthogonal family of special functions on the graph. This section discusses the fast DST. We begin by giving an expanded review of the group theoretic background (Section 3.1), sufficient to present the fast algorithm in Section 3.2.

3.1. Background and Notation. In the interest of making this paper as self-contained as possible, we sketch the group theoretic background and notation. We mainly follow Stanton's expositions [St1, St3] which are very accessible and provide a wealth of references. For the necessary graph theoretic terminology, with special attention paid to distance transitive graphs, see Biggs [Bi]. Serre's book [S] provides a nice introduction to the representation theory of finite groups.

Throughout, X is a **distance transitive graph**: Thus, X is a finite metric space with **integer-valued** distance d (taken to be the usual distance of shortest path on the graph) with G a group of isometries of X (acting on the left) satisfying the property of **two-point homogeneity**:

If $x, y, x', y' \in X$ are such that $d(x, y) = d(x'y')$, then there exists $g \in G$ such that $gx = x'$ and $gy = y$.

In this case, X is also said to be a **two-point homogeneous space** (with respect to G and d).

It is perhaps instructive at this point to recall the analogy here with the usual 2-sphere in $\mathbf{R}^3$. In this case we know that any pair of points on the sphere which are a fixed distance apart can be moved into any other two such points by a rotation - or isometry of $\mathbf{R}^3$. Thus, the rotation group $SO(3)$ acts two-point homogeneously on the 2-sphere. Stanton's papers [St1, St3] do a terrific job of spelling out these analogies.

Thus, since $d(x, x) = 0 = d(x', x')$ for every $x, x' \in X$, there is a $g \in G$ such that $gx = x'$, i.e., G acts **transitively** on X. Let $x_0 \in X$ denote a fixed **basepoint** and $H = \{g \in G \mid gx_0 = x_0\}$ denote the stabilizer subgroup of x_0. Since G acts transitively on X, any element $x \in X$ can be written as $x = sx_0$ for some $s \in G$ and if $sx_0 = x = tx_0$, then $s^{-1}tx_0 = x_0$ and hence $s^{-1}t \in H$ and $t \in sH$. Thus, there is a natural correspondence between X and the coset space G/H, associating any element $x = sx_0$ with the coset sH. Keeping in mind the analogy with the 2-sphere, consider the subgroup $H = SO(2) < SO(3)$ of rotations which fix the north pole. Cosets are represented by circles of latitude, with coset representatives in 1:1 correspondence with a choice of points at all possible latitudes. Any two points of the same latitude differ only by a rotation about the north pole.

Under the correspondence between points in X and cosets in G/H the vector space of complex-valued functions on X, $L^2(X)$ is isomorphic to the vector space of complex-valued functions on G/H, $L^2(G/H)$, by defining

$$f(sH) \equiv f(sx_0).$$

Any function on G/H then naturally extends to $\tilde{f}$, a function defined on the entire group G, by declaring it to be constant on cosets,

$$\tilde{f}(s) \equiv f(sH).$$

It is not hard to check that $\tilde{f}$ is well-defined and that

$$\tilde{f}(sh) = \tilde{f}(s). \tag{3.1}$$

Thus, $\tilde{f}$ is a **(right) H-invariant function on G.** Conversely, it is easy to see that for any subgroup $H < G$ the subspace of $L^2(G)$ satisfying (3.1)) is equivalent to the

$L^2(G/H)$. Following along the analogy with the 2-sphere, $L^2(SO(2)\backslash SO(3)/SO(2))$ can be identified with the subspace of functions on the 2-sphere which are constant on latitudes.

The action of G on X gives rise to a **representation** of G in $L^2(X)$ by translation. More precisely, for every $s \in G$ and $f \in L^2(X)$ a new function $\rho(s)f \in L^2(X)$ can be defined by

$$\left[\rho(s)f\right](x) \equiv f(s^{-1}x).$$

In this manner, each $\rho(s)$ defines a linear operator on $L^2(X)$ such that $\rho(st) = \rho(s)\rho(t)$ and thus is a **representation** of the group G. This representation is in general **reducible** in the sense that there exist proper subspaces of $W_1, \ldots, W_r$ such that each W_i is G**-invariant** (i.e. $\rho(s)W_i \subset W_i$ for all $s \in G$) and $L^2(X) = W_1 \oplus \cdots \oplus W_r$. A subspace W is G**-irreducible** if it contains no proper G-invariant subspaces. For $L^2(X)$, an irreducible decomposition can be understood by considering the action of H.

