JOURNAL OF COMPUTATIONAL PHYSICS **147,** 594–609 (1998)
ARTICLE NO. CP986104

A Generalized One-Dimensional Fast Multipole Method with Application to Filtering of Spherical Harmonics[1]

Norman Yarvin[2] and Vladimir Rokhlin

Department of Computer Science, Yale University, P.O. Box 208285 Yale Station, New Haven, Connecticut 06520-8285
E-mail: yarvin@cs.yale.edu, rokhlin@cs.yale.edu

Received June 1, 1998; revised September 22, 1998

The need to filter functions defined on the sphere arises in a number of applications, such as climate modeling, electromagnetic and acoustic scattering, and several other areas. Recently, it has been observed that the problem of uniform resolution filtering on the sphere can be performed efficiently via the fast multipole method (FMM) in one dimension. In this paper, we introduce a generalization of the FMM that leads to an accelerated version of the filtering process. Instead of multipole expansions, the scheme uses special-purpose bases constructed via the singular value decomposition of appropriately chosen submatrices of the filtering matrix. The algorithm is applicable to a fairly wide class of projection operators; its performance is illustrated with several numerical examples.

Key Words: singular value decompositions; fast algorithms; spherical harmonics.

1. INTRODUCTION

The fast multipole method (FMM) [6] is an $O(n)$ algorithm for calculating electrostatic potentials at n points due to a set of n charges. Variants of it exist in one [3, 15], two [6, 9], and three [7] dimensions. While the two- and three-dimensional variants have found direct uses, the one-dimensional version is normally used as a step in the solution of other numerical problems (see, for example, [3]). One such use of the one-dimensional FMM has recently been published by Jakob-Chien and Alpert [10], in an algorithm for the rapid uniform resolution filtering and interpolation of functions on the sphere; that algorithm has uses in the solution of partial differential equations on the sphere [13], in fast algorithms for

[1] Supported in part by DARPA/AFOSR under Grant F49620-97-1-0011, and in part by ONR under Grant N00014-96-1-0188.

[2] Corresponding author.

0021-9991/98 $25.00

electromagnetic scattering [4], and in several other environments. In this paper, we describe a version of the one-dimensional FMM which has been generalized so as to calculate not only electrostatic potentials, but a wide class of similar kernels, and we describe an accelerated version of the algorithm of [10] in which two subroutine calls to the original one-dimensional FMM are replaced by one call to the generalized FMM.

Formally, this paper describes an algorithm for the following task: given an $n \times m$ matrix P of a certain structure and given a desired accuracy ε, compress P so that its product with a vector can be efficiently computed to that accuracy. The structure the algorithm requires of P is as follows: there must exist numbers $x_1 < x_2 < \cdots < x_m$ and $y_1 < y_2 < \cdots < y_n$ such that, roughly speaking, any submatrix of P which is separated in index space from the line $x_i = y_j$ by a distance greater than its own size has a rank less than some (reasonably small) number r, to the precision ε; the CPU time taken by the algorithm for multiplication of P by a vector is then $O(nr)$. (A rigorous accounting of the execution time of the algorithm is somewhat complicated and is given in Section 3.2.6.) One matrix $P = [p_{ij}]$ which has such a structure is given by the formula

$$p_{ij} = \frac{1}{y_i - x_j} \tag{1}$$

and is the matrix whose multiplication by a vector is implemented by the original one-dimensional versions of the FMM.

This paper is arranged as follows. Section 2 briefly reviews numerical tools used by the algorithm. Section 3 describes the generalized FMM in its basic form. Section 4 describes modifications to the algorithm of Section 3, the principal one of which is the diagonalization of roughly a third of the interaction matrices. Section 5 contains numerical results for the generalized FMM applied to the matrix (1). Section 6 describes modifications to the algorithm of [10] which incorporate the generalized FMM. Finally, Section 7 examines generalizations of the schemes presented in this paper.

2. NUMERICAL PRELIMINARIES

2.1. *Singular Value Decomposition*

The singular value decomposition (SVD) is a ubiquitous tool in numerical analysis, given for the case of real matrices by the following lemma (see, for instance, [14] for more details).

LEMMA 2.1. *For any $n \times m$ real matrix A, there exist an integer p, an $n \times p$ real matrix U with orthonormal columns, an $m \times p$ real matrix V with orthonormal columns, and a $p \times p$ real diagonal matrix $S = [s_{ij}]$ whose diagonal entries are nonnegative, such that $A = USV^*$ and that $s_{ii} \geq s_{i+1,i+1}$ for all $i = 1, \ldots, p-1$.*

The diagonal entries s_{ii} of S are called singular values of A; the columns of the matrix V are called right singular vectors; the columns of the matrix U are called left singular vectors.

2.2. *Least Squares Approximation*

This section contains three lemmas on the least squares approximation of matrices, proven in a more general setting in [15]. In this section and in the remainder of the paper $\mathbb{R}^{n,m}$ will

denote the space of all real $n \times m$ matrices, and the matrix norm used will be the Schur or Frobenius norm; that is, for an $n \times m$ real matrix $A = [a_{ij}]$,

$$\|A\| = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij}^2}. \tag{2}$$

LEMMA 2.2. *Suppose A is a $p \times n$ real matrix, B is an $m \times k$ real matrix, and C is a $p \times k$ real matrix, for some m, p, n, and k. Let $A = \tilde{U}_A \tilde{S}_A \tilde{V}_A^*$ be a singular value decomposition of A, and let $B = \tilde{U}_B \tilde{S}_B \tilde{V}_B^*$ be a singular value decomposition of B. Let r be the number of nonzero singular values of A, and let q be the number of nonzero singular values of B. Let U_A and V_A consist of the first r columns of $\tilde{U}_A$ and $\tilde{V}_A$, respectively, and let S_A consist of the first r rows of the first r columns of $\tilde{S}_A$. Let U_B and V_B consist of the first q columns of $\tilde{U}_B$ and $\tilde{V}_B$, respectively, and let S_B consist of the first q rows of the first q columns of $\tilde{S}_B$. Then the solution $\hat{X}$ of the minimization problem,*

$$\min_{X \in \mathbb{R}^{n,m}} \|AXB - C\|, \tag{3}$$

is given by

$$\hat{X} = V_A S_A^{-1} U_A^* C V_B S_B^{-1} U_B^*. \tag{4}$$

Furthermore,

$$\|A\hat{X}B - C\| = \|C - U_A U_A^* C V_B V_B^*\|. \tag{5}$$

The following lemma provides a bound, in certain situations, on the error of the approximation given by Lemma 2.2.

