Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 195 (2004) 790–803
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Another preprocessing algorithm for generalized one-dimensional fast multipole method ☆

Reiji Suda *,[1], Shingo Kuriyama [2]

Department of Computational Science and Engineering, Nagoya University, Japan

Received 19 September 2002; received in revised form 21 October 2003; accepted 22 October 2003

Abstract

The fast multipole method (FMM), which is originally an algorithm for fast evaluation of particle interactions, is also effective for accelerating several numerical computations. Yarvin and Rokhlin proposed "generalized" FMM using the singular value decomposition (SVD), which gives the optimum low-rank approximation. Their algorithm reduces the computational costs of the FMM evaluation and frees the FMM from analytical approximation formulae. However, the computational complexity of the preprocessing for an $N \times N$ matrix is $O(N^3)$ because of the SVD, and it requires orthogonal matrices of the low-rank approximations. In this paper we propose another preprocessing algorithm for the generalized FMM. Our algorithm runs in time $O(N^2)$ even with the SVD and releases the low-rank approximations from orthogonal matrices. The triangularization by the QR decomposition with sparsification, which reduces the costs of the FMM more than the diagonalization, is enabled. Although the algorithm by Yarvin and Rokhlin can be accelerated to $O(N^2)$ using the QR decomposition, our preprocessing algorithm outperforms it in fast spherical filter, fast polynomial interpolation and fast Legendre transform.

AMS: 65F30; 65G99; 65Y20; 43A90/33C55; 86A10

Keywords: Fast multipole method; Low-rank approximation; Computational complexity; Fast matrix–vector multiplication; Fast spherical harmonic transform; Fast spherical filter; Fast polynomial interpolation

1. Introduction

The fast multipole method (FMM) [1] is originally an algorithm for fast approximate evaluation of particle interactions, and is also effective for accelerating several numerical computations, such as poly-

☆ This research is partly supported by Research for the Future Program of JSPS and Grants-in-Aid for Scientific Research of MEXT.
* Corresponding author. Present address: Department of Computer Science, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan. Tel./fax: +81-3-5841-4099.
E-mail address: reiji@is.s.u-tokyo.ac.jp (R. Suda).
[1] Currently with the University of Tokyo.
[2] Currently with Nippon Steel.

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2003.10.018

nomial computations [2] and integral transforms [3,4]. Recently, two algorithms using one-dimensional FMM are proposed for fast solution of partial differential equations on spherical geometries – fast spherical filter [5] and fast spherical harmonic transform [6] – which are expected to be breakthroughs to very large-scale climate simulations.

Yarvin and Rokhlin [7] proposed a "generalized" one-dimensional FMM, meaning no use of analytic formula but of the optimum algebraic approximation by the singular value decomposition (SVD). The generalized FMM not only reduces the computational costs of the FMM evaluation, but also has potential of higher applicability to accelerating numerical computations.

However, the preprocessing algorithm by Yarvin and Rokhlin requires $O(N^3)$ time for an $N \times N$ matrix, while the evaluation of the FMM requires only $O(N)$ time in the best cases. That high computational complexity of the preprocessing, which comes from the complexity of the SVD, will be prohibiting in large-scale applications where the speed of the FMM is most effective. The pivoted QR decomposition [8] must be used as the low-rank approximation to reduce the computational complexity into $O(N^2)$.

In this paper we propose another preprocessing algorithm for the generalized FMM. Our preprocessing algorithm runs in time $O(N^2)$ even with the SVD on the assumption that the FMM evaluation runs in time $O(N)$. It does not require orthogonal matrices of the low-rank approximations unlike the Yarvin–Rokhlin algorithm, and thus provides more freedom to the low-rank approximations. We can utilize that freedom in *triangularization* by the QR decomposition with sparsification, which reduces the floating point operation count of the FMM. The diagonalization [7] is also known as of similar effects, but the triangularization outperforms it. We will evaluate the performance of our algorithm in fast spherical filter, fast polynomial interpolation and fast Legendre transform.

2. Our preprocessing algorithm

This section describes our preprocessing algorithm. The reader is assumed to be familiar with the original analytical FMM [1]. The preprocessing algorithm of the generalized FMM by Yarvin and Rokhlin will also be mentioned in contrast with our algorithm.

The FMM is essentially a fast multiplication of a matrix and a vector, and this paper denotes the matrix to be computed P and assumes its size be $N \times N$. There is no difficulty to apply our algorithm to rectangular matrices.

2.1. Low-rank approximation and the SVD

The generalized FMM is based on low-rank approximations. For an $n \times m$ matrix A, a low-rank approximation is a decomposition of the form

$$A \approx XY,$$

where X is an $n \times p$ matrix, Y is a $p \times m$ matrix, and p is the *rank* of the approximation. Then a matrix–vector multiplication Ab can be approximated as

$$Ab \approx X(Yb).$$

The intermediate vector Yb corresponds to an *expansion* of the FMM, and thus the matrices Y and X *creates* and *evaluates* the expansion, respectively. The complexity of the computation $X(Yb)$ is $O(p(m+n))$. It is smaller than $O(nm)$, which is the complexity of the direct computation of Ab, if the rank p is small enough compared to n and m.

