ISSN 1995-0802, Lobachevskii Journal of Mathematics, 2019, Vol. 40, No. 11, pp. 1771–1780. © Pleiades Publishing, Ltd., 2019.

Low Rank Methods of Approximation in an Electromagnetic Problem

A. A. Aparinov[1*], A. V. Setukha[2], and S. L. Stavtsev[3***]**

(Submitted by E. E. Tyrtyshnikov)

[1] *Central Aerohydrodynamic Institute (TsAGI), Zhukovsky, Moscow oblast, 140180 Russia*

[2] *Lomonosov Moscow State University, Moscow, 119991 Russia*

[3] *Marchuk Institute of Numerical Mathematics, Russian Academy of Science, Moscow, 119333 Russia*

Received May 20, 2019; revised June 26, 2019; accepted July 11, 2019

Abstract—In this article authors present a new method to construct low-rank approximations of dense huge-size matrices. The method develops mosaic-skeleton method and belongs to kernel-independent methods. In distinction from a mosaic-skeleton method, the new one utilizes the hierarchical structure of matrix not only to define matrix block structure but also to calculate factors of low-rank matrix representation. The new method was applied to numerical calculation of boundary integral equations that appear from 3D problem of scattering monochromatic electromagnetic wave by ideal-conducting bodies. The solution of model problem is presented as an example of method evaluation.

DOI: 10.1134/S1995080219110064

Keywords and phrases: *parallel algorithms, fast matrix algorithms, integral equations, scattering problems.*

1. INTRODUCTION

Methods of boundary integral equations are widespread use in problems of electromagnetic wave scattering. Significant efficiency these methods show in external problems because they require to build grid only on surfaces of irradiating bodies, so there is no problem of expanding of the grid to fulfill conditions on infinity. Nevertheless, in problems with complex geometry or with huge electrodynamic size (the relation of body size to wavelength is big) one have to solve large-scale systems of linear equations, moreover, matrices of these systems are dense. So for the solution of such kind of problems special systems with distributed memory are used as well as numerical methods based on approximations of dense matrices.

Different methods are known to approximate matrices growing while discretization of integral equations. The most widespread are multipole methods [1, 2]. Their parameters can be tuned according to problem type. Other methods for approximating large dense matrices utilize pure algebraic approaches. These methods use function of matrix-element calculation like black-box, so that is why they called kernel-independent methods.For example, method of mosaic-skeleton [3–5] approximations is kernel-independent method, as well as method utilising $\mathcal{H}^2$ matrices [6–8]. All methods of matrix approximations allow to construct approximations and to multiply matrix of size $N \times N$ by vector of size N in $\mathcal{O}(N\log(N))$ or even $\mathcal{O}(N)$ operations instead of $\mathcal{O}(N^2)$ operations.

In this article authors present a new method to construct approximations of a large dense matrix. The presented method is the evolution of mosaic-skeleton method, and so it is kernel-independent. In

*E-mail: andrey.aparinov@gmail.com

**E-mail: setuhaav@rambler.ru

***E-mail: sstass2000@mail.ru

distinction from a mosaic-skeleton method, the new one utilizes a hierarchical structure of matrix not only to define matrix block structure but also to calculate factors of low-rank matrix representation.

The method was applied to an approximate matrix that appears in the problem of scattering electromagnetic wave by the ideal-conducting body with complex geometry. To solve the diffraction problem with large wave number by the method of integral equations, one should solve the system of linear equations with a large dense matrix. This task needed utilization of low-rank approximations methods and supercomputer calculations as well. In distinction with [7] in this method, low-rank approximations were calculated without iterations. New non-iterative parallel algorithm of matrix approximation is better scalable than an iterative algorithm of [7].

This article is organized as follows: the electrodynamic problem is described in Section 2, Section 3 includes an overview of the mosaic-skeleton method, Section 4 presents the main ideas of a new method, Section 5 shows valuable details of a new method, that makes it robust, and in Section 6 the calculation examples are presented.

2. ELECTRODYNAMICS PROBLEM

Let us consider the problem of monochromatic electromagnetic wave scattering by a perfectly conducting surface Σ, which can be either closed or open.

A monochrome wave with a frequency ω satisfies the Maxwell equations,

$$\nabla \times \vec{E} = i\mu\mu_0\omega\vec{H}; \nabla \times \vec{H} = -i\varepsilon\varepsilon_0\omega\vec{E}.$$

On a perfectly conducting surface the following boundary condition holds,

$$\vec{n} \times (\vec{E}_0 + \vec{E}) = 0,$$

where $\vec{E}_0$ is a given function, defined by the incident wave (we assume that the incident wave is planar), and $\vec{n}$ is a normal vector to the surface.