LEMMA 2.3. *Under the conditions of Lemma* 2.2, *suppose that there exist an $n \times k$ matrix D and an $p \times m$ matrix E such that*

$$\|AD - C\| < \varepsilon_1 \tag{6}$$

and

$$\|EB - C\| < \varepsilon_2. \tag{7}$$

Then

$$\|A\hat{X}B - C\| < \varepsilon_1 + \varepsilon_2. \tag{8}$$

As shown by the following lemma, the error bound of Lemma 2.3 also applies when a different formula for the minimizing matrix is used.

LEMMA 2.4. *Under the conditions of Lemma* 2.3, *let the $n \times m$ matrix Y be given by the formula*

$$Y = DV_B S_B^{-1} U_B^*. \tag{9}$$

Then

$$\|AYB - C\| < \varepsilon_1 + \varepsilon_2. \tag{10}$$

3. BASIC FMM

This section describes the generalized FMM of this paper. It is described as a set of modifications to the FMM of [6, 3]; the reader is assumed to be familiar with that algorithm.

The overall FMM structure of an upward pass for creation of far field expansions, followed by a pass which computes local expansions from far field expansions, followed by a downward pass which propagates local expansions to lower levels and evaluates them, is retained. However, all the expansions are different, being based on singular value decompositions rather than on analytical formulae. In addition, the hierarchical subdivision scheme is different, being performed according to matrix indices rather than according to point locations. (The expansions used permit almost any subdivision scheme, whether adaptive as in [15], or nonadaptive as in [3]; the present scheme was chosen solely for its simplicity.)

3.1. *Subdivision Scheme*

The hierarchical subdivision is performed on column indices of the matrix P, as follows:

- Each interval of column indices, if it is divided, is divided into two intervals of equal size (or differing in size by one, if the number of indices in the interval is odd).
- The subdivision is uniform; either all the intervals at any given depth of the tree are subdivided, or none are.
- The subdivision process continues until the lowest-level intervals are as close as possible to a user-chosen size.

For each interval $[j_1, j_2]$ of column indices produced by the above process, a corresponding interval $[i_1, i_2]$ of row indices is chosen such that the portion of P addressed by the two intervals of indices contains as much as possible of the line $x_i = y_j$. The precise criterion used to choose the interval $[i_1, i_2]$ is that it should be the interval of maximal size such that

$$\left(x_{j_1-1} + x_{j_1}\right)/2 \le y_{i_1} < \cdots < y_{i_2} < \left(x_{j_2} + x_{j_2+1}\right)/2. \tag{11}$$

(If x_{j_1-1} or x_{j_2+1} does not exist, the corresponding inequality in the above equation is not enforced. The quantities $x_1 < x_2 < \cdots < x_m$ and $y_1 < y_2 < \cdots < y_n$ were, in the present implementation, user-provided; in an environment where they are not readily available, they can be determined by numerically searching P for areas of high numerical rank.)

3.2. *Expansions*

This section describes the expansions used in the generalized FMM. Submatrices of P will be designated as follows: $P_{a,b}$ denotes the portion of P whose column indices are in b and whose row indices are in a, where a and b are either intervals of indices into P, or sets thereof.

For each interval, the FMM divides the intervals at the same depth in the tree into two sets:

- 1. The *near field* region, consisting of the interval itself and the two adjacent intervals at the same depth in the tree of intervals.
- 2. The *far field* region, consisting of all remaining intervals at the same depth in the tree. We denote the far field region of the i'th interval by F_i.

A third set is also required: the *interaction list* of an interval i is the set of intervals at the same depth in the tree which are in the far field of i and which are not in the far field of the parent of i.

3.2.1. Far-field expansions. The original FMM [6] relies on the fact that the electrostatic potential due to a set of charges can be represented to high precision, at points distant from those charges, by a multipole expansion of relatively few terms. In the generalized FMM described in this paper, the output (no longer necessarily the electrostatic potential, although we will continue to use the terms "potential" and "charge" for convenience) does not need to be describable by a multipole expansion, but can be describable by an arbitrary expansion, provided that the expansion coefficients are linear functions of the charge magnitudes and that the potential is a linear function of the expansion coefficients. The creation and evaluation matrices for this expansion, which we will call a far-field expansion, do not need to be furnished as such by the user; they are computed from the matrix P using the singular value decomposition. This computation is performed for each interval i for which a far-field expansion is needed and is as follows: Let $n_i \times m_i$ be the dimensions of the matrix $P_{F_i,i}$, let the singular value decomposition of $P_{F_i,i}$ be denoted by $\tilde{U}\tilde{S}\tilde{V}^*$, the number of singular values by $\tilde{p}$, and the singular values by $s_1 \geq s_2 \geq \cdots \geq s_{\tilde{p}}$. Let p_i be the minimum integer such that

$$\sum_{j=p_i+1}^{\tilde{p}} s_j^2 < \varepsilon^2 \|P\|^2 \frac{n_i m_i}{nm}. \tag{12}$$

Let the $m_i \times p_i$ matrix V_i consist of the first p_i columns of $\tilde{V}$ and let the $p_i \times n_i$ matrix E_i consist of the first p_i columns of the product $\tilde{U}\tilde{S}$. We will refer to V_i^* as the far-field expansion creation matrix for interval i and to E_i as the far-field evaluation matrix; the latter is not used explicitly in the algorithm.

As shown in [8], the product $E_i V_i^*$ is, among matrices of rank p_i, the closest approximation to the matrix $P_{F_i,i}$ in the norm (2). Thus the number of terms in any known expansion for $P_{F_i,i}$ (such as a multipole expansion) is an upper bound for the number of terms p_i in the far-field expansion of the same accuracy computed as above.