The SVD provides a low-rank approximation of the minimum rank for a prescribed precision in the 2-norm (and in the Frobenius norm) [8]

$$\|A - XY\|_2 \leqslant \epsilon.$$

The most frequently used procedure to compute the SVD is a combination of the Householder bidiagonalization and the QR iteration [9]. The former consumes most of the computational time, and its computational complexity is $O(nm^2)$ for an $n \times m$ matrix.

In the preprocessing algorithm by Yarvin and Rokhlin, submatrices of P are compressed by the SVD. The size of the largest submatrix is $n \approx m = O(N)$, thus the computational complexity of the SVD for that submatrix is $O(N^3)$. Our preprocessing algorithm applies the SVD to matrices whose numbers of columns m are bounded by a constant. This is the reason of the low computational complexity of our algorithm.

2.2. Construction of the forest of the subdomains

The first step of the preprocessing is the definition of the forest of the subdomains. Here the simplest scheme is presented. Some modifications can be introduced to enhance the performance of the FMM.

We use terminology of graph theory without notice. Since one-dimensional FMM is discussed, we use the terms "subdomain" and "interval" interchangeably, and also the corresponding set of indices of the matrix P is meant by those words.

Assume that the entries of the matrix P are defined by points of one-dimensional space as

$$P_{ij} = f(x_i, y_j).$$

All the examples in Section 3 and in the paper by Yarvin and Rokhlin are of this kind. Let $a = \min\{x_i, y_j\}$ and $b = \max\{x_i, y_j\}$ so that the interval $[a, b]$ contains all the points. That interval is divided into four intervals (subdomains) of equal sizes, which become the roots of the forest of subdomains. Each of those root intervals is divided into two intervals of equal sizes, which are the children of that interval. The resulting intervals are recursively divided until the number of points included in each interval becomes less than a prescribed threshold. That process results in a forest of subdomains with four binary trees.

The parent of an interval I is denoted by I_P, and its two children are denoted by I_{CL} and I_{CR}. For each interval I the *far-field region* F_I is defined. F_I includes the intervals of the same depth as I but excludes I and the two adjacent intervals.

2.3. The definitions of the expansions

The second step of the preprocessing is the definitions of the matrices to generate and evaluate expansions. In our preprocessing algorithm the matrices are defined in the same order as the FMM evaluation procedure: The far-field expansions from the leaves to the roots, and the local expansions from the roots to the leaves.

The expansions approximate submatrices of P. To facilitate presentation this section uses the following notations of submatrices. For a matrix A, $[A]_R$ denotes the submatrix with rows restricted to region R, where R consists of an interval or some intervals. Similarly $[A]^R$ denotes a submatrix whose columns are restricted to region R, and $[A]_R^{R'}$ denotes a submatrix that is restricted in rows and columns. Using those notations, the far-field expansion of an interval I is to approximate $[P]_{F_I}^{I}$, and the local expansion approximates $[P]_{I}^{F_I}$.

2.3.1. Far-field expansions for leaf intervals

The *far-field expansion* of an interval I approximates the submatrix $[P]_{F_I}^{I}$. The far-field expansion of each leaf interval is defined by approximating that submatrix as a low-rank approximation as

$$[P]^I_{F_I} \approx E_I C_I. \tag{1}$$

The matrix C_I is called the *far-field expansion creation matrix* and E_I is called the *far-field expansion evaluation matrix*.

Those matrices are exactly the same as the Yarvin–Rokhlin algorithm, except that Yarvin and Rokhlin restrict C_I of orthogonal rows. The evaluation matrix E_I is not used in the algorithm by Yarvin and Rokhlin, but we use it to define other expansions.

The precision of the low-rank approximation (1) should be controlled as

$$\|E_I C_I - [P]^I_{F_I}\| \leqslant \epsilon,$$

where ϵ is the precision parameter.

Remark. In our algorithm the error of every approximation is bounded by ϵ. Thus the total error of the FMM can be bounded by $\alpha\epsilon$, where α is the number of approximations. However, the actual total error is usually much smaller than the bound $\alpha\epsilon$. A practical scheme to control the total error is reconstructing the data structure of the FMM with modifying the parameter ϵ so as to bring the total error closer to the prescribed value.

Yarvin and Rokhlin scale the approximation error bound ϵ by the matrix norm $\|P\|$ and the size of the submatrix nm, but they did not show any reasoning or optimality of that scaling. In place of scaling by $\|P\|$, we introduce an application-driven scaling in Section 2.4. We do not scale the error bound by the size of the submatrix (like the analytical FMM).

2.3.2. Far-field expansions for non-leaf intervals

Next, the far-field expansion is defined on each non-leaf interval I. While the algorithm by Yarvin and Rokhlin approximates the submatrix $[P]^I_{F_I}$ directly, our algorithm uses the expansions of the children I_{CL} and I_{CR}. The key equation here is

$$[P]^I_{F_I} = \left([[P]^{I_{CL}}_{F_{I_{CL}}}]_{F_I} \quad [[P]^{I_{CR}}_{F_{I_{CR}}}]_{F_I} \right).$$

To see that it is enough to note $F_I \subset F_{I_{\mathrm{CL}}}$, $F_I \subset F_{I_{\mathrm{CR}}}$ and $I = I_{\mathrm{CL}} + I_{\mathrm{CR}}$.