Let Ω_k denote the sphere of radius k about x_0,

$$\Omega_k = \{x \in X \mid d(x, x_0) = k\}. \tag{3.2}$$

Then the Ω_k are exactly the H-orbits in X. That is X is the disjoint union of the Ω_k and if $x, y \in \Omega_k$ then $hx = y$ for some $h \in H$ and conversely, if $hx = y$, then $x, y \in \Omega_k$ for some k. In the case of the 2-sphere, the associated "Ω_k" are the circles of latitude.

A subspace W of $L^2(X)$ is H**-invariant** if for all $f \in W$, $\rho(h)f \in W$ for all $h \in H$. Thus, a function constant on each of the Ω_k is H-invariant and conversely. Hence, if N is the maximum distance occurring in X, then the subspace of H-invariant functions in $L^2(X)$ is of dimension $N+1$. This may be immediately translated into a statement about functions on G: Under the association of $L^2(X)$ with right H-invariant functions on G, the H-invariant functions of $L^2(X)$ become functions which are **both** left and right H-invariant, i.e., functions $f \in L^2(G)$ such that

$$f(h_1 s h_2) = f(s)$$

for all $h_1, h_2 \in H$. Such H-biinvariant functions are then naturally associated with the space of functions constant on double cosets $H\backslash G/H$ and thus are denoted as $L^2(H\backslash G/H)$. Hence we see that $L^2(H\backslash G/H)$ is of dimension $N+1$.

Note that the subspaces $L^2(X)$ and $L^2(H\backslash G/H)$ are in fact algebras under convolution: if $f, g \in L^2(X)$ then define $f \star g \in L^2(X)$ by

$$f \star g(x) = \sum_{s \in G} \tilde{f}(s) g(s^{-1}x) \tag{3.3}$$

where $\tilde{f}$ is the function on G derived from f by extending it to be constant on cosets of H as in (3.1). It is easy to check that if f and g are H-invariant then their convolution is as well.

As a complex representation space for a finite group G, $L^2(X)$ may be decomposed into G-irreducible subspaces (cf. [S], Section 1.4, Theorem 2). In the general situation of decomposing the permutation representation arising from a finite group G acting transitively on a set X, this irreducible decomposition need not be unique. However, under the assumption of distance transitivity, an irreducible decomposition is indeed unique.

THEOREM 3.1. *(**[St1], Theorem 2.6**) Let all notation be as above. Then as a representation of G, the space $L^2(X)$ has a unique decomposition into irreducible subspaces as*

$$L^2(X) = V_0 \oplus V_1 \oplus \cdots \oplus V_N.$$

where the V_i are all pairwise inequivalent - that is, the representation of G in $L^2(X)$ is **multiplicity-free.**

The proof of this theorem is not crucial for the main results, but it is worth remarking that it depends only on the fact that G acts two-point homogeneously on X and as such is a general fact about permutation representations (cf. [W], Chapter 5, Section 29). In this context a proof follows from the fact that Theorem 3.1 is equivalent to the statement that the **intertwining algebra** of the permutation representation is commutative. (The intertwining algebra is the algebra of linear operators T that commute with the permutation representation ρ - i.e., the set of T such that $T\rho(s) = \rho(s)T$ for all $s \in G$.) To show this commutativity, choose a basis for $L^2(X)$ consisting of "delta functions" or point masses concentrated at single points. For $0 \leq k \leq N$, define the $|X| \times |X|$ matrices D_k by

$$D_k(x, y) = \begin{cases} 1 & \text{if } d(x, y) = k \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

Straightforward combinatorial arguments (eg. [St1, St3]) show that the D_k commute and span the intertwining algebra, which must then be of dimension $N+1$. Even more, the algebra is generated by D_1 since the D_k satisfy the following three-term recurrence ([St1, St3]),

$$D_1 D_i = c_{i+1} D_{i+1} + a_i D_i + b_{i-1} D_{i-1} \tag{3.5}$$

where

$$\begin{aligned} c_{i+1} &= |\{z : d(x, z) = 1, d(y, z) = i\}| \text{ for any fixed } x, y \text{ with } d(x, y) = i + 1, \\ a_i &= |\{z : d(x, z) = 1, d(y, z) = i\}| \text{ for any fixed } x, y \text{ with } d(x, y) = i, \\ b_{i-1} &= |\{z : d(x, z) = 1, d(y, z) = i\}| \text{ for any fixed } x, y \text{ with } d(x, y) = i - 1. \end{aligned}$$