3.2.2. Local expansions. Using far-field expansions alone, an $O(n \cdot \log n)$ version of the FMM can be produced (for an overview of the various versions see [7]). The $O(n)$ version of the FMM requires additional numerical machinery, namely local expansions, which approximate the potential on a region due to charges on distant regions. In the original FMM, local expansions were harmonic expansions; in the generalized FMM, creation and evaluation matrices for local expansions are computed from the matrix P using the singular value decomposition, as follows. Let $n_i' \times m_i'$ be the dimensions of the matrix P_{i,F_i}; let the singular value decomposition of P_{i,F_i} be denoted by $\tilde{U}\tilde{S}\tilde{V}^*$, the number of singular values by $\tilde{r}$, and the singular values by $s_1 \geq s_2 \geq \cdots \geq s_{\tilde{r}}$. Let r_i be the minimum integer such that

$$\sum_{j=r_i+1}^{\tilde{r}} s_j^2 < \varepsilon^2 \|P\|^2 \frac{n_i' m_i'}{nm}. \tag{13}$$

Let the $m_i' \times r_i$ matrix U_i consist of the first r_i columns of $\tilde{U}$. We will refer to U_i as the local expansion evaluation matrix for interval i.

3.2.3. Far-field translation matrices. The FMM does not compute far-field expansions for intervals at high levels in the tree directly from the charges in the interval, but rather computes them from far-field expansions at lower levels. Associated with each interval i whose parent interval j has a far-field expansion is a translation matrix T_i which takes as input a far-field expansion for i and produces as output a far-field expansion for j which evaluates to the same potential. Let V_i^* be the far-field creation matrix for interval i, and let $V_{j,i}^*$ be the far field creation matrix for interval j, with columns deleted such that it only accepts input from the interval i. Clearly the translation matrix T_i should be such that for any m_i-vector q, the vector $T_i V_i^* q$ is as close as possible, by some measure, to the vector $V_{j,i}^* q$. The measure we use is the least squares measure; in particular, T_i is chosen so as to minimize the quantity $\|V_{j,i}^* - T_i V_i^*\|$. The formula for such minimization is given by Lemma 2.2; using the fact that the singular value decomposition of any matrix with orthogonal columns consists of that matrix multiplied by two identity matrices, it reduces in this case to

$$T_i = V_{j,i}^* V_i. \tag{14}$$

We will refer to T_i as the far-field expansion translation matrix for interval i.

Lemma 2.4 gives a bound for the error associated with using the translation matrix T_i. Suppose $E_{j,k}$ and $E_{i,k}$ are matrices which take as input the far-field expansions on interval j and on interval i, respectively, and use them to evaluate the potential on some other interval k and are such that

$$\|P_{i,k} - E_{j,k} V_{j,i}^*\| < \varepsilon_1 \tag{15}$$

$$\|P_{i,k} - E_{i,k} V_i^*\| < \varepsilon_2. \tag{16}$$

Using (15), (16), and Lemma 2.4, we get that

$$\|P_{i,k} - E_{j,k} T_i V_i^*\| < \varepsilon_1 + \varepsilon_2. \tag{17}$$

3.2.4. Local expansion translation matrices. The FMM does not evaluate local expansion for intervals at high levels in the tree directly at each of the points at which the potential is to be evaluated, but rather transforms them into local expansions for intervals at lower levels. Associated with each interval i, whose parent interval j has a local expansion, is a translation matrix M_i which takes as input a local expansion on j and produces as output a local expansion on i. M_i is computed as follows. Let U_i be the local expansion evaluation matrix for interval i, and let $U_{j,i}$ be the local expansion evaluation matrix for interval j, with rows deleted so that it only produces output on the interval i. Clearly the translation matrix M_i should be such that for any r_i-vector α, the vector $U_i M_i \alpha$ is as close as possible, by some measure, to the vector $U_{j,i}\alpha$. The measure we use is the least squares measure; in particular, M_i is chosen so as to minimize the quantity $\|U_{j,i} - U_i M_i\|$. The formula for such minimization is given by Lemma 2.2. Using the fact that the singular value decomposition of any matrix with orthogonal columns consists of that matrix multiplied by two identity matrices, it reduces in this case to

$$M_i = U_i^* U_{j,i}. \tag{18}$$

The error incurred by using M_i is bounded by Lemma 2.4; the analysis is almost identical to that presented in Section 3.2.3 for the far-field translation matrix T_i and is omitted. We will refer to M_i as the local expansion translation matrix for interval i.

3.2.5. Far-field to local interaction matrices. A far-field to local interaction matrix $E_{j,i}$ takes as input a far-field expansion on an interval i and produces as output a local expansion on another interval j. Such matrices are constructed only for pairs of intervals (i, j) such that j is in the interaction list of i. The matrix $E_{j,i}$ should be such that for all m_i-vectors q the product $U_j E_{j,i} V_i^* q$ is as close as possible, by some measure, to the product $P_{j,i} q$. We choose $E_{j,i}$ so as to minimize the quantity

$$\varepsilon_{j,i} = \| U_j E_{j,i} V_i^* - P_{j,i} \|. \tag{19}$$

The formula for such minimization is given by Lemma 2.2; using the fact that the singular value decomposition of any matrix with orthogonal columns consists of that matrix multiplied by two identity matrices, it reduces in this case to

$$E_{j,i} = U_j^* P_{j,i} V_i. \tag{20}$$

Lemma 2.3, combined with (12) and (13), gives a bound for $\varepsilon_{j,i}$:

$$\varepsilon_{j,i} < \varepsilon \| P \| \left(\sqrt{\frac{n_i m_i}{nm}} + \sqrt{\frac{n_i' m_i'}{nm}} \right). \tag{21}$$

We will refer to $E_{j,i}$ as the far field to local interaction matrix from interval i to interval j.

Remark 3.1. A brief inspection of the above formulae for the creation, translation, and evaluation matrices $\{U_i\}$, $\{V_i\}$, $\{T_i\}$, $\{M_i\}$, and $\{E_{j,i}\}$ shows that the same matrices are generated, in different roles, if the input matrix to the algorithm is the adjoint P^* of P, provided that the hierarchical subdivision is retained: the far field expansion creation matrices for P are identical to the local expansion evaluation matrices for P^*, and vice versa; the far field translation matrices for P are identical to the local expansion translation matrices for P^*, and vice versa; and the far field to local matrices for P are the adjoints of the far field to local matrices for P^*. Thus the matrices precomputed for P can also be used for multiplying by P^*.

