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

Norman Yarvin and Vladimir Rokhlin
Research Report YALEU/DCS/RR-1142
January 1998

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

An algorithm is described for the rapid application to arbitrary vectors of a fairly broad class of matrices. The scheme can be viewed as a generalized (and somewhat accelerated) version of the Fast Multipole Method in one dimension. In place of multipole expansions, the new scheme uses singular value decompositions of appropriately chosen submatrices of the matrix to be applied. The scheme is applied to the uniform resolution filtering of functions on the sphere, in a modification of the recently published scheme by Jakob-Chien and Alpert. The performance of the method is illustrated with numerical examples.

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

Norman Yarvin and Vladimir Rokhlin
Research Report YALEU/DCS/RR-1142
January 1998

The authors were supported in part by DARPA/AFOSR under Grant F49620-97-1-0011, and in part by ONR under Grant N00014-96-1-0188.
Approved for public release: distribution is unlimited.
Keywords: *Singular Value Decompositions, Fast Algorithms, Spherical Harmonics.*

1 Introduction

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 $x_1 < x_2 < \ldots < x_m$ and $y_1 < y_2 < \ldots < y_n$ 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.) A typical matrix $P = [p_{ij}]$ having such a structure is given by the formula

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

the multiplication of that matrix by a vector can be accomplished efficiently by the one-dimensional versions [3, 9] of the Fast Multipole Method (FMM) [4]. The algorithm described in this paper is also organized along the lines 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). In Section 6, we use the constructed scheme as a tool for the rapid uniform resolution filtering and interpolation of functions on the sphere. It should be observed that the latter algorithm is a fairly straightforward modification of that published by Jakob-Chien and Alpert in [7].

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, [8] 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 [9]. In this section, and in the remainder of the paper, $\mathbf{R}^{n,m}$ will denote

the space of all real $n \times m$ matrices, and the matrix norm used will be the Schur 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 the formula

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

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*

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

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 [4, 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 far field 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 [9], or non-adaptive 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

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

(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 $x_1 < x_2 < \ldots < x_m$ and $y_1 < y_2 < \ldots < y_n$ 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 all other intervals at the same depth in the tree into two sets:

- 1. The *near field* region, consisting of the two adjacent intervals at the same depth in the tree of intervals.

with $a^m_{k,l} = 0$, *for all* $k \leq 0$, *or* $l \leq 0$, *or* $m \leq 1$. *Then*

$$A^m_k(p) = \frac{(-1)^{\frac{m-1}{2}+k}}{(\frac{m-1}{2}+k)!(\frac{m-1}{2}-k)!} \sum_{l=1}^{\frac{m-1}{2}} a^m_{k,l} p^l, \tag{16}$$

for any odd $m \geq 3$, $1 \leq k \leq \frac{m-1}{2}$ *and* $A^m_k(p)$ *is defined by (11).*

Proof. Due to (11),

$$A^m_k(p) = \frac{(-1)^{\frac{m-1}{2}+k}}{(\frac{m-1}{2}+k)!(\frac{m-1}{2}-k)!} C^m_k(p), \tag{17}$$

where

$$C^m_k(p) = \frac{1}{(p-k)} \prod_{t=0}^{m-1} (p + \frac{m-1}{2} - t). \tag{18}$$

Thus it is sufficient to show that

$$C^m_k(p) = \sum_{l=1}^{\frac{m-1}{2}} a^m_{k,l} p^l. \tag{19}$$

This will be shown by induction. Indeed, if $m = 3$ then, due to (18),

$$C^3_1(p) = p^2 + p, \tag{20}$$

which is equivalent to (15-a),(15-b).
Assume now that for some m, k such that $-\frac{m-1}{2} \leq k \leq \frac{m-1}{2}$,

$$C^m_k(p) = \sum_{l=1}^{\frac{m-1}{2}} a^m_{k,l} p^l. \tag{21}$$

Combining (18) and (21), we have

$$\begin{aligned} C^{m+2}_k(p) &= (p + \frac{m+1}{2})(p - \frac{m+1}{2}) \sum_{l=1}^{\frac{m-1}{2}} a^m_{k,l} p^l \\ &= (p^2 - (\frac{m+1}{2})^2) \sum_{l=1}^{\frac{m-1}{2}} a^m_{k,l} p^l \\ &= \sum_{l=1}^{\frac{m-1}{2}} a^m_{k,l} p^{l+2} - (\frac{m+1}{2})^2 \sum_{l=1}^{\frac{m-1}{2}} a^m_{k,l} p^l, \end{aligned} \tag{22}$$