First assume that the children are leaves. From (1) the children have approximations:

$$[P]^{I_{\mathrm{CL}}}_{F_{I_{\mathrm{CL}}}} \approx E_{I_{\mathrm{CL}}} C_{I_{\mathrm{CL}}},$$

$$[P]^{I_{\mathrm{CR}}}_{F_{I_{\mathrm{CR}}}} \approx E_{I_{\mathrm{CR}}} C_{I_{\mathrm{CR}}}.$$

Therefore, we have

$$[P]^I_{F_I} \approx \left([E_{I_{\mathrm{CL}}}]_{F_I} \quad [E_{I_{\mathrm{CR}}}]_{F_I} \right) \begin{pmatrix} C_{I_{\mathrm{CL}}} & 0 \\ 0 & C_{I_{\mathrm{CR}}} \end{pmatrix}. \tag{2}$$

In our preprocessing algorithm the first factor of the right-hand side

$$Q_I = \left([E_{I_{\mathrm{CL}}}]_{F_I} \quad [E_{I_{\mathrm{CR}}}]_{F_I} \right) \tag{3}$$

is approximated as a low-rank approximation

$$Q_I \approx E_I \bar{M}_I. \tag{4}$$

The matrix $\bar{M}_I$ generates the far-field expansion of I from those of the children.

The *far-field translation matrices* $M_{I_{\mathrm{CL}}}$ and $M_{I_{\mathrm{CR}}}$ are obtained by dividing $\bar{M}_I$ according to the assembly of Q_I (3) as

$$\bar{M}_I = \begin{pmatrix} M_{I_{\mathrm{CL}}} & M_{I_{\mathrm{CR}}} \end{pmatrix}.$$

The *far-field expansion creation matrix* C_I, which represents the computation of the far-field expansion from the input vector, should be defined as

$$C_I = \bar{M}_I \bar{C}_I, \tag{5}$$

where $\bar{C}_I$ is the second factor of the right-hand side of (2)

$$\bar{C}_I = \begin{pmatrix} C_{I_{\mathrm{CL}}} & 0 \\ 0 & C_{I_{\mathrm{CR}}} \end{pmatrix}.$$

From (2) and (4) the submatrix $[P]^I_{F_I}$ is approximated as

$$[P]^I_{F_I} \approx Q_I \bar{C}_I \approx E_I \bar{M}_I \bar{C}_I,$$

where the second approximation is the one defined in (4). Thus the error of the low-rank approximation of (4) should be controlled as

$$\|E_I \bar{M}_I \bar{C}_I - Q_I \bar{C}_I\| \leqslant \epsilon. \tag{6}$$

A scheme to control the error of the low-rank approximation in the above form will be discussed in Section 2.4.

In the algorithm by Yarvin and Rokhlin, the far-field expansion creation matrix C_I is first generated using the SVD as (1), then the translation matrices are defined as

$$\bar{M}_I = C_I \bar{C}_I^{\mathrm{T}}, \tag{7}$$

which gives the best approximation of $M_I \bar{C}_I \approx C_I$ in Frobenius norm on the condition that $\bar{C}_I$ has orthogonal rows. They showed that it gives reasonably good approximation for $[P]^I_{F_I} \approx E_I \bar{M}_I \bar{C}_I$.

In the Yarvin–Rokhlin algorithm $\bar{C}_I$ must be of orthogonal rows, but in our algorithm the low-rank approximation (4) can be anything that satisfies (6). Thus our algorithm gives more freedom to the low-rank approximations than that of the Yarvin–Rokhlin algorithm.

In both algorithms the creation matrix C_I and the evaluation matrix E_I are defined so that the submatrix $[P]^I_{F_I}$ is approximated as

$$[P]^I_{F_I} \approx E_I C_I \tag{8}$$

that coincides with (1). Thus the far-field expansions of the intervals with non-leaf children can be defined in the same way as the above.

Repeating the above procedure from the intervals next to the leaves to the root intervals, the far-field expansion creation, translation and evaluation matrices are defined for all the intervals.

The algorithm by Yarvin and Rokhlin computes a low-rank approximation for $[P]^I_{F_I}$, which becomes larger for a larger interval. For a root interval I, the size of $[P]^I_{F_I}$ is about $N/4 \times N/2$ or $N/4 \times N/4$, and the SVD requires time of $\mathrm{O}(N^3)$. Our algorithm computes low-rank approximations for Q_I, whose columns are much fewer than that of $[P]^I_{F_I}$.

2.3.3. Local expansions for root intervals

The *local expansion* of an interval I approximates the submatrix $[P]^{F_I}_I$. First consider the case of a root interval. The local expansion of a root interval I can be computed from the far-field expansions of the

intervals in its far-field region F_I. Here we assume that F_I consists of two intervals I_0 and I_1: The procedure for general case is easily understood from this example.