3.2.6. Execution time. The FMM performs one matrix–vector multiplication for each instance of the matrices $\{U_i\}$, $\{V_i\}$, $\{T_i\}$, $\{M_i\}$, and $\{E_{j,i}\}$. Thus the CPU time which it consumes is proportional to the total number of elements in all instances of the matrices. The sizes of the matrices depend on the numerical ranks p_i and r_i, as defined by (12) and (13). We analyze the execution time further only in the case that all those ranks are all bounded by some number r. In that case, the computation of far-field expansions from the input takes $O(mr)$ time, the computation of the output from local expansions takes $O(nr)$ time, and the computations of expansions from other expansions take $O(kr^2)$ time, where k is the total number of intervals produced by the subdivision process. Assuming that m is proportional to n, the total execution time is $O(nr + kr^2)$. The quantity $nr + kr^2$ is minimized (with respect to k) when n/k is equal to r. Since n/k is proportional to the size of the lowest-level intervals, the minimum execution time occurs when the size of the lowest-level intervals is proportional to r, with the constant of proportion depending on the details of the computer involved.

4. TECHNICAL IMPROVEMENTS

4.1. *Diagonalization of Far Field to Local Matrices*

A certain amount of freedom is present in the definition of far field and local expansions: the results of the FMM are clearly unaffected if the far-field expansion creation matrix V_i^* for an interval i is multiplied on the left by any orthogonal matrix W, its far field translation matrix T_i is multiplied on the right by W^*, and its far field to local matrices $E_{j,i}$ for all j are multiplied on the right by W^*. Similarly, the results of the FMM are unaffected if the local expansion evaluation matrix U_i for an interval i is multiplied on the right by any orthogonal matrix W, its local expansion translation matrix M_i is multiplied on the left by W^*, and its far field to local matrices $E_{i,j}$ for all j are multiplied on the left by W^*.

We use this freedom to diagonalize one of the (usually three) far field to local matrices for each interval. Suppose that $E_{i,j}$ for some intervals i and j is the matrix to be diagonalized. Let its singular value decomposition be denoted by $E_{i,j} = USV^*$. Then we multiply V_j^* on the right by V^*, and multiply U_i on the left by U, also changing translation matrices and far field to local matrices as indicated in the previous paragraph so that the results of the FMM are unaffected.

Far field to local matrices are chosen for diagonalization in such a way that each expansion redefined by this process is redefined only once. The scheme used is as follows: each level of intervals is divided into blocks of four adjacent intervals; inside each block the interactions chosen for diagonalization are: $1 \to 3$, $2 \to 4$, $3 \to 1$, and $4 \to 2$ (as depicted in Fig. 1).

4.2. *Splits by Factors Other Than Two*

Another modification which was made to the above FMM is to split intervals into more than two pieces. This clearly can be done to any interval, at any level in the tree. However, the only use which was made of this flexibility was to alter the top of the tree of intervals slightly, so as to control better the size of the lowest-level intervals in the tree. The top interval was split either into two, three, or five pieces; if three, its subintervals might each

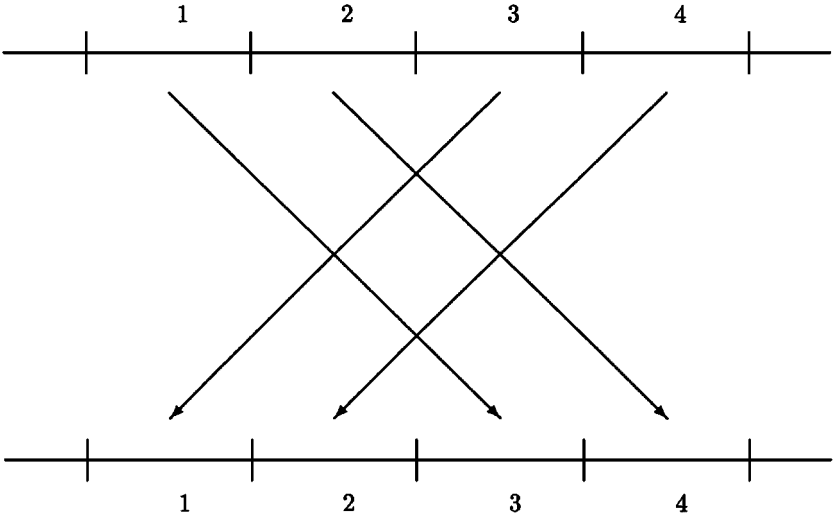

FIG. 1. Far field to local operators which are diagonalized.

TABLE I
Double Precision Timings for the $1/x$ Kernel

N	Error (L^2 norm)	Times (seconds) Init	Eval	Direct	Ratio eval/FFT	Memory (REAL*8 spaces)
64	0.35477E-15	0.070	0.001	0.001	5.21	3852
128	0.92042E-15	0.820	0.003	0.005	7.31	10407
256	0.23512E-14	6.620	0.007	0.019	8.93	26205
512	0.16144E-13	39.700	0.013	0.073	5.60	52263
1024	0.21925E-13	214.710	0.031	0.730	4.16	117881

be split into three parts, the remaining intervals in the tree all being split into two parts. This permits a choice of the size of the lowest-level intervals not only of $n/2^k$ for any k, but also of $n/(3 \times 2^k)$, $n/(5 \times 2^k)$, or $n/(9 \times 2^k)$.

5. NUMERICAL RESULTS

For comparison against the older one-dimensional FMMs of [3, 15], the generalized FMM was applied to the $1/x$ kernel; that is, the input matrix $P = [p_{ij}]$ was given by (1). Timings for various numbers of points n are listed in Tables I and II for double and single precision (that is, with the parameter ε set to 10^{-14} and 10^{-7}). In all cases, the parameter m was set to be equal to n, the nodes $\{x_i\}$ were identical to the nodes $\{y_i\}$, being slightly perturbed equispaced nodes. All timings were performed on a Sun Sparcstation 10 in double precision (Fortran `REAL*8`) arithmetic. Also included in the tables are ratios of the execution time of the algorithm to the execution time of a standard SLATEC FFT of size n.