Consequently, D_i is a polynomial in D_1,

$$D_i = p_i(D_1). \tag{3.6}$$

Since D_1 is real symmetric and generates an algebra of dimension $N+1$ it has distinct real nonzero eigenvalues $\{\lambda_0 < \cdots < \lambda_N\}$. Also, since D_1 is in the intertwining algebra for the representation ρ and the intertwining algebra is commutative, the λ_i eigenspaces must be the G-irreducible subspaces.

The importance of Theorem 3.1 is that it shows that in this special case, the isotypic and irreducible decompositions coincide, so that the irreducible decomposition is independent of choice of basis. It is a direct consequence of Theorem 3.1 that in each V_i there exists a unique one-dimensional H-fixed subspace (e.g. see [D], p.54, Theorem 9). We choose a basis vector ϕ_i for this subspace by demanding that $\phi_i(x_0) = 1$. Note that this is possible, since the previous reference - or Frobenious Reciprocity (cf. [S]) - shows the existence of some nonzero H-fixed element (hence constant on each of the Ω_k) $\phi_i \in V_i$. Since

$$D_1 \phi_i = \lambda_i \phi_i \tag{3.7}$$

and ϕ_i is constant on each Ω_k, the fact that ϕ_i is not identically zero implies that $\phi_i(x_0) \neq 0$. Hence ϕ_i can be normalized so as to assume $\phi_i(x_0) = 1$.

Note that ϕ_i may be viewed as either an H-invariant function on X or an H-biinvariant function on G. As an H-invariant function on X it is constant on the spheres Ω_k. We call ϕ_i the i^{th} **spherical function.** By counting we see that the spherical functions give a basis for the H-invariant functions on X.

For distance transitive graphs, the polynomial nature of the spherical functions is derived from the selfsame property of the commuting algebra for the representation of G in $L^2(X)$.

As shown in [St1], by evaluating the eigenvalue equation (3.7) at Ω_k we move the recurrence (3.5) to the ϕ_i:

$$\lambda_i \phi_i(\Omega_k) = \gamma_k \phi_i(\Omega_{k+1}) + \alpha_k \phi_i(\Omega_k) + \beta_k \phi_i(\Omega_{k-1}), \tag{3.8}$$

where for any $x \in \Omega_k$,

$$\begin{aligned} \gamma_k &= |\ \{z : z \in \Omega_{k+1} \text{ and } d(x,z) = 1\}\ |, \\ \alpha_k &= |\ \{z : z \in \Omega_k \text{ and } d(x,z) = 1\}\ |, \text{ and} \\ \beta_k &= |\ \{z : z \in \Omega_{k-1} \text{ and } d(x,z) = 1\}\ |. \end{aligned}$$

Examination of the combinatorics yields,

$$|\,\Omega_k\,|\ \phi_i(\Omega_k) = p_k(\lambda_i). \tag{3.9}$$

The orthogonality relations for the spherical functions take on the form

$$\frac{1}{|\,X\,|} \sum_{k=0}^{N} \phi_i(\Omega_k)\phi_j(\Omega_k)\ |\,\Omega_k\,| = \delta_{ij} \frac{1}{dim(V_j)} \tag{3.10}$$

where $\phi_i(\Omega_k)$ makes sense as ϕ_i is constant on Ω_k. In addition we have a "dual orthogonality relation",

$$\frac{1}{|\,X\,|} \sum_{i=0}^{N} \phi_i(\Omega_k)\phi_i(\Omega_j) dim(V_i) = \delta_{kj} \frac{1}{|\,\Omega_k\,|}. \tag{3.11}$$

We summarize this with the following theorem.

