Adv Comput Math (2020) 46: 52
https://doi.org/10.1007/s10444-020-09794-y

Simple non-extensive sparsification of the hierarchical matrices

Daria A. Sushnikova[1] · **Ivan V. Oseledets**[2,3]

Received: 30 April 2019 / Accepted: 7 May 2020 /
Published online: 8 June 2020

Abstract
In this paper, we consider the matrices approximated in $\mathscr{H}^2$ format. The direct solution, as well as the preconditioning of systems with such matrices, is a challenging problem. We propose a non-extensive sparse factorization of the $\mathscr{H}^2$ matrix that allows to substitute direct $\mathscr{H}^2$ solution with the solution of the system with an equivalent sparse matrix of the same size. The sparse factorization is constructed of parameters of the $\mathscr{H}^2$ matrix. In the numerical experiments, we show the consistency of this approach in comparison with the other approximate block low-rank hierarchical solvers, such as HODLR [3], H2Lib [5], and IFMM [11].

Keywords $\mathscr{H}^2$ matrix · Sparse factorization · Preconditioning

Mathematics Subject Classification (2010) 65F05 · 15A23 · 65R20 · 65F50

1 Introduction

Problems arising in the discretization of boundary integral equations (and several other problems with approximately separable kernels) lead to matrices that can be well-approximated by hierarchical block low-rank ($\mathscr{H}$ matrices [15, 17], mosaic

Communicated by: Leslie Greengard

✉ Daria A. Sushnikova
daria.sushnikova@nyu.edu

Ivan V. Oseledets
i.oseledets@skolkovotech.ru

[1] Courant Institute of Mathematical Sciences New York University, 251 Mercer Street, New York, NY 10012-1185, USA

[2] Skolkovo Institute of Science and Technology, Nobel St. 3, Skolkovo Innovation Center, Moscow, Moscow Region, 143025, Russia

[3] Institute of Numerical Mathematics Russian Academy of Sciences, Gubkina St. 8, Moscow, 119333, Russia

skeleton [28]) matrices. These are the matrices hierarchically divided into blocks, some of which have low rank. The development of the $\mathscr{H}$ matrices is the $\mathscr{H}^2$ matrices [6, 16], which are the hierarchical block low-rank matrices with nested bases. The nested basis property leads to the additional improvement in terms of storage and complexity of matrix-vector products. Approximate solution and preconditioning of systems with $\mathscr{H}^2$ matrices is a rapidly developed area [3, 9, 11, 19, 23]; however, construction of the accurate, time-efficient and memory-efficient factorization that leads to approximate solution is still a challenging problem. In this paper, we propose a new representation of $\mathscr{H}^2$ matrices. Namely, we show that $\mathscr{H}^2$ factorization of matrix $A \in \mathbb{R}^{N\times N}$ is equivalent to the factorization

$$A = USV^\top, \tag{1}$$

where $S \in \mathbb{R}^{N\times N}$ is a sparse matrix. Note that the size of matrix S matches the size of matrix A. $U \in \mathbb{R}^{N\times N}$ and $V \in \mathbb{R}^{N\times N}$ are orthogonal matrices that are products of block-diagonal and permutation matrices. Once the factorization (1) is built, we can substitute a solution of the system

$$Ax = b,$$

by a solution of the system with the sparse matrix:

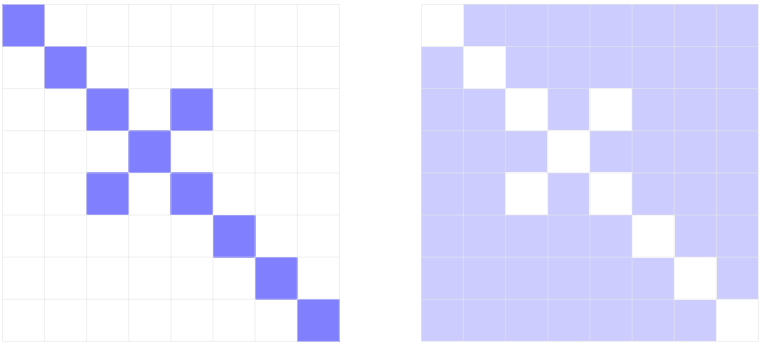
$$Sy = U^\top b, \tag{2}$$

where $x = Vy$. The system (2) can be solved using modern, well-developed sparse tools. In this paper, we propose:

- Sparse *non-extensive*[1] factorization for $\mathscr{H}^2$ matrix that leads to the solver and the preconditioner.
- The algorithm that allows to construct factors U, S and V from parameters of $\mathscr{H}^2$ matrix.
- Numerical comparison of the proposed method with HODLR [1], IFMM [11] and H2Lib [5] packages.

The main idea of the sparsification algorithm is the compression of the low-rank blocks (the very close approach of the compression of the fill-in blocks during the block Cholesky factorization of a sparse matrix is presented in works [26, 30]. Also, a similar approach is used in coupled sparsification and factorization algorithms for hierarchical matrices [9, 19, 23].) The main difference between the presented sparse factorization and the other methods of sparsification [2, 11, 27] is a size of factors. In the other methods, the hierarchical matrix is factorized into the sparse matrices of *larger sizes* [2, 27]). The matrix extension is a major drawback since it increases the complexity of matrix computations. We propose the factorization that takes $\mathscr{H}^2$ matrix and returns the sparse factors of the *same size*. Another drawback of the extended sparse factorizations is that the resulting sparse matrix may lose a

[1]The term *non-extensive* means that the sizes of the factors S, U and V are equal to the size of the matrix A (in opposition to extensive [2, 11, 27] sparse factorizations of $\mathscr{H}^2$ matrix).



(a) Matrix C_0 (b) Matrix F_0

Fig. 1 Close and far blocks of example matrix A at level $l = 0$. **a** Matrix C_0. **b** Matrix F_0

positive definiteness of the original $\mathscr{H}^2$ matrix. Proposed sparsification preserves symmetry and positive definiteness of the $\mathscr{H}^2$ matrix (see Proposition 1).

2 Compression algorithm

Consider the dense matrix $A \in \mathbb{R}^{N\times N}$ that can be approximated in $\mathscr{H}^2$ format (has corresponding low-rank blocks). Formal definition of the $\mathscr{H}^2$ matrix is presented in Section 4.1, here we give basic facts that are used in the current section.

Matrix A is a matrix with a hierarchical block division; the number of levels is L. Block size of zero level is B. On level 0, the matrix A is represented by

$$A = C_0 + F_0.$$

Matrices C_0 and F_0 (called "close" and "far") are two block matrices with the same block division that have no intersecting nonzero blocks. Matrix $C_0 \in \mathbb{R}^{N\times N}$ is block-sparse[2] matrix that consists of full-rank close blocks (for the rigorous definition of the close matrix, see Section 4.1).

Remark 1 Close block in the grid-based[3] matrix is the following: if a group of grid points corresponding to row indexes of the block is geometrically close to the points corresponding to column indexes of the block, then the block is called *close*. If the matrix is not a grid-based, then close blocks are ones close to the diagonal.

See the example of C_0 in Fig. 1a. Matrix F_0 is the matrix with approximately low-rank block rows and block columns. F_0 has additional low-rank property, that will be described during the algorithm. See the example of F_0 in Fig. 1b.

[2]The *block-sparse* matrix is a block matrix that has $M \times M$ blocks and $\mathscr{O}(M)$ of them are nonzero.
[3]The grid-based matrix is a result of the discretization of some equation on the grid.

2.1 Compression at zero level

First consider the compression procedure at zero block level ($l = 0$). Assume that the number of block rows and columns at zero level is M. Nonzero block $F_{ij} \in \mathbb{R}^{B\times B}$ of far matrix F_0 has approximate low rank:

$$F_{ij} \approx \dot{U}_i \dot{F}_{ij} \dot{V}_j^\top, \quad \forall i, j = 1, \ldots, M$$

where $\dot{F}_{ij} \in \mathbb{R}^{B\times B}$ is the compressed far block with the following structure:

$$\dot{F}_{ij} = \begin{bmatrix} D_{ij} & 0 \\ 0 & 0 \end{bmatrix},$$

where $D_{ij} \in \mathbb{R}^{r\times r}$. Matrices $\dot{U}_i \in \mathbb{R}^{B\times B}$ and $\dot{V}_j \in \mathbb{R}^{B\times B}$ are orthogonal. All blocks in ith row have the same left orthogonal compression factor $\dot{U}_i$ and all blocks in jth column have the same right factor $\dot{V}_j$.

The goal of the compression procedure is to sparsify the matrix A by obtaining the compressed blocks $\dot{F}_{ij}$ instead of original blocks F_{ij}. One can achieve this by finding $\dot{U}_i$ and $\dot{V}_j$ compression matrices and applying matrix $\dot{U}_i$ to ith row and $\dot{V}_j$ to jth column. We introduce the block-diagonal orthogonal compression matrix

$$U_0^\top = \begin{bmatrix} \dot{U}_0^\top & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \dot{U}_M^\top \end{bmatrix}, \tag{3}$$

where $U_0^\top \in \mathbb{R}^{N\times N}$. Similarly, for block columns we obtain the block-diagonal orthogonal compression matrix

$$V_0 = \begin{bmatrix} \dot{V}_0 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \dot{V}_M \end{bmatrix}. \tag{4}$$

where $V_0 \in \mathbb{R}^{N\times N}$. Applying matrices U_0 and V_0 to the matrix A we obtain the matrix A_1 with *compressed* far matrix:

$$A_1 = U_0^\top A V_0.$$

The process is illustrated in Fig. 2.

