A Geometric Mapping Cross Approximation Method

Jianming Zhang[*], Xingshuai Zheng, Chenjun Lu，Guizhong Xie，Guangyao Li

State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body,

College of Mechanical and Vehicle Engineering,

Hunan University, Changsha 410082, China

Correspondence to: Jianming Zhang

College of Mechanical and Vehicle Engineering, Hunan University, Changsha 410082, China

Telephone: +86-0731-88823061

E-mail: zhangjianm@gmail.com

ABSTRACT

The matrices of Boundary Element Method (BEM) are fully populated and require special compression techniques for the efficient treatment. In this article, the $\mathcal{H}$-matrix representation is used to approximate the dense stiffness matrix in admissible blocks by low-rank matrices. This paper presents a Geometric Mapping Cross Approximation (GMCA) algorithm to compute the low-rank matrices. Compared with the Adaptive Cross Approximation (ACA), the GMCA determines the skeleton points from the two interacting groups of nodes by their spacing characteristics directly and, thus, has a remarkable non-iterative nature and requires some simple geometric transformations, only. Numerical examples show that the new algorithm is feasible.

Keywords: $\mathcal{H}$-matrix; low-rank matrices; adaptive cross approximation; skeleton point; geometric mapping

1. INTRODUCTION

In the applied and engineering sciences, Boundary Element Method (BEM) plays an important role, especially in the problems with infinite domain. However, the resulting matrices are in general dense and require computer memory consumption scale quadratically with respect to the degrees of freedom. The efficient treatment of dense matrices requires special compression techniques to reduce the storage requirement and speed up the arithmetic (e.g., inversion).

Several fast solutions of boundary integral equation have been developed in the last two decades. Well known are panel clustering[1,2], multipole expansions [3-8] and (adaptive) cross approximation[9-12,17,18]. Even the wavelet technique can also be used to compress the resulting dense matrix, when the underlying geometry can be described by a few of smooth maps[13].

The panel clustering and fast multipole method (FMM) are both based on an explicit Taylor series expansion of the kernel function. However, the former neglects the special structure of the kernel function as a result that the separation rank produced by this method is rather large[12]. Compared with the panel clustering, the fast multipole method not only requires explicit expansion but also exploits this special structure. Obviously, these two methods are limited to the standard kernel functions where they are known and the expansions of the kernel functions in different problems are not uniform.

A new approximation method is proposed by Bebendorf (cf.(1)). As can be seen in Eq. 1, this method takes the single-variable function which is the result of fixing some source points or field points of the kernel function as the basis of the interpolation space. The selected points are called skeleton points. Hence, the low rank approximation can be assembled when we have found an optimal set of the skeleton points. Obviously, the optimal set of the skeleton points is not unique.

$$\kappa(P,Q) \approx \{\kappa(P,Q_1) \quad \ldots \quad \kappa(P,Q_k)\} \begin{bmatrix} \kappa(P_1,Q_1) & \cdots & \kappa(P_1,Q_k) \\ \vdots & & \vdots \\ \kappa(P_k,Q_1) & \cdots & \kappa(P_k,Q_k) \end{bmatrix}^{-1} \begin{Bmatrix} \kappa(P_1,Q) \\ \vdots \\ \kappa(P_k,Q) \end{Bmatrix} \tag{1}$$

An algorithm called adaptive cross approximation (ACA) is proposed by Bebendorf and Rjasanow[9,10,17]. This algorithm determines the skeleton points similarly to Gram-Schmidt orthogonalization. It uses only entries from the original matrix for the approximation of each block. This algorithm can be regarded as the algebraic counterparts of panel clustering and fast multipole methods. There is no need for the expansion of the kernel function. Compared with the FMM, the ACA is the easier parallelization of the algorithm[14] and much simpler to be implemented. Meanwhile, as an advantage over the FMM, an approximation of the global matrix can be used for preconditioning besides the near-field matrix[15]. The Hybrid Cross Approximation (HCA) which combines the ACA algorithm and the interpolation-based separation of the kernel function is proposed by Steffen and Lars[12]. An algorithm which constructs nested bases approximations in the spirit of ACA is proposed by Bebendorf and Venn[19] to farther improve the storage complexity of low rank approximation. Likewise, an algorithm which determines the skeleton points using random sampling technique is proposed by P.G. Martinsson[16].