which is equivalent to (15-d).
Now, assume that for some k

$$C^{2k+1}_k(p) = \sum_{l=1}^{k} a^{2k+1}_{k,l} p^l. \tag{23}$$

Combining (23) and (18), we have

$$\begin{aligned} C_k^{2k+3}(p) &= (p-k)(p-(k+1))\sum_{l=1}^{k} a_{k,l}^{2k+1}p^l \\ &= (p^2+p-(k^2+k))\sum_{l=1}^{k} a_{k,l}^{2k+1}p^l \\ &= \sum_{l=1}^{k} a_{k,l}^{2k+1}p^{l+2} + \sum_{l=1}^{k} a_{k,l}^{2k+1}p^{l+1} - (k^2+k)\sum_{l=1}^{k} a_{k,l}^{2k+1}p^l, \end{aligned} \tag{24}$$

which is equivalent to (15-c). □

Lemma 2.5 *Suppose that $m \geq 3$ is odd. Then,*

$$D_{i,k}^m = \frac{(-1)^{\frac{m-1}{2}+k}}{(\frac{m-1}{2}+k)!(\frac{m-1}{2}-k)!} a_{k,2i-1}^m (2i-1)!, \tag{25}$$

for any k, i such that $-\frac{m-1}{2} \leq k \leq \frac{m-1}{2}$, and $1 \leq i \leq \frac{m-1}{2}$,

with the coefficients $a_{k,l}^m$ defined by the recurrence relation in Lemma 2.4.
Proof. Substituting (16) into (13), we immediately obtain

$$\begin{aligned} D_{i,k}^m &= \frac{(-1)^{\frac{m-1}{2}+k}}{(\frac{m-1}{2}+k)!(\frac{m-1}{2}-k)!} \frac{\partial^{(2i-1)}}{\partial p^{(2i-1)}} \sum_{l=1}^{m-1} a_{k,l}^m p^l \mid^{p=0} \\ &= \frac{(-1)^{\frac{m-1}{2}+k}}{(\frac{m-1}{2}+k)!(\frac{m-1}{2}-k)!} a_{k,2i-1}^m (2i-1)!. \end{aligned} \tag{26}$$

□

The following six lemmas provide identities which are used in the proof of Theorem 3.1.

Lemma 2.6 *If $k \geq 2$ is an integer and $a_{k,l}^m$ is defined in Lemma 2.4.*

$$\mid (l)! \cdot a_{k,l}^{2k+1} \mid \;<\; \mid (l+2)! \cdot a_{k,l+2}^{2k+1} \mid, \tag{27}$$

for all $l = 1, 2, ..., 2k-3$.

Proof.
If $k = 2$, and $l = 1$ then $\mid (1)! \cdot a_{2,1}^5 \mid = 2$, $\mid (3)! \cdot a_{2,3}^5 \mid = 12$, and therefore (27) is obviously true.
Now, assume that

$$\mid (l)! \cdot a_{k,l}^{2k+1} \mid \;<\; \mid (l+2)! \cdot a_{k,l+2}^{2k+1} \mid, \tag{28}$$

for some $k \geq 2$ and all $l = 1, 2, ..., 2k-3$.
Now, due to (15-a), (15-b), (15-c), and (15-d),

$$(l)! \cdot a_{k+1,l}^{2k+3} = (((k+1)-(k+1)^2)a_{k,l}^{2k+1} + a_{k,l-1}^{2k+1} + a_{k,l-2}^{2k+1}) \cdot (l!), \tag{29}$$

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, $M_{i,j}$ 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 degree 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^*.

