A SVD accelerated kernel-independent fast multipole method and its application to BEM

Yanchuang Cao, Lihua Wen & Junjie Rong
College of Astronautics, Northwestern Polytechnical University, P. R. China

Abstract

The kernel-independent fast multipole method (KIFMM) proposed by L. Ying *et al.* is of almost linear complexity. In the original KIFMM the time-consuming M2L translations are accelerated by FFT. However, when more equivalent points are used to achieve higher accuracy, the efficiency of the FFT approach tends to be lower because more auxiliary volume grid points have to be added. In this paper, all the translations of the KIFMM are accelerated by using the singular value decomposition (SVD) based on the low-rank property of the translating matrices. The acceleration of M2L is realized by first transforming the associated translating matrices into more compact form, and then using low-rank approximations. By using the transform matrices for M2L, the orders of the translating matrices in upward and downward passes are also reduced. The improved KIFMM is then applied to accelerate BEM. Numerical results show that, compared with the original KIFMM, the present method can reduce about 40% of the iterating time and 25% of the memory requirement.
Keywords: boundary element method, kernel-independent fast multipole method, singular value decomposition, matrix compression

1 Introduction

The boundary element method (BEM) has become a promising numerical method in computational science and engineering. Despite many unique advantages, like the dimension reduction, high accuracy and suitability for treating infinite domain problems, a major disadvantage of the BEM is its dense system matrix which solution cost is prohibitive in large-scale problems. During the past three decades, several acceleration methods have been proposed to circumvent this disadvantage.

Representative examples are the fast multipole method (FMM) [1], wavelet compression method [2], $\mathcal{H}$-matrix [3], adaptive cross approximation (ACA) [4], pre-corrected FFT [5], etc. Among them the FMM is no doubt the most outstanding one.

The conventional FMM is originally proposed to accelerate the N-body simulations, which requires the analytical expansions of the kernel functions. This poses a severe limitation on its applications to many problems where the analytical expansions are hard to be obtained. Besides, this makes it difficult to develop a universal FMM code for real-world applications. To overcome this drawback, the kernel-independent FMM (KIFMM) has been proposed in the past decade [6–8]. A salient feature of the KIFMM is that the expansion of the kernel function is no longer required. Instead, only the kernel value evaluations are needed. Therefore, the structure of the FMM acceleration algorithm is in common for many typical problems.

In this paper, the KIFMM proposed by Ying *et al.* [6] is concerned. This method uses equivalent densities in lieu of the analytical expansions. It provides a unified framework for fast summations with the Laplace, Stokes, Navier and similar kernel functions. Due to its ease-of-use and high efficiency, it has attracted the attention of many researchers [9–11].

The moment-to-local (M2L) translation is the most time-consuming part of the FMM [7, 8, 12–15]. In the KIFMM [6] the M2L translation is accelerated by the fast Fourier transform (FFT), leading to $\mathcal{O}(p^3 \log p)$ computational complexity, where p is the number of equivalent points along the cube side. However, one should note that the efficiency of the FFT approach tends to become lower when p increases. This is because the equivalent points lie only on the boundary of each box, while to use the FFT Cartesian grid points interior the box must be considered as well. In this paper, the M2L in KIFMM is compressed and accelerated using the singular value decomposition (SVD); see Section 3. This method is built on the fact that the M2L matrices are typically of very low numerical ranks. Our numerical experiments, including those in Section 5, show that the proposed method is more efficient than the FFT approach. Another advantage of the SVD accelerating approach is that it is more flexible than the FFT approach, because the later requires the equivalent and check points to be equally spaced while this is not needed in the SVD approach. Moreover, the orders of the translating matrices in the upward and downward passes can also be reduced by using the compressing matrices for M2L, leading to further reduction of the CPU time and memory usage.

The original KIFMM in [6] is designed to accelerate the potential evaluation for particle simulations. Recently, it was applied to solve boundary integral equations (BIEs) in, e.g., blood flow, molecular electrostatic problems [16–18]. It is noticed that the central idea of all those works is to translate the far-field interactions to a particle summation formulation so that the original KIFMM can be used in a *straightforward* manner. For example, in [17], the Nyström method is used to discretize the BIE in order to obtain the particle summation form.

In this paper, the KIFMM is used to accelerate the BEM. This work is nontrivial since the KIFMM can not be straightly used in BEM due to the presence of elements, let alone to maintain the accuracy and efficiency. For example, the equivalent and check surfaces are crucial components of the KIFMM. In the original KIFMM these surfaces can be set as the surfaces of each cube. However, in BEM setting this choice would deteriorate the accuracy, because the boundary elements belonging to a cube can often extrude from the cube; see Section 4 for the details in choosing those surfaces.