To find a unique solution it is necessary to pose additional conditions $\vec{E} \in L_2^{\mathrm{loc}}(\Omega)$ and

$$\frac{d}{d\tau}\begin{pmatrix}\vec{E}\\ \vec{H}\end{pmatrix} - ik\begin{pmatrix}\vec{E}\\ \vec{H}\end{pmatrix} = o\left(\frac{1}{|\vec{x}|}\right),\ \tau = |\vec{x}|,\ \tau \to \infty.$$

The problem can be reduced to the electric field integral equation in the unknown tangential vector field $\vec{j}(y)$ to the surface Σ:

$$\vec{n} \times \iint\limits_{\Sigma} \vec{j}(y)\Big(\text{grad div}F(x-y) + k^2F(x-y)\Big)d\sigma_y = -\vec{n} \times \vec{E}_0(x),\ x \in \Sigma, \tag{1}$$

where $k = \omega\sqrt{\varepsilon\varepsilon_0\mu\mu_0}$ is the wave number, and

$$F(R) = \frac{\exp{(ikR)}}{R},\ R = |x-y|.$$

In equation (1) the integral can be understood in the sense of the Hadamard finite part.

For the numerical solution of the equation (1) we use a numerical scheme described in [8]. In this scheme the surface is uniformly divided into cells σ_i, $i = \overline{1,n}$, and for each cell an orthonormal basis $\vec{e}_{i1}$, $\vec{e}_{i2}$ is introduced. For each cell σ_i it is assumed that $\vec{j}_i = \vec{j}(x_i)$, where x_i is the center of mass of the cell. Each cell is considered to be planar. Discretization of the integral operator produces a matrix that consists of 2×2 blocks:

$$A_{ij} = \begin{pmatrix}\vec{E}_{1j}(x_i)\cdot\vec{e}_{1i} & \vec{E}_{2j}(x_i)\cdot\vec{e}_{1i}\\ \vec{E}_{1j}(x_i)\cdot\vec{e}_{2i} & \vec{E}_{2j}(x_i)\cdot\vec{e}_{2i}\end{pmatrix},$$

$$\vec{E}_{1j}(x_i) = \int\limits_{\partial\sigma_j}\vec{Q}(x_i)(\vec{s},\vec{e}_2)dl + k^2\vec{e}_{1j}\int\limits_{\sigma_j}\frac{\exp{(ikR)}}{R}d\sigma;$$

$$\vec{E}_{2j}(x_i) = -\int\limits_{\partial\sigma_j} \vec{Q}(x_i)(\vec{s},\vec{e}_1)dl + k^2\vec{e}_{2j}\int\limits_{\sigma_j} \frac{\exp{(ikR)}}{R}d\sigma, \vec{Q}(x) = \nabla_y \frac{\exp{(ik|x-y|)}}{|x-y|}, \tag{2}$$

where $\vec{s}$ is a unit tangent vector to a curve. In (2) the contour and surface integrals are calculated numerically.

A variation of the algorithm for calculating blocks A_{ij} of matrix A is described in details in papers [9, 10].

3. MOSAIC-SKELETON METHOD

Let's apply mosaic-skeleton approximations method [3] to large dense matrix $\bar{A}$ (2) of size N. This method is based on a hierarchical decomposition of the matrix to blocks. If the block corresponds to separated domains, then we can construct low-rank approximation for it with algorithm 1 (incomplete low-rank approximation).

Algorithm 1

The building of low-rank approximation of block $\tilde{A}$ of matrix A with predefined precision.

On entrance: Matrix element calculation function $A = (a_{ij})$, size of block $m \times n$, precision ε. On exit: factors $U = (u_{ik})$ and $V = (v_{jk})$ in format $m \times r$ and $n \times r$ respectively with the property that matrix $\tilde{A} = UV^T$ is approximation of matrix A minimizing expression $||A - \tilde{A}||_F$, where $|||\dot{|}|_F$ is some matrix norm (usually Frobenius norm) (r is rank of created approximation).