Our contribution is a new non-iterative method that implements the low rank approximation of each admissible block in the original matrix of BEM based on the geometric mapping. It is well known that the matrices of BEM are derived from the boundary integrals of the kernel functions. In contrast with implementations by constant elements where analytical integration is feasible, numerical integrations are inevitable when the higher order elements are used. However, it would be difficult and considerably inefficient to encapsulate the numerical integration into the iteration procedure of the ACA. The goal of the proposed method is separating the procedure of determining the skeleton points from the integral operation of the kernel function. The skeleton points are determined in advance from the perspective of the geometric characteristics. Then, the boundary integrals which correspond to the skeleton points in the far-field can be computed together with those for the near-field interaction. Integrals for both near-field and far-field interactions are computed in a unified framework in the same way as the traditional BEM.

The rest of this article is organized as follows: in Section 2, we introduce a simple model problem and describe the $\mathcal{H}$-matrix format and the low rank approximation in short. In Section 3, we introduce the Geometric Mapping Cross Approximation algorithm and provide numerical examples that show this algorithm is feasible in Section 4. Finally, the paper ends with conclusions in Section 5.

2. THE H-MATRIX FORMAT

2.1. Model problem: integral equation

Large dense matrices are usually derived from the BEM without additional structure. These matrices arise after the discretisation of the boundary integral equation

$$A[u](x) = \int_{\Gamma} K(x,y)u(y)d\Gamma(y) \tag{2}$$

where $\Gamma \subset R^3$ is the boundary of the computational domain and $\boldsymbol{K}: \ \Gamma \times \Gamma \to R$ is the kernel function. The kernel function might be the single or double layer kernel for the Laplacian:

$$g^{SLP}(x,y) := \frac{1}{4\pi\|x-y\|}, \qquad g^{DLP}(x,y) := \frac{\langle x-y, n(x)\rangle}{4\pi\|x-y\|^3} = \frac{\partial g^{SLP}(x,y)}{\partial n(x)}$$

The kernel functions $g^{SLP}(x,y)$ and $g^{DLP}(x,y)$ are asymptotically smooth. In fact, H-matrix is based on the typical kernel functions, singularities only occur at the diagonal and the function is smooth everywhere else[12].

A standard Galerkin discretisation of A$[u](x)$ for a basis $(\varphi_j)_{j\in I}, I = \{1,...,n\}$, $V_n := span\{\varphi_1,...,\varphi_n\}$, yields a matrix A with entries

$$a_{i,j} = \int_{\Gamma}\int_{\Gamma} \varphi_j(x)K(x,y)\psi_i(y)d\Gamma(x)d\Gamma(y) \tag{3}$$

The support of the kernel function *K*(*x,y*) is in general not local. So, a dense matrix **A** arises. The computational complexity for computing and the storage requirement for storing a dense matrix are quadratic to the number of the degrees of freedom. Hence, an efficient approximation method has to be developed.

2.2. Low-rank approximation

The rank of a matrix **A** is defined as the number of linearly independent columns or rows of **A**. For a matrix $\mathbf{A} \in C^{m\times n}$, the rank of **A** is bounded by the minimum value of the ***m*** and ***n***. Although the matrices in the BEM usually have full rank, they can often be approximated by matrices having a much lower rank. For example, a matrix $\mathbf{A} \in C^{m\times n}$, the rank of **A** is ***k***. A low rank approximation of **A** can be represented as a factorisation of the form $A = UV^T$ with matrices $U \in R^{n\times k}, V \in R^{m\times k}$ (cf. Fig.1).

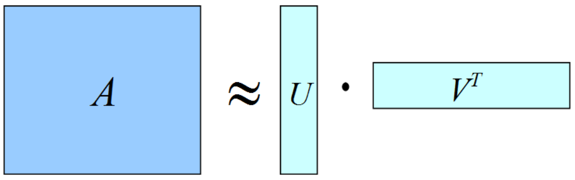


Fig. 1. The schematic diagram of the low rank approximation

The storage requirement in low rank approximation is $k(m+n)$ in contrast to the quadratic cost mn for standard full matrix. Hence, if the k is small and the condition $k(m+n) < mn$ is satisfied, **A** matrix is said to be of low rank. These matrices are favorably stored in the outer-product form.

The operational complexity in the matrix-vector multiplication is also efficiently improved. In the standard representation, the number of dominant arithmetic operations is mn. In contrast to this, only $2k(m+n)$-k-n additions and multiplications of real numbers are necessary in the outer-product form.