2 Basic idea of the KIFMM

The KIFMM was proposed in [6] to solve the potential problems for particles. Here its framework is briefly reviewed.

As the original FMM, the KIFMM is also implemented on a spatial tree structure, which can be constructed by first defining a root level cube containing all the particles, then subdividing it recursively until each leaf cube contains non more than s particles. For each cube C, we define its near field $\mathscr{N}^C$ as the union of the cubes in the same level that share at least one vertex with C, its far field $\mathscr{F}^C$ as the complement of $\mathscr{N}^C$, and its interaction field $\mathscr{I}^C = \mathscr{F}^C \backslash \mathscr{F}^B$, where B is the parent cube of C. The union of the cubes at the same level in $\mathscr{I}^C$ is called its interaction list.

Generally, in a FMM, the potentials induced by the sources in the near field are computed directly, which is named as S2T translations. The potentials induced by the sources in the far field are efficiently evaluated by a series of translations, named as S2M, M2M, M2L, L2L and L2T translations. The main feature of the KIFMM lies in that the above translations are performed using equivalent densities on the equivalent points and check potentials on the check points, while in the conventional FMM the translations are performed using the multipole expansions and local expansions. The equivalent points and check points are sampled on the equivalent surface and check surface, respectively, which can be defined as the surfaces of cubes, as suggested in [6]. In the upward pass of KIFMM, for each cube C, if C is a leaf cube, its upward equivalent densities are translated from the sources inside C by the S2M operator $\mathbf{S}$; otherwise they would be translated from the upward equivalent densities of its C's children by the M2M operator $\mathbf{M}$. In the downward pass, the M2L operator $\mathbf{K}$ translates the upward equivalent densities of C's interaction list into C's downward check potentials. And for each non-leaf cube C, the L2L operator $\mathbf{L}$ translates the downward check potentials of C into that of C's children. When C is a leaf cube, the contribution of the sources in $\mathscr{F}^C$ for the potentials on the target points inside C are translated from the downward check potentials of C by the L2T operator $\mathbf{T}$. For the detail of the translations, we refer to Ref. [6].

In summary, the potential $\mathbf{p}$ on the target points in a leaf cube induced by the source densities $\mathbf{q}$ in its far field can be computed by

$$\mathbf{p} = \mathbf{TLKMSq}. \tag{1}$$

where, $\mathbf{S}, \mathbf{M}, \mathbf{K}, \mathbf{L}, \mathbf{T}$ are the translation operators for S2M, M2M, M2L, L2L, L2T, respectively.

The M2L translation is the most time-consuming step in the KIFMM. It is accelerated by FFT in the original KIFMM [6]. In its implementation auxiliary points must be added *inside* the upward equivalent surface and the downward check surface, although one only needs these on the surfaces. This makes the FFT approach less efficient when the number of the equivalent points and check points are large because the auxiliary points would account for a large proportion. To overcome this drawback, in the next section, a SVD approach is proposed which requires no auxiliary points, and can accelerate all the translations in KIFMM.

3 SVD-based acceleration for translations

Our new SVD-based accelerating technique consists of two steps, namely (1) the matrix reduction for all the translation operators and (2) the low rank approximation for the compressed M2L matrices. These are explained in detail respectively in the following.

3.1 Matrix reduction

The compressing matrices for the matrix reduction step is constructed by the scheme in [7] which is originally used to accelerate the M2L translations of the black-box FMM. It should be noted our matrix reduction is distinguished from [7] by that, the other translation operators are compressed into more compact form as well, which could further improve the efficiency and reduce the memory requirement.

Following [7], the compressing matrices are computed using the M2L matrices. Suppose that the kernel function is translational invariant. The union of unique translating matrices over all cubes in each level forms a set of 316 matrices. To compress these matrices, first collect them into a fat matrix $\mathbf{K}_{\text{fat}}$ in which they are aligned in a single row and a thin matrix $\mathbf{K}_{\text{thin}}$ in which they are aligned in a single column, then perform SVD

$$\begin{aligned}\mathbf{K}_{\text{fat}} &= \begin{bmatrix}\mathbf{K}^{(1)} & \mathbf{K}^{(2)} & \dots & \mathbf{K}^{(316)}\end{bmatrix} \\ &= \mathbf{U\Sigma}\begin{bmatrix}\mathbf{V}^{(1)^{\mathrm{T}}} & \mathbf{V}^{(2)^{\mathrm{T}}} & \dots & \mathbf{V}^{(316)^{\mathrm{T}}}\end{bmatrix},\end{aligned} \tag{2a}$$

$$\begin{aligned}\mathbf{K}_{\text{thin}} &= \begin{bmatrix}\mathbf{K}^{(1)}; & \mathbf{K}^{(2)}; & \dots; & \mathbf{K}^{(316)}\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{Q}^{(1)}; & \mathbf{Q}^{(2)}; & \dots; & \mathbf{Q}^{(316)}\end{bmatrix}\mathbf{\Lambda R}^{\mathrm{T}},\end{aligned} \tag{2b}$$