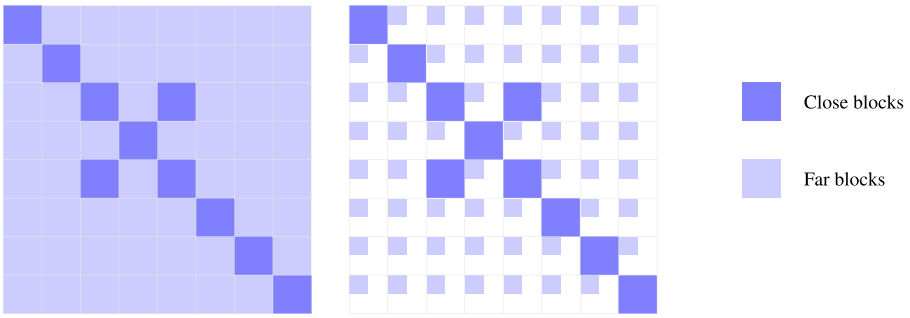
Thus, we obtain:

$$A_1 = U_0^\top (C_0 + F_0) V_0 = U_0^\top C_0 V_0 + U_0^\top F_0 V_0 \approx U_0^\top C_0 V_0 + \dot{F}_1, \tag{5}$$

where $\dot{F}_1 \in \mathbb{R}^{N\times N}$ is a compressed far matrix which consists of blocks $\dot{F}_{ij}$. Note that the matrix C_0 is available as one of the parameters of the $\mathscr{H}^2$ format, it is a so-called close matrix.

2.2 Compression at the first level ($l = 1$)

Definition 1 For each row of matrix A_1, if the corresponding row of matrix $\dot{F}_1$ is zero, then we denote it by "non-basis" row, otherwise we denote it by



(a) Matrix A (b) Matrix A_1

Fig. 2 Zero level compression. **a** Matrix A. **b** Matrix A_1

"first level basis" row. Analogously, we denote first level basis and non-basis columns (see the illustration in Fig. 3).

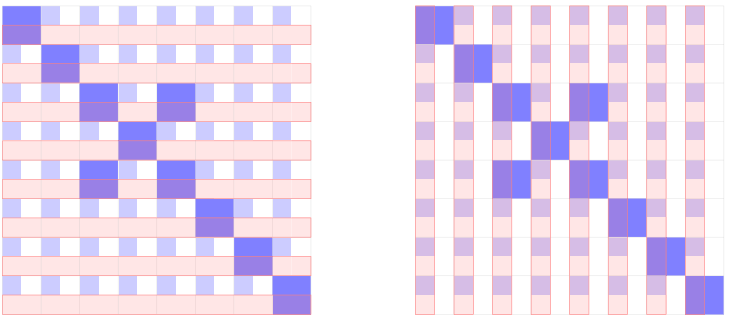
Assume that each block row (column) has r basis rows (columns) and $(B-r)$ non-basis. Introduce the permutation P_{r1} that puts non-basis block rows before the basis ones preserving the row order and permutation P_{c1} that does the same for columns, see Fig. 5a. For the permuted matrix

$$\widetilde{A}_1 = P_{r1} A_1 P_{c1} \tag{6}$$

we obtain

$$\widetilde{A}_1 = \begin{bmatrix} A_{\mathbf{n_1 n_1}} & A_{\mathbf{n_1 b_1}} \\ A_{\mathbf{b_1 n_1}} & A_{\mathbf{b_1 b_1}} \end{bmatrix},$$

where $A_{\mathbf{n_1 n_1}} \in \mathbb{R}^{M(B-r)\times M(B-r)}$ is a submatrix on the intersection of non-basis rows and non-basis columns, $A_{\mathbf{b_1 n_1}} \in \mathbb{R}^{Mr\times M(B-r)}$ is on the intersection of basis rows and non-basis columns and so on (see Fig. 5a).



(a) Non-basis rows of matrix A_1 (b) First level basis columns of matrix A_1

Fig. 3 Basis and non-basis rows and columns of matrix A_1. **a** Non-basis rows of matrix A_1. **b** First level basis columns of matrix A_1

Permutations P_{r1} and P_{c1} concentrate all nonzero blocks of compressed far matrix $\dot{F}_1$ inside the submatrix $A_{\mathbf{b_1 b_1}}$.

Remark 2 Note that the matrix $A_{\mathbf{b_1 b_1}} \in \mathbb{R}^{Mr \times Mr}$ has the same close and far block structure as the matrix A, but the block size in $A_{\mathbf{b_1 b_1}}$ is r, compare Figs. 2a and 4a.

Now we join block rows and columns of the matrix $A_{\mathbf{b_1 b_1}}$ by groups of J blocks (e.g. $J = 2$ in Fig. 4a). Assume that $Jr = B$.

Definition 2 We will call the grouped blocks "big blocks" (see Fig. 4a). Among these blocks, the big block that consists only of far sub-blocks will be called far (light blue blocks in Fig. 4b), the big block that contains at least one close small block will be referred to as close (dark blue plus green blocks in Fig. 4b).

Definition 3 Define operator $(\cdot)_{\mathbf{b_1 b_1}}$ that takes a matrix and returns its submatrix on basis rows and basis columns.

Note that $(A_1)_{\mathbf{b_1 b_1}} = A_{\mathbf{b_1 b_1}}$. Denote blocks of the far matrix $(\dot{F}_1)_{\mathbf{b_1 b_1}}$ that become close after grouping by $\widehat{F}_{\mathbf{ml1}} \in \mathbb{R}^{Mr \times Mr}$ (green blocks in Fig. 4b), and the remaining blocks by $\widehat{F}_1 \in \mathbb{R}^{Mr \times Mr}$.

$$(\dot{F}_1)_{\mathbf{b_1 b_1}} = \widehat{F}_{\mathbf{ml1}} + \widehat{F}_1$$

Remark 3 Note that the matrix $\widehat{F}_{\mathbf{ml1}}$ is close in meaning to the first level interaction list in the corresponding $\mathscr{H}^2$ matrix; or to the first level m2l interaction in the FMM [13, 14].

Denote new close matrix with big blocks by

$$\widehat{C}_1 = (U_0^\top C_0 V_0)_{\mathbf{b_1 b_1}} + \widehat{F}_{\mathbf{ml1}}. \tag{7}$$

where $\widehat{C}_1 \in \mathbb{R}^{Mr \times Mr}$.

The joining procedure for the block $A_{\mathbf{b_1 b_1}}$ is

$$A_{\mathbf{b_1 b_1}} = (U_0^\top C_0 V_0)_{\mathbf{b_1 b_1}} + (\dot{F}_1)_{\mathbf{b_1 b_1}} = (U_0^\top C_0 V_0)_{\mathbf{b_1 b_1}} + \widehat{F}_{\mathbf{ml1}} + \widehat{F}_1 = \widehat{C}_1 + \widehat{F}_1.$$

We assume that block rows and columns of the matrix $\widehat{F}_1$ have low rank. For matrix $\widehat{F}_1$ we compute orthogonal block-diagonal compression matrices $U_{\mathbf{b_1}}, V_{\mathbf{b_1}} \in \mathbb{R}^{Mr \times Mr}$, similarly to (3).

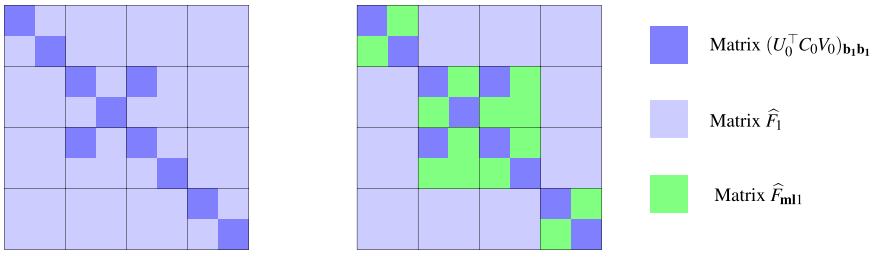
The next step is to return to the matrix $\widetilde{A}_1$ of size $N \times N$. Define the following matrices:

$$F_{\mathbf{ml1}} = \begin{bmatrix} 0_{(N-Mr)\times(N-Mr)} & 0 \\ 0 & \widehat{F}_{\mathbf{ml1}} \end{bmatrix}, \tag{8}$$

$$F_1 = \begin{bmatrix} 0_{(N-Mr)\times(N-Mr)} & 0 \\ 0 & \widehat{F}_1 \end{bmatrix}, \tag{9}$$

$$\widetilde{C}_1 = P_{r1}(U_0^\top C_0 V_0) P_{c1}, \tag{10}$$

$$\widetilde{F}_1 = P_{r1}(\dot{F}_1) P_{c1}, \tag{11}$$



(a) $A_{\mathbf{b_1 b_1}} = (U_0^\top C_0 V_0)_{\mathbf{b_1 b_1}} + (\dot{F}_1)_{\mathbf{b_1 b_1}}$ (b) $A_{\mathbf{b_1 b_1}} = \widehat{C}_1 + \widehat{F}_1$

Fig. 4 Matrix $A_{\mathbf{b_1 b_1}}$. **a** $A_{\mathbf{b_1 b_1}} = (U_0^\top C_0 V_0)_{\mathbf{b_1 b_1}} + (\dot{F}_1)_{\mathbf{b_1 b_1}}$. **b** $A_{\mathbf{b_1 b_1}} = \widehat{C}_1 + \widehat{F}_1$

where $F_{\mathbf{ml}1}, F_1, \widetilde{F}_1, \widetilde{C}_1 \in \mathbb{R}^{N\times N}$. Then, we put (5), (8), (9), (10) and (11) into (6) and obtain

$$\widetilde{A}_1 = P_{r1}(U_0^\top C_0 V_0 + \dot{F}_1)P_{c1} = \widetilde{C}_1 + \widetilde{F}_1 = \widetilde{C}_1 + F_{\mathbf{ml}1} + F_1 = C_1 + F_1,$$

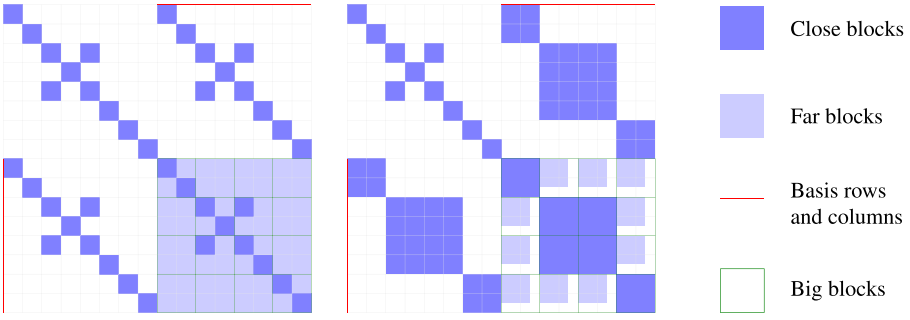
where $C_1 = \widetilde{C}_1 + F_{\mathbf{ml}1}$, $C_1 \in \mathbb{R}^{N\times N}$. Now we introduce extended compression matrices U_1 and V_1:

$$U_1 = \begin{bmatrix} I_{(N-Mr)\times(N-Mr)} & 0 \\ 0 & U_{\mathbf{b_1}} \end{bmatrix}, \quad V_1 = \begin{bmatrix} I_{(N-Mr)\times(N-Mr)} & 0 \\ 0 & V_{\mathbf{b_1}} \end{bmatrix}. \tag{12}$$

Applying matrices U_1 and V_1 to matrix $\widetilde{A}_1$ we obtain the matrix with compressed first level far matrix:

$$A_2 = U_1^\top \widetilde{A}_1 V_1 = U_1^\top (C_1 + F_1)V_1 \approx U_1^\top C_1 V_1 + \dot{F}_2,$$

where $\dot{F}_2 \in \mathbb{R}^{N\times N}$ is a compressed far matrix. The process of the first level compression is shown in Fig 5b.