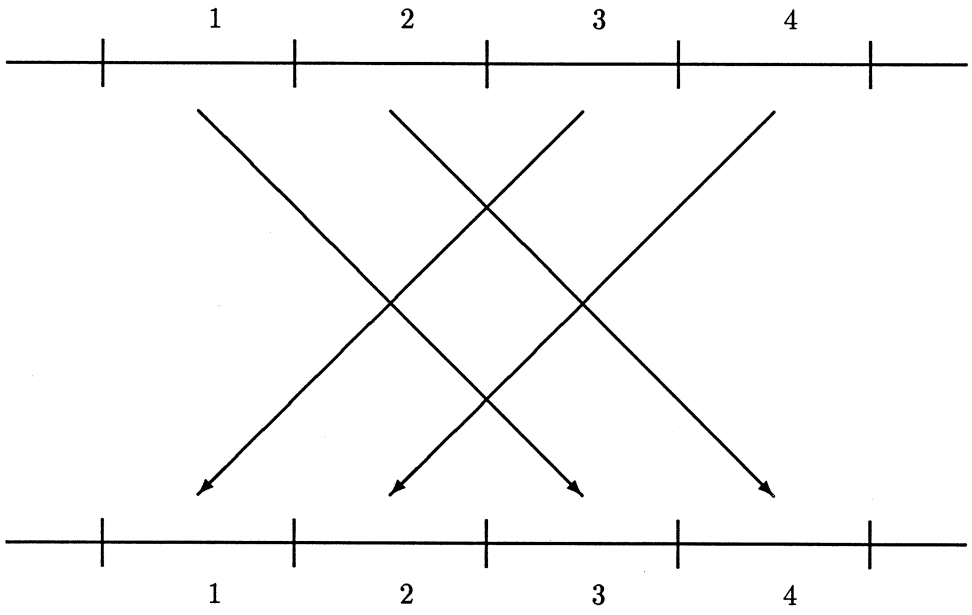
We use this freedom to diagonalize one of the (usually three) far field to local matrices for each interval. Suppose that $E_{i,j}$, for some intervals i and j, is the matrix to be diagonalized. Let its singular value decomposition be denoted by $E_{i,j} = USV^*$. Then we multiply V_j^* on the right 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, and inside each block, the interactions chosen for diagonalization are: $1 \rightarrow 3$, $2 \rightarrow 4$, $3 \rightarrow 1$, and $4 \rightarrow 2$ (as depicted in Figure 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

Figure 1: Far field to local operators which are diagonalized



split either into two, three, or five pieces, and if three, its subintervals might each 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, 9], 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 1 and 2 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 1–D FMM (that of [9]) known to the authors. It is, however, far inferior to the FMMs of [3, 9] in the time spent in the precomputation stage; initialization times for those algorithms did not exceed execution time by more than a factor of ten, whereas the initialization time for the generalized FMM exceeds the execution time by factors of thousands. 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 1: 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

Table 2: 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 [7] for uniform resolution filtering of functions on the sphere. Their algorithm as a whole performs the following task: given numbers $f(\phi_i, \theta_j)$, for $i = 1, \ldots, I$ and $j = 1, \ldots, J$, such that

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

compute 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 [7] for details).

We modify only the core of the algorithm of [7], which performs the following one-dimensional filtering operation: given numbers $f^m(\theta_1), \ldots, f^m(\theta_J)$ such that

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

Combining (14) and (58), we have

$$T^n_{\beta^m}(f) = T_n(f) + \sum_{l=1}^{\frac{m-1}{2}} \frac{h^{2l}B_{2l}}{(2l)!}(f^{(2l-1)}(b) - f^{(2l-1)}(a) - 2R^{(2l-1)}_{m-1}). \tag{59}$$

Finally, combining (59) with Lemma 2.1, we observe that for some $a < \xi < b$,

$$T^n_{\beta^m}(f) = \int_a^b f(x)dx + 2R^{(2l-1)}_{m-1} + \frac{h^m B_m}{m!} f^m(\xi), \tag{60}$$

and the theorem immediately follows from (60). □

Remark 3.2 It is easy to see that for $m \geq 3$ and odd, and any k such that $-\frac{m-1}{2} \leq k \leq \frac{m-1}{2}$, $D^m_{i,-k} = -D^m_{i,k}$, and $D^m_{i,0} = 0$ (due to (13)), and hence $\beta^m_{-k} = -\beta^m_k$ and $\beta^m_0 = 0$ (due to 52). Now, instead of (51) one could define the end-point corrected trapezoidal rule by the formula

$$T^n_{\beta^m}(f) = T_n(f) + h\sum_{k=1}^{\frac{m-1}{2}}(f(b+kh) - f(b-kh) - f(a+kh) + f(a-kh))\beta^m_k. \tag{61}$$

4 End-point Corrections for Singular Functions

In this section we construct a group of quadrature formulae for end-point singular functions, generalizing the classical end-point corrected trapezoidal rules. The actual values of end-point corrections are obtained for each singularity as a solution of a system of linear algebraic equations. All the rules developed in this section are simple extensions of the corrected trapezoidal rule $T^n_{\beta^m}$ developed in the preceding section.