The best possible low rank approximation of the matrix $\mathbf{A} \in C^{m\times n}$ can be obtained by its singular value decomposition (SVD). However, the cost of computing an SVD for a general matrix is expensive. For this reason, some cross approximation algorithms are applied [9-12, 17, 18]. In Section 3 we will construct a low rank approximation of the matrix **A** (cf.(3)) by a non-iterative cross approximation algorithm which is based on the geometric mapping.

3. GEOMETRIC MAPPING CROSS APPROXIMATION ALGORITHM

The cross approximation algorithm can be regarded as the procedure of searching the skeleton points from the two admissible interacting groups of nodes. The higher approximation accuracy can be obtained if the columns or rows in the two low rank matrices which are computed by the skeleton points have the larger linear independence. In the BEM, the kernel functions are asymptotically smooth, and are the decay functions of the distance between the source point and the field points. Thus, we try to determine the skeleton points based on the relative geometric relationship between the boundary nodes.

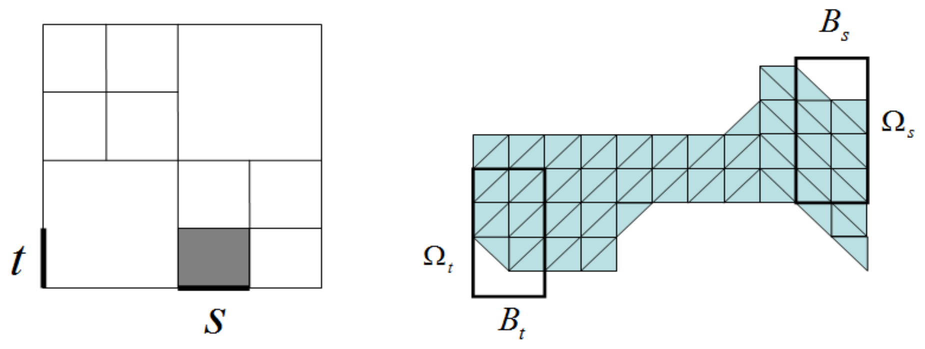


Fig. 2. The admissible block $t \times s$ corresponds to a subset $\Omega_t \times \Omega_s$

3.1. The geometric mapping

In this section, we will introduce the geometric mapping for obtaining the skeleton points. The same as the ACA, this algorithm requires that the strong η-admissibility condition holds [9,10,17]. In other words, it will be effective only on those parts of the domain that are far from the singularity as well. We fix axially parallel boxes B_s and B_t which bound the nodes of the BEM and hold the strong η-admissibility condition(cf. Fig.2).

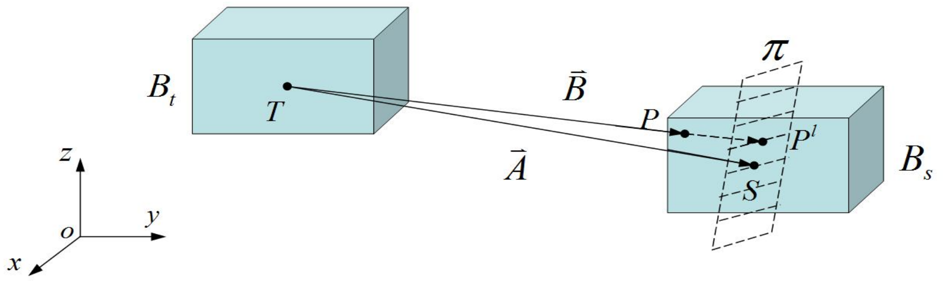


Fig. 3. The boxes B_t and B_s in the three dimensions and the geometric mapping

The image of the parallel boxes B_t and B_s in three dimensions are shown in Fig.3. Now, let us implement the geometric mapping in the box B_s. Firstly, We denote the centers of the two boxes by T and S. All nodes in the box B_s are the object nodes. The projection plane π is defined in the box B_s, which is perpendicular to the vector $\boldsymbol{A}$ drawn from point T to S, and passes through the point S.

Secondly, we set the local coordinate system (x^*, y^*, z^*) on the projection plane in which the axis z^* is reversed to the vector $\boldsymbol{A}$ and the origin is coincidence with the point S (cf. Fig.4).