1. $p := 1$; $j_1 := \lceil n/2 \rceil$; $\tilde{A} := 0$ initialization: p is iteration index; j_1 is initial column;

2. REPEAT;

3. $u_{ip} := a_{ij_p} - \sum\limits_{k=1}^{p-1} u_{ik}v_{j_pk}, i = 1, \ldots, m$ (calculating matrix column and basing on it recurrent vector $u_p = (u_{ip})$ in U);

4. $i_p = \max\limits_i u_{ip}$ (defining recurrent row index i_p);

5. $v_{ip} := a_{i_pj} - \sum\limits_{k=1}^{p-1} u_{i_pk}v_{jk}, i = 1, \ldots, n$ (calculating matrix row and basing on it recurrent vector $v_p = v_{ip}$ in V);

6. $j_{p+1} = \max\limits_i v_{ip}$ (defining column index j_{p-1} for next iteration);

7. $U_p = \begin{pmatrix} U_{p-1} & u_p \end{pmatrix}$; $V_p = \begin{pmatrix} V_{p-1} & v_p \end{pmatrix}$;

8. $p := p + 1$;

9. UNTIL $\left(\sqrt{\min\{m,n\} - (p-1)}\,\left(\left|\left|U_pV_p^T\right|\right|_F - \left|\left|U_{p-1}V_{p-1}^T\right|\right|_F\right) \geq \varepsilon\left|\left|U_pV_p^T\right|\right|_F\right)$ AND$((p + r_0) < \min\{m,n\})$;

10. RETURN $U := U_p$; $V := V_p$.

Algorithm 1 is closely connected with skeleton decomposition [3, 11, 12] of matrix $A = UV^T$. Skeleton decomposition is one of the main decompositions in linear algebra and it is based on choosing linear-independent rows and columns of matrix. Skeleton decomposition for matrix $A \in \mathbb{C}^{n\times m}$, rank$(A) = r$ looks like

$$A = \tilde{C}\hat{A}^{-1}\tilde{R}, \tag{3}$$

where matrix $\tilde{C} \in \mathbb{C}^{n\times r}$ consists of basis columns of matrix A, $\tilde{R} \in \mathbb{C}^{r\times m}$—of basis rows, and $\hat{A}^{-1}$—submatix, consisting of elements placed on interception of chosen rows and columns. In Algorithm 1 besides decomposition factors U and V the indexes of rows $I = \{i_p\}$, $p = 1, \ldots, r$, and columns $J = \{j_p\}$, $p = 1, \ldots, r$, of skeleton decomposition (3) are calculated. According to these index sets with help of function for calculating matrix element one can calculate matrices $\tilde{C} \in \mathbb{C}^{n\times r}$, $\tilde{R} \in \mathbb{C}^{r\times m}$ and $\hat{A}^{-1} \in \mathbb{C}^{r\times r}$ of decomposition (3).

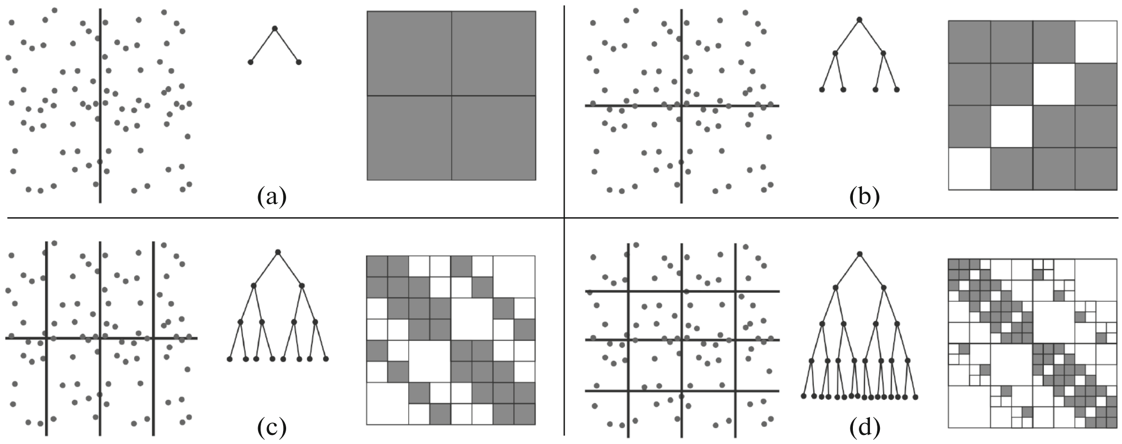

Fig. 1. Cluster tree and block matrix structure: (a) level 1; (b) level 2; (c) level 3; (d) level 4.

Basing on skeleton decomposition (3) Algorithm 1 is interpreted as interpolation algorithm that thins out the set of row and columns of matrix A, in other words find sets consisting of rows $I = \{i_p\}$, $p = 1, \ldots, r$, and columns $J = \{j_p\}$, $p = 1, \ldots, r$, that are enough to approximate the whole matrix with formula (3). Further we will use formula (3) in following representation

$$A = C\hat{A}R, \tag{4}$$

where $C = \tilde{C}\hat{A}^{-1}$, $R = \hat{A}^{-1}\tilde{R}$.