Our algorithm is based on the equation

$$[P]_I^{F_I} = \left([[P]_{F_{I_0}}^{I_0}]_I \quad [[P]_{F_{I_1}}^{I_1}]_I \right)$$

which comes from $F_I = I_0 + I_1$, $I \subset F_{I_0}$ and $I \subset F_{I_1}$. From (8) we have

$$[P]_I^{F_I} \approx \left([E_{I_0}]_I \quad [E_{I_1}]_I \right) \begin{pmatrix} C_{I_0} & 0 \\ 0 & C_{I_1} \end{pmatrix}.$$

Our preprocessing algorithm approximates the first factor of the right-hand side

$$R_I = \left([E_{I_0}]_I \quad [E_{I_1}]_I \right) \tag{9}$$

as a low-rank approximation

$$R_I \approx X_I \bar{L}_I. \tag{10}$$

The matrix $\bar{L}_I$ generates the local expansion of I from the far-field expansions of I_0 and I_1. It is divided according to the assembly of R_I (9) as

$$\bar{L}_I = \left(T_{I,I_0} \quad T_{I,I_1} \right)$$

to obtain T_{I,I_0} and T_{I,I_1}, which are called the *far-field to local interaction matrices*. The matrix X_I is called the *local expansion evaluation matrix*.

The *local expansion creation matrix* H_I is defined as

$$H_I = \bar{L}_I \bar{H}_I, \tag{11}$$

where $\bar{H}_I$ is defined as

$$\bar{H}_I = \begin{pmatrix} C_{I_0} & 0 \\ 0 & C_{I_1} \end{pmatrix},$$

so that the approximation is expressed as

$$[P]_I^{F_I} \approx X_I H_I. \tag{12}$$

Since the approximation of the submatrix $[P]_I^{F_I}$ is

$$[P]_I^{F_I} \approx R_I \bar{H}_I \approx X_I \bar{L}_I \bar{H}_I, \tag{13}$$

the error of the low-rank approximation (10) should be controlled as

$$\| X_I \bar{L}_I \bar{H}_I - R_I \bar{H}_I \| \leqslant \epsilon. \tag{14}$$

In the Yarvin–Rokhlin algorithm, the local expansion evaluation matrix is first computed as (12) with restriction that X_I has orthogonal columns using the SVD. Then the far-field to local interaction matrices are computed as

$$\bar{L}_I = X_I^{\mathrm{T}} [P]_I^{F_I} \bar{H}_I^{\mathrm{T}}, \tag{15}$$

which gives the best approximation for $[P]_I^{F_I} \approx X_I \bar{L}_I \bar{H}_I$ on the assumption that X_I has orthogonal columns and H_I has orthogonal rows. That directly corresponds with (13).

2.3.4. Local expansions for non-root intervals

The local expansion of a non-root interval I is computed from the local expansion of the parent I_P and the far-field expansions of the intervals in $F_I - F_{I_\mathrm{P}}$ (which is called the *interaction list*). We assume that $F_I - F_{I_\mathrm{P}}$ contains three intervals I_0, I_1 and I_2 as an example.

Based on the equation

$$[P]_I^{F_I} = \left(\left[[P]_{I_\mathrm{P}}^{F_{I_\mathrm{P}}}\right]_I \quad \left[[P]_{F_{I_0}}^{I_0}\right]_I \quad \left[[P]_{F_{I_1}}^{I_1}\right]_I \quad \left[[P]_{F_{I_2}}^{I_2}\right]_I \right)$$

and using (8) and (12) we have

$$[P]_I^{F_I} \approx R_I \bar{H}_I, \quad R_I = \left([X_{I_\mathrm{P}}]_I \quad [E_{I_0}]_I \quad [E_{I_1}]_I \quad [E_{I_2}]_I \right), \ \bar{H}_I = \begin{pmatrix} H_{I_\mathrm{P}} & 0 & 0 & 0 \\ 0 & C_{I_0} & 0 & 0 \\ 0 & 0 & C_{I_1} & 0 \\ 0 & 0 & 0 & C_{I_2} \end{pmatrix} \tag{16}$$

(permutation of columns may be needed, but we ignore that for simplicity). Eqs. (10)–(12) and (14) are used in the same ways. The matrix $\bar{L}_I$ defined at (10) is divided according to the assembly of R_I (16) as

$$\bar{L}_I = \left(L_I \quad T_{I,I_0} \quad T_{I,I_1} \quad T_{I,I_2} \right),$$

where L_I is called the *local expansion translation matrix*.

This procedure is repeated from the intervals next to the roots to the leaf intervals. The preprocessing is finished when the local expansion interaction, translation and evaluation matrices are computed for all the intervals.

In the Yarvin–Rokhlin algorithm, first the local expansion evaluation matrix X_I is computed by the SVD as (12), and then the local expansion translation matrix is defined as

$$L_I = X_I^{\mathrm{T}} [X_{I_\mathrm{P}}]_I, \tag{17}$$

which gives the best approximation for $X_I L_I \approx [X_{I_\mathrm{P}}]_I$, assuming that X_I has orthogonal columns. That gives reasonably good approximation for $X_I L_I H_{I_\mathrm{P}} \approx [X_{I_\mathrm{P}}]_I H_{I_\mathrm{P}}$, which coincides with our approximation.

2.4. The error control and scaling

The low-rank approximation in our algorithm should satisfy (6) and (14). However, it is difficult to control the error of a low-rank approximation in the form of matrix multiplication as those. Although we have

$$\|E_I \bar{M}_I \bar{C}_I - Q_I \bar{C}_I\| \leqslant \|E_I \bar{M}_I - Q_I\| \|\bar{C}_I\|$$

for consistent matrix norms, that inequality is usually too loose.