THEOREM 3.2. *Let X be a finite distance transitive graph with respect to a group of isometries G. Let*

$$L^2(X) = V_0 \oplus \cdots \oplus V_N$$

be the isotypic, and hence irreducible, decomposition of $L^2(X)$ so that N is the maximum distance occurring in X. Let ϕ_i be the spherical function for V_i and λ_i as in (3.7) and Ω_k as in (3.2). Then

$$|\,\Omega_k\,|\ \phi_i(\Omega_k) = p_k(\lambda_i)$$

for some set of orthogonal polynomials $\{p_k(x) \mid 0 \leq k \leq N\}$ such that p_k is of degree k and the polynomials satisfy a three-term recurrence (3.5).

Remarks. 1. It is worth noting that while the spherical functions ϕ_i are determined by the polynomial functions described above, Theorem 3.2 does not say that $\phi_i(\Omega_k)$ is polynomial in k (i.e., equal to some fixed polynomial evaluated at $N+1$ fixed points). Rather this is a statement that the dual functions p_k are an orthogonal polynomial system with respect to the weights $dim(V_i)$, although for many examples this will also be true of the spherical functions (cf. section 4). As a consequence, in general the inverse spherical transform is always projection onto polynomials, and would thus benefit from general results on fast projection onto polynomials. In fact such an algorithm can then be transposed to obtain an algorithm for the direct transform with the same complexity, whether or not the direct transform is obtained by projection onto polynomials.

2. The existence of spherical functions depended only on the fact that the permutation representation of G on $L^2(G/H)$ was multiplicity-free. This is often summarized by saying that (G,H) form a **Gelfand pair**. Gelfand pairs have been much studied of late. See Gross [Gr] for a survey with applications to number theory and Diaconis [D] for applications to statistics and probability as well as an extensive bibliography. Helgason's book [He] gives a thorough introduction to the study of spherical functions for compact and locally compact groups with a full bibliography.

As remarked, the polynomial nature follows from the polynomial relation of the D_i. This is true in a slightly more general setting than distance transitive graphs. It can be extended to finite two-point homogeneous spaces in which the metric satisfies some technical properties (cf. [St1], p. 90).

Spherical functions may also be computed by character theoretic methods. Travis [Tr] generalizes this approach to construct "generalized" spherical functions attached to any pair of characters ψ and χ for representations of a finite group G and subgroup H respectively.

3. While the existence of the spherical functions depends only on the multiplicity-free nature of the representation, their expression as certain sampled values of orthogonal polynomial sets, and consequent relations via the recurrence (3.8) uses the integer-valued property of the metric.

4. We should point out that many of the problems and results discussed here may be phrased in the language of association schemes. Bannai and Ito [BI] give a beautiful treatment of this subject. We have not pursued this connection.

3.2. Fast spherical transforms on distance transitive graphs. The terminology is that of Section 3.1. In particular, recall that $H < G$ is the isotropy group of a fixed basepoint $x_0 \in X$ so that $L^2(H\backslash G/H)$ is identified with the subspace of $L^2(X)$ of functions constant on spheres around x_0.

THEOREM 3.3. *For distance transitive graphs with maximum distance N, the spherical transform and its inverse can be computed in at most $O(N\log^2 N)$ operations.*

Proof. Let $f \in L^2(H\backslash G/H)$. Then the components of $\widehat{f}$ are

$$\widehat{f}(\phi_i) = \sum_{k=0}^{N} f(\Omega_k)\phi_i(\Omega_k) \mid \Omega_k \mid = \sum_{k=0}^{N} f(x_k)p_k(\lambda_i). \tag{3.12}$$

using

$$\mid \Omega_k \mid \phi_i(\Omega_k) = p_k(\lambda_i). \tag{7}$$

from Section 3.1.

This has the form of polynomial evaluation, or multiplication by the transpose of the generalized Vandermonde matrix associated with the polynomials p_k and the evaluation points λ_i :

$$\begin{pmatrix} \widehat{f}(\phi_0) \\ \widehat{f}(\phi_1) \\ \vdots \\ \widehat{f}(\phi_N) \end{pmatrix} = \begin{pmatrix} p_0(\lambda_0) & p_1(\lambda_0) & \cdots & p_N(\lambda_0) \\ p_0(\lambda_1) & p_1(\lambda_1) & \cdots & p_N(\lambda_1) \\ \vdots & \vdots & \ldots & \vdots \\ p_0(\lambda_N) & p_1(\lambda_N) & \cdots & p_N(\lambda_N) \end{pmatrix} \begin{pmatrix} f(\Omega_0)\,|\Omega_0| \\ f(\Omega_1)\,|\Omega_1| \\ \vdots \\ f(\Omega_N)\,|\Omega_N| \end{pmatrix} = P(\Lambda)^t \mathbf{f}_\Omega.$$