Reliability if approximation (4) calculation increases if algorithm maxvol from [13] is applied after Algorithm 1.

The description of the parallel method of mosaic-skeleton approximations is published in work [14].

4. HIERARCHICAL BASIS

Let's construct matrix approximation utilizing hierarchical decomposition of a matrix to calculate factors of low-rank approximation. Let consider that low-rank approximation for low-level blocks uses a low-rank approximation of higher-level blocks.

Let's see the example of approximation construction following one special case of matrix decomposition to blocks. Fig. 1 represents the correspondence of domain with grid cells centers to cluster tree and matrix decomposition to blocks. In the example in Fig. 1, there are 4 levels of decomposition, grey color marks dense blocks, data from other blocks are compressed in low-rank matrix approximation. Figure 2 shows decomposition of matrix on level 3. Basing on this block structure let's represent the method of low-rank matrix decomposition calculation with hierarchical basis.

Let $A_0 \in \mathbb{C}^{N\times N}$ is matrix, consisting of elements of dense blocks of matrix $\bar{A}$, and other elements of A_0 are equal to zero. According to figure 2 matrix A_0 consists of small number of non-zero elements, that means A_0 is sparse.

Let's construct multipliers of low-rank representation of 3rd level. To get row multipliers we divide matrix so as it is shown in figure 3 (a) (all low-rank block-rows united into one block and consists of all row elements except blocks of matrix A_0). According to Fig. 1 row blocks include data from low-rank blocks constructed on levels equal or higher than 3. For each row block with the help of Algorithm 1 and formula (4) we calculate matrices C, which we will represent as $L_{1\,p} \in \mathbb{C}^{n_p\times r_{1p}}$, $p = 1, \ldots, 8$, where r_{1p} is rank of low-rank decomposition in Algorithm 1. Matrix

$$L_1 = \mathrm{diag}(L_{1\,1}, \ldots, L_{1\,8}), \tag{5}$$

we will name row basis of 3 level. Value $r_1 = \sum_{i=1}^{8} t_{1i}$ we will call row rank of 3 level.

Using the same approach we construct the column blocks (see Fig. 3b), where with Algorithm 1 and formula (4) matrices R are calculated. These matrices, that we'll denote as $R_{1\,p} \in \mathbb{C}^{t_{1p}\times m_p}$, $p = 1, \ldots, 8$

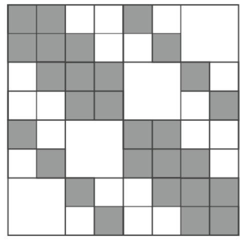

Fig. 2. Block matrix structure on the level 3.

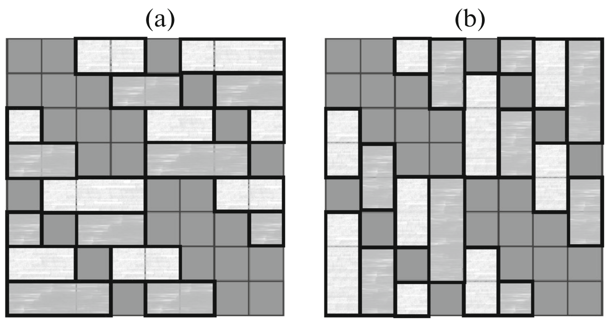

Fig. 3. United blocks of matrix A for constructing row (a) and column (b) basises.

(t_{1p} is rank of low-rank decomposition of column-block p in Algorithm 1), are members of column basis of 3rd level,

$$R_1 = \mathrm{diag}(R_{1\,1}, \ldots, R_{1\,8}). \tag{6}$$

Value $q_1 = \sum\limits_{i=1}^{8} r_{1i}$ we will call column rank of 3rd level.

Let us denote $\tilde{A}_1 \in \mathbb{C}^{r_1 \times q_1}$—matrix, consisting of matrices $\hat{A}$ from formula (4) for low-rank blocks, constructed on levels not lower than 3. In Fig. 4 (a) matrices $\hat{A}$ are shown with grey squares in low-rank blocks. According the our construction process elements of matrix $\hat{A}$ are defined by sets of indices of rows $I = \{i_p\}$ when constructing matrix R_1 and columns $J = \{j_p\}$ when constructing matrix L_1 with Algorithm 1. So Algorithm 1 allowed us to approximate matrix A with row and column basises of 3rd level

$$A = A_0 + L_1 \tilde{A}_1 R_1. \tag{7}$$

Let's mark with $A_1 \in \mathbb{C}^{r_1 \times q_1}$ matrix, which holds elements of matrix $\tilde{A}_1$ of low-rank blocks of 3rd level and other elements are equal to zero (see figure 4 (b)). In matrix $\tilde{A}_1 - A_1$ we consolidate blocks of third and second levels. Then using Algorithm 1 for matrix $\tilde{A}_1 - A_1$ we construct row basis of second level L_2 and column basis of second level R_2:

$$\tilde{A}_1 = A_1 + L_2 \tilde{A}_2 R_2, \tag{8}$$

following the same way as for matrices L_1 and R_1 from (7).