From the timings, it can be seen that the generalized FMM is similar in execution speed to the best previous 1D FMM (that of [15]) known to the authors. It is, however, far inferior to the FMMs of [3, 15] in the time spent in the precomputation stage; initialization times for those algorithms did not exceed execution time by more than a factor of 10, whereas the initialization time for the generalized FMM exceeds the execution time by factors of 1000s. Effectively, it limits the usefulness of the procedure of this paper to problems of sufficient importance that the initialization data can be precomputed and stored. The following section discusses one such case.

TABLE II
Single Precision Timings for the $1/x$ Kernel

N	Error (L^2 norm)	Times (seconds) Init	Eval	Direct	Ratio eval/FFT	Memory (REAL*8 spaces)
64	0.25040E-08	0.040	0.001	0.001	4.74	3500
128	0.23352E-07	0.440	0.002	0.005	5.90	8465
256	0.19125E-06	3.580	0.005	0.018	6.13	17803
512	0.64886E-06	22.710	0.010	0.074	4.03	36911
1024	0.28910E-06	124.690	0.021	0.590	2.77	79407

6. APPLICATION TO FILTERING

This section describes a use of the generalized FMM, in an algorithm recently published by Jakob-Chien and Alpert [10] for uniform resolution filtering of functions on the sphere. Their algorithm as a whole performs the following task: given numbers $f(\phi_i, \theta_j)$, $i = 1, \ldots, I$; $j = 1, \ldots, J$, such that

$$f(\phi_i, \theta_j) = \sum_{n=0}^{K} \sum_{m=-n}^{n} f_n^m Y_n^m(\phi_i, \theta_j), \tag{22}$$

computes numbers $\tilde{f}(\tilde{\phi}_i, \tilde{\theta}_j)$ such that

$$\tilde{f}(\tilde{\phi}_i, \tilde{\theta}_j) = \sum_{n=0}^{N} \sum_{m=-n}^{n} f_n^m Y_n^m(\tilde{\phi}_i, \tilde{\theta}_j), \tag{23}$$

where the functions Y_n^m are the surface harmonics and where $\{\phi_i\}$, $\{\theta_j\}$, $\{\tilde{\phi}_i\}$, and $\{\tilde{\theta}_j\}$ are appropriately chosen grid points (see [10] for details).

We modify only the core of the algorithm of [10], which performs the following one-dimensional filtering operation: given numbers $f^m(\theta_1), \ldots, f^m(\theta_J)$ such that

$$f^m(\theta_i) = \sum_{j=m}^{J-1} f_j^m \bar{P}_j^m(\mu_i), \quad i = 1, \ldots, J, \tag{24}$$

compute numbers $\tilde{f}^m(\tilde{\theta}_1), \ldots, \tilde{f}^m(\tilde{\theta}_N)$ such that

$$\tilde{f}^m(\tilde{\theta}_i) = \sum_{j=m}^{N} f_j^m \bar{P}_j^m(\tilde{\mu}_i), \quad i = 1, \ldots, N, \tag{25}$$

where the functions $\bar{P}_n^m$ are the normalized associated Legendre functions, $\mu_i = \sin\theta_i$ and $\tilde{\mu}_i = \sin\tilde{\theta}_i$.

Due to the orthonormality of the functions $\bar{P}_n^m$ for fixed m and integer $n \geq m$, if the nodes $\mu_1, \ldots, \mu_J$ are Legendre nodes (nodes of the Gaussian quadrature corresponding to the weight function $\omega(x) = 1$; see, for instance, [14]), then the coefficients f_m^m, $f_{m+1}^m, \ldots, f_N^m$ are given by

$$f_n^m = \sum_{j=1}^{J} f^m(\theta_j) \bar{P}_n^m(\mu_j) w_j, \tag{26}$$

where $w_1, \ldots, w_J \in \mathbb{R}$ are the Gaussian weights corresponding to the nodes $\mu_1, \ldots, \mu_J$. Combining (25) and (26) yields an equation for the entire filtering operation:

$$\tilde{f}^m(\tilde{\theta}_i) = \sum_{k=1}^{J} f^m(\theta_k) w_k \sum_{j=m}^{N} \bar{P}_j^m(\mu_k) \bar{P}_j^m(\tilde{\mu}_i). \tag{27}$$

Equation (27) constitutes a linear transformation from $f^m(\theta_1), \ldots, f^m(\theta_J)$ to $\tilde{f}^m(\tilde{\theta}_1), \ldots, \tilde{f}^m(\tilde{\theta}_N)$; we will refer to the matrix of this transformation as the filtering matrix and will

denote it by P. Using the Christoffel–Darboux formula for the associated Legendre functions (see, for instance, [1, Section 8.9.1]), which is

$$(\tilde{\mu} - \mu) \sum_{n=|m|}^{N} \bar{P}_n^m(\tilde{\mu}) \bar{P}_n^m(\mu) = \varepsilon_{N+1}^m \big(\bar{P}_{N+1}^m(\tilde{\mu}) \bar{P}_N^m(\mu) - \bar{P}_N^m(\tilde{\mu}) \bar{P}_{N+1}^m(\mu) \big), \tag{28}$$

where

$$\varepsilon_n^m = \sqrt{(n^2 - m^2)/(4n^2 - 1)}, \tag{29}$$

the filtering operation can be written as

$$\frac{\tilde{f}^m(\tilde{\theta}_j)}{\varepsilon_{N+1}^m} = \bar{P}_{N+1}^m(\tilde{\mu}_j) \sum_{i=1}^{J} \frac{f^m(\theta_i) w_i \bar{P}_N^m(\mu_i)}{\tilde{\mu}_j - \mu_i} - \bar{P}_N^m(\tilde{\mu}_j) \sum_{i=1}^{J} \frac{f^m(\theta_i) w_i \bar{P}_{N+1}^m(\mu_i)}{\tilde{\mu}_j - \mu_i}. \tag{30}$$

From (30) it immediately can be seen that the filtering matrix consists of the sum of two matrices of the form (1), each multiplied on the left and the right by a diagonal matrix. Thus, the filter can be implemented using two calls to an FMM for the $1/x$ kernel; this is the method presented in [10] (from where the above analysis is copied). It also follows that, if the generalized FMM of this paper is applied to the filtering matrix, the numerical ranks $\{r_i\}$ and $\{p_i\}$ (see (13) and (12)) are no more than twice the corresponding ranks when the generalized FMM is applied to a matrix of the form (1). Thus, the filter can be implemented efficiently via a single call to the generalized FMM.