Likewise, the inverse spherical transform can be written in terms of $(P(\Lambda)^t)^t, =$ $P(\Lambda)$ itself; by the dual orthogonality equation (3.11)

$$f(\Omega_k) = \frac{1}{|X|} \sum_{i=0}^{N} \widehat{f}(\phi_i)\phi_i(\Omega_k) dim(V_i).$$

Using equation (3.10) we rewrite this as

$$|\Omega_k| f(\Omega_k) = \frac{|\Omega_k|}{|X|} \sum_{i=0}^{N} \widehat{f}(\phi_i) p_k(\lambda_i) dim(V_i).$$

Up to scaling, this has the form of projection onto the polynomials p_k, or multiplication by the generalized Vandermonde matrix $P(\Lambda)$. The three-term recurrence relation is already known explicitly; consequently, the methods of Section 2 apply directly to this computation. Thus, the inverse spherical transform can be done in $O(N \log^2 N)$ operations.

We can also conclude that the transpose problem, the direct spherical transform, is also fast. This follows, for instance, from the results of N. Bshouty et. al. [BKK]. Roughly speaking they observe that if a straight line algorithm can compute a matrix product $M \cdot \mathbf{v}$ in time $O(T(n))$ (for M an $n \times n$ matrix and $\mathbf{v}$ a column vector), then there exists an algorithm computing the transposed product $M^t \mathbf{v}$ in the same time. They note that any such algorithm may be encoded in a directed graph and in this context the "transposed" algorithm is essentially given by reversing all arrows on the graph.

In our case, the inverse spherical transform algorithm amounts to a factorization of the matrix $P(\Lambda)$. The order reversed and transposed factors comprise a factorization of $P(\Lambda)^t$, which effects the direct transform by matrix multiplication. The individual factors have block structure with Toeplitz blocks, and this does not change upon their transposition. Therefore, the direct transform is computed with the same complexity as the inverse. □

Finally, consider convolution of two functions $f, g \in L^2(H\backslash G/H)$. Recall that this is simply the convolution over G (cf. equation (3.3) of H-biinvariant functions which is again H-biinvariant. As such it makes sense to compute the DST of the convolution. It can be shown that for such functions

$$|X|\, \widehat{f \star g}(\phi_i) = \widehat{f}(\phi_i)\widehat{g}(\phi_i)$$

with a quick proof using the multiplicative properties of a Fourier transform on a finite group and the fact that a spherical function is a particular matrix coefficient for the symmetry group (cf. [D], pp. 54–56). Thus, we obtain

THEOREM 3.4. *Let $f, g \in L^2(H\backslash G/H)$. Given as initial input the spherical transforms $\{\widehat{f}(\phi_i)\}_i$, the function f may be recovered in $O(N \log^2 N)$ operations. The* *convolution $f \star g$ can be computed in at most $O(N \log^2 N)$ operations. The implied constants here are universal and depend only on the universal constant for the FFT.*

EXAMPLE 2. *Some particular distance transitive graphs.*

1. *K-sets of an N-set*: This is the collection of size K subsets $x \subset \{1, 2, ..., N\}$, $\mid x \mid = K$, with metric $d(x, y) = K - |x \cap y|$, S_N as symmetry group, and made into a graph in the usual way: put edges between those K-sets whose distance is 1. Assuming $2K < N$, the largest distance is K, occurring for disjoint K-sets. Picking a basepoint K-set x_0, the stabilizer is $\cong S_K \times S_{N-K}$, so we may identify this graph with $S_N/(S_K \times S_{N-K})$.