Matrix $\tilde{A}_2 \in \mathbb{C}^{r_2 \times q_2}$ is analogical to matrix $\tilde{A}_1$ from (7) (see Fig. 5). As $\tilde{A}_2$ holds data of all low-rank blocks of 2nd level lets denote it $A_2 = \tilde{A}_2$ according to (7) and (8) low-rank representation of A takes form

$$A = A_0 + L_1 \Big(A_1 + L_2 A_2 R_2 \Big) R_1. \tag{9}$$

Figure 6 shows scheme of representation of matrices from (9).

Now then, in approximation (9) matrices L_k and R_k, $k = 1, 2$, define basises and matrices A_k, $k = 1, 2$, contains coordinates of initial data in hierarchical basises.

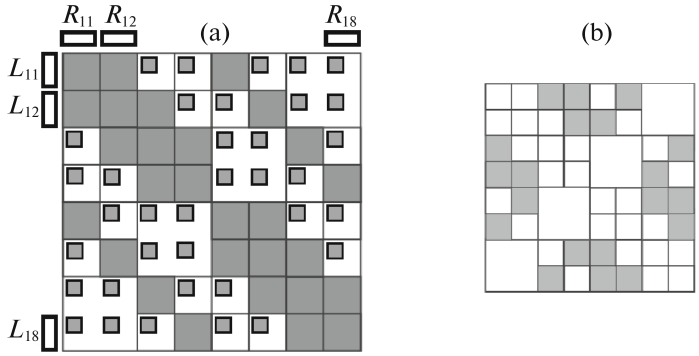

Fig. 4. (a) Blocks of matrices L_1, R_1 and $\tilde{A}_1$; (b) blocks of matrix A_1.

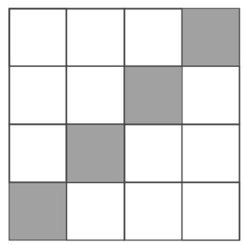

Fig. 5. Matrix structure $\tilde{A}_2$.

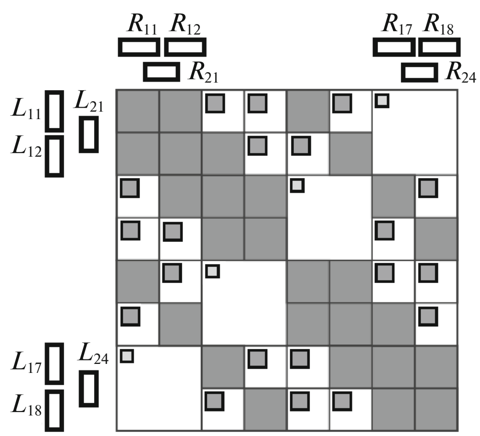

Fig. 6. Blocks of matrices A_0, L_1, R_1 A_1, L_2, R_2, A_2.

What is the advantage of this low-rank matrix representation of A to mosaic-skeleton representation? Blocks of matrices L_1, R_1 are common for both low-rank matrices A of level 3 and level 2, so for low-rank blocks of level 2 we can keep in memory $\mathcal{O}(r^2)$ elements instead of $\mathcal{O}(nr)$.

5. BASISES CALCULATION

To calculate matrices L_k, R_k, $k = 1, \ldots,$ first using Algorithm 1 we calculate sets of indices of basis rows $I = \{i_p\}$ and basis columns $J = \{j_p\}$. Then with formula (4) we calculate matrices L_k, R_k, $k = 1, \ldots$.

If we try to apply Algorithm 1 to row and column blocks represented in Fig. 3 it requires the calculation of long rows and columns and sequentially to huge expenses in memory and processor time. To fasten the calculation of indices sets $I = \{i_p\}$ and $J = \{j_p\}$ it is better first to apply mosaic-skeleton approximations to each low-rank block to define temporal sets of indices $\tilde{I} = \{i_p\}$ and $\tilde{J} = \{j_p\}$.

