Hybrid Cross Approximation for a Magnetostatic Formulation

Jonathan Siau, Olivier Chadebec, Ronan Perrussel, Jean-René Poirier

Hybrid Cross Approximation for a Magnetostatic Formulation

Jonathan Siau[1,2,5], Olivier Chadebec[1,2], Ronan Perrussel[3,4], Jean-René Poirier[3,4]

[1] Univ. Grenoble Alpes, G2Elab, F-38000 Grenoble, France
[2] CNRS, G2Elab, F-38000 Grenoble, France
[3] Université de Toulouse; INPT, UPS; LAPLACE; F-31071 Toulouse, France
[4] CNRS; LAPLACE; F-31071 Toulouse, France.
[5] CEDRAT S.A., Grenoble, France

A particular integral equation is considered for magnetostatic problems. A comparison of the adaptive cross approximation and the hybrid cross approximation, and the fast multipole method is done. The relevancy and the versatility of the HCA is assessed on magnetostatic examples.

***Index Terms*—$\mathcal{H}$-matrix, Hybrid Cross Approximation (HCA), Integral equation, Magnetostatic.**

I. Introduction

INACTIVE regions, like the air, have to be taken into account in modeling electromagnetic devices. The integral equations enable us to avoid the discretization of these regions, but keeping the possibility to compute the magnetic field at any point in these inactive regions. However an integral equation leads to a *dense* matrix $G \in \mathbb{R}^{n\times n}$ of the form

$$G_{ij} = \int_\Omega \int_\Omega \varphi_i(\mathbf{x}) g(\mathbf{x},\mathbf{y}) \psi_j(\mathbf{y}) \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} \tag{1}$$

with n the number of Degrees Of Freedom (DOF), φ_i and ψ_j respectively test and trial functions and g a kernel function. Matrix G *a priori* requires a storage and a matrix-vector product complexity proportional to n^2.

To avoid this n^2 complexity, solutions have been proposed for integral equations. Most of these solutions approximate the kernel g by a degenerate kernel $\tilde{g}$. The multipole method is such a solution but requiring an explicit expansion of g. This expansion is known for the kernel considered here, but we prefer an algebraic approach that adaptively controls the error.

We have retained the $\mathcal{H}$-matrix format that enables the compression of the matrix G block-wise: the blocks of G that correspond to distant interactions are approximated by low-rank matrices. This method constrains the storage and complexity to scale as $n \log n$, and permits in particular to compute efficient preconditioners. The Adaptive Cross Approximation (ACA) [1] is the most classical technique to build the low-rank approximations. Many articles show great compression results for classical kernels, as the Single Layer Potential (SLP) or the Double Layer Potential (DLP). However some limitations of the ACA are pointed out with simple examples (see [2] for instance): it is shown that we do not have a reliable error estimation for geometries with edges and kernels that contain a differential operator. The solution we promote instead is the Hybrid Cross Approximation (HCA) [3].

The purpose of this paper is to compare the well-known ACA to the HCA on a simple example, and to show how HCA can be straightforwardly implemented on a more complicated kernel. Section II introduces the clustering method and the two cross approximation techniques. Then, in Section III a comparison between the HCA and the Multi-Level Fast Multipole Method (MLFMM) is performed for the solution of an electrostatic problem on a thick sphere. A magnetostatic formulation is presented in Section IV and the ACA and HCA are considered for tackling this formulation: first on a spherical shell and finally on a more complicated geometry.

Manuscript received June 8, 2014; revised XX, XX. Corresponding author: J. Siau (email: jonathan.siau@g2elab.grenoble-inp.fr).

II. $\mathcal{H}$-matrix

The $\mathcal{H}$-matrix representation is a data-sparse matrix format that enables the memory storage and the complexity of the arithmetic operations, as the matrix-vector product, to scale as $n\log(n)$. Thus, it allows solving large-scale problems efficiently using integral equation methods.

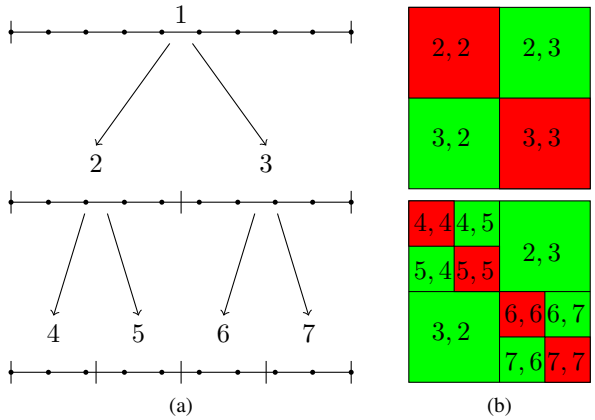
A. Clustering and low-rank approximation

The $\mathcal{H}$-matrices, like the FMM, use a clustering technique to define which