This is a distance transitive graph of size $\binom{N}{K}$. The weights in the spherical function orthogonality relations are the sizes of the spheres at fixed distances from the basepoint: $|\Omega_j| = \binom{N-K}{j}$. Recall that the spherical functions satisfy

$$\mid \Omega_j \mid \phi_i(\Omega_j) = p_j(\lambda_i)$$

for a family of orthogonal polynomials defined on the collection of eigenvalues for the adjacency operator. We saw that the spherical functions are eigenvectors of the adjacency operator, and that this provides the three-term recurrence from which the λ_i and the p_j may be determined:

$$\begin{aligned}\lambda_i p_j(\lambda_i) = j^2 p_{j-1}(\lambda_i) + (K-j)(N-K-j)p_{j+1}(\lambda_i)\\ +[K(N-K) - j^2 - (K-j)(N-K-j)]p_j(\lambda_i).\end{aligned}$$

This is the recurrence for the Eberlein, or dual Hahn polynomials. From the case $j = 1$ we get $\lambda_i = K(N-K) - i(N+1-i)$. We can now compute the spherical functions as

$$\phi_i(\Omega_j) = c \sum_{l=0}^{i} \frac{(N-K-i+1)_l}{(-K)_s} \binom{K-j}{l} \binom{j}{-i-l}$$

This is the Hahn polynomial $Q_i(j; K-N-1, -K-1, K)$ [KM, St1]. As we have already seen, these are orthogonal polynomials satisfying a three-term recurrence (2.27). The norms can be determined from $dim V_i = \binom{N}{i} - \binom{N}{i-1}$

In this case we have the recurrence relation needed to make the forward spherical transform fast, as discussed in Example 1 of the last section. On the other hand, we know that in every case we have the recurrence relation for the p_k needed to make the inverse transform fast.

2. *The hypercube*: The hypercube $\mathbf{Z}_2^N$ has the Hamming metric and symmetries consisting of the hyperoctohedral group, the semidirect product $\mathbf{Z}_2$ wr S_N. This is a distance transitive graph. The three-term recurrence from the adjacency operator eigenequation is

$$\lambda_i p_j(\lambda_i) = j\lambda_i p_{j-1}(\lambda_i) + (N-j)p_{j+1}(\lambda_i)$$

from which we determine the eigenvalues $\lambda_i = N - 2i$, and the spherical functions

$$\phi_i(\Omega_j) = c \sum_{l=0}^{i} \binom{j}{l} \binom{N-j}{i-l} (-1)^{i-l}.$$

Again, these are orthogonal polynomials with respect to weights $|\Omega_j| = \binom{N}{j} = dimV_j$, known as the Krawtchouk polynomials $K_i(j, 1/2, N)$. This is a particularly nice case in that the dual polynomials and the spherical polynomials are the same up to a constant. So the forward spherical transform and its inverse are effectively the same, and can be done fast using the three-term recurrence as before.

4. Computation of Isotypic Projections. As remarked in Section 1, a fast DST algorithm has applications in spectral analysis for data on distance transitive graphs. In this section we wish to explain this in a little more detail. Diaconis' book ([D], especially Chapter 8) is an excellent introduction to these ideas and also gives many pointers to the existing literature. (See [DR] for a more thorough account of the following discussion as well as for other approaches to this problem.)

In general, let G be a finite group acting transitively on a set

$$X = \{x_0, x_1, \ldots, x_n\}.$$

The action of G on X then determines the associated permutation representation ρ of G in $L^2(X)$ given by translation,

$$(\rho(s)f)(x) = f(s^{-1}x).$$

If η and η' are two representations of G, then we will write $\eta \sim \eta'$ if η is equivalent to η'. Recall that the **isotypic decomposition** of $L^2(X)$,

$$L^2(X) = V_0 \oplus \cdots \oplus V_N$$

is defined by

(1) Each V_i is G-invariant.
(2) If $\rho^{(i)} := \rho\,|_{V_i}$ then,

$$\rho^{(i)} \sim m_i \eta_i$$

where the η_i are irreducible representations of G, m_i some positive integer and $i \neq j$ implies that $\eta_i \not\sim \eta_j$.

The isotypic decomposition is canonical in the sense that it is independent of choice of basis for $L^2(X)$.

In appropriate settings, data on a such a finite homogeneous space X is viewed as a vector $f \in L^2(X)$. The relevant statistics then become the projections of f onto the isotypic components. To state things a bit more sharply the computational problem is

Given as input $f = (f(x_0), \ldots, f(x_n))$, compute for $i = 0, \ldots, N$ the projection of f, into the i^{th} isotypic, denoted $f^{(i)} (\in V_i)$ as

$$f^{(i)} = (f^{(i)}(x_0), \ldots, f^{(i)}(x_n)).$$

One way to proceed here is via character theory. Let χ_i be the character corresponding to η_i. Then ([S], Theorem 2.7),

$$f^{(i)}(x) = \frac{\chi_i(1)}{|G|} \sum_{s \in G} \chi_i(s) f(s^{-1}x). \tag{4.1}$$

This gives a naive upper bound of $|X||G|$ operations to compute all projections. Note that in the example of the California Lottery of Section 1, this would give $49!\binom{49}{6}$ operations, which is beyond the capabilities of any machine.