where $\mathbf{K}^{(i)}$ is the i-th translating matrix. Then it is explained that the M2L matrices can be approximated by

$$\mathbf{K} = \mathbf{U}(\mathbf{U}^{\mathrm{T}}\mathbf{K}\mathbf{R})\mathbf{R}^{\mathrm{T}} \approx \tilde{\mathbf{U}}(\tilde{\mathbf{U}}^{\mathrm{T}}\mathbf{K}\tilde{\mathbf{R}})\tilde{\mathbf{R}}^{\mathrm{T}} = \tilde{\mathbf{U}}\tilde{\mathbf{K}}\tilde{\mathbf{R}}, \tag{3}$$

where, the compressing matrices $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{R}}$ are consisted by columns corresponding with dominant singular values that are not less than $\varepsilon_1\|\mathbf{K}_{\text{fat}}\|_2 = \varepsilon_1\boldsymbol{\Sigma}_{0,0} = \varepsilon_1\boldsymbol{\Lambda}_{0,0}$, and $\tilde{\mathbf{K}}$ is the compressed translating matrix.

Now consider the other translations in the upward and downward passes. Since the columns of $\tilde{\mathbf{R}}$ are orthonormal, thus $\tilde{\mathbf{R}}^{\mathrm{T}}\tilde{\mathbf{R}} = \mathbf{I}$. The potentials in $\mathscr{I}^B$ generated by the upward equivalent densities $\mathbf{q}^{B,\mathrm{u}}$ can be written as follows

$$\mathbf{K}\mathbf{q}^{B,\mathrm{u}} \approx \tilde{\mathbf{U}}\tilde{\mathbf{K}}\tilde{\mathbf{R}}^{\mathrm{T}}\mathbf{q}^{B,\mathrm{u}} = \tilde{\mathbf{U}}\tilde{\mathbf{K}}(\tilde{\mathbf{R}}^{\mathrm{T}}\tilde{\mathbf{R}})\tilde{\mathbf{R}}^{\mathrm{T}}\mathbf{q}^{B,\mathrm{u}} = \tilde{\mathbf{U}}\tilde{\mathbf{K}}\tilde{\mathbf{R}}^{\mathrm{T}}\mathbf{q}_1^{B,\mathrm{u}}, \tag{4}$$

where $\mathbf{q}_1^{B,\mathrm{u}} = \tilde{\mathbf{R}}\tilde{\mathbf{R}}^{\mathrm{T}}\mathbf{q}^{B,\mathrm{u}}$ is the projection of $\mathbf{q}^{B,\mathrm{u}}$ to the space spanned by the columns of $\tilde{\mathbf{R}}$. This suggests that $\mathbf{q}_1^{B,\mathrm{u}}$ can approximately reproduce the potential field in $\mathscr{I}^C$ excited by $\mathbf{q}^{B,\mathrm{u}}$. In other words, $\mathbf{q}_1^{B,\mathrm{u}}$ can be taken as the new upward equivalent densities for the potential field in $\mathscr{I}^C$.

For each cube B and its parent cube C, since $\mathscr{I}^C$ lies outside $\mathscr{I}^B$, from potential theory we know that $\mathbf{q}_1^{B,\mathrm{u}}$ can also be used to reproduce the potential field in $\mathscr{I}^C$, ie.,

$$\begin{aligned} \tilde{\mathbf{U}}\tilde{\mathbf{K}}\tilde{\mathbf{R}}^{\mathrm{T}}\mathbf{M}\mathbf{q}^{B,\mathrm{u}} &\approx \tilde{\mathbf{U}}\tilde{\mathbf{K}}\tilde{\mathbf{R}}^{\mathrm{T}}\mathbf{M}\mathbf{q}_1^{B,\mathrm{u}} \\ &= \tilde{\mathbf{U}}\tilde{\mathbf{K}}\tilde{\mathbf{R}}^{\mathrm{T}}\mathbf{M}\tilde{\mathbf{R}}\tilde{\mathbf{R}}^{\mathrm{T}}\mathbf{q}^{B,\mathrm{u}} = \tilde{\mathbf{U}}\tilde{\mathbf{K}}\tilde{\mathbf{R}}^{\mathrm{T}}\mathbf{M}\tilde{\mathbf{R}}\tilde{\mathbf{R}}^{\mathrm{T}}\mathbf{S}\mathbf{q} \\ &= \tilde{\mathbf{U}}\tilde{\mathbf{K}}\tilde{\mathbf{M}}\tilde{\mathbf{S}}\mathbf{q}, \end{aligned} \tag{5}$$

where, $\tilde{\mathbf{M}} = \tilde{\mathbf{R}}^{\mathrm{T}}\mathbf{M}\tilde{\mathbf{R}}$ is the new translating matrix for M2M; $\tilde{\mathbf{S}} = \tilde{\mathbf{R}}^{\mathrm{T}}\mathbf{S}$ is the new translating matrix for S2M.