A right-end corrected trapezoidal rule $T^n_{R\beta^m}$ is defined by the formula

$$T^n_{R\beta^m}(f) = h(\frac{f(x_{n-1})}{2} + \sum_{i=1}^{n-2} f(x_i)) \quad + \quad h\sum_{k=1}^{\frac{m-1}{2}}(f(b+kh) - f(b-kh))\beta^m_k, \tag{62}$$

where $f(0, b+mh] \rightarrow R^1$ is an integrable function, n, m are a pair of natural numbers with $m \geq 3$ and odd, the coefficients β^m_k are given by (52), and

$$\begin{aligned} h &= \frac{b}{n-1}, \\ x_i &= ih. \end{aligned} \tag{63}$$

We will say that the rule $T^n_{R\beta^m}$ is of right-end order $m \geq 3$ if for any $f \in c^{m+1}[0, b+mh]$ such that $f(0)[214z = f'(0) = ... = f^{(m)}(0) = 0$, there exists $c > 0$ such that

$$\mid T^n_{R\beta^m}(f) - \int_0^b f(x)dx \mid < \frac{c}{n^m}. \tag{64}$$

It easily follows from Theorem 3.2 that $T^n_{R\beta^m}$ is of right-end order m.
Similarly, a left-end corrected trapezoidal rule $T^n_{L\beta^m}$ is defined by the formula

$$T^n_{L\beta^m}(f) = h(\frac{f(x_{-(n-1)})}{2} + \sum_{i=1}^{n-2} f(x_{-i})) \quad + \quad h \sum_{k=1}^{\frac{m-1}{2}} (-f(-b+kh) + f(-b-kh))\beta^m_k, \qquad (65)$$

where $f[-b-mh, 0) \to R^1$ is an integrable function, n, m are a pair of natural numbers with $m \geq 3$ and odd, the coefficients β^m_k are given by (52), and h, x_i are defined by (63). We will say that the rule $T^n_{L\beta^m}$ is of left-end order $m \geq 3$ if for any $f \in c^{m+1}[-b-mh, 0)$ such that $f(0) = f'(0) = ... = f^{(m)}(0) = 0$, there exists $c > 0$ such that

$$| T^n_{L\beta^m}(f) - \int_{-b}^{0} f(x)dx | < \frac{c}{n^m}. \qquad (66)$$

It also easily follows from Theorem 3.2 that $T^n_{L\beta^m}$ is of left-end order m.

Suppose now that the function $f(-kh, b+mh] \to R^1$ is of the form

$$f(x) = \phi(x)s(x) + \psi(x), \qquad (67)$$

with $\phi, \psi \in c^k(-kh, b+mh]$, and $s \in c(-kh, b+mh]$ an integrable function with a singularity at 0. For a finite sequence $\alpha = (\alpha_{-k}, \alpha_{-(k-1)}, \alpha_{-1}, \alpha_1, ..., \alpha_k)$ and $T^n_{R\beta^m}$ defined in (62), we define the end-point corrected rule $T^n_{\alpha\beta^m}$ by the formula

$$T^n_{\alpha\beta^m}(f) = T^n_{R\beta^m}(f) + h \sum_{j=-k, j\neq 0}^{k} \alpha_j f(x_j), \qquad (68)$$

with $h = b/(n-1)$, $x_j = jh$.
We will use the expression $T^n_{\alpha\beta^m}$ with appropriately chosen α as quadrature formulae for functions of the form (67), and the following construction provides a tool for finding α once $\beta^m = (\beta^m_1, \beta^m_2, ..., \beta^m_{\frac{m-1}{2}})$ is given, so that the rule is of order k, i.e., there exists a $c > 0$ such that

$$| T^n_{\alpha\beta^m}(f) - \int_{0}^{b} f(x)dx | < \frac{c}{n^k}. \qquad (69)$$