(a) Matrix $\widetilde{A}_1$ (b) Matrix A_2

Fig. 5 The first level compression. **a** Matrix $\widetilde{A}_1$. **b** Matrix A_2

2.3 Compression at all levels

We apply permutation and repeat this procedure L times and obtain:

$$\begin{aligned} A_1 &= U_0^\top A V_0 \approx U_0^\top C_0 V_0 + \dot{F}_1 \\ A_2 &= U_1^\top P_{1r} U_0^\top A V_0 P_{1c} V_1 \approx U_1^\top C_1 V_1 + \dot{F}_2 \\ &\vdots \\ A_L &= \left(\prod_{k=1}^{L} U_k^\top P_{kr}\right) U_0^\top A V_0 \left(\prod_{k=L}^{1} P_{kc} V_k\right) \approx U_{L-1}^\top C_{L-1} V_{L-1} + \dot{F}_L = S, \end{aligned} \tag{13}$$

thus

$$A \approx U_0 \left(\prod_{k=L}^{1} P_{1r}^\top U_k\right) S \left(\prod_{k=1}^{L} V_k^\top P_{kc}^\top\right) V_0^\top.$$

If we denote

$$U = U_0 \left(\prod_{k=L}^{1} P_{1r}^\top U_k\right), \quad V^\top = \left(\prod_{k=1}^{L} V_k^\top P_{kc}^\top\right) V_0^\top, \tag{14}$$

then the final result of the algorithm is a sparse approximate factorization

$$A \approx U S V^\top, \tag{15}$$

where S is a matrix of the same size as matrix A; U and V are orthogonal matrices that are products of permutation and block-diagonal orthogonal matrices.

Remark 4 If matrix A is approximated into $\mathscr{H}^2$ format, then matrices S, U and V can be constructed from parameters of the representation (see details in Section 4).

Remark 5 In Section 3 we will prove that the matrix S sparse. U and V are also sparse, thus the factorization (15) if a sparse factorization.

Proposition 1 *If the matrix A is symmetric and positive definite, then the factors U and V are equal and the matrix S is symmetric and positive definite.*

Proof If the matrix A is symmetric and positive definite, then from the steps of compression algorithm, compression matrices U_i and V_i, $i = 0, \ldots, L$ are equal. Permutation matrices P_{ir} and P_{ic}, $i = 0, \ldots, L$ are equal. Thus, by (14), $U = V$. Since $S = U^\top A V$, $U = V$ and A is symmetric and positive definite, then S is symmetric and positive definite matrix. □

Remark 6 The proposed sparsification algorithm is applicable to the special cases of $\mathscr{H}^2$ matrices such as HSS (Hierarchically Semiseparable) and HOLDR (Hierarchical Off-Diagonal Low-Rank).

2.4 Pseudo code of the compression algorithm

Algorithm 1: Compression algorithm.

Input:
- $A \in \mathbb{R}^{N\times N}$ - matrix with $\mathscr{H}^2$ structure
- L - number of levels

Compression:
- **for** $k = 0, \ldots, L$ **do**
  - M_k - number of blocks on level k
  - P_{rk}, P_{ck} basis-non-basis permutations
  - $\widetilde{A}_k = P_{rk} A_1 P_{ck}$, $(P_{r0} = I,\ P_{c0} = I)$
  - Compute U_k, V_k
  - $A_{k+1} \approx U_k \widetilde{A}_k V_k^\top$

Output: *Factorization* $A \approx USV^\top$
- $U = U_0 \left(\prod_{k=L}^{1} P_{1r}^\top U_k \right)$
- $V = \left(\prod_{k=1}^{L} V_k^\top P_{kc}^\top \right) V_0^\top$,
- $S = A_L$

3 Sparsity of the matrix *S*

First, define the block sparsity pattern of a block-sparse matrix.

Definition 4 For a matrix A with M_1 block rows of block size B_1, and M_2 block columns of size B_2 define $\mathbf{bsp}(A)_{B_1\times B_2}^{M_1\times M_2}$ (block sparsity pattern) as a function

$$\mathbf{bsp} : \mathbb{R}^{M_1 B_1 \times M_2 B_2} \to \mathbb{B}^{M_1 \times M_2},$$

where $\mathbb{B} = \{0, 1\}$. The function takes block matrix $A \in \mathbb{R}^{M_1 B_1 \times M_2 B_2}$ as input and returns as output the matrix $R = \mathbf{bsp}(A)_{B_1\times B_2}^{M_1\times M_2} \in \mathbb{B}^{M_1\times M_2}$ such that

$$\begin{cases} R_{ij} = 1, & \text{if} \quad A_{ij} \in \mathbb{R}^{B_1\times B_2} \quad \text{is nonzero block,} \\ R_{ij} = 0, & \text{if} \quad A_{ij} \in \mathbb{R}^{B_1\times B_2} \quad \text{is zero block.} \end{cases}$$

Remark 7 The block structure of the matrix A in Definition 4 is given by the parameters M_1, M_2, B_1, B_2. Thus, for one matrix A, we can consider different block sparsity patterns for different block structures.

By $\#\mathbf{bsp}(A)_{B_1\times B_2}^{M_1\times M_2}$ define the number of nonzero blocks of matrix A and the number of ones in matrix R.

Proposition 2 *If the matrix A has* $\mathscr{H}^2$ *structure, the compression Algorithm 1 has L levels, the block size on each level is B, the matrix A has block-sparse close matrix*

C, and the far blocks are compressed with rank $r = B/2$, then the compression algorithm for the matrix A leads to the factorization:

$$A \approx USV^{\top}$$

where S is a ***sparse*** *matrix that has*

$$\#S \leqslant \left(4L + \frac{3}{2^L} - 2\right)\#\mathbf{bsp}(C)_{B\times B}^{M\times M}$$

nonzero blocks[4] *of size* $(r \times r)$.

Proof Consider the matrix S from (15). Let matrix S_{ij} correspond to ith level non-basis hyper row and jth level non-basis hyper column. Thanks to basis-non-basis row and column permutations P_{ri} and P_{ci} matrix S is separated into blocks S_{ij}, where $i, j = 0, \ldots, L$.

The number of nonzero blocks in S is equal to sum of nonzero blocks in S_{ij}:

$$\#S = \sum_{i=0}^{L}\sum_{j=0}^{L}\#S_{ij}. \tag{16}$$

Let us compute the number of nonzero blocks in block S_{ij}. Since on each level we join block rows by groups of 2 blocks, we obtain:

$$\mathbf{bsp}(S_{ij})_{r\times r}^{M/2^i\times M/2^j} = \mathbf{bsp}(C)_{2^iB\times 2^jB}^{M/2^i\times M/2^j}, \quad i, j = 0, \ldots, L.$$

Therefore, in terms of the number of nonzero blocks in bsp:

$$\#\mathbf{bsp}(S_{ij})_{r\times r}^{M/2^i\times M/2^j} = \#\mathbf{bsp}(C)_{2^iB\times 2^jB}^{M/2^i\times M/2^j}, \quad i, j = 0, \ldots, L. \tag{17}$$

We substitute (17) into (16) and obtain

$$\#S = \sum_{i=0}^{L}\sum_{j=0}^{L}\#\mathbf{bsp}(C)_{2^iB\times 2^jB}^{M/2^i\times M/2^j}. \tag{18}$$

Block grouping procedure leads to the following inequality

$$\#\mathbf{bsp}(C)_{2^iB\times 2^jB}^{M/2^i\times M/2^j} \leqslant \frac{\#\mathbf{bsp}(C)_{B\times B}^{M\times M}}{2^{\min(i,j)}}, \quad i, j = 0, \ldots, L. \tag{19}$$

We substitute (19) into (18) and obtain

$$\#S \leqslant \sum_{i=0}^{L}\sum_{j=0}^{L}\frac{\#\mathbf{bsp}(C)_{B\times B}^{M\times M}}{2^{\min(i,j)}} = \#\mathbf{bsp}(C)_{B\times B}^{M\times M}\sum_{i=0}^{L}\sum_{j=0}^{L}\frac{1}{2^{\min(i,j)}}. \tag{20}$$

We twice apply the formula of the sum of a geometric series to (20) and obtain

$$\#S \leqslant \#\mathbf{bsp}(C)_{B\times B}^{M\times M}\sum_{i=0}^{L}\left(2 + \frac{L-i-1}{2^i}\right) = \#\mathbf{bsp}(C)_{B\times B}^{M\times M}\left(4L + \frac{3}{2^L} - 2\right).$$

[4]The symbol # before the matrix means the number of nonzero $(r \times r)$ blocks in this matrix.

Thus, the number of nonzero blocks in matrix S is less than the number of nonzero blocks in close matrix C multiplied by constant $\left(4L + \frac{3}{2^L} - 2\right)$, if matrix C is block-sparse, and all proposition conditions are met, then matrix S is also sparse. □

Corollary 1 *For a matrix with non-uniform ranks, if ranks are bounded by $r_i \leqslant B/2$, then the factorization is sparse.*

Remark 8 The Proposition 2 can be generalized on the case of non-uniform block size B as well as on the general case of a tree.