Remark 6.1. If N is larger than J, the operation (30) amounts to interpolation rather than filtering. If the output nodes $\{\tilde{\mu}_i\}$ are the Legendre nodes of order N, then the filtering matrix from J nodes to N nodes is, except for the multiplication of the input by Gaussian weights, the adjoint of the interpolation matrix from N nodes to J nodes; this can easily be seen by inspection of (30). Thus, the matrices $\{U_i\}$, $\{V_i\}$, $\{T_i\}$, $\{M_i\}$, and $\{E_{j,i}\}$, precomputed for the purpose of filtering, can also be used for interpolation (see Remark 3.1).

6.1. *General Nodes*

If the nodes $\mu_1, \ldots, \mu_J$ are not Legendre nodes, then the coefficients $f_m^m, \ldots, f_N^m$ cannot be computed by direct use of the formula (26). In this case, two methods of performing the filtering operation are available. First, Eq. (24) can be solved for the coefficients $f_m^m, \ldots, f_J^m$. Alternatively, the function can be interpolated onto Legendre nodes, following which the filtering matrix for Legendre nodes (30) can be used. We use the second method to show that the filtering matrix for general nodes can be compressed by the generalized FMM; we used the first method in our implementation.

As is well known (see, for instance, [1]), each of the associated Legendre functions P_n^m is either a polynomial or a polynomial multiplied by $\sqrt{1 - x^2}$, depending on whether m is even or odd. Thus the interpolation onto Legendre nodes is a polynomial interpolation, which, if m is odd, is preceded by a division by $\sqrt{1 - x^2}$ and followed by a multiplication by $\sqrt{1 - x^2}$. As shown in [3], polynomial interpolation can be performed in $O(n)$ time using an FMM. The filtering matrix for general nodes is the product of the interpolation matrix and the filtering matrix for Legendre nodes; since each of these can be compressed by a generalized FMM, their product also can be compressed by a generalized FMM (see [2]).

Remark 6.2. In the solution of Eq. (24) for the coefficients $f_m^m, \ldots, f_N^m$, when $m > 0$, there are more equations than unknowns. The definition of the problem is such that there is an exact solution; however, numerically, this issue was dealt with by solving the equation in the least squares sense.

6.2. *Optimizations*

The above filtering algorithm admits several optimizations. We describe them only for the case when the nodes $\mu_1, \ldots, \mu_J$ are Legendre nodes; however, all of them have also been implemented in the case of general nodes.

First, when m is close to N, the number of coefficients f_j^m to be extracted is small; thus direct computation of (26) followed by (25) is the most efficient algorithm for the filter.

Second, portions of the filtering matrix have negligible norm and can be discarded. This can be easily seen by examination of (30), using the fact that the functions P_n^m take on small values near the endpoints of the interval $[-1, 1]$. The fraction of the matrix which can be discarded increases with increasing m, to as much as eight ninths. This optimization is clearly not specific to the generalized FMM; it can be applied equally well to the direct method or to the unaltered algorithm of [10] and was applied to the direct method code which was used in the timings presented below.

Third, the filter can be speeded up slightly by splitting the input function into odd and even parts, and filtering them separately. Each of the associated Legendre functions P_n^m is either odd or even, with functions of successive degree n being alternately odd and then even. Thus the filter, applied to an odd function, yields an odd function and, applied to an even function, yields an even function. This implies that the filtering matrix is block-diagonalized (into two blocks) by the separation of odd functions from even functions. We address only the case in which the separation can be done trivially, that is, when each of the sets of nodes $\{\mu_i\}$ and $\{\tilde{\mu}_i\}$ is symmetric around zero; for brevity of explanation, we further assume that N and J are even. In this case the separation of odd functions from even functions is accomplished by the usual formulae

$$f_{\text{odd}}(x) = (f(x) - f(-x))/2, \tag{31}$$

$$f_{\text{even}}(x) = (f(x) + f(-x))/2, \tag{32}$$

where, as usual, each of the functions f_{odd} and f_{even} are symmetric around zero and, thus, need only be stored at half the nodes. It is easily shown, using (30) and (31), that in the case that the nodes $\mu_1, \ldots, \mu_J$ are Legendre nodes, each block $\hat{P} = [\hat{p}_{ij}]$ of the block-diagonalized filtering matrix is given by

$$\begin{aligned}\hat{p}_{ij} = &\frac{\bar{P}_{N+1}^m(\tilde{\mu}_j)\bar{P}_N^m(\mu_i)w_i - \bar{P}_N^m(\tilde{\mu}_j)\bar{P}_{N+1}^m(\mu_i)w_i}{\tilde{\mu}_j - \mu_i} \\ &\pm\frac{\bar{P}_{N+1}^m(\tilde{\mu}_j)\bar{P}_N^m(\mu_i)w_i + \bar{P}_N^m(\tilde{\mu}_j)\bar{P}_{N+1}^m(\mu_i)w_i}{\tilde{\mu}_j + \mu_i},\end{aligned} \tag{33}$$

where, for the block which filters even functions, the "$\pm$" sign is an addition, and, for the block which filters odd functions, it is a subtraction. An inspection of (33) immediately shows that each block is compressible by a generalized FMM.

Remark 6.3. Experimentally, the ranks produced by the generalized FMM when applied to the block-diagonalized matrix are almost identical to the ranks produced when applied to the original filtering matrix, except near the point $\mu = 0$, where the ranks are slightly smaller in the block-diagonalized version.

Remark 6.4. Since the generalized FMM is, when applied to matrices of this form, an $O(n)$ procedure, splitting the problem into two problems of half the size does not produce any asymptotic improvement in execution time, although it does produce an improvement for small to medium-sized n. By contrast, applying this optimization to the direct method (as was done in the code used in the timings presented below) reduces the execution time by a factor of 2 asymptotically, since the direct method is $O(n^2)$.