From the symmetry of the algorithm, ie., the upward pass and the downward pass playing the same role in the algorithm, we know that the downward pass can be transformed by $\tilde{\mathbf{U}}$ similarly. Thus, $\tilde{\mathbf{L}} = \tilde{\mathbf{U}}^{\mathrm{T}}\mathbf{L}\tilde{\mathbf{U}}$ is the new translating matrix for L2L; $\tilde{\mathbf{T}} = \tilde{\mathbf{T}}\tilde{\mathbf{U}}$ is the new translating matrix for L2T.

Since both the transformation matrices $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{R}}$ are thin matrices, the new translating matrices $\tilde{\mathbf{S}}$, $\tilde{\mathbf{M}}$, $\tilde{\mathbf{L}}$ and $\tilde{\mathbf{T}}$ are smaller than their original forms, and thus the computational cost of the upward and downward passes can be reduced.

The threshold ε_1 affects the balance between the computational cost and the accuracy of the algorithm. The induced error in each M2L translation is of order ε_1, and the total error is approximately $L\varepsilon_1$ [6]. In order to maintain the error decreasing rate of BEM with piecewise constant element, $L\varepsilon_1$ should decrease by a factor of 2 with each mesh refinement $L\varepsilon_1 \sim 2^{-L}$. In this paper, ε_1 is chosen by

$$\varepsilon_1 = C_1\frac{2^{-L}}{L}, \tag{6}$$

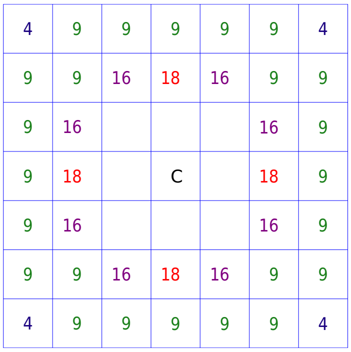
where, C_1 is a constant coefficient.

3.2 Low rank approximation for M2L

After the dimension reduction, we found that most of the compressed M2L matrices $\tilde{\mathbf{K}}$ are still of low numerical ranks, as shown in Figure 1. This fact indicates that the computational cost of M2L can be further reduced by using the low rank decomposition of compressed M2L matrices $\tilde{\mathbf{K}}^{(i)}$. Here the low rank decomposition is computed by SVD, so that optimal rank can be obtained

$$\tilde{\mathbf{K}}^{(i)} \approx \hat{\mathbf{U}}^{(i)}\hat{\mathbf{S}}^{(i)}(\hat{\mathbf{Q}}^{(i)})^{\mathrm{T}} = \hat{\mathbf{U}}^{(i)}\hat{\mathbf{V}}^{(i)}, \tag{7}$$

where, the matrices with hat is the truncated matrices consisted by the columns corresponding with dominant singular values that is no smaller than $\varepsilon_2\|\mathbf{K}_{0,\mathrm{fat}}\|_2$. Since the number of the translating matrices is $\mathcal{O}(1)$, this computational overhead is small.



4	9	9	9	9	9	4
9	9	16	18	16	9	9
9	16				16	9
9	18		C		18	9
9	16				16	9
9	9	16	18	16	9	9
4	9	9	9	9	9	4

Figure 1: Numerical rank distribution of M2L matrices $\tilde{\mathbf{K}}_{84\times 84}$ in numerical example 5.1 with $N = 2097152, p = 8, C_1 = 0.1, C_2 = 100$.

The error introduced by this approximation is determined by ε_2. Denote $\hat{\mathbf{K}}^{(i)} = \hat{\mathbf{U}}^{(i)}\hat{\mathbf{V}}^{(i)}$. From the truncating scheme, there exists

$$\|\hat{\mathbf{K}}^{(i)} - \tilde{\mathbf{K}}^{(i)}\|_2 \le \varepsilon_2\|\mathbf{K}_{\mathrm{fat}}\|_2.$$

Since $\|\mathbf{A}\|_{\max} \le \|\mathbf{A}\|_2 \le \sqrt{mn}\|\mathbf{A}\|_{\max}$ for arbitrary $m \times n$ matrix $\mathbf{A}$, thus

$$\|\hat{\mathbf{K}}^{(i)} - \tilde{\mathbf{K}}^{(i)}\|_{\max} \le \varepsilon_2\|\mathbf{K}_{\mathrm{fat}}\|_2.$$