We propose to tighten the above inequality by *scaling*

$$\|E_I \bar{M}_I \bar{C}_I - Q_I \bar{C}_I\| \leqslant \|(E_I \bar{M}_I - Q_I) S\| \|S^{-1} \bar{C}_I\|, \tag{18}$$

where S is a diagonal matrix whose elements are the norms of the rows of $\bar{C}_I$. Making $S^{-1}\bar{C}_I$, that is, making the norms of the rows of $\bar{C}_I$ unity, is the simplest scaling strategy to improve the condition number of $\bar{C}_I$ [10]. We found that this simple scaling makes the inequality (18) tight enough for practical use. Now we can control the precision of the low-rank approximation as

$$\|E_I \bar{M}_I S - Q_I S\| \leqslant \epsilon / \|S^{-1} \bar{C}_I\|, \tag{19}$$

where the multiplication by S causes no difficulty since it is easy to invert.

To reduce the computational cost the value $\|S^{-1}\bar{C}_I\|$ may be replaced by its lower or upper bound. They are 1 and $\sqrt{m}$ in the 2-norm, where m is the number of rows of $\bar{C}_I$.

The above scheme of error control is applicable to the FMM itself as follows. Assume that the matrix P is used in a linear computation Z as

$$Z = APB,$$

where A and B represents the computations after and before the FMM. Let us represent the computation of P via FMM as $\tilde{P}$. Then the error of the fast computation $\tilde{Z} = A\tilde{P}B$ can be bounded as

$$\|\tilde{Z} - Z\| \leqslant \epsilon$$

by controlling the error of the FMM as

$$\|S_A\tilde{P}S_B - S_APS_B\| \leqslant \epsilon/(\|AS_A^{-1}\|\|S_B^{-1}B\|),$$

where S_A and S_B are diagonal matrices which makes the norms of the columns of AS_A^{-1} and those of the rows of $S_B^{-1}B$ unity. The above scheme determines the precision of the FMM from the requirements of the application. If some entries of S_A and/or S_B are small, then lower relative precision is allowed in the corresponding part of the FMM. The numerical examples in Section 3 use this scheme to control the errors of the FMM.

2.5. The computational complexity

Next consider the computational complexity of our preprocessing algorithm. We assume that the FMM runs most efficiently, as is the case of an $N \times N$ Cauchy matrix

$$P_{ij} = \frac{1}{y_i - x_j},$$

where the ranks of the low-rank approximations are bounded by a constant p, the number of the intervals is $O(N)$, and the number of the points contained in a leaf interval is $O(p) = O(1)$. Then our preprocessing algorithm runs in time $O(N^2)$ even using the SVD.

Construction of the forest of the subdomains. The subdomains, the forest of them and the far-field regions are defined in the same way as the original FMM, and they require $O(N \log N)$ time.

Low-rank approximations. Next consider the computational complexity of the computations of the low-rank approximations (1), (4) and (10).

Let $n \times m$ be the size of the matrix to be approximated. For (1) the matrix to be approximated is $[P]^I_{F_I}$. The size of the matrix is given as

$$n = |F_I| = O(N),$$

$$m = |I| = O(1),$$

where the latter comes from the assumption that the number of the points in a leaf interval is $O(1)$. For Q_I of (4), $n = |F_I| = O(N)$ is clear and $m = O(p) = O(1)$ comes from the assumption that the ranks of the low-rank approximations $E_{I_{CL}}$ and $E_{I_{CR}}$ are bounded by a constant p. Similarly for R_I of (10) we have $n = O(N)$ and $m = O(1)$.

Thus we have $n = O(N)$ and $m = O(1)$ for all matrices to be approximated. Because the SVD requires time $O(nm^2)$, each low-rank approximation is computed in time $O(N)$.

The low-rank approximations are computed twice (for the far-field expansion and the local expansion) for each subdomain. Since we assume the number of the subdomains be $O(N)$, the total computational complexity of computing the low-rank approximations can be bounded by $O(N^2)$.

Computations of the creation matrices. Next consider the costs of the computations of the creation matrices (5) and (11).

First consider (5), that is a multiplication of $\bar{M}_I$ and $\bar{C}_I$. Let $n \times m$ be the size of $\bar{M}_I$ and $m \times l$ be the size of $\bar{C}_I$. Because we have $n \leqslant p = O(1)$, $m \leqslant 2p = O(1)$ and $l \leqslant N$, that multiplication can be computed in time $O(N)$.

Similarly H_I can be computed in time $O(N)$. Since there are $O(N)$ subdomains, the total time to compute the creation matrices is bounded by $O(N^2)$.

Computations of the scaling matrices. We need the scaling matrix S in the error control scheme of the low-rank approximation (18).

Letting $m \times l$ be the size of $\bar{C}_I$, we have $m = O(1)$ and $l = O(N)$ as is discussed in the previous paragraph. Thus the direct computation of the norms of the rows of $\bar{C}_I$ requires $O(N)$ time.

We can prove that the scaling for a local expansion is computed in time $O(N)$ as well. Since there are $O(N)$ expansions, the total computational complexity for the scaling is $O(N^2)$.

The value $\|S^{-1}\bar{C}_I\|$ is used in (19). We have proposed to neglect it in Section 2.4, because it lies in the range of $[1, \sqrt{m}]$ and we have $m = O(1)$. It could be computed using the power method, but that seems to cost too much.