Careful analysis of this formula permits significant speed-ups, even in the general case.

Theorem *([DR], Theorem 2.4) For any fixed i, the projection onto the i^{th} isotypic can be computed in at most $|X|^2$ operations. Consequently, all projections can be computed in at most $(N+1)|X|^2$ operations.*

Let us now specialize the case of interest, that in which $L^2(X)$ has a multiplicity-free decomposition so that the isotypic decomposition is actually an irreducible decomposition. As previously, let H denote the isotropy subgroup of the chosen basepoint x_0 and let $\{s_0 = 1, \ldots, s_n\}$ denote a fixed set of coset representatives. Let $\{\phi_i\}_{i=0}^N$ denote the corresponding spherical functions. In this case the character formula (4.1) can be rewritten as

$$f^{(i)}(x_k) = \frac{|H|}{|G|} \chi_i(1) \sum_{j=0}^{N} f_k(\Omega_j)\phi_i(\Omega_j)\,|\Omega_j| \tag{4.2}$$

where

$$f_k(\Omega_j) = \frac{1}{|\Omega_j|} \sum_{x \in \Omega_j} f(s_k^{-1}x). \tag{4.3}$$

Computation of the f_k is a preprocessing of the original data whose complexity will depend on the geometry of X. In any event this always may be done in at most $|X|^2$ additions.

Finally, using the notation of the previous sections, we rewrite (4.2) as

$$f^{(i)}(x_k) = \frac{|H|}{|G|} \chi_i(1) \widehat{f_k}(\phi_i).$$

Consequently using the results of Section 3, we have the following

THEOREM 4.1. *Let X be a distance transitive graph for G with maximum distance N. Then the set of projections $f^{(i)} \in V_i$, $(i = 0, \ldots, N)$ may be computed in at most*

$$O(|X|^2 + |X|\, N \log^2 N)$$

operations.

Remark. In some cases the projections can also be computed combinatorially using variations of the discrete Radon transform (cf. [D, DR, BDRR]).

5. Final Remarks. Of course, distance transitive graphs are not the only source of orthogonal polynomials. Another example closely related to this setting is the construction of orthogonal polynomial systems from group actions on posets [St2]. If P is a ranked poset, then $L^2(P)$ has a natural decomposition into "harmonics". In [St2] Stanton shows that under certain assumptions about the automorphism group G of P, the space of functions on the maximal elements of P, gives a multiplicity-free representation of G. Again these functions can be written in terms of discretely sampled orthogonal polynomials.

More generally, one might consider other special function systems satisfying recurrence relations that arise in a continuum setting. Results similar to those of this paper can be obtained, provided an appropriate sampling theorem is available to reduce the computations to finite ones. Some initial work along this line, for the case of the homogeneous space $SO(3)/SO(2)$ may be found in [DrH]. Dave Maslen has recently extended these ideas to more general compact groups [M].

Beyond the example of spectral analysis, we are actively seeking other applications for the techniques presented here. A recent book by Nikiforov, Suslov and Uvarov [NSU] cites a large number of tantalizing possibilities.

REFERENCES

[BI] E. Bannai and T. Ito, *Algebraic Combinatorics I: Association Schemes.* Benjamin/Cummings Publishing Co., Inc., Menlo Park, CA, 1984.

[BDRR] R. Bailey, P. Diaconis, D. Rockmore and C. Rowley, *Representation theory and ANOVA*, Technical Report, Department of Mathematics, Harvard University, 1994.

[Bi] N. Biggs, *Algebraic Graph Theory*, Cambridge Tracts in Mathematics No. 67, Cambridge University Press, Cambridge, 1974.

[BM] A. Borodin and I. Munro, *The Computational Complexity of Algebraic and Numeric Problems*, Elsevier, New York, 1975.

[Bo] J. Boyd, *Chebyshev and Fourier Spectral Methods*, Springer-Verlag, New York, 1989.