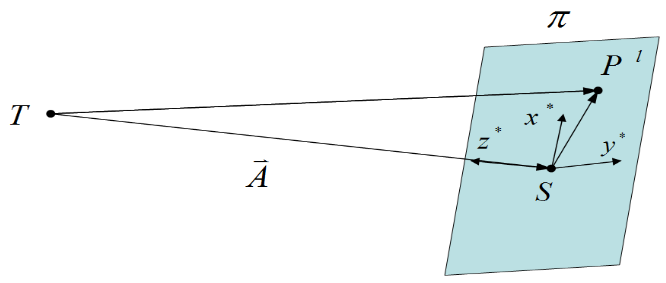


Fig. 4. The local coordinate system on the projection plane

Thirdly, all object nodes are projected to the projection plane π along the mapping vectors drawn from the center T to the object nodes, respectively. For example, a node in the box B_s is denoted by P. Then, the projection point P^l corresponding to the object node P is the intersection of the mapping vector $\boldsymbol{B}$ with the projection plane π (cf. Fig.3). So, we can obtain the coordinates of the projection point P^l by

$$t = \frac{\left\|\overrightarrow{TS}\right\|}{\left\|\overrightarrow{TP}\right\| \bullet \cos\angle PTS} = \frac{\left\|\vec{A}\right\|}{\left\|\vec{B}\right\| \bullet \cos\angle PTS} \quad , \qquad P^l = T + \overrightarrow{TP} \cdot t \tag{4}$$

in which P^l and T denote the coordinates of the point P^l and T, respectively. The vectors $\boldsymbol{A}$ and $\boldsymbol{B}$ are shown in the Fig.3.

Next, we can get the local coordinates of all the projection points on the projection plane π . For example, we assume the vector drawn from the center S to the projection point P^l to be (x_1, y_1, z_1) (cf. Fig.4). Then, the local coordinates of the point P^l denoted by (x_1*, y_1*, z_1*) are given by

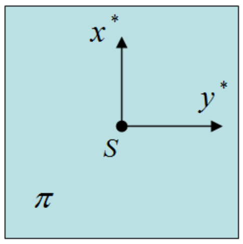
$$\begin{bmatrix} x_1^* \\ y_1^* \\ z_1^* \end{bmatrix} = \begin{bmatrix} \frac{\partial x^*}{\partial x} & \frac{\partial x^*}{\partial y} & \frac{\partial x^*}{\partial z} \\ \frac{\partial y^*}{\partial x} & \frac{\partial y^*}{\partial y} & \frac{\partial y^*}{\partial z} \\ \frac{\partial z^*}{\partial x} & \frac{\partial z^*}{\partial y} & \frac{\partial z^*}{\partial z} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = J \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \tag{5}$$

in which $\boldsymbol{J}$ is the transformation matrix expressing the local Cartesian variables in terms of the global variables. The entries of the matrix $\boldsymbol{J}$ are the direction cosines between the global and local coordinate axes in the 3D Cartesian system.

3.2. The search tree on the projection plane for determining the skeleton points

In the preceding section, we project all the nodes in the box B_s on the projection plane π and compute the local coordinates of all the projection points. In the course of this section, we will determine the skeleton points by constructing the search tree on the projection plane.

Firstly, we construct a general quadtree on the projection plane with the depth to be 3 which covers all the projection points. The box of the quadtree parallel to the axes x^* and y^*, is the minimal square bounding all the projection points and centered by the point S (cf.Fig.5). The square box is called the cell of level 0. Then, start dividing the parent cell into four equal child cells of level 1. Continue dividing in this way, that is, take a parent cell of level k and divide it into four child cells of level k+1. A cell having no child cells is called a leaf. Therefore, we can get sixty-four leaves in this quadtree structure.



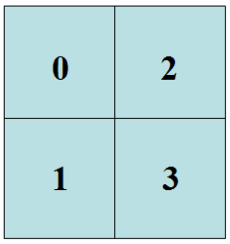
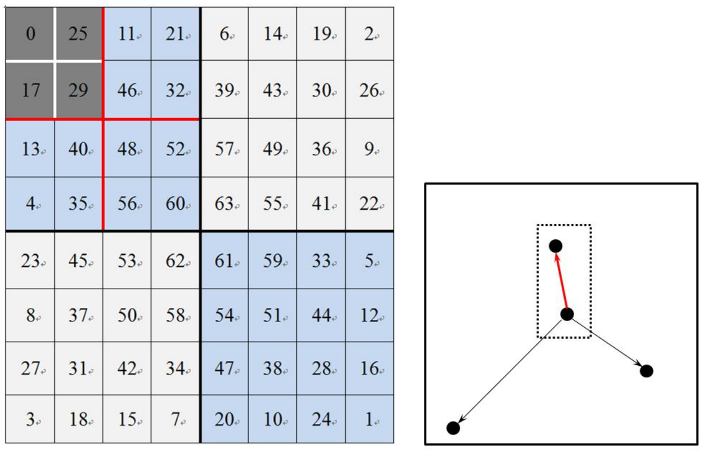