Table 1. Memory required to store full and compressed matrices (in megabytes)

N	8448	34 122	94 872	167 676	376 800
Full	544	8883	68 670	214 502	1 083 208
MS	190	979	3304	6160	15 210
HB	124	409	756	1390	4330

In mosaic-skeleton method only indices of driving rows and columns are calculated but not factors themselves. Then we apply Algorithm 1 only to row and column blocks shown in Fig. 3, but for high dimensionality we look for driving indices only among calculated earlier indices $\tilde{I} = \{i_p\}$ and $\tilde{J} = \{j_p\}$.

A parallel algorithm for calculating matrices L_k, R_k is based on the distribution of row and column blocks of the minimal level to processors. In the example in Fig. 1, minimal level where low-rank blocks appear is 2. The number of low-rank blocks is equal to 4, so a parallel algorithm for calculating matrices L_k, R_k easily scales to the number of processes less or equal than 4. If the number of available processors is more than the number of blocks on a minimal level, then we increase the minimal level basing on the number of processors. For example, if we have 8 processors, the minimal level, where we construct low-rank and dense blocks, we choose 3.

The matrices L_k, R_k, A_k, $k = 1, \dots$, are calculated by Algorithm 2.

Algorithm 2

1. Matrix is divided into blocks with mosaic-skeleton method, blocks are spreaded to processors, for each block leading rows and columns are calculated;

2. Blocks data including sets of leading rows and columns are copying to all processors;

3. According to algorithm described in this section row and column blocks of the minimal level are spreaded to processors. If the number of available processors is more than the number of blocks on a minimal level, then we increase the minimal level basing on the number of processors;

4. Independently on each processor basing on leading rows and columns that were calculated on step 1 we calculate with Algorithms 1, maxvol and formula (4) matrix blocks L_1 (R_1) of matrix C (R) from (4) and find sets of basis rows (columns) to calculate blocks of matrices L_2 (R_2);

5. Applying Algorithm 1, algorithm maxvol and formula (4) to submatrix which rows (columns) are defined by set of basis rows (columns) found on step 4 and colums (rows) are defined with set of leading columns (rows), found on step 1, we calculate matrix blocks L_2 (R_2) and find sets of basis rows (columns) to calculate blocks of matrices L_3 (R_3);

6. Repeat step 5 to calculate blocks of matrix L_k (R_k) basing on matrix L_{k-1} (R_{k-1}) data;

7. Send found sets of basis rows and columns between processors and basing on them calculate blocks of matrices A_k, $k = 1, \dots$.

Note that constructed algorithm include mosaic-skeleton algorithm (step 1).

6. NUMERICAL EXAMPLES

In this section we apply the presented method to the electrodynamic problem. For example, let's consider plane-wave scattering by a perfectly conducting round cylinder with diameter 15 cm and height 25 cm (see Fig. 7).

In Table 1 N is matrix size, in row Full the memory required for full matrix storage is shown, in row MS is memory required for storage in mosaic-skeleton format, in HB is memory required for matrix storage in hierarchical basis format. Memory expressed in megabytes. Irradiation frequency is equal to 8 GHz. Approximation accuracy is 10^{-2}.

For the cylinder with 125594 cells and frequency 8 GHz and approximation accuracy 10^{-2} the results for scalability are shown in Table 2. In table n_p is number of processors. In Table 2 are shown coefficients of acceleration for calculation of matrix in mosaic-skeleton format and hierarchical basis format for the different number of processors. The calculations were done on Intel Xeon E5-2670v3 2.30 GHz

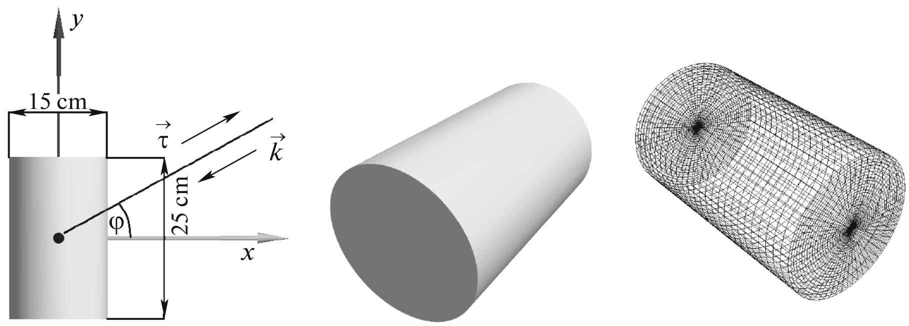

Fig. 7. Geometry and grid of the object.