4 Building sparse factorization from $\mathscr{H}^2$ coefficients

4.1 Definition of $\mathscr{H}^2$ matrix

In this section we consider the matrix A approximated in $\mathscr{H}^2$ format. There exist a number of efficient ways to build this approximation [6, 22]. In this paper, we do not consider the process of building the $\mathscr{H}^2$ matrix and assume that it is given. Let us explain in details how to construct factors in the decomposition (15) from parameters of $\mathscr{H}^2$ matrix A. First, we give the definition of $\mathscr{H}^2$ matrix, the more detailed definition can be found in [6].

Definition 5 (Row and column cluster trees) Cluster trees of rows and columns $\mathscr{T}_r$ and $\mathscr{T}_c$ define the hierarchical division of block rows and columns. At each level of the row cluster tree $\mathscr{T}_r$, each node corresponds to a block row of the matrix A, child nodes correspond to the subrows of this row. Same for the column cluster tree $\mathscr{T}_c$.

Definition 6 (Block cluster tree) Let $\mathscr{T}_{rc}$ be a tree. $\mathscr{T}_{rc}$ is a block cluster tree for $\mathscr{T}_r$ and $\mathscr{T}_c$ if it satisfies the following conditions:

- root($\mathscr{T}_{rc}$) = (root($\mathscr{T}_r$),root($\mathscr{T}_c$)).
- Each node $b \in \mathscr{T}_{rc}$ has the form $b = (t, s)$ for $t \in \mathscr{T}_r$ and $s \in \mathscr{T}_c$.
- Let $b = (t, s) \in \mathscr{T}_{rc}$. If sons$(b) \neq \varnothing$, then

$$\text{sons}(b) = \begin{cases} \{t\} \times \text{sons}(s) & \text{if sons}(t) = \varnothing, \text{sons}(s) \neq \varnothing, \\ \text{sons}(t) \times \{s\} & \text{if sons}(t) \neq \varnothing, \text{sons}(s) = \varnothing, \\ \text{sons}(t) \times \text{sons}(s) & otherwise. \end{cases}$$

Definition 7 (Admissibility condition) Let $\mathscr{T}_r$ and $\mathscr{T}_r$ be a row and column cluster trees. A predicate

$$\mathscr{A} = \mathscr{T}_r \times \mathscr{T}_c \to \{\text{True,False}\}$$

is an admissibility condition for $\mathscr{T}_r$ and $\mathscr{T}_r$ if

$$\mathscr{A}(t, s) \Longrightarrow \mathscr{A}(t', s) \quad \text{holds for all } t \in \mathscr{T}_r, s \in \mathscr{T}_c, t' \in \text{sons}(t)$$

and

$$\mathscr{A}(t, s) \Longrightarrow \mathscr{A}(t, s') \quad \text{holds for all } t \in \mathscr{T}_r, s \in \mathscr{T}_c, s' \in \text{sons}(s).$$

If $\mathscr{A}(t, s)$, the pair (t,s) is called admissible.

Assume that $t \in \mathscr{T}_r$ and $s \in \mathscr{T}_c$ are associated with groups of source and target points on some grid in $\mathbb{R}^d$, $d = 1, 2, 3$.

Definition 8 (Strong/weak admissibility condition) The admissibility condition $\mathscr{A}(t, s)$ is called strong, if

$$\mathscr{A}(t, s) = (max(\text{diam}(s), \text{diam}(t)) \leq \eta \text{dist}(s, t)),$$

where $\text{diam}(s)$, $\text{diam}(s)$ are sizes of bounding boxes of source and target points associated with s and t; $\text{dist}(s, t)$ is a distance between centers of corresponding bounding boxes, η is constant.

The admissibility condition $\mathscr{A}(t, s)$ is called weak, if

$$\mathscr{A}(t, s) = \left(\frac{\text{diam}(s) + \text{diam}(t)}{2} \leq \text{dist}(s, t) \right).$$

Definition 9 (Farfield and nearfield) Let $\mathscr{T}_{rc}$ be a block cluster tree for $\mathscr{T}_r$ and $\mathscr{T}_c$, let $\mathscr{A}$ be an admissibility condition. The index set

$$\mathscr{I}^+_{N\times N} := \{(t, s) \in \mathscr{I}_{N\times N} \quad : \mathscr{A}(t, s) = \text{True}\}$$

is called set of farfield blocks. The index set

$$\mathscr{I}^-_{N\times N} := \{(t, s) \in \mathscr{I}_{N\times N} \quad : \mathscr{A}(t, s) = \text{False}\}$$

is called set of nearfield blocks.

Definition 10 (Cluster basis) Let $K = (K_t)_{t\in\mathscr{T}_r}$ be a family of finite index sets (*rank distribution for* $\mathscr{T}_r$). Let $R = (R_t)_{t\in\mathscr{T}_r}$ be a family of matrices satisfying $R_t \in \mathbb{R}^{N\times K_t}$ for all $t \in \mathscr{T}_r$. Then R is called row cluster basis and the matrices R_t are called row cluster basis matrices. Analogically for column cluster basis E and column cluster basis matrices E_s, $s \in \mathscr{T}_c$.

Definition 11 (Close matrix) The block matrix $C \in \mathbb{R}^{N\times N}$ has nearfild blocks equal to corresponding blocks of matrix A and farfield blocks equal to zero.

Definition 12 ($\mathscr{H}^2$ matrix) The matrix $A \in \mathbb{R}^{N\times N}$ is approximated in $\mathscr{H}^2$ format the if following exist:

- $\mathscr{T}_r$ and $\mathscr{T}_c$ are block cluster trees of rows and columns of matrix A,
- $K = (K_t)_{t\in\mathscr{T}_r}$ is a family of finite index sets $\mathscr{T}_r$, $L = (L_s)_{s\in\mathscr{T}_c}$ is a family of finite index sets $\mathscr{T}_c$,
- row cluster basis R (row transition matrices),
- column cluster basis E (column transition matrices),
- a family $D = (D_b)_{b\in\mathscr{I}^+_{N\times N}}$ of matrices satisfying $D_b \in \mathbb{R}^{K_t\times L_s}$ for all $b = (s, t) \in \mathscr{I}^+_{N\times N}$ (interaction list),
- close matrix C.

And if

$$A \approx C + \sum_{b=(s,t)\in\mathscr{J}^{+}_{N\times N}} R_t D_b E_s^*.$$

4.2 Construction of matrices *U* and *V* from the coefficients of $\mathscr{H}^2$ matrix

Let us first construct orthogonal matrices U and V from the factorization (15). According to (14):

$$U = U_0 \left(\prod_{k=L}^{1} P_{1r}^{\top} U_k \right), \quad V = \left(\prod_{k=1}^{L} V_k^{\top} P_{kc}^{\top} \right) V_0^{\top}, \tag{21}$$

Note that matrices U_k and V_k, $k = 0, \ldots, L$, are very close in their meaning to cluster basis matrices R_t and E_s both are level compression matrices. The difference between these matrices is that the diagonal blocks of matrices U_k and V_k are square orthogonal blocks, and diagonal blocks of matrices R_t and E_s are rectangular non-orthogonal blocks.

Thus, we can take matrices R_t and E_s, orthogonalize blocks, complete each block to square orthogonal block and obtain the matrices U_k and V_k. The algorithm that orthogonalizes blocks of matrices R_t and E_s is known as the $\mathscr{H}^2$ compression algorithm, it can be found in [6, 7]. Compression of blocks can be done by QR decomposition of blocks with the square Q factor. Permutations P_{rk} and P_{ck} can be constructed from the cluster trees.

4.3 Construction of the matrix *S* from the coefficients of $\mathscr{H}^2$ matrix

According to (13), (10) and (7):

$$\begin{aligned} S &= U_{L-1}^{\top} C_{L-1} V_{L-1} + \dot{F}_L = \\ &= U_{L-1}^{\top} \left(P_{r(L-1)} (U_{L-2}^{\top} C_{L-2} V_{L-2}) P_{c(L-1)} + F_{\mathrm{ml}(L-1)} \right) V_{L-1} + \dot{F}_L = \\ &= U_{L-1}^{\top} \left(P_{r(L-1)} U_{L-2}^{\top} \left(\ldots \left(P_{r1} (U_0^{\top} C_0 V_0) P_{c1} + \right. \right. \right. \\ &+ \left. \left. \left. F_{\mathrm{ml}1} \right) \ldots \right) V_{L-2} P_{c(L-1)} + F_{\mathrm{ml}(L-1)} \right) V_{L-1} + \dot{F}_L \end{aligned} \tag{22}$$

Construction of matrices U_i and V_i is shown in the previous subsection, matrix C_0 is stored in the $\mathscr{H}^2$ matrix explicitly as close matrix C, matrices $F_{\mathrm{ml}i}$ are exactly matrices D_i from interaction list, matrix $\dot{F}_L$ is matrix D_L. Permutations P_{ri} and P_{ci} can be built using the lists of basis rows and columns on each level. Thus, matrix S can be computed from the coefficients of the $\mathscr{H}^2$ matrix.

5 Numerical experiments

Sparsification algorithm is implemented in the Python programming language. For the $\mathscr{H}^2$ matrix approximation we use the h2tools [22] library.

Remark 9 Note that we assume strong admissibility 8 for all $\mathscr{H}^2$ matrices used in numerical experiments.

All computations are performed on MacBook Air with a 1.3GHz Intel Core i5 processor and 4 GB 1600 MHz DDR3 RAM.

First, we numerically show that the matrix S in factorization (15) is indeed sparse. Then we give the timing and storage requirements of the sparse factorization. Thereafter we consider the sparse factorization of the $\mathscr{H}^2$ matrix combined with a sparse direct solver as a direct solver for the system with the $\mathscr{H}^2$ matrix and compare this approach with HODLR direct solver and $\mathscr{H}^2$-LU solver from $\mathscr{H}$2Lib library. Finally, we consider sparse factorization with the matrix S factorized by the ILUt method as a preconditioner for GMRES solver.

We want to note that the sparsification algorithm *does not worsen* the accuracy of the $\mathscr{H}^2$ approximation. It follows from the algorithm and it is confirmed in the experiments. Therefore, in the experiments below, we omit the accuracy of the sparse factorization and show only the $\mathscr{H}^2$ approximation accuracy to avoid redundancy.