The total computational complexity. Summing up the above, the computational complexity of our preprocessing algorithm is bounded by $O(N^2)$ for a Cauchy matrix. If the FMM evaluation is not efficient as $O(N)$, then the computational costs of our algorithm increase accordingly, but we did not perform any analysis for such cases.

2.6. Low-rank approximations by the QR decomposition

Low-rank approximations can be obtained using the pivoted QR decomposition (QRD) [8]. The pivoted QRD does not always give a low-rank approximation of the minimum rank, but its computational complexity is much lower than the SVD. If it terminates at rank p for an $n \times m$ matrix, then the computational complexity is $O(nmp)$, which is much smaller than that of the SVD $O(nm^2)$ assuming that the rank p is smaller than m (and n).

2.6.1. Triangularization and sparsification

We noted that the matrix R of a low-rank approximation by the QR decomposition $A \approx QR$ is triangular, and thus the computational costs of the FMM evaluations can be reduced by skipping computations of the zeros. Let us call that reduction of computational costs "triangularization".

The diagonalization [7] is an improvement of similar effects. It diagonalize one of the far-field to local interaction matrix of each interval by multiplying appropriate matrices to the related translation/interaction matrices.

The effects of the diagonalization and the triangularization can be compared as follows. Choose an interaction matrix, and let the rank of its far-field expansion be p_F and that of the local expansion be p_L. The diagonalization removes the off-diagonal entries of the interaction matrix of size $p_F \times p_L$, thus the number of the generated zeros is $p_F p_L - \min\{p_F, p_L\}$. The triangularization removes only lower diagonal entries, but it is applicable to the both of the far-field and local expansions independently, and thus generates $p_F(p_F - 1)/2 + p_L(p_L - 1)/2$ zeros. Therefore, the two schemes are of equal performance for $p_F = p_L$, but otherwise the triangularization works better.

In many cases several elements of the last row of R in a QRD have very small magnitudes. Dropping such elements the computational costs can be further reduced. We call this scheme the *QRD with*

sparsification (QRS). The QRS is not useful in the algorithm by Yarvin and Rokhlin where the matrices are multiplied as (7), (15) and (17) and the zero elements disappear. Our preprocessing algorithm uses the matrix as it is, and thus the sparsification can reduce the computational costs.

2.6.2. Combinations of the SVD and the QRD

The pivoted QRD does not always give the optimum rank. This disadvantage can be remedied using the SVD. After computing the low-rank approximation by the SVD

$$A \approx XY,$$

the QRS is computed for Y,

$$Y \approx QR.$$

Letting $Z = XQ$, we have a low-rank approximation $A \approx ZR$, whose rank is that of the SVD. The precision of the QRS should be controlled as

$$\|X(Y - QR)\| \leqslant \epsilon - \|A - XY\|.$$

We call this scheme *SVD-QRS*, which is *SVD with triangularization and sparsification*. The SVD-QRS requires more computational time, but the efficiency of the resulting low-rank approximation is higher than the QRS, as is shown in the next section.

The combination in the reverse order, the *QR-SVD*, is also possible and results in *the SVD accelerated by the QRD*. In this case, the QR decomposition $A \approx QR$ is computed first as

$$\|A - QR\| \leqslant \epsilon_Q, \tag{20}$$

then the SVD $R \approx XY$ is computed as

$$\|Q(R - XY)\| \leqslant \epsilon - \|A - QR\| \geqslant \epsilon - \epsilon_Q. \tag{21}$$

If ϵ_Q is small enough, e.g. 0.01ϵ, then the influence of the approximation errors of the QRD is very small, and almost the same rank is attained as the direct SVD for A. The computational complexity is almost the same as the QRD. Assume that the rank of the QRD be p_Q, then the QRD runs in time $O(nmp_Q)$ and the SVD runs in time $O(mp_Q^2)$. Thus the costs of the SVD is smaller than those of the QRD if p_Q is much less than n. Thus the QR-SVD gives approximations of almost minimum rank with computational costs similar to the QRD.

Using the QR-SVD, the preprocessing algorithm by Yarvin and Rokhlin runs in time $O(N^2)$. Thus the Yarvin–Rokhlin algorithm and ours are actually of the same order of computational complexity. In the next section, we will compare these algorithms in some numerical experiments.

3. Numerical results

This section reports the results of some numerical experiments. We implement the preprocessing algorithm by Yarvin and Rokhlin and ours in C language. The programs are compiled by `gcc` with option `-O4` and run on a machine with COMPAQ alpha 21264A 750 MHz CPU. We have three applications of FMM to compare preprocessing algorithms.

The first application is the fast spherical filter [5,7] for $m = 0$. It consists of a pair ($l = 0, 1$) of FMM applications

$$P_{ij}^{(l)} = \frac{\gamma_{ij}^{(l)}}{x_i^2 - x_j^2},$$

which are Cauchy matrices scaled by

$$\gamma_{ij}^{(0)} = cw_i\bar{P}_n(x_i)\bar{P}_{n+2}(x_j),$$
$$\gamma_{ij}^{(1)} = -cw_i\bar{P}_{n+2}(x_i)\bar{P}_n(x_j),$$

where $\bar{P}_n(x)$ is the normalized Legendre polynomial, w_i is the Gaussian weight and c is a constant. The diagonal entries $P_{ii}^{(l)}$ are defined by l'Hospital's rule. The points x_i are the Gaussian points, and thus the distribution of the evaluation points $\{x_i^2\}$ is not uniform but regular.