Let $\hat{\mathbf{K}}_{\mathrm{fat}}$ and $\tilde{\mathbf{K}}_{\mathrm{fat}}$ denote the fat matrices for $\hat{\mathbf{K}}$ and $\tilde{\mathbf{K}}$, respectively, which are constructed similarly as $\mathbf{K}_{\mathrm{fat}}$. It is easy to know that

$$\|\hat{\mathbf{K}}_{\mathrm{fat}} - \tilde{\mathbf{K}}_{\mathrm{fat}}\|_{\max} \le \varepsilon_2\|\mathbf{K}_{\mathrm{fat}}\|_2.$$

Since the dimension of $\tilde{\mathbf{K}}_{\mathrm{fat}}$ is $\tilde{p} \times 316\tilde{p}$, where $\tilde{p}$ is the dimension of $\tilde{\mathbf{K}}$, and

$$\|\hat{\mathbf{K}}_{\mathrm{fat}} - \tilde{\mathbf{K}}_{\mathrm{fat}}\|_{\max} \ge \frac{1}{\sqrt{316\tilde{p}^2}}\|\hat{\mathbf{K}}_{\mathrm{fat}} - \tilde{\mathbf{K}}_{\mathrm{fat}}\|_2,$$

one has

$$\|\hat{\mathbf{K}}_{\mathrm{fat}} - \tilde{\mathbf{K}}_{\mathrm{fat}}\|_2 \le \varepsilon_2\sqrt{316\tilde{p}^2}\|\mathbf{K}_{\mathrm{fat}}\|_2.$$

Therefore, the error introduced by the low rank approximation is ensured to be of same order as ε_1 by letting

$$\varepsilon_2 \sim \frac{\varepsilon_1}{\sqrt{316\tilde{p}^2}} \sim \frac{\varepsilon_1}{\tilde{p}}.$$

In our scheme, it is defined by

$$\varepsilon_2 = C_2 \frac{\varepsilon_1}{\tilde{p}}, \tag{8}$$

where C_2 is a constant coefficient.

Generally the compression (3) is performed for M2L translating matrices at all levels. When the kernel is homogeneous, the M2L matrices in different levels can be scaled to each other. Therefore, the matrix reduction and low rank approximation has only to be implemented on the M2L operators of one level, and the operators in other levels can be obtained by scaling.

4 KIFMM for BEM

Now let us discussion the application issues of KIFMM to accelerating BEM. The main difference with the KIFMM for particle summations lies in the definition of equivalent and check surfaces, since now the sources distribute continuously on the boundary instead of on discrete points.

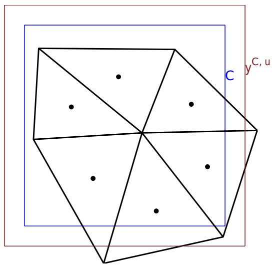


Figure 2: The elements and the upward equivalent surface related to a leaf cube.

In this paper, the BIE is discretized with piecewise constant triangular elements. The centroids of the triangles are used as the reference points to construct the octree. Figure 2 illustrates a leaf cube C in the octree. The union of all the elements whose centroids lying in C is denoted by $\Gamma(C)$. Obviously in the S2M translation, the upward check potentials of each leaf cube C has to be computed by quadrature over the triangles on $\Gamma(C)$.

It is shown in Figure 2 that $\Gamma(C)$ may extrude from C. In order to ensure the existence of the equivalent densities and the accuracy of the far field approximation, the definition of the upward equivalent surface $\mathbf{y}^{C,\mathrm{u}}$ and the upward check surface $\mathbf{x}^{C,\mathrm{u}}$ has to satisfy the following restrictions:

1. $\mathbf{y}^{C,\mathrm{u}}$ and $\mathbf{x}^{C,\mathrm{u}}$ lie between $\Gamma(C)$ and $\mathscr{F}^C$; $\mathbf{x}^{C,\mathrm{u}}$ encloses $\mathbf{y}^{C,\mathrm{u}}$;
2. $\mathbf{y}^{C,\mathrm{d}}$ and $\mathbf{x}^{C,\mathrm{d}}$ lie between C and $\Gamma(\mathscr{F}^C)$, with $\Gamma(\mathscr{F}^C)$ being the union of all elements that belongs to $\mathscr{F}^C$; $\mathbf{y}^{C,d}$ encloses $\mathbf{x}^{C,\mathrm{d}}$.