5.1 Sparsity of the factor *S*

In Section 3 we have studied the sparsity of the factor S from the factorization (15) analytically. Here we present the numerical illustration that matrix S is indeed sparse. Tests are performed for two $\mathscr{H}^2$ matrices.

Example 1 $\mathscr{H}^2$ approximation of the matrix

$$A_{ij} = \begin{cases} \frac{1}{|r_i - r_j|} & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases}, \tag{23}$$

where $r_i \in \mathbb{R}^2$ or $r_i \in \mathbb{R}^3$ is the position of the ith element. Elements are randomly distributed in identity square $\Omega = [0,1] \times [0,1]$ in $\mathbb{R}^2$ case and in identity cube $\Omega = [0,1] \times [0,1] \times [0,1]$ in $\mathbb{R}^3$ case. The $\mathscr{H}^2$ approximation accuracy is $\epsilon = 10^{-6}$. In Fig. 6 the factor S of the sparse factorization of the $\mathscr{H}^2$ matrix with the kernel (23) is marked by "inv."

Example 2 $\mathscr{H}^2$ approximation of the matrix

$$A_{ij} = 2\delta_{ij} + \exp(-||r_i - r_j||^2), \tag{24}$$

where $r_i \in \mathbb{R}^2$ or $r_i \in \mathbb{R}^3$ is the position of the ith element. Elements are randomly distributed in identity square $\Omega = [0,1] \times [0,1]$ in $\mathbb{R}^2$ case and in identity cube $\Omega = [0,1] \times [0,1] \times [0,1]$ in $\mathbb{R}^3$ case. The $\mathscr{H}^2$ approximation accuracy is $\epsilon = 10^{-6}$, In Fig. 6 the factor S of the sparse factorization of the $\mathscr{H}^2$ matrix with the kernel (24) is marked by "exp."

We consider the sparse factorization (15) of the $\mathscr{H}^2$ matrices from Examples 1 and 2 in 2D and 3D. In Fig. 6 we show the number of nonzero elements *per row* in the factor S. Since for all considered matrices the number of nonzero elements

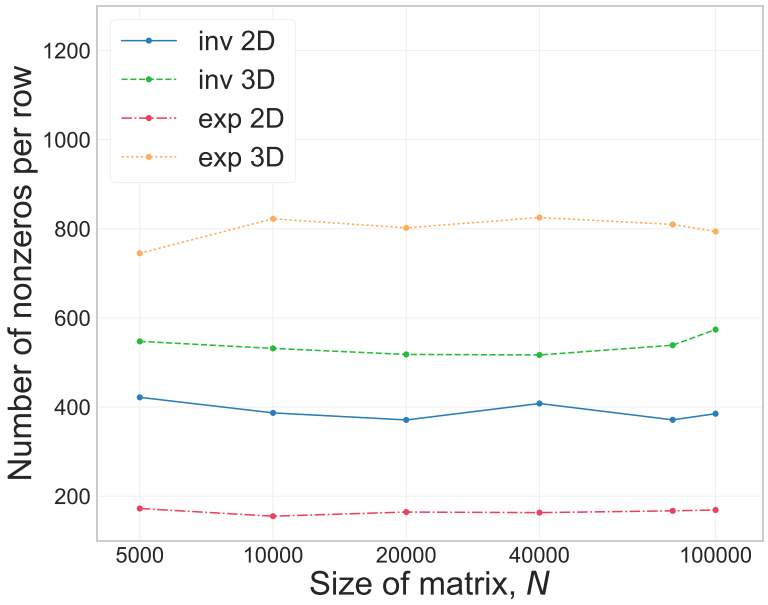


Fig. 6 Number of nonzero elements per row in factor S for $\mathscr{H}^2$ matrices from Examples 1 and 2 in 2D and 3D

per row is constant (and does not grow with the matrix size), we conclude that the matrix S in factorization (15) is indeed sparse.

5.2 The storage requirements and timing of the sparsification algorithm

In Fig. 7 we show the time of the sparse factorization of $\mathscr{H}^2$ matrices with the kernels (23) and (24) (approximation accuracy is $\varepsilon = 10^{-6}$) denoted by "inv" and "exp."

Sparsification time grows almost linearly for 2D and 3D matrices with the kernel (24) and for 2D matrix with kernel (23). For the 3D matrix with kernel (23) (green line in Fig. 7) time grows as $\mathscr{O}(N^{1.3})$. In Fig. 8 we show the memory requirements of the matrix (23) in 2D and 3D, also we show the memory requirements of its $\mathscr{H}^2$ approximation ($\varepsilon = 10^{-6}$), of the sparse factorization (15) (sum of U, S, and V), and of the factor S separately, to illustrate that storage of S much higher than the storage of $U + V$.

In Fig. 9 we show the memory requirements of the matrix (24) in 2D and 3D, also we show the memory requirements of it's $\mathscr{H}^2$ approximation ($\varepsilon = 10^{-6}$), of the sparse factorization (sum of U, S, and V) and of the factor S separately, to illustrate that storage of S much higher than the storage of $U + V$.

For all examples, the memory requirements of both $\mathscr{H}^2$ and the sparse factorization grow almost linearly, unlike the memory requirements of the original matrix which scale quadratically. Note that the matrices U and V have the same structure as m2m and l2l factors in $\mathscr{H}^2$ matrix (and thus almost the same memory requirements), while the S factor is up to 3 times larger then close+m2l factors due to the blocks

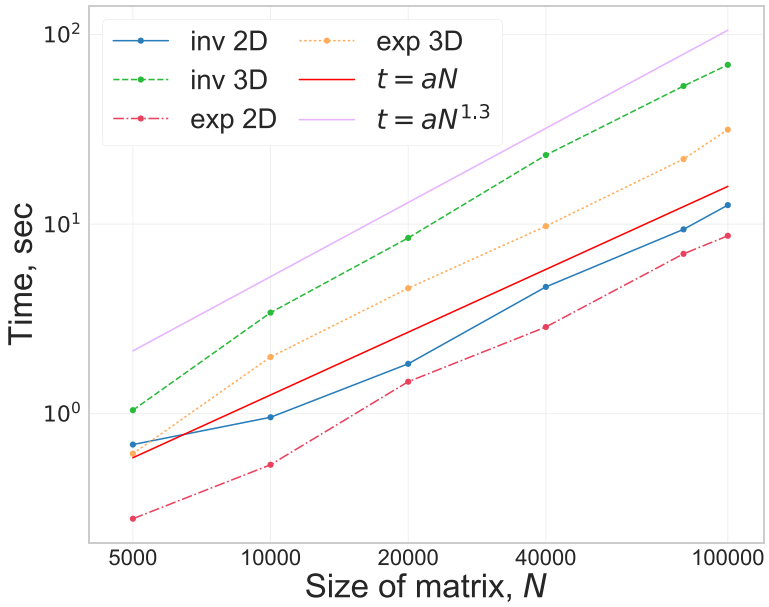
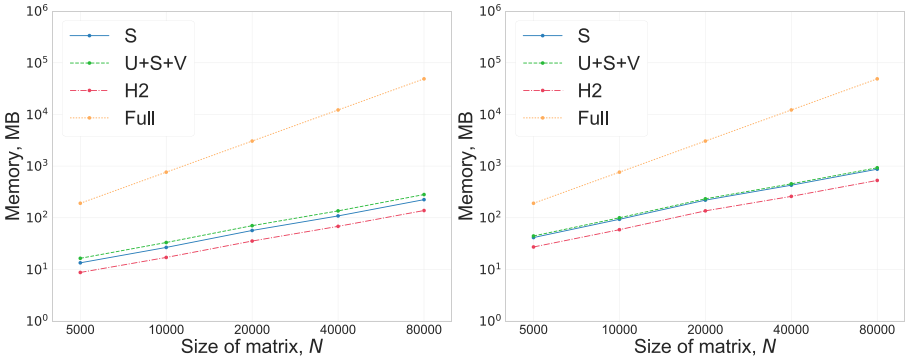


Fig. 7 Timing of sparsification building for matrices from Examples 1 and 2 in 2D and 3D

extension (see Section 2.2). That is why the storage of S is almost same as $U+S+V$ and is several times larger than $\mathscr{H}^2$.

5.3 Comparison with HODLR

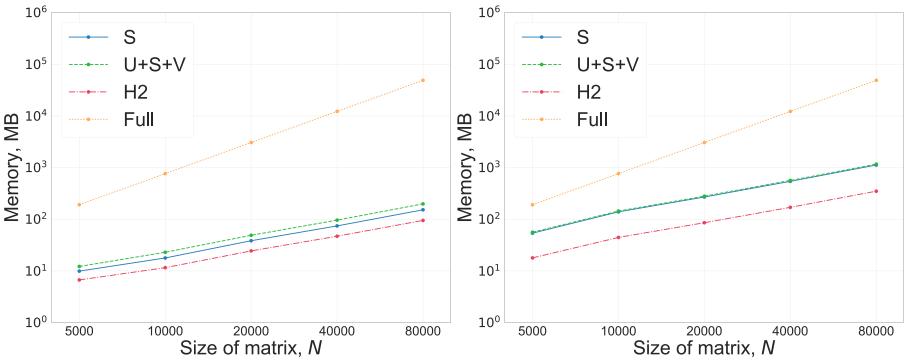
In this subsection, we compare the sparsification-based solver with the direct solver for HODLR matrix [1]. Paper [3] considers the HODLR direct solver as an efficient way to compute the determinant of the dense matrix. We propose the $\mathscr{H}^2$



(a) 2D case

(b) 3D case

Fig. 8 Storage requirements for the original matrix (23), it's $\mathscr{H}^2$ approximation and the sparse factorization. **a** 2D case. **b** 3D case



(a) 2D case (b) 3D case

Fig. 9 Storage requirements for the original matrix (24), its $\mathscr{H}^2$ approximation and the sparse factorization. **a** 2D case. **b** 3D case

approximation of the dense matrix, its sparsification and factorization of the sparse matrix as an alternative. The triangular factorization of the sparse matrix is computed by CHOLMOD [12] package. Tests are performed for 3D data, for matrix

$$A_{ij} = 2\delta_{ij} + \exp(-||r_i - r_j||^2),$$

where $r_i \in \mathbb{R}^3$ is the position of the ith element. Both HODLR and $\mathscr{H}^2$ approximation accuracy is $\varepsilon = 10^{-6}$. The HODLR factorization accuracy is $\vartheta_{\text{HOLDR}} = 10^{-5}$, the cholesky factorization is exact up to machine precision.