Fig. 5. The box of the quadtree on the projection plane and the numbering scheme for the child cells of any given parent cell

Secondly, we number the sixty-four leaves in a special way shown in the Fig.6, and take the serial number as the priority order of the leaves. For example, we take the leaf at the left-top corner of the bounding box as the first priority with zero. The leaf at the right-bottom corner which is farthest from the first leaf is taken as the second priority with one. The third leaf which is farthest from the first and second leaves as possible is taken as the third priority with two. In this way, we can ensure that the distance in the projection plane between any two leaves whose numbers are adjacent is farthest as possible. Here, we try to get the columns or rows in the two low rank matrices which are computed by the skeleton points and have the larger linear independence based on the farther distance within the projection plane. In Section 3.1

we project all the nodes in the box B_s on the projection plane π and the projection points can be obtained. After the construction of the quadtree, all of the projection points are clustered by the leaves.



0	25	11	21	6	14	19	2
17	29	46	32	39	43	30	26
13	40	48	52	57	49	36	9
4	35	56	60	63	55	41	22
23	45	53	62	61	59	33	5
8	37	50	58	54	51	44	12
27	31	42	34	47	38	28	16
3	18	15	7	20	10	24	1

Fig. 6. The priority order of all the leaves and the recorded projection point in a leaf

Thirdly, we sequentially search the projection points in the leaves based on the priority order, and only one projection point will be recorded in a leaf where more than one projection point is existent. The projection point closest to the center of the leaf will be recorded when there are more than one projection points (cf.Fig.6). For example, we record the first projection point in the leaf with first priority, then, the leaf with second priority will be considered. If there is no projection point, we will record the second projection point in the leaf with next priority. In this way, several projection points can be obtained. Finally, we can take the nodes which correspond to the recorded projection points in the box B_s as the skeleton points.

The same as the procedure of determining the skeleton points in the box B_s, we can obtain the skeleton points in the box B_t. Take the minimum value of the numbers of skeleton points in boxes B_t and B_s as the rank k of the low rank matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ (cf Sect. 2.2). We take the same number of skeleton points according to the rank k in the boxes B_t and B_s, respectively.

Using the above procedure, we can obtain a set of skeleton points in boxes B_t and B_s, respectively. The distances between the skeleton points are farthest as possible in the projection plane, so that the columns or rows in low-rank matrices which correspond to the skeleton points can be linearly independent as large as possible.

3.3. Assembling the low rank approximation

In Section 3.2, we have obtained the skeleton points in the two boxes. Then, the low rank approximation represented as the out-product form (cf Sect. 2.2) can be obtained as follows:

Firstly, the low rank matrix ***U*** is assembled as its columns derived from the matrix **A** which correspond to the skeleton points in the box B_s. Secondly, we construct the low rank matrix $\boldsymbol{V}^T$ as its rows derived from the matrix **A** which correspond to the skeleton points in the box B_t. At last, we obtain the low rank approximation of the matrix **A**. However, the amount of storage required for the low rank approximation can still be reduced, for the low rank matrices ***U*** and ***V*** usually have orthonormal columns.

Hence, we assemble the low rank approximation based on the pseudo-skeleton representation as follows[17,18]:

$$UV^T = A_{1:m,j_{1:k}} A_k^{-1} A_{i_{1:k},1:n} \tag{6}$$

where $A_k := A_{i_{1:k},j_{1:k}}$. The parts $A_{1:m,j_{1:k}}$ and $A_{i_{1:k},1:n}$ are derived from the original matrix **A** the same as the matrices ***U*** and ***V***. In this way, we construct the matrix ***V***

$$V^T = A_k^{-1} A_{i_{1:k},1:n} \tag{7}$$

where A_k^{-1} assembled by the SVD of the matrix A_k .

4. NUMERICAL EXAMPLES

The purpose of this section is to validate the new algorithm. In Sect.4.1, we apply the GMCA and the ACA in the examples with the single layer potential for comparison. For the original ACA fails to converge in these two situations shown in Fig.8, the HCA is applied to compare with the GMCA in Sect.4.2. In Sect.4.3, we take another example to show the comparison of convergence history between GMCA and ACA.

4.1. Examples with single layer potential

Here, we consider the matrix **A** with entries

$$A_{ij} := \frac{1}{\|x_i - y_j\|} \quad i \in m, j \in n$$

where the vertices $x_i \in X$, $y_j \in Y$, are chosen as in the following figures.