6.3. *Numerical Results*

Table III contains experimental results for the filter for functions tabulated at Legendre nodes. The filter was run for several values of J, with $N = J/2$ and for each $m = 1, \ldots, N$; the average initialization and execution times, the average L^2 error, and the average amount of memory used for precomputed data (for all values of m) are tabulated. The quantity labeled as initialization time is, as before, the amount of time taken to compute the matrices which comprise the generalized FMM; this task only needs to be performed once for any combination of J and N, since the precomputed matrices can be stored. All figures were produced by an implementation in double precision (Fortran `REAL*8`) arithmetic on a Sun Sparcstation 10. The table also contains the amount of time taken by the direct method and the ratio of the execution time of the FMM-based filter to the execution time of a standard

TABLE III
Filter Timings for Points Tabulated at Legendre Nodes

	Average time per m (seconds) for					
J	Direct	FMM eval	FMM init	Ratio: eval/FFT	Average error (L^2)	Average memory used (`REAL*8` spaces)
			Requested accuracy 10^{-3}			
64	0.00014	0.00021	0.038	1.10	0.87216E-04	637
128	0.00059	0.00063	0.173	1.73	0.21141E-03	1814
256	0.00239	0.00172	0.861	2.25	0.35270E-03	4684
512	0.00916	0.00406	4.528	1.64	0.55393E-03	10586
1024	0.15601	0.00930	22.708	1.26	0.72021E-03	22799
			Requested accuracy 10^{-7}			
64	0.00016	0.00020	0.035	1.05	0.62995E-09	715
128	0.00069	0.00068	0.145	1.84	0.89805E-08	2351
256	0.00272	0.00199	0.749	2.61	0.20946E-07	7074
512	0.01015	0.00545	4.480	2.21	0.35158E-07	18763
1024	0.17623	0.01351	25.102	1.84	0.50011E-07	45001
			Requested accuracy 10^{-12}			
64	0.00017	0.00018	0.035	0.97	0.64733E-13	712
128	0.00078	0.00070	0.118	1.88	0.36187E-12	2604
256	0.00312	0.00221	0.630	2.90	0.13528E-12	8496
512	0.01102	0.00656	3.752	2.64	0.30608E-12	26072
1024	0.19227	0.01763	26.347	2.37	0.14238E-11	66714

SLATEC FFT of size J. The direct method for which timings are listed is a modestly optimized variant: the filtering matrix it used was precomputed; certain optimizations used for the FMM-based method were also applied to it, as described in Section 6.2.

The filter was also implemented for functions tabulated at general nodes (Section 6.1) and was tested on Chebyshev nodes. The timings are almost identical, with the only major difference being that considerably more time was required to compute the filtering matrix; they are omitted.

Remark 6.5. The implausibly large CPU times taken by the direct method for $J = 1024$ are the result of the problem size exceeding the size of the cache; on the machine on which timings were run, only two double precision vectors of length 1024 fit in the data cache. Such a jump in timings is not expected to occur on most machines and, in any case, could be eliminated by use of a blocked matrix–vector multiplication routine.

Figure 2 is a graph of the average numerical rank of interaction found by the filter for Legendre nodes (the average of the ranks $\{p_i\}$), plotted as a function of m, for $J = 1024$ and $\varepsilon = 10^{-12}$. (The ranks for the filter for arbitrary nodes, when applied to Chebyshev nodes,

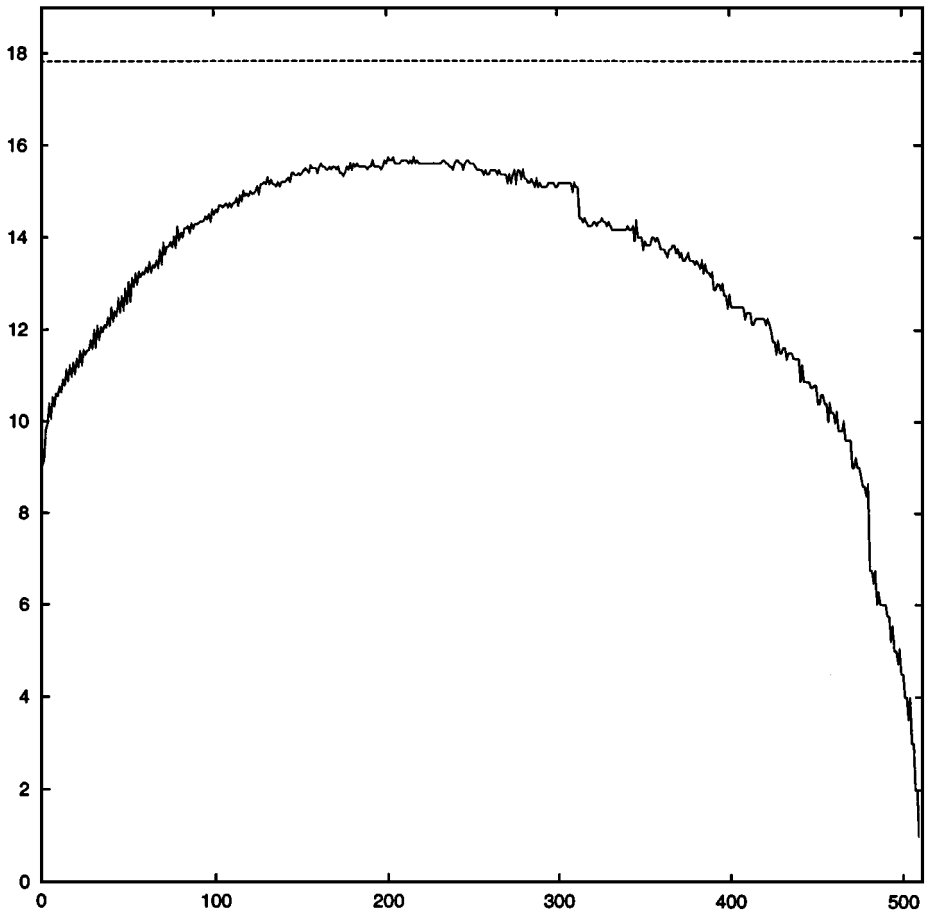

FIG. 2. Average numerical rank of interaction, as a function of m, for $J = 1024$ and $\varepsilon = 10^{-12}$. The dashed line is the theoretical bound on the rank.

were nearly identical.) Also plotted in Fig. 2 is the theoretical upper bound for the average rank, that is, twice the average rank of an FMM for the $1/x$ kernel of the same accuracy. Since most of the ranks were close to their average, the execution time of the FMM is roughly proportional to the average rank. (See Section 3.2.6 for an analysis of the case of all ranks being equal; a similar analysis applies to other variants of the 1D FMM.) Thus, Fig. 2 provides a rough indication of the amount of speedup that is obtained by switching from the scheme of [10] to the generalized FMM: to a first approximation, if the average rank were equal to its upper bound for all m, the two schemes would be of equal speed; to the extent that it is lower, the generalized FMM is faster. (However, it should be noted that the generalized FMM requires more precomputed data and is, thus, more vulnerable to caching effects.)