The equivalent and check surfaces for BEM are defined in a similar way with that in [6]. That is, for each cube C with side length $2r$, $\mathbf{y}^{C,\mathrm{u}}$ is defined as the surface of the concentric cube with halfwidth $(1 + d)r$, and $\mathbf{x}^{C,\mathrm{u}}$ is defined as the surface of the concentric cube with halfwidth $(3 - 2d)r$. The difference is that, in [6], d is chosen as a user-defined small value. However, in the KIFMM accelerated BEM, d has to be chosen large enough to satisfy the above restrictions. For a quasi-uniform element partition, assume that the size of the element is h and each leaf cube contains at most s elements, then the halfwidth of the leaf cubes in the finest level is proportional with $\sqrt{s}h$. The distance between the out-most vertex and the cube surface is no larger than h, thus d is of order

$$d \sim \mathcal{O}\left(\frac{(\sqrt{s}+1)h}{\sqrt{s}h} - 1\right) = \mathcal{O}\left(\frac{1}{\sqrt{s}}\right).$$

So, in this paper d is evaluated as

$$d = C_d \frac{1}{\sqrt{s}}, \tag{9}$$

where, C_d is user-defined constant. Our numerical experience indicates that $C_d = 0.5$ is proper for most problems.

The M2M, M2L, L2L and L2T translations in the KIFMM BEM is exactly the same with the KIFMM for particle summations. Obviously, the total computational complexity of our KIFMM BEM remains $\mathcal{O}(N)$.

5 Numerical examples

The performance of our SVD-based accelerating technique and the kernel-independent fast multipole BEM for Laplace BIEs is demonstrated by two numerical examples. The resultant linear systems are solved by GMRES solver. All simulations are carried out on a computer with a Xeon 5440 (3.00 GHz) CPU and 28 GB RAM.

5.1 Electrostatic problem

In this subsection, the electric charge density on an ellipsoidal conductor is computed by

$$\int_\Gamma G(\mathbf{x},\mathbf{y})q(\mathbf{y})\mathrm{d}\mathbf{y} = f(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \tag{10}$$

where, $G(\mathbf{x},\mathbf{y}) = 1/(4\pi|\mathbf{x}-\mathbf{y}|)$ is the fundamental solution of the Laplace equation. The ellipsoid can be described by $(x_1/2)^2 + x_2^2 + (x_3/3)^2 = 1$. The analytic solution can be expressed analytically using ellipsoidal coordinates. The convergence tolerance for GMRES solver is set to be 10^{-6}. The surface of the ellipsoid is first discretized into $N = 2048$ triangular elements, then the mesh is refined 5 times. The finest mesh has $N = 2097152$ elements.

Table 1: Errors obtained with $N = 2097152$ and different p, C_1 and C_2.

p	FFT	$C_1 = 0.1$ $C_2 = 0$	$C_1 = 0.5$ $C_2 = 0$	$C_1 = 0.1$ $C_2 = 10$	$C_1 = 0.1$ $C_2 = 100$	$C_1 = 0.1$ $C_2 = 500$
4	0.004 880	0.004 880	0.004 887	0.004 883	0.004 896	0.005 296
6	0.000 790	0.000 791	0.001 227	0.000 791	0.000 793	0.000 932
8	0.000 789	0.000 790	0.001 014	0.000 790	0.000 793	0.000 988

Table 2: CPU times in each iteration T_{it} and the total memory usage with $N = 2097152$, $C_1 = 0.1$ and different p, C_2

p	T_{it} (s)			Memory usage (MB)		
	FFT	$C_2 = 10$	$C_2 = 100$	FFT	$C_2 = 10$	$C_2 = 100$
4	18.25	30.12	24.32	7 504.3	7 598.9	7 597.7
6	76.16	45.51	34.79	11 669.0	8 861.7	8 859.8
8	198.03	44.89	34.73	17 924.3	8 866.4	8 864.6

Table 3: Results of the fast BEM accelerated by FFT and SVD with $p = 6$. In the SVD accelerating technique, $C_1 = 0.1$ and $C_2 = 10$.

N	Relative error		T_{it} (s)		Memory usage (MB)	
	FFT	SVD	FFT	SVD	FFT	SVD
2 048	0.032 901	0.033 082	0.05	0.01	19.8	8.1
8 192	0.014 001	0.014 081	0.31	0.06	55.8	28.3
32 768	0.006 641	0.006 681	1.11	0.30	186.4	119.5
131 072	0.003 182	0.003 220	4.75	1.78	736.5	493.4
524 288	0.001 579	0.001 587	13.26	8.56	2 917.5	2 062.5
2 097 152	0.000 790	0.000 791	76.16	45.51	11 669.0	8 861.7

The accuracy and efficiency of the present KIFMM BEM are mainly determined by parameters C_1 in (6) and C_2 in (8). Basically, with larger C_1 and C_2, the computational cost would be lower, while on the other hand the error would become greater. Consequently, the choices of C_1 and C_2 are determined by the tradeoff between the accuracy and the efficiency.