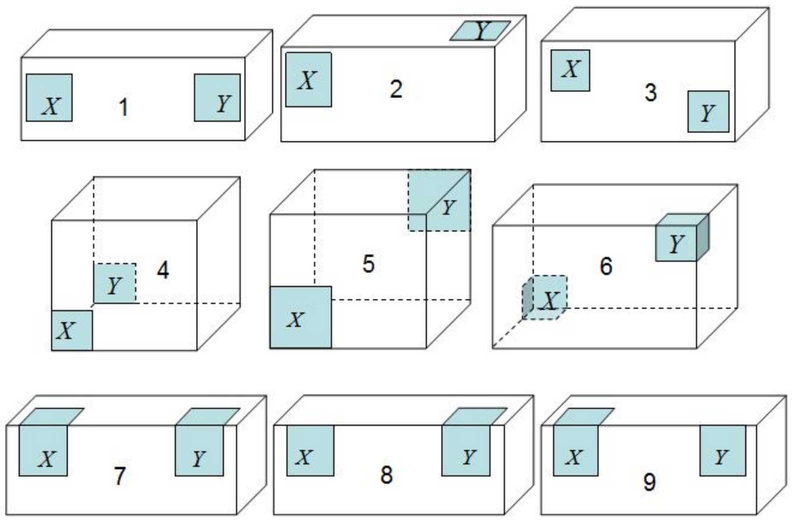


Fig. 7. The geometries for the examples with the single layer potential

Table 1. Numerical results for the examples with the comparison of the GMCA and ACA

Example	m	n	GMCA		ACA	
			k	Accuracy	k	Accuracy
1	1600	1600	3	3.864×10^{-4}	4	1.073×10^{-5}
2	3600	3600	5	1.617×10^{-5}	5	1.266×10^{-5}
3	1600	1600	4	6.52×10^{-4}	4	8.821×10^{-5}
4	3600	3600	4	4.895×10^{-6}	4	2.026×10^{-5}
5	3600	3600	3	2.803×10^{-6}	3	3.869×10^{-5}
6	2700	2025	4	6.658×10^{-4}	4	7.309×10^{-5}
7	3200	3200	5	1.181×10^{-4}	6	2.418×10^{-5}
8	3200	1600	4	3.747×10^{-4}	4	1.042×10^{-4}
9	1600	3200	5	1.168×10^{-5}	5	4.348×10^{-5}

The numerical results for the examples shown in Fig.7 are listed in Table 1. In the Table, m and n denote the numbers of the vertices in the domains X and Y, respectively. The approximation accuracy and the corresponding rank k obtained by the GMCA and the ACA are listed in column 4, 5, 6, 7, respectively. It is seen that the accuracy for some examples of the new algorithm is somewhat worse than that of the ACA. However, the necessary ranks for them are less and the storage requirement can be further reduced.

4.2. Examples with single layer potential gradient

In this section, we consider the matrix **A** with entries

$$A_{ij} := \frac{\langle n(x_i), x_i - y_i \rangle}{\| x_i - y_j \|} \quad i \in m, j \in n$$

where the vertices $x_i \in X \cup X^*$, $y_j \in Y \cup Y^*$, are chosen as in the following figures , $n(x_i)$ is the outer unit normal vector to $X \cup X^*$. When X and Y lie in the same plane, the outer unit normal $n(x_i)$ and the vector drown from y_i to x_i are perpendicular. Thus, all entries A_{ij} with $x_i \in X$ and $y_j \in Y$ are zero.

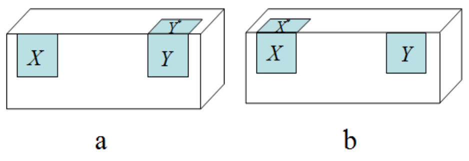


Fig. 8. The geometries for the examples with single layer potential gradient

Table 2. Numerical results for the examples with the comparison of the GMCA and HCA

Example	m	n	GMCA		HCA	
			k	Accuracy	k	Accuracy
a	400	800	2	3.137×10^{-4}	4	4.606×10^{-6}
b	800	400	2	2.448×10^{-5}	4	7.943×10^{-6}

The numerical results for the examples with the single layer potential gradient are listed in Table 2. In these situations, the original ACA fails to converge[12]. Therefore, we apply the HCA and the GMCA for comparison. The approximation accuracy and the corresponding rank k obtained by the GMCA and the HCA are listed in column 4, 5, 6, 7, respectively. The comparative accuracy for the examples is similar to that presented in Sect.4.1. As we can see, however, if we increase the rank of the approximation by the GMCA, the equivalent accuracy to the HCA can be achieved.