7. GENERALIZATIONS

In this paper, we have presented a scheme for the efficient filtering of functions on the two-dimensional sphere. The approach is based on two observations. The first observation is that in the fast multipole method (see, for example, [3, 6]) potential kernels can be replaced with functions from a much more general class, using the standard singular value decomposition, and that this yields a fairly efficient implementation. The second observation is that the Christoffel–Darboux formula (28) provides a straightforward proof that the filtering operator on the sphere (27) can be compressed by FMM-type techniques. Both observations admit far-reaching generalizations, outlined below.

1. The fast multipole method used in this paper is a special case of an extremely general procedure. Particular versions of this procedure have been used repeatedly (see [11, 12]); it is effective in all situations when the operator can be compressed by wavelet techniques. The following is a brief outline of the approach.

Given a matrix to be rapidly applied to arbitrary vectors, examine it (either analytically or numerically), identifying large submatrices that are of low rank. When the coefficients of a submatrix are a sufficiently smooth function of its indices, such a submatrix is guaranteed to have a low rank (this is the environment where wavelets and wavelet-type techniques can be used); another frequently encountered situation involves submatrices that are not smooth, but are smooth matrices multiplied by diagonal matrices from the left and/or from the right (as in the case of the filtering operator (30)). Any matrix whose rank is much lower than its dimensionality is "compressed" by its singular value decomposition; applying this procedure to a sufficiently large collection of submatrices of some matrix, we obtain a primitive "fast" algorithm for applying it to arbitrary vectors. The scheme is further accelerated by recursive application of this approach.

A strong argument can be made that the SVD of a matrix is its "optimal" low-rank representation; in this sense, SVD-based implementations of FMM-type algorithms are "optimal." Indeed, schemes have been constructed using the SVD to further compress multipole expansions (see, for example, [3, 9]); the resulting procedures tend to be more efficient than the original FMM. In addition, the FMM for potential kernels has been accelerated (dramatically so, in higher dimensions) by using diagonal forms of translation operators (see [7, 15]). Possible hybrid algorithms combining the latter with SVD-based compression of more general kernels are currently under investigation in one, two, and three dimensions.

2. Formula (28) in the present paper is a special case of the well-known Christoffel–Darboux formula,

$$\sum_{k=0}^{n} p_k(x) \cdot p_k(y) = \frac{q_n}{q_{n+1}} \cdot \frac{p_{n+1}(x) \cdot p_n(y) - p_{n+1}(y) \cdot p_n(x)}{x - y}, \tag{34}$$

where p_k are polynomials orthogonal with *some* weight function w on *some* interval, q_k is the coefficient at the term x^k in the polynomial p_k, and n is an arbitrary positive integer (see, for example, [5, Section 8.902]). It is immediately clear from (34) that the algorithm of this paper can be used to evaluate rapidly the projections in spaces of polynomials on subspaces consisting of polynomials of reduced rank, in the norm associated with the weight w. There are a number of other projections that can be evaluated rapidly using the FMM scheme of this paper, or its variants. The operators we have experimented with include projections on subspaces in the space of polynomials in two dimensions, projections on subspaces spanned by appropriately chosen Bessel functions, and several others. In some cases, we have determined experimentally that the scheme works, but have not constructed the underlying mathematics. This whole class of issues is currently under investigation.

REFERENCES

1. M. Abramowitz and I. Stegun, *Handbook of Mathematical Functions*, Appl. Math. Ser. (Natl Bureau of Standards, Washington, DC, 1964).
2. B. Alpert, G. Beylkin, R. Coifman, and V. Rokhlin, Wavelet-like bases for the fast solution of second-kind integral equations, *SIAM J. Sci. Comput.* **14**(1), 159 (1993).
3. A. Dutt, M. Gu, and V. Rokhlin, Fast algorithms for polynomial interpolation, integration, and differentiation, *SIAM J. Numer. Anal.* **33**(5), (1996).
4. M. A. Epton and B. Dembart, Multipole translation theory for the three-dimensional Laplace and Helmholtz equations, *SIAM J. Sci. Comput.* **16**(4), 865 (1995).
5. I. S. Gradshteyn and I. M. Ryzhik, *Table of Integrals, Series, and Products*, 5th ed. (Academic Press, New York, 1994).
6. L. Greengard and V. Rokhlin, A fast algorithm for particle simulations, *J. Comput. Phys.* **73**(2), 325 (1987).
7. L. Greengard and V. Rokhlin, A new version of the fast multipole method for the Laplace equation in three dimensions, *Acta Numer.* 229 (1997).
8. V. H. Golub and C. H. Van Loan, *Matrix Computations* (Johns Hopkins Univ. Press, Baltimore, 1983).
9. T. Hrycak and V. Rokhlin, *An Improved Fast Multipole Algorithm for Potential Fields*, Research Report 1089, Computer Science Department, Yale, 1995.
10. R. Jakob-Chien and B. Alpert, A fast spherical filter with uniform resolution, *J. Comput. Phys.* **136**(2), 580 (1997).
11. S. Kapur and D. E. Long, IES^3: A fast integral equation solver for efficient 3-dimensional extraction, in *37th International Conference on Computer Aided Design, Nov. 1997*.
12. S. Kapur, D. E. Long, and J. Zhao, Efficient full-wave simulation in layered, lossy media, in *Proceedings of the IEEE Custom Integrated Circuits Conference, May 1998*.
13. S. A. Orszag, Fourier series on spheres, *Mon. Weather Rev.* **102**, 56 (1974).
14. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, 2nd ed. (Springer-Verlag, New York/Berlin, 1993).
15. N. Yarvin and V. Rokhlin, An improved fast multipole algorithm for potential fields on the line, *SIAM J. Numer. Anal.*, to appear.