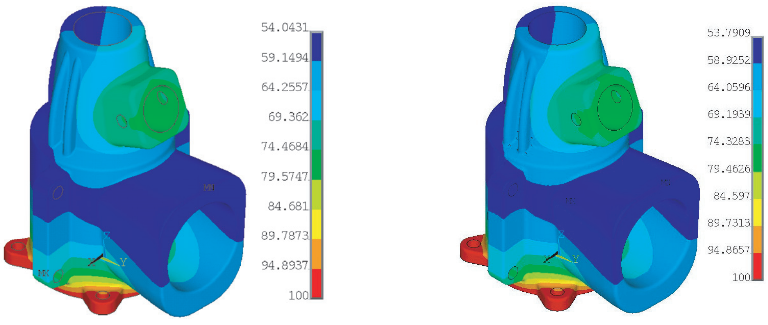
First the influence of C_1 on the accuracy of the algorithm is tested by the model with the finest mesh, which could get the best accuracy. Three cases with C_1 being 0, 0.1 and 0.5 and $C_2 = 0$ are computed. The results corresponding to $C_1 = 0$ are computed using the original FFT-accelerating scheme in [6]. The resulting errors are listed in Table 1. It is shown that $C_1 = 0.1$ is nearly optimal to maintain the accuracy. The error with $p = 4$ is much larger, this is because the error of the algorithm is also bounded by p, see [6] for the details. The errors with $p = 6$ and $p = 8$ are almost the same, which indicates for this numerical example, $p = 6$ is sufficient to maintain the accuracy of the BEM.

The influence of C_2 is studied by setting $C_2 = 10$, 100, 500 while $C_1 = 0.1$. In Table 1 it can be seen that for $C_2 = 10$ and $C_2 = 100$ the results keep almost the same errors; while for $C_2 = 500$ the errors increase. Therefore, C_2 should be chosen between 10 and 100 to maintain the accuracy. The CPU times T_{it} in each iteration and the total memory usage of the two methods, FFT-accelerating approach and the SVD accelerating approach, are listed in Table 2. It is shown that the iteration with the SVD approach can be considerably more efficient comparing with the FFT approach, especially for large p. The reason is that, when p is increased, more auxiliary points has to be added in the FFT approach, which makes it less efficient. However, the efficiency of the SVD accelerated algorithm is mainly determined by ε_1 and ε_2, which is independent with p. Besides the CPU time, the memory usage can also be considerably reduced in the SVD approach, since the translating matrices used in S2M and L2T are compressed into more condensed form by the scheme in Section 3.1.

Now the results with $p = 6$, $C_1 = 0.1$, $C_2 = 10$ and different DOFs are listed in Table 3. It is shown that the computational cost increases linearly with the DOF. The CPU time cost in each iteration can be reduced about 40% and the memory cost can be reduced about 25% by SVD approach compared with the original FFT approach while maintaining the accuracy of BEM. These parameters will be used in the next numerical example.

5.2 Heat conduction problem

To demonstrate the ability of the present KIFMM BEM for solving real-world problems, a steady-state heat conduction analysis of a engine block is solved here; see Figure 5.2. The temperature field is governed by the Laplace equation. The conductivity of the engine block is $\lambda = 80\mathrm{W}/(\mathrm{m}\cdot{}^\circ\mathrm{C})$. The temperature of the inner surface of the oblique tube and the temperature of the bottom surface are set to be $75^\circ\mathrm{C}$ and $100^\circ\mathrm{C}$, respectively. Convective condition with constant film coefficient $h = 10\mathrm{W}/(\mathrm{m}^2 \cdot{}^\circ\,\mathrm{C})$ and constant bulk temperature $T_0 = 22^\circ\mathrm{C}$ are applied to the other surfaces. The KIFMM BEM is applied to compute the temperature field with nearly 4754670 elements. For comparison, this problem is also solved by finite element method (FEM) with 698317 tetrahedral elements, 1015653 nodes. The converging tolerance for GMRES solver is 10^{-4}. It converged after 97 iterations. Each iteration cost about 108 s, and the overall computing time is 5.09 h. The memory consumption is 24.4 GB. The result is exhibited in Figure 3(b). It can be seen that the temperature distribution obtained by the KIFMM BEM agrees very well with that by FEM in Figure 3(a).



(a) FEM result (b) KIFMM result

Figure 3: Temperature contours computed by FEM and KIFMM BEM.

6 Conclusion

The FMM is one of the most successful fast algorithms for BEM acceleration. But it requires the analytical expansion of the kernel function, which makes it difficult to be applied to some complicated problems. Recently, various kernel-independent FMMs were developed to overcome this drawback. Among them the KIFMM proposed in [6] has high efficiency and accuracy, and thus has been extensively used [16–18]. The time consuming M2L translations are accelerated by using the FFT. However, it is noticed that when more equivalent and check points are sampled to get higher accuracy, the efficiency of the FFT approach tends to be lower because more auxiliary volume grid points have to be added in order to do FFT.