4.3. The comparison of convergence history between GMCA and ACA

In this section, we show the comparison of convergence history between GMCA and ACA by approximating matrix $A \in R^{M \times N}$ resulting from the single layer potential operator (cf.Sec.4.1) between two point-sets which hold the strong admissibility condition (cf.Fig.9).

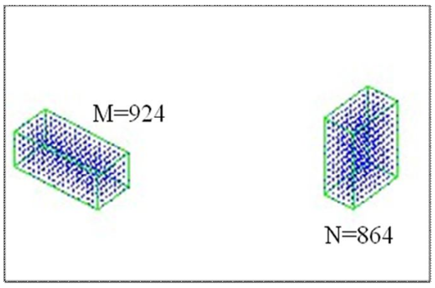


Fig. 9. The two point-sets for matrix $A \in R^{M \times N}$

In Fig.10 we compare the quality of the approximation obtained by GMCA and ACA to show the convergence history of both methods. Here, we show the approximation results of GMCA by choosing the crosses depending on the priority order of the skeleton points. With the rank of approximation increasing, the approximation accuracies obtained by GMCA and ACA are improved.

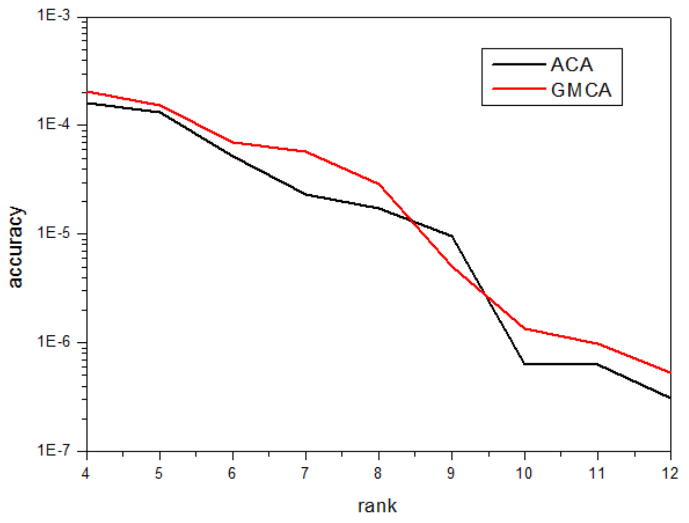


Fig. 10. Comparison of the convergence history between GMCA and ACA

In Fig.11 we show the quality of the approximation obtained by GMCA and ACA, with the number of the matrix entries ($M \times N$) increasing. Here, we just increase the number of points in the two point-sets(cf.Fig.9). With the $M \times N$ increasing, the approximation accuracy obtained by GMCA is somewhat better than that of ACA.

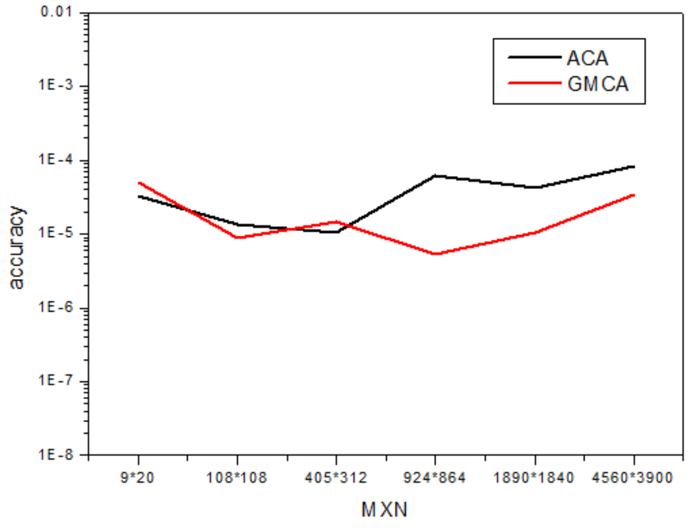


Fig. 11. Comparison of the quality of approximation between GMCA and ACA with $M \times N$ increasing

5. CONCLUSIONS AND FUTURE WORK

A new non-iterative low rank approximation algorithm is proposed in this paper. This algorithm determines the skeleton points based on the geometric mapping which just needs some sample geometry transformations. As it has been detailed, we can obtain the skeleton points before assembling the low rank approximation. The procedure of determining the skeleton points and the integral operation of the kernel function are separated. In contrast with our method, the original ACA performs the integral operations which correspond to the entries of the selected column or row in the matrices of BEM in each iterative process when the higher order elements are used. This is why so far most fast BEM implementations are based on constant elements. Therefore, we try to determine the skeleton points by the geometric mapping in this paper.