[BKK] N. Bshouty, M. Kaminski, and D. Kirkpatrick, *Addition requirements for matrix and transposed matrix products*, J. of Algorithms, 9 (1988), pp. 354–364.

[C] T. S. Chihara, *An Introduction to Orthogonal Polynomials*, Gordon and Breach, New York, 1978.

[D] P. Diaconis, *Group Representations in Probability and Statistics*, Institute for Mathematical Statistics, Hayward, CA, 1988.

[DR] P. Diaconis and D. Rockmore, *Efficient computation of isotpic projections for the symmetric group*, in DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 11, L. Finkelstein and W. Kantor, eds., AMS, Providence, RI, 1992, pp. 87–104.

[DrH] J. R. Driscoll and D. Healy, *Computing Fourier transforms and convolutions on the 2-sphere*, Adv. in Appl. Math., 15 (1994), pp. 202–250.

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

[Ga1] W. Gautschi, *On the construction of Gaussian quadrature rules from modified moments*, Mathematics of Computation 24 (1970), pp. 245–260.

[Ga2] ———, *Questions of numerical condition related to polynomials*, in Studies in Numerical Analysis, G. Golub, ed., MAA, Washington DC, 1984, pp. 140–177.

[Gr] B. Gross, *Some applications of Gelfand pairs to number theory*, Bull. of A.M.S., 24 (1991) pp. 277–301.

[HMR] D. Healy, S. Moore, and D. Rockmore, Efficiency and stability issues in the numerical computation of Fourier transforms on the 2-sphere. *Technical Report PCS-TR94-222, Dept. of Math. and Computer Science, Dartmouth College*, 1993.

[HMMRT] D. Healy, D. Maslen, S. Moore, D. Rockmore and M. Taylor, *Applications of FFT's on the 2-sphere*, in preparation.

[He] S. Helgason, *Groups and Geometric Analysis*, Academic Press, New York, 1984.

[Hi] N. J. Higham, *Fast solution of Vandermonde-like systems involving orthogonal polynomials*, IMA J. Numer. Anal., 8 (1988), pp. 473–486.

[KM] S. Karlin and J. L. McGregor, *The Hahn polynomials, formulas and an application*, Scripta Math., 26 (1961), pp. 33–46.

[M] D. Maslen, *Fast Transforms and Sampling for Compact Groups*, Ph.D. Thesis, Department of Mathematics, Harvard University, 1993.

[MHR] S. Moore, D. Healy, and D. Rockmore, *Symmetry stabilization for polynomial evaluation and interpolation*, Lin. Alg. Appl., 192 (1993), pp. 249–299.

[NSU] A. F. Nikiforov, S. K. Suslov and V. B. Uvarov, *Classical Orthogonal Polynomials of a Discrete Variable*, Springer-Verlag, New York, 1991.

[N] H. J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms*, Springer-Verlag, New York, 1982.

[OS] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, NJ, 1989.

[P] V. Pan, *Matrix and polynomial computations*, SIAM Review, 34 (1992), pp. 225–262.

[S] J. P. Serre, *Linear Representations of Finite Groups*, Springer-Verlag, New York, 1986.

[St1] D. Stanton, *Orthogonal polynomials and Chevalley groups*, in Special Functions: Group Theoretical Aspects and Applications, R. Askey, T. Koornwinder, and W. Schempp eds., D. Reidel Publishing Co., Dordrecht, 1984, pp. 87–128.

[St2] ———, *Harmonics on posets*, J. Comb. Theory, Series A, 40 (1985), pp. 136–149.

[St3] ———, *An introduction to group representations and orthogonal polynomials* in Orthogonal Polynomials, P. Nevai, ed., Kluwer Academic Publishers, 1990, pp. 419–433.

[Te] C. Temperton, *On scalar and vector transform methods for global spectral models*, Monthly Weather Review 119 (1991), pp. 1303–1307.

[TAL] R. Tolimieri, M. An, and C. Lu, *Algorithms for Discrete Fourier Transform and Convolution*, Springer-Verlag, New York, 1989.

[Tr] D. Travis, *Spherical functions on finite groups*, J. of Algebra 29 (1974), pp. 65–76.

[W] H. Wielandt, *Finite Permutation Groups*, Academic Press, New York, 1964.