In this paper, the low rank property of the translating matrices in KIFMM is sufficiently exploited by SVD (called SVD approach in this paper) to accelerate all the translations, including the most time-consuming M2L. It consists of two steps. First *all the translation matrices* are compressed into more compact form, then the compressed M2L matrices are approximated by low rank approximations. Finally, the above improved KIFMM is applied to accelerate BEM, leading to a highly efficient KIFMM BEM for solving large-scale problems.

The accuracy and efficiency of the SVD approach and the KIFMM BEM are demonstrated by numerical examples. It is shown that, when compared with the

FFT-accelerated KIFMM, the SVD approach can reduce about 40% of the iterating time and 25% of the total memory requirement. The presented KIFMM BEM is of $\mathcal{O}(N)$ complexity. By using this method Laplace problem with nearly 5 million unknowns can be successfully solved within 5 hours on a Xeon-5440 2.83 GHz CPU and 28 GB RAM.

Acknowledgements

This work was supported by the Doctorate Foundation of Northwestern Polytechnical University under Grant No. CX201220, National Science Foundations of China under Grants 11074201 and 11102154, and Funds for Doctor Station from the Chinese Ministry of Education under Grants 20106102120009 and 20116102110006.

References

[1] L. Greengard and V. Rokhlin. A fast algorithm for particle simulation. *Journal of Computational Physics*, 73:325–348, 1987.
[2] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms. *Pure Appl. Math.*, 37:141–183, 1991.
[3] W Hackbusch. A sparse matrix arithmetic based on h-matrices. part I: Introduction to h-matrices. *Computing*, 62:89–108, 1999.
[4] M Bebendorf and S Rjasanow. Adaptive low-rank approximation of collocation matrices. *Computing*, 70:1–24, 2003.
[5] Joel R. Phillips and Jacob K. White. A precorrected-fft method for electrostatic analysis of complicated 3-d structures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(10):1059–1072, 1997.
[6] Lexing Ying, George Biros, and Denis Zorin. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics*, 196:591–626, 2004.
[7] William Fong and Eric Darve. The black-box fast multipole method. *Journal of Computational Physics*, 228(23):8712–8725, 2009.
[8] Pierre-David Létourneau, Christopher Cecka, and Eric Darve. Generalized fast multipole method. *IOP Conference Series: Materials Science and Engineering*, 10, 2010.
[9] Yuchun Lin, Andrij Baumketner, Shaozhong Deng, Zhenli Xu, Donald Jacobs, and Wei Cai. An image-based reaction field method for electrostatic interactions in molecular dynamics simulations of aqueous solutions. *The Journal of chemical physics*, 131(15):154103, 2009.
[10] Ming Xiang, Shaozhong Deng, and Wei Cai. A sixth-order image approximation to the ionic solvent induced reaction field. *Journal of scientific computing*, 41(3):411–435, 2009.

[11] S. N. Razavi, N. Gaud, N. Mozayani, and A. Koukam. Multi-agent based simulations using fast multipole method: application to large scale simulations of flocking dynamical systems. *Artificial Intelligence Review*, 35(1), 2011.

[12] L. Greengard and V. Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numerica*, 229–269, 1997.

[13] W. Elliott and J. Board. Fast fourier transform accelerated fast multipole algorithm. *SIAM J. Sci. Comput.*, 17(2):398–415, 1996.

[14] Junichiro Makino. Yet another fast multipole method without multipoles — pseudoparticle multipole method. *Journal of Computational Physics*, 151:910–920, 1999.

[15] R. Yokota. An fmm based on dual tree traversal for many-core architectures. *arXiv: 1209.3516*, 2012.

[16] Abtin Rahimian, Ilya Lashuk, Shravan K. Veerapaneni, Aparna Chandramowlishwaran, Dhairya Malhotra, Logan Moon, Rahul Sampath, Aashay Shringarpure, Jeffrey Vetter, Richard Vuduc, Denis Zorin, and George Biros. Petascale direct numerical simulation of blood flow on 200k cores and heterogeneous architectures. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, New Orleans, Louisiana, USA, November 2010. IEEE.

[17] Lexing Ying, George Biros, and Denis Zorin. A high-order 3d boundary integral equation solver for elliptic pdes in smooth domains. *Journal of Computational Physics*, 219:247–275, 2006.

[18] Chandrajit Bajaj, Shun-Chuan Chen, and Alexander Rand. An efficient higher-order fast multipole boundary element solution for Poisson-Boltzmann-based molecular electrostatics. *SIAM J. Sci. Comput.*, 33(2):826–848, 2011.