Our method can determine the skeleton points in the admissible interacting blocks in advance, and the boundary integrals which correspond to the skeleton points in the far-field can be computed together with those for the near-field interaction. Integrals for both near-field and far-field interactions are computed in a unified framework in the same way as the traditional BEM. It will be convenient and effective to use higher order elements in the fast BEM. A nested bases approximation can also benefited from our algorithm [19]. Moreover, it may be more effective when we implement the parallel computation based on the GPUs.

The original ACA fails to converge in some situations where the ACA needs to be modified [12]. However, the new algorithm proposed in this paper can still get the low rank approximations in these situations.

Numerical examples have shown that the new algorithm can obtain the low rank approximation but with somewhat lower accuracy compared with the ACA. Concerning its advantages stated above, however, our method deserves further consideration.

ACKNOWLEDGEMENTS

This work was supported in part by National Science Foundation of China under grant numbers 11172098 and in part by National 973 Project of China under grant number 2010CB328005.

REFERENCES

[1] W. Hackbusch, Z.P. Nowak. On the fast matrix multiplication in the boundary element method by panel clustering. Numer. Math., 1989; 54:463–491.

[2] S. Sauter. Variable order panel clustering. Computing., 2000; 64:223-261.

[3] V. Rokhlin. Rapid solution of integral equations of classical potential theory. J. Comput. Phys., 1985; 60:187–207.

[4] L. Greengard, V. Rokhlin. A new version of the fast multipole method for the Laplace in three dimensions. Acta Numer., 1997 ; 6:229–269.

[5] J.M. Zhang, Masa. Tanaka, M. Endo. The hybrid boundary node method accelerated by fast multipole method for 3-D potential problems. Int. J. Num. Meth. Eng., 2005; 63:660-680.

[6] J.M. Zhang, Masa. Tanaka. Adaptive spatial decomposition in fast multipole method. J. Comput. Phys., 2007; 226:17-28.

[7] J.M. Zhang, Masa. Tanaka. Systematic study of thermal properties of CNT composites by a fast multipole hybrid boundary node method. Eng. Anal. Bound. Elem., 2007; 31:388-401.

[8] J.M. Zhang, C. Zhuang, X.Y. Qin, G.Y. Li, X.M. Sheng. FMM-accelerated hybrid boundary node method for multi-domain problems, Eng. Anal. Bound. Elem., 2010; 34:433-439.

[9] M. Bebendorf. Approximation of boundary element matrices. Numer. Math., 2000; 86: 565–589.

[10] M. Bebendorf, S. Rjasanov. Adaptive Low-Rank Approximation of Collocation Matrices. Computing., 2003; 70:1–24.

[11] E. Tyrtyshnikov. Incomplete cross approximation in the mosaic-skeleton method. Computing., 2000; 64:367–380.

[12] S. Börm, L. Grasedyck. Hybrid cross approximation of integral operators. Numer. Math., 2005; 101:221–249.

[13] W. Dahmen, R. Schneider. Wavelets on manifolds I: Construction and domain decomposition. SIAM J. Math. Anal., 1999; 31:184–230.

[14] M. Bebendorf, R. Kriemann. Fast parallel solution of boundary integral equations and related problems. Comp. Vis. Sci., 2005; 8:121–135.

[15] M. Bebendorf. Hierarchical LU decomposition-based preconditioners for BEM. Computing., 2005; 74:225–247.

[16] P.G. Martinsson, V. Rokhlin, M. Tygert. A randomized algorithm for the decomposition of matrices. Appl. Comput. Harmon. Anal., 2011; 30:47–68.

[17] M. Bebendorf. Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems Lecture Notes in Computational Science and Engineering (LNCSE), vol. 63. Springer, Berlin (2008). ISBN: 978-3-540-77146-3

[18] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. A theory of pseudoskeleton approximations. Linear Algebra Appl., 1997; 261:1–21.

[19] M. Bebendorf, R. Venn. Constructing nested bases approximations from the entries of non-local operators. Numer. Math., 2012; 121(4):609-635.

List of Tables

List of Figures