For a pair of natural numbers k, m, with $k \geq 1$ and $m \geq 3$ and odd, we will consider the following system of linear algebraic equations with respect to the unknowns α^n_j, with $j = 0, \pm 1, \pm 2, ..., \pm k$:

$$\sum_{j=-k, j\neq 0}^{k} x_j^{i-1} \alpha^n_j = \frac{1}{h} \int_0^b x_j^{i-1} dx - T^n_{R\beta^m}(x^{i-1}), \qquad (70)$$

for $i = 1, 2, ..., k$, and

$$\sum_{j=-k, j\neq 0}^{k} x_j^{i-k-1} s(x_j) \alpha^n_j = \frac{1}{h} \int_0^b x_j^{i-k-1} s(x) dx - T^n_{R\beta^m}(x^{i-k-1} s(x)), \qquad (71)$$

even. In this case the separation of odd functions from even functions is accomplished by the usual formulae

$$f_{odd}(x) = (f(x) - f(-x))/2, \tag{29}$$
$$f_{even}(x) = (f(x) + f(-x))/2, \tag{30}$$

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 (27) and (29), 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 the equation

$$\begin{aligned} \hat{p}_{ij} &= \frac{\bar{P}^m_{N+1}(\tilde{\mu}_j)\bar{P}^m_N(\mu_i)w_i - \bar{P}^m_N(\tilde{\mu}_j)\bar{P}^m_{N+1}(\mu_i)w_i}{\tilde{\mu}_j - \mu_i} \\ &\pm \frac{\bar{P}^m_{N+1}(\tilde{\mu}_j)\bar{P}^m_N(\mu_i)w_i + \bar{P}^m_N(\tilde{\mu}_j)\bar{P}^m_{N+1}(\mu_i)w_i}{\tilde{\mu}_j + \mu_i}, \end{aligned} \tag{31}$$

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 (31) 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 an $O(n)$ procedure, splitting the problem into two problems of half the size does not produce any asymptotic improvement in execution time, though it does so 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 two asymptotically, since the direct method is $O(n^2)$.*

6.3 Numerical results

Table 3 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 SLATEC FFT of size J. The direct method for which timings are listed is a modestly optimized variant: the

Table 3: Filter timings for points tabulated at Legendre nodes

J	Average time per m (seconds), for:			Ratio: Eval /FFT	Average Error (L^2)	Average memory used (REAL*8 spaces)
	Direct	FMM eval	FMM init			
Requested accuracy 1E-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 1E-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 1E-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

filtering matrix it used was precomputed, and 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.*

References

[1] M. ABRAMOWITZ, I. STEGUN, *Handbook of Mathematical Functions*, Applied Mathematics Series, National Bureau of Standards, 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 Journal on Scientific Computing, Vol. 14, No. 1, pp. 159-184, January 1993

[3] A. DUTT, M. GU, AND V. ROKHLIN, *Fast Algorithms for Polynomial Interpolation, Integration, and Differentiation*, SIAM Journal on Numerical Analysis, Vol. 33, No. 5, Oct 1996

[4] L. GREENGARD AND V. ROKHLIN, *A Fast Algorithm for Particle Simulations*, Journal Of Computational Physics, Vol. 73, No.2, Dec 1987

[5] L. GREENGARD AND V. ROKHLIN, *A new version of the Fast Multipole Method for the Laplace Equation in three dimensions*, Acta Numerica, 1997, pp. 229-269

[6] GOLUB, V. H., AND VAN LOAN, C. H., *Matrix Computations*, Johns Hopkins University Press, Baltimore, 1983

[7] R. JAKOB-CHIEN AND B. ALPERT, *A Fast Spherical Filter with Uniform Resolution*, Journal of Computational Physics, Vol. 136, No. 2, September 15, 1997, p 580-584

[8] J. STOER AND R. BULIRSCH, *Introduction to Numerical Analysis, Second Edition*, Springer-Verlag, 1993

[9] N. YARVIN AND V. ROKHLIN, *An Improved Fast Multipole Algorithm for Potential Fields on One-Dimensional Structures*, Research Report 1119, Yale Computer Science Department, 1997 (submitted to the SIAM Journal on Numerical Analysis)