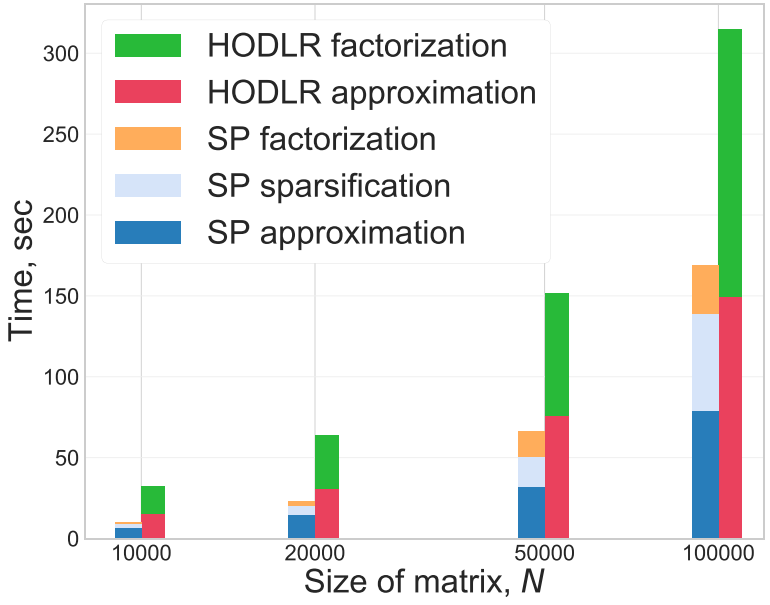


Fig. 10 Comparison of the $\mathscr{H}^2$ sparsification approach with HODLR solver in 3D

In Fig. 10 we show the time comparison for these two approaches.
Total solution time comparison in 2D and 3D is presented in Fig. 11.

5.4 Comparison with H2Lib library

In this subsection we compare the approach proposed in this paper (sparse non-extensive factorization of the $\mathscr{H}^2$ matrix, triangular factorization of the sparse matrix and then the solution of the system) with the approach based on $\mathscr{H}^2$-LU factorization, proposed in the work [9] and implemented in H2lib package [5]. The $\mathscr{H}^2$-LU factorization takes the $\mathscr{H}^2$ matrix and returns the triangular factors L and U in $\mathscr{H}^2$ format. The triangular factorization of the sparse matrix is computed by CHOLMOD [12] package. Tests are performed on the following problem.

Example 3 Consider the Dirichlet boundary value problem for Laplace's equation

$$\begin{cases} -\Delta u(x) = 0 & x \in \Omega, \\ u(x) = f(x) & x \in \Gamma = \partial\Omega, \end{cases} \tag{25}$$

where $\Omega = [-1, 1]^3$ is a cube. The standard technic: using the single layer potential we obtain boundary integral formulation of the (25).

$$\int_\Gamma G(x - y)\varphi(y)\, ds_y = f(x), \tag{26}$$

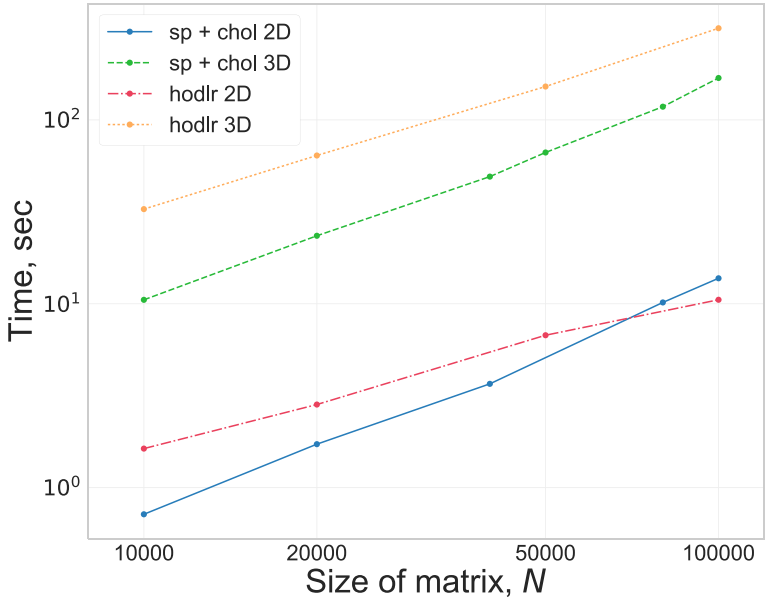


Fig. 11 Comparison of the $\mathscr{H}^2$ sparsification approach with HODLR solver in 2D and 3D

where $G(x) = \frac{1}{4\pi}\frac{1}{|x|}$ is the fundamental solution for the Laplace operator. Then we discretize the integral (26) on the triangular grid on Γ using the Galerkin method. Obtained dense matrix is approximated in $\mathscr{H}^2$ format with accuracy $\varepsilon = 10^{-6}$.

The accuracy of the $\mathscr{H}^2$-LU factorization is $\vartheta_{\text{H2LU}} = 10^{-6}$, the cholesky factorization is exact up to machine precision.

is $\vartheta_{\text{sp}} = 10^{-10}$. In Table 1 we show the time comparison of the solution of the system with matrix from Example 3, using $\mathscr{H}^2$-LU and sparsification approaches. For the $\mathscr{H}^2$-LU we show the approximation in $\mathscr{H}^2$ format and factorization time, for the sparsification we show the approximation in $\mathscr{H}^2$ format, sparsification and sparse factorization time. In both cases, time of the solution of the system with factorized matrix is negligible, so we do not show it.

In Fig. 12 we show the comparison of the total time, required for the system solution.

The H2lib package has $\mathscr{O}(N)$ asymptotics. The sparsification is also $\mathscr{O}(N)$, but the cholesky factorization of the sparse matrix is $\mathscr{O}(N^{1.5})$, which makes the sparsification solver $\mathscr{O}(N^{1.5})$ in total.

Even though the sparsification solver has worse asymptotics, it has a much smaller constant, which makes it beneficial for the tested problem sizes.

5.5 Sparsification method as a preconditioner

$\mathscr{H}^2$ matrix is an efficient tool to multiply a matrix by a vector. This allows to apply iterative solvers like GMRES to solution of the systems with $\mathscr{H}^2$ matrix. But the preconditioning is still a challenging problem due to complexity of the factorization of the $\mathscr{H}^2$ matrix. We propose to use the approximately factored sparsification of the $\mathscr{H}^2$ matrix as a preconditioner to iterative method. For tests we use randomly distributed 3D data with following interaction matrix.

Example 4

$$A_{ij} = \begin{cases} 1 & \text{if } i = j \\ \frac{|r_i - r_j|}{d} & \text{if } 0 < |r_i - r_j| < d \\ \frac{d}{|r_i - r_j|}, & \text{if } |r_i - r_j| \geqslant d \end{cases},$$

where $r_i \in \mathbb{R}^3$ is the position of the ith element.

Table 1 Comparison with H2Lib

N	3072	12,288	49,152	196,608
H2Lib approx. (s)	6.72	24.26	104.97	487.24
H2Lib factor. (s)	13.56	164.00	1712.17	10,970.43
Sp. approx. (s)	4.8	26.34	110.91	399.43
Sp. sp. (s)	2.16	10.12	63.47	323.23
Sp. factor. (s)	0.19	1.30	7.83	56.89

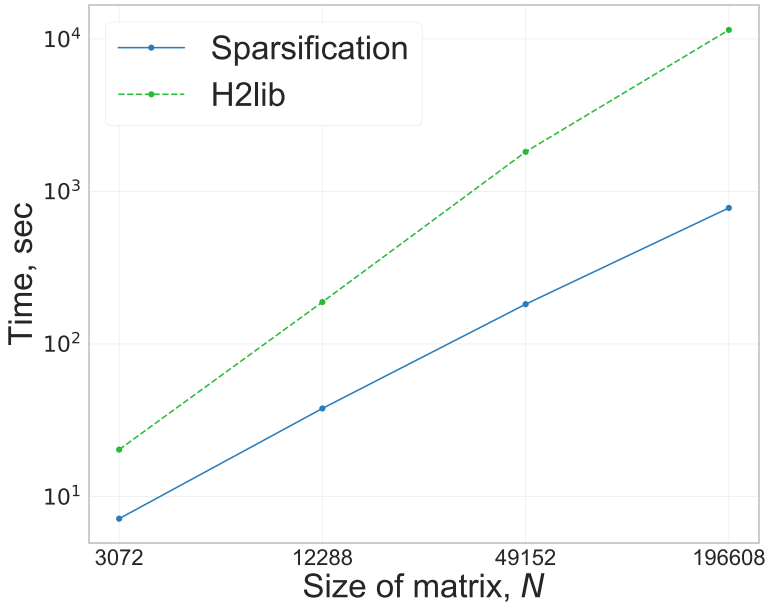


Fig. 12 Total time required for the system solution using $\mathscr{H}^2$ sparsification and H2lib solver

This example is used for testing of IFMM (Inverse Fast Multipole Method) method as a preconditioning in [11], so we have chosen this example for the convenient comparison.

This matrix is useful for the iterative tests since condition number of this matrix significantly depends on the parameter d: the larger d is, the larger condition number is (for the fixed N).

Example with well-conditioned matrix First, consider the matrix from Example 4 with $d = 10^{-3}$, condition number for $N = 5000$ cond$(A) = 10$, the larger N is, the larger condition number is. We solve this system using GMRES iterative solver for the matrix approximated in $\mathscr{H}^2$ format with accuracy $\epsilon = 10^{-9}$ and as a preconditioner we use $\mathscr{H}^2$ approximation of the matrix A with accuracy $\epsilon = 10^{-3}$ sparsified and factorized with ILUt decomposition. Figure 13 shows convergence of the GMRES method for $N = 10^5$ with different ILUt threshold parameters. The required residual of the GMRES method $r = 10^{-10}$.

The standard trade-off: the more time on the preconditioner building we spend, the faster iterations converge. Figure 14 illustrates the total time required for the system solution (including the sparsification construction).