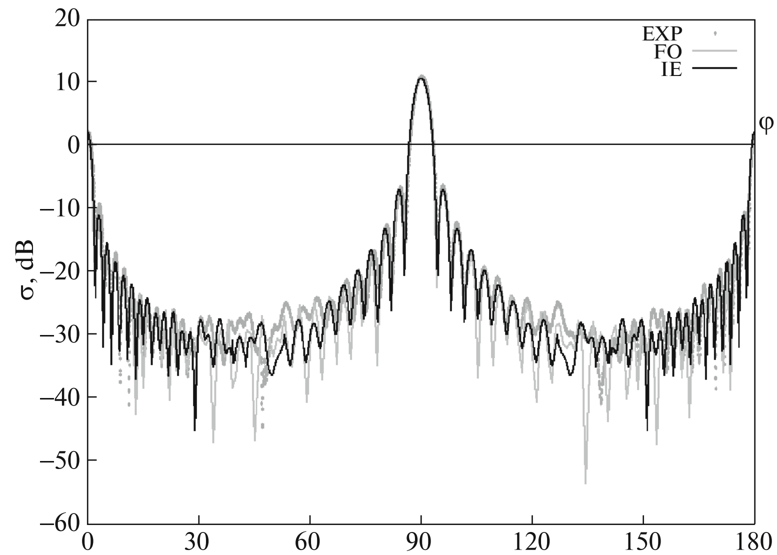

Fig. 8. RCS, 16 GHz, vertical polarization.

processors of INM RAS cluster (http://cluster2.inm.ras.ru/). We used Intel Fortran Compiler 9.0 for Linux (9.0.033). The results show that the scalability of HB algorithm is lower.

The scalability of algorithms for HB is lower than of MS algorithm. In fact, Algorithm 1 is done on the first step of Algorithm 2 and then sequential steps 2 and 7 decrease it scalability.

Unlike the example in Fig. 2 in diffraction problem the blocks of minimal level which are spreaded among processors have different size so the calculation times of matrices L_k, R_k, $k = 1, \ldots,$ on steps 3–6 of Algorithm 2 differ on different processors. This time difference in calculating steps 3–6 influences the scalability of Algorithm 2.

In diffraction problem the scalability of Algorithm 2 is not an issue because the approximation itself is times faster than linear system solution. The problem solution first of all needs huge amount of memory and the number of processors is not of that importance. According to 1 Algorithm 2 better compresses matrix so it seems to be preferable in solution of diffraction problem than known before algorithm of mosaic-skeleton approximations.

Table 2. Scalability of different algorithms for low-rank approximations

n_p	1	2	4	8	16	32	64	128
MS	1	1.91	3.61	7.24	13.38	24.51	41.31	51.84
HB	1	1.88	3.51	7.15	10.26	16.27	23.17	32.86

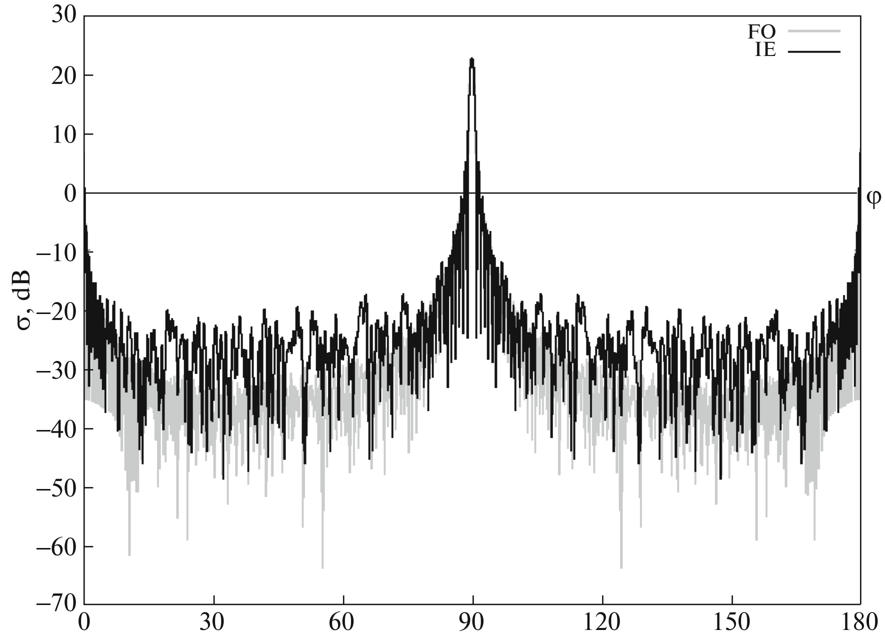

Fig. 9. RCS, 64 GHz, horizontal polarization.