The second application is the Lagrange polynomial interpolation, on which the fast spherical harmonic transform is based [6]. The matrix P is as

$$P_{ij} = \frac{\alpha_i\beta_j}{y_i - x_j},$$

which is a Cauchy matrix scaled by

$$\alpha_i = \prod_{j=1}^{N}(y_i - x_j),$$

$$\beta_j = \prod_{1\leqslant k\leqslant N, k\neq j}(x_j - x_k)^{-1}.$$

The evaluation points $\{x_j, y_i\}$ are equispaced, and the sampling points $\{x_j\}$ are chosen so that the numerical stability is optimized [6]. Many sampling points $\{x_j\}$ are close to the either ends, and the interpolation points $\{y_i\}$ are located around the center. Thus the distributions of the points are quite non-uniform.

Table 1 tabulates the results for the fast spherical filter, which are the average of the two FMM applications. The absolute errors of the filter are shown in the column E_A. We made several experiments with varying the parameter ϵ around the prescribed value E_A and the error $\|\tilde{P} - P\|_2$ is computed by the power method.

Table 2 tabulates the results for the fast polynomial interpolation. Here E_R represents the relative error $\|\tilde{P} - P\|_2/\|P\|_2$ computed by the power method. The parameter ϵ is set around $E_R/\|P\|_2$.

Table 1
The experimental results for the fast spherical filter

E_A	N	Yarvin–Rokhlin			Ours				
		T_{YS}	T_{YQ}	S_{YS}	T_{SS}	T_{SQ}	S_{SS}	S_{SQ}	S_{S0}
10^{-6}	50	0.002	0.002	0.69	0.002	0.001	0.74	0.74	0.69
	100	0.010	0.008	0.93	0.009	0.004	1.05	1.04	0.98
	200	0.044	0.026	1.47	0.038	0.016	1.65	1.64	1.54
	400	0.235	0.096	2.67	0.122	0.055	2.89	2.87	2.80
	800	1.693	0.415	5.07	0.429	0.206	5.39	5.36	5.32
10^{-10}	50	0.002	0.002	0.60	0.002	0.001	0.64	0.63	0.60
	100	0.007	0.005	0.76	0.005	0.002	0.83	0.83	0.78
	200	0.035	0.023	1.05	0.031	0.012	1.19	1.19	1.10
	400	0.210	0.100	1.73	0.129	0.054	1.89	1.88	1.82
	800	1.616	0.483	3.12	0.508	0.253	3.32	3.31	3.28

Table 2
The experimental results for the fast polynomial interpolation

E_R	N	Yarvin–Rokhlin			Ours				
		T_{YS}	T_{YQ}	S_{YS}	T_{SS}	T_{SQ}	S_{SS}	S_{SQ}	S_{S0}
10^{-6}	50	0.010	0.007	1.42	0.009	0.003	1.56	1.56	1.49
	100	0.047	0.026	2.38	0.035	0.015	2.59	2.58	2.52
	200	0.215	0.087	4.35	0.115	0.049	4.63	4.62	4.54
	400	1.537	0.367	8.16	0.397	0.192	8.55	8.50	8.44
	800	17.16	2.206	15.63	1.490	0.938	16.30	16.23	16.21
10^{-10}	50	0.007	0.006	1.09	0.005	0.002	1.21	1.21	1.15
	100	0.034	0.023	1.67	0.029	0.012	1.80	1.80	1.75
	200	0.201	0.096	2.76	0.122	0.052	3.00	2.99	2.93
	400	1.547	0.448	5.11	0.453	0.226	5.39	5.38	5.35
	800	17.32	2.750	9.62	2.460	1.245	10.11	10.09	10.09

In both tables N represents the problem size. The number of the root intervals and the depth of the forest are optimized for each instance. The diagonalization is used in the Yarvin–Rokhlin algorithm.

The average times (in seconds) for the preprocessing are shown in the columns T_*, where the used algorithm is designated in the subscription:

- YS for the original Yarvin–Rokhlin algorithm,
- YQ for the Yarvin–Rokhlin algorithm accelerated by QR-SVD,
- SS for ours with SVD-QRS,
- SQ for ours with QRS.

We can see that the preprocessing times are $O(N^3)$ for the Yarvin–Rokhlin algorithm with SVD and $O(N^2)$ for the others. The times for the Yarvin–Rokhlin algorithm accelerated by the QR-SVD and ours with the SVD-QRS are similar, and those of our algorithm with the QRS is about half of them. The preprocessing time for the fast spherical filter is shorter than that for the polynomial interpolation, perhaps because of the difference of the distribution of the points.