We show the total time required for $\mathscr{H}^2$ matrix sparsification, time required for building the ILUt preconditioner with dropping tolerance $\tau = 2 \times 10^{-2}$ and iterations timing as a function of N in Fig. 15a.

Example with ill-conditioned matrix Consider the matrix from Example 4 with $d = 10^{-2}$, condition number for $N = 5000$ cond$(A) = 10^4$, the larger N is, the

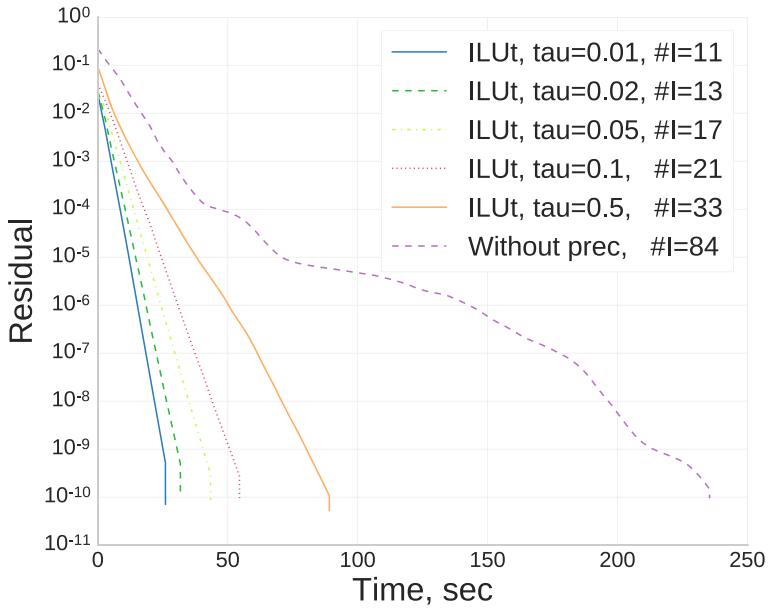


Fig. 13 Convergence of GMRES with different drop tolerance parameter of the ILUt preconditioner, $N = 10^5$, $\#I$ is number of iterations

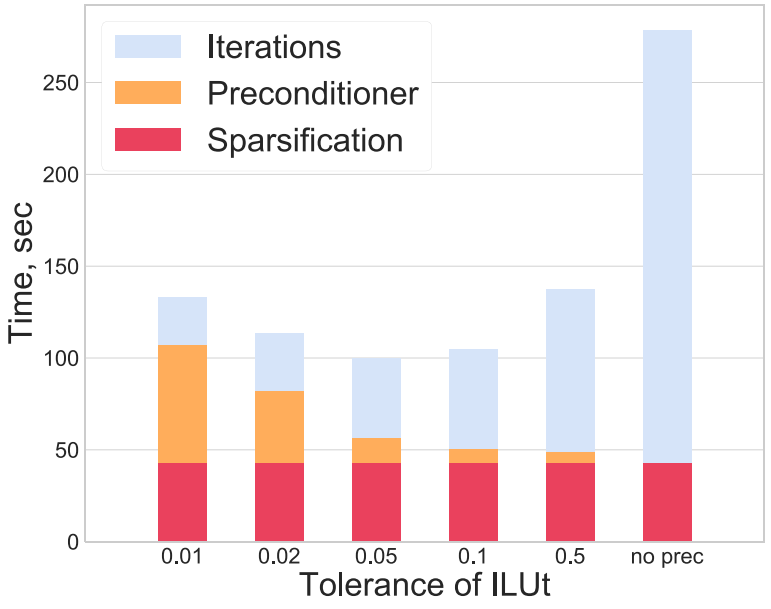
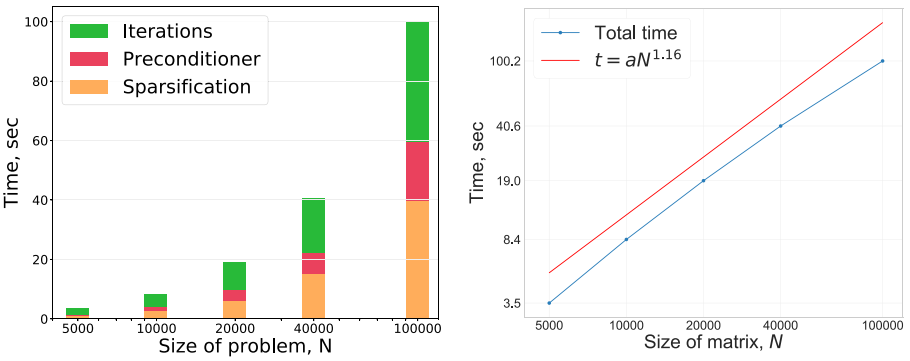


Fig. 14 Contribution into total time of sparsification, preconditioner, and iterations



(a) Contribution into total time of sparsification, preconditioner, and iterations

(b) Total time

Fig. 15 Total solution time. **a** Contribution into total time of sparsification, preconditioner, and iterations. **b** Total time

larger condition number is. As in the previous paragraph, we solve this system using GMRES iterative solver for the matrix approximated in $\mathscr{H}^2$ format with accuracy $\epsilon = 10^{-9}$ and as a preconditioner we used $\mathscr{H}^2$ approximation of the matrix A with accuracy $\epsilon = 10^{-3}$ sparsified and factorized with ILUt decomposition. In Fig. 16 convergence until the tolerance 10^{-10} for different ILUt parameters is shown. The number of iterations is limited by 200.

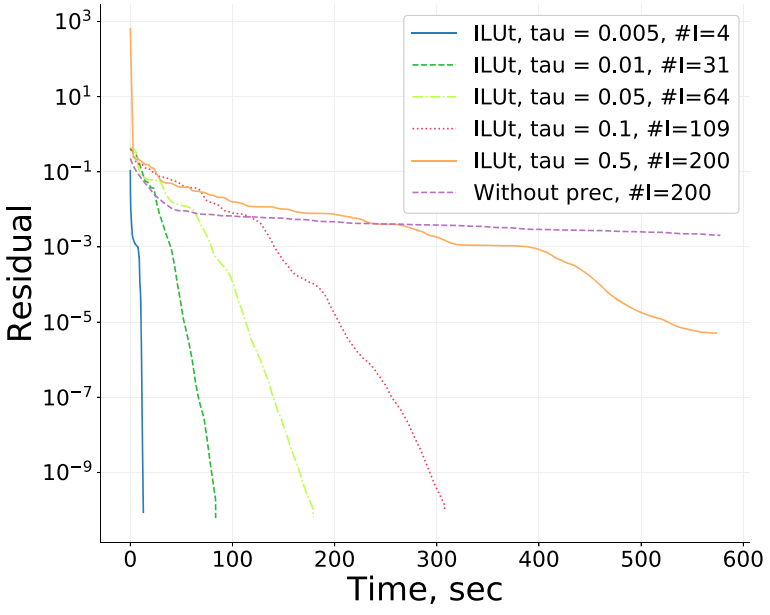


Fig. 16 Convergence of GMRES with different preconditioners, $N = 10^5$, $\#I$ is number of iterations

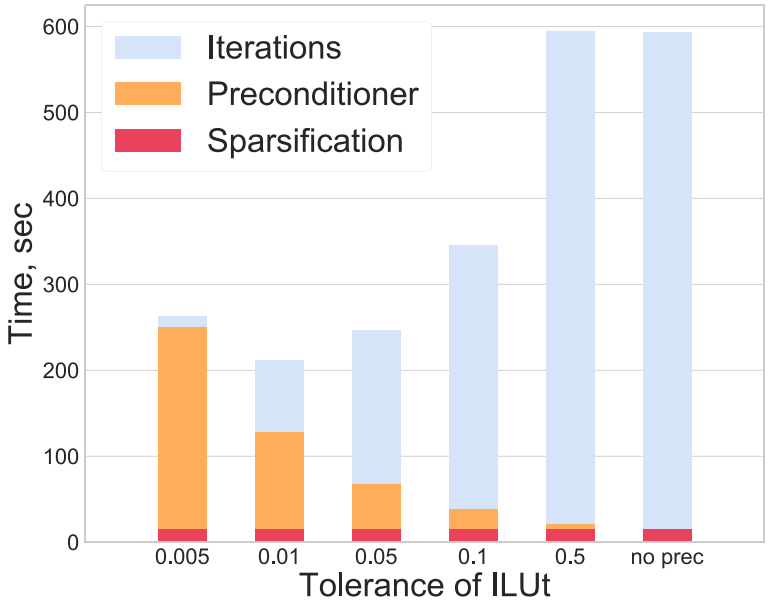
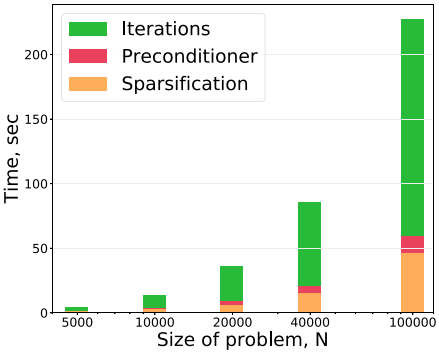


Fig. 17 Contribution into total time of sparsification, preconditioner, and iterations, $N = 10^5$

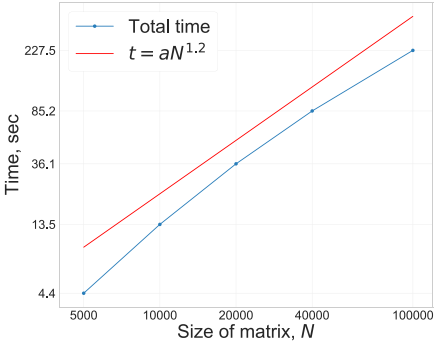
In Fig. 17 the total solution time for different ILUt parameters is shown.