In Fig. 8 backscattering RCS for frequency 16GHz is shown. Value σ for different unit vectors $\vec{\tau}$ of wave vectors (see Fig. 7) of incident wave was calculated by formula

$$\sigma(\vec{\tau}) = \frac{4\pi}{|\vec{E}_0|^2} \left| \sum_{i=1}^{n} \left(\vec{j}_i - \vec{\tau} \cdot (\vec{\tau} \cdot \vec{j}_i) \right) k^2 \exp\left(-ik\vec{\tau} \cdot x_i\right) \sigma_i \right|^2 .$$

Figure 8 shows RCSs, got with different methods: experiment (EXP), integral equations (IE), physical optics (according to [15]) (FO). All results are close to each other.

Straight solution without low-rank approximations allowed to calculate RCS on the cylinder for frequencies up to 16 GHz on INM RAS cluster. The described method of approximation allowed to calculate RCSs for frequencies up to 64 GHz (Fig. 9).

So, the presented methods better compress matrix, than known mosaic-skeleton method. The parallel implementation allowed to find RCS for high frequencies.

ACKNOWLEDGMENTS

The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University.

FUNDING

This work was supported by the Russian Science Foundation, grant no. 19-11-00338.

REFERENCES

1. R. Coifman, V. Rokhlin and S. Wanzura "The fast multipole method for the wave equation: a pedestrian prescription," IEEE Antennas Propag. Mag. **35** (3), 7–12 (1993).
2. J. M. Song and W. C. Chew, "Multilevel fast multipole algorithm for solving combined field integral equations of electromagnetic scattering," Microw. Opt. Technol. Lett. **10**, 14–19 (1995).
3. E. Tyrtyshnikov, "Mosaic-skeleton approximations," Calcolo **33**, 47–57 (1996).
4. S. L. Stavtsev, "Block LU preconditioner for the electric field integral equation," in *Proceedings of Progress in Electromagnetics Research Symposium, 2015,* pp. 1523–1527.

5. A. Aparinov, A. Setukha, and S. Stavtsev, “Supercomputer modelling of electromagnetic wave scattering with boundary integral equation method,” Commun. Comput. Inform. Sci. **793**, 325–336 (2017).
6. W. Hackbusch, *Hierarchical Matrices: Algorithms and Analysis,* Springer Series in Computational Mathematics (Springer, Berlin, Heidelberg, 2015).
7. A. Yu. Mikhalev and I. V. Oseledets, “Iterative representing set selection for nested cross approximation,” Numer. Linear Algebra Appl. **23**, 230–248 (2016).
8. S. L. Stavtsev, “$\mathcal{H}^2$ matrix and integral equation for electromagnetic scattering by a perfectly conducting object,” in *Integral Methods in Science and Engineering,* Ed. by by C. Constanda, Dalla M. Riva, P. Lamberti, and P. Musolino (Springer, Birkhauser, Cham, 2017), vol. 2, pp. 255–264. https://doi.org/10.1007/978-3-319-59387-6_25
9. E. V. Zakharov, G. V. Ryzhakov, and A. V. Setukha, “Numerical solution of 3D problems of electromagnetic wave diffraction on a system of ideally conducting surfaces by the method of hypersingular integral equations,” Difer. Equat. **50**, 1240–1251 (2014).
10. A. Setukha and S. Fetisov, “The method of relocation of boundary condition for the problem of electromagnetic wave scattering by perfectly conducting thin objects,” J. Comput. Phys. **373**, 631–647 (2018).
11. N. L. Zamarashkin and A. I. Osinsky, “New accuracy estimates for pseudoskeleton approximations of matrices,” Dokl. Math. **94**, 643–645 (2016).
12. A. I. Osinsky and N. L. Zamarashkin, “Pseudo-skeleton approximations with better accuracy estimates,” Linear Algebra Appl. **537**, 221–249 (2018).
13. S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin “How to find a good submatrix,” in *Matrix Methods: Theory, Algorithms and Applications, 2010,* pp. 247–256.
14. A. Aparinov, A. Setukha, and S. Stavtsev, “Parallel implementation for some applications of integral equations method,” Lobachevskii J. Math. **39**, 477–485 (2018).
15. F. Saez de Adana, O. Gutierrez, I. Gonzalez, M. F. Catedra, and L. Lozano, *Practical Applications of Asymptotic Techniques in Electromagnetics* (Artech House, Boston, London, 2011).