The columns S_* provide the speed-up rates of the fast algorithms against the direct computation in the number of floating point operations. Here the costs of the preprocessing are not included, and this index is intended to evaluate the efficiencies of the resulting FMM, especially of the effects the diagonalization and the triangularization. The precise speed-up rate for the prescribed precision E_A or E_R is estimated by interpolating the experimental results. The speed-up rates for YQ are not shown because they are almost the same as those of YS, which means that the QR-SVD gives almost the same ranks as the SVD. The QRS gives better speed-up rates than the diagonalization. The differences between the speed-up rates for the SVD-QRS and the QRS are small, perhaps because the QRD gives good low-rank approximations for Cauchy matrices and the sparsification reduces the difference between the QRD and the SVD.

Our algorithm includes another improvement that some local expansions are computed directly (not from far-field expansions) and some far-field expansions are evaluated directly (without transformed into local expansions). The speed-up rates without this improvement (using QRS) are shown in the column S_{S0}. This improvement is effective, but the effects get smaller in large problems.

We do not compare the speed-up rates of the two problems or with those of the other papers [5,7] because the definitions of the speed-up rates and the errors are different.

The third application is the fast Legendre transform, that is, the fast spherical harmonic transform for $m = 0$ [6]. Table 3 gives the results, where N is the total wave number, E_* is the observed relative error, and S_* is the speed-up rate of the Legendre transform. The value of τ_* gives the time for the whole preprocessing of the fast Legendre transform, which includes the computations of the Legendre functions, the approximation error control, the optimization of the numerical stability, the generation of the matrices for the

Table 3
The experimental results for the fast Legendre transform

N	Yarvin–Rokhlin				Ours			
	E_{YS}	τ_{YS}	T_{YS}	S_{YS}	E_{SS}	τ_{SS}	T_{SS}	S_{SS}
1023	1.06e − 10	260	31.2	1.564	1.04e − 10	370	15.3	1.616
1365	1.13e − 10	771	84.8	1.664	1.45e − 10	935	29.7	1.790
2047	1.34e − 10	2170	347	2.049	1.37e − 10	2297	69.1	2.172
2730	1.60e − 10	6158	945	2.353	1.86e − 10	5760	157	2.538
4095	3.56e − 10	22,004	5572	3.029	3.65e − 10	16,051	452	3.233

FMM, the preprocessing of those FMM, and so forth. The fast Legendre transform algorithm contains many FMM instances, and T_* gives the sum of the preprocessing times of the FMM instances in the fast Legendre transform. The evaluation points are the Gaussian points, and the number of points is approximately $3N/2$.

The total preprocessing time of our algorithm τ_{SS} is larger than that with the Yarvin–Rokhlin algorithm for $N \leqslant 1365$. The authors guess that the longer τ comes from the *higher* efficiency of our FMM implementation, because the preprocessing algorithm of the fast Legendre transform uses the branch-and-bound algorithm to minimize the computational costs of the transform, and it searches for deeper branches when the FMM gives higher efficiency. However, the situation is reversed for larger problems because of the high computational complexity of the Yarvin–Rokhlin algorithm. Our algorithm gives better speed-up for every size with almost the same precision. The superiority of our algorithm is clear.

4. Summary

This paper has proposed another preprocessing algorithm for the generalized FMM. The preprocessing algorithm by Yarvin and Rokhlin requires time $O(N^3)$ when it is coupled with the SVD, but ours runs in time $O(N^2)$ even with the SVD. We have proposed the triangularization by the QR decomposition with sparsification, which enhances the efficiency of the FMM evaluation. Our algorithm enables this by freeing the low-rank approximation from the orthogonality.

The triangularization by the QRD can be introduced to the Yarvin–Rokhlin algorithm in place of the diagonalization, and the sparsification is also possible. However, the error controls (6) and (14) are inconsistent against the error controls of the expansions of the Yarvin–Rokhlin algorithm.

This paper and the one by Yarvin and Rokhlin [7] apply the ''generalized'' FMM to Cauchy matrices to which the analytical FMM is applicable. We plan to apply the generalized FMM to the fast spherical harmonic transform [6], which splits the associated Legendre functions to enable the fast polynomial interpolation. Because the split doubles the computational costs, the performance will be improved if the generalized FMM accelerates the non-polynomial interpolation.

References

[1] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 73 (1987) 325.
[2] A. Dutt, M. Gu, V. Rokhlin, Fast algorithms for polynomial interpolation, integration, and differentiation, SIAM J. Numer. Anal. 33 (1996) 1689.
[3] J.P. Boyd, Multipole expansions and pseudospectral cardinal functions: a new generation of the fast Fourier transform, J. Comput. Phys. 103 (1992) 184.
[4] B. Alpert, V. Rokhlin, A fast algorithm for the evaluation of Legendre expansions, J. Sci. Stat. Comput. 12 (1991) 158.
[5] R. Jakob-Chien, B.K. Alpert, A fast spherical filter with uniform resolution, J. Comput. Phys. 136 (1997) 580.

[6] R. Suda, M. Takami, A fast spherical harmonics transform algorithm, Math. Comp. 71 (2002) 703.
[7] N. Yarvin, V. Rokhlin, A generalized one-dimensional fast multipole method with application to filtering of spherical harmonics, J. Comput. Phys. 147 (1998) 594.
[8] G.W. Stewart, Matrix Algorithms, Basic Decompositions, vol. 1, SIAM, Philadelphia, PA, 1998.
[9] G.H. Golub, C.F. van Loan, Matrix Computations, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[10] N.J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, Philadelphia, PA, 1996.