Figure 18 b presents the total time required for system solution as a function of N using $\tau = 10^{-2}$

For the GMRES iterative solver, we compare the sparsification preconditioner and IFMM preconditioner [11]. For test we used matrix from Example 4 with 3D data, matrix size $N = 10^5$, iterative method GMRES until the residual $r = 10^{-10}$, parameters $d = \{10^{-3}, 10^{-2}\}$ for well- and ill-conditioned systems. Sparsification accurasy



(a) Contribution into total time of sparsification, preconditioner, and iterations



(b) Total time

Fig. 18 Total solution time. **a** Contribution into total time of sparsification, preconditioner, and iterations. **b** Total time

Table 2 Comparison with IFMM method

	$d = 10^{-3}$	$d = 10^{-2}$
IFMM solution time (s)	118	301
Sparsification solution time (s)	96	218

is $\varepsilon = 10^{-3}$, ILUt threshold is $\tau = 2 \times 10^{-2}$. IFMM rank of approximation of the farfield interactionsis is $r = 3$. We present the total solution time for both methods in Table 2.

Note that sparsification method is implemented in Python programming language and can be significantly improved by applying Cython or switching to another programing language (as C++ or Fortran).

6 Related work

Hierarchical low-rank matrix formats such as $\mathscr{H}$ [8, 15–17], HODLR [1, 3] (Hierarchical Off-Diagonal Low-Rank), HSS [10, 21, 24] (Hierarchically Semiseparable), and $\mathscr{H}^2$ [6, 16] matrices, which are matrix analogies of the fast multipole method [13, 14], have two significant features: they do store information in data-sparse formats and they provide the fast matrix by vector product. Fast ($\mathcal{O}(N)$, where N is size of the matrix) matrix by vector product allows to apply iterative solvers. Data-sparse representation allows to store matrix in $\mathcal{O}(N)$ cells of memory, but storage scheme is usually complicated.

If the hierarchical matrix is ill-conditioned, then pure iterative solver fails and it is required to apply either approximate direct solver or preconditioner (that is also approximate direct solver, probably with lower accuracy). Due to complex storage schemes of hierarchical matrices, construction of the approximate direct solver is a challenging problem. There exist two general approaches to direct solution of system with $\mathscr{H}^2$ matrix: factorization of hierarchical matrix [4, 19, 21, 23, 24], and sparsification of the hierarchical matrix followed by factorization of the sparse matrix [2, 27].

The factorization approach is more popular for hierarchical matrices with strong low-rank structure, also known as hierarchical matrices with weak admissibility criteria [18] ($\mathscr{H}$ [15, 17], HODLR [1, 3], HSS [10, 21, 24] matrices). For the $\mathscr{H}$ matrix, the algorithm $\mathscr{H}$-LU [4] with almost linear complexity was proposed. This algorithm has been successfully applied to many problems. The major drawback of the $\mathscr{H}$-LU algorithm is that factorization time and memory required for L and U factors can be quite large. Approximate direct solvers based on factorization of HSS and HODLR matrices are also well studied and found many [1, 3, 20, 21, 24, 25, 29] successful applications. The factorization approach for strong-admissible hierarchical matrices ($\mathscr{H}^2$ matrices) is also a rapidly developed area. The factorizations are based on coupled sparsification and factorization approach [9, 19, 23]. Coupled solvers eliminate

each row after its compression, which leads to almost linear algorithms, but the constant is, typically, big.

The sparsification-based solvers first perform all compressions (which leads to sparse factorization) and then factorize the sparse matrix. On the one hand, sparsification and factorization coupled solvers have better asymptotics; on the other hand, the sparsification-based solver has better constant and guarantees that far-field ranks will not be spoiled.

The sparsification approach is usually applied to hierarchical matrices with strong-admissibility criteria ($\mathscr{H}^2$ matrices). Sparsification algorithms transform $\mathscr{H}^2$ matrix into the sparse matrix and then factorize the sparse matrix. Algorithm, proposed in this paper, is the sparsification algorithm. The main difference between presented work and the other sparse factorizations [2, 11, 27] is a size of sparse factors. The main benefit of the non-extensive sparsification is that it preserves the size of the factorized $\mathscr{H}^2$ matrix, while the other sparsification algorithms return extended factors.

7 Conclusions

We have proposed a new approach to the solution of the systems with $\mathscr{H}^2$ matrices that is based on sufficient non-extensive sparsification of the $\mathscr{H}^2$ matrix. Proposed sparsification is suitable for any $\mathscr{H}^2$ matrices (including non-symmetric) and preserves such essential properties of the matrix as its size, symmetry (if it exists) and positive definite (if it exists). Since the method has lower constant, but worse scalability than existing direct solvers for $\mathscr{H}^2$ matrices, the non-extensive sparsification-based solver can be successfully applied to the 2D and 3D problem of arbitrary size up to some limit. The sparsification-based solver can also be useful for parallel implementation, and matrices with fill-in sensitive far matrix.

Funding information Section 2 is supported by Russian Foundation for Basic Research grant 17-01-00854, Sections 3 and 4 are supported by Russian Foundation for Basic Research grant 16-31-60095, Sectiom 5 is supported by Russian Science Foundation grant 15-11-00033.

References

1. Ambikasaran, S., Darve, E.: An $\mathscr{O}(n \log n)$ fast direct solver for partial hierarchically semi-separable matrices. J. Sci. Comput. **57**(3), 477–501 (2013)
2. Ambikasaran, S., Darve, E.: The inverse fast multipole method. arXiv:1309.1773 (2014)
3. Ambikasaran, S., Foreman-Mackey, D., Greengard, L., Hogg, D.W., O'Neil, M.: Fast direct methods for gaussian processes. IEEE T Pattern Anal. **38**(2), 252–265 (2016)
4. Bebendorf, M.: Hierarchical LU decomposition-based preconditioners for BEM. Computing **74**(3), 225–247 (2005)
5. Börm, S.: $\mathscr{H}$2lib package. http://www.h2lib.org/
6. Börm, S.: Efficient numerical methods for non-local operators: $\mathscr{H}^2$-matrix compression, algorithms and analysis, vol. 14. European Mathematical Society (2010)
7. Börm, S.: $\mathscr{H}^2$-matrix compression. In: New Developments in the Visualization and Processing of Tensor Fields, pp. 339–362. Springer (2012)

8. Börm, S., Grasedyck, L., Hackbusch, W.: Introduction to hierarchical matrices with applications. Eng. Anal. Bound. Elem. **27**(5), 405–422 (2003). https://doi.org/10.1016/S0955-7997(02)00152-2
9. Börm, S., Reimer, K.: Efficient arithmetic operations for rank-structured matrices based on hierarchical low-rank updates. Comput. Vis. Sci. **16**(6), 247–258 (2013)
10. Chandrasekaran, S., Dewilde, P., Gu, M., Lyons, W., Pals, T.: A fast solver for HSS representations via sparse matrices. SIAM J. Matrix Anal. A. **29**(1), 67–81 (2006)
11. Coulier, P., Pouransari, H., Darve, E.: The inverse fast multipole method: using a fast approximate direct solver as a preconditioner for dense linear systems. arXiv:1508.01835 (2015)
12. Davis, T.A., Hager, W.W.: Dynamic supernodes in sparse Cholesky update/downdate and triangular solves. ACM T Math. Softw. **35**(4), 27 (2009)
13. Greengard, L., Rokhlin, V.: A fast algorithm for particle simulations. J. Comput. Phys. **73**(2), 325–348 (1987)
14. Greengard, L., Rokhlin, V.: The Rapid Evaluation of Potential Fields in Three Dimensions. Springer, Berlin (1988)
15. Hackbusch, W.: A sparse matrix arithmetic based on $\mathscr{H}$-matrices. Part I: introduction to $\mathscr{H}$-matrices. Computing **62**(2), 89–108 (1999)
16. Hackbusch, W., Khoromskij, B., Sauter, S.: On $\mathscr{H}^2$-Matrices. In: Bungartz, H.-J. et al. (eds.) Lectures on Applied Mathematics, pp. 9–30. Springer, Berlin (2000)
17. Hackbusch, W., Khoromskij, B.N.: A sparse h-matrix arithmetic. Computing **64**(1), 21–47 (2000)
18. Hackbusch, W., Khoromskij, B.N., Kriemann, R.: Hierarchical matrices based on a weak admissibility criterion. Computing **73**(3), 207–243 (2004)
19. Ma, M., Jiao, D.: Accuracy directly controlled fast direct solutions of general $\mathscr{H}$,^{2}-matrices and its application to electrically large integral-equation-based electromagnetic analysis. arXiv:1703.06155 (2017)
20. Martinsson, P.G.: A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix. SIAM J. Matrix Anal. A. **32**(4), 1251–1274 (2011)
21. Martinsson, P.G., Rokhlin, V.: A fast direct solver for boundary integral equations in two dimensions. J. Comput. Phys. **205**(1), 1–23 (2005)
22. Mikhalev, A.Y., Oseledets, I.V.: Iterative representing set selection for nested cross approximation. Numer. Linear Algebra Appl. **23**(2), 230–248 (2016)
23. Minden, V., Ho, K.L., Damle, A., Ying, L.: A recursive skeletonization factorization based on strong admissibility. Multiscale Model. Simul. **15**(2), 768–796 (2017)
24. Sheng, Z., Dewilde, P., Chandrasekaran, S.: Algorithms to solve hierarchically semi-separable systems. In: System Theory, the Schur Algorithm and Multidimensional Analysis, pp. 255–294. Springer (2007)
25. Solovyev, S.: Multifrontal hierarchically solver for 3D discretized elliptic equations. In: International Conference on Finite Difference Methods, pp. 371–378. Springer (2014)
26. Sushnikova, D.A., Oseledets, I.V.: "Compress and eliminate" solver for symmetric positive definite sparse matrices. arXiv:1603.09133 (2016)
27. Sushnikova, D.A., Oseledets, I.V.: Preconditioners for hierarchical matrices based on their extended sparse form. Russ. J. Numer. Anal. M. **31**(1), 29–40 (2016)
28. Tyrtyshnikov, E.E.: Mosaic-skeleton approximations. Calcolo **33**(1), 47–57 (1996). https://doi.org/10.1007/BF02575706
29. Xia, J., Chandrasekaran, S., Gu, M., Li, X.S.: Superfast multifrontal method for large structured linear systems of equations. SIAM J. Matrix Anal. A. **31**(3), 1382–1411 (2009)
30. Yang, K., Pouransari, H., Darve, E.: Sparse hierarchical solvers with guaranteed convergence. arXiv:1611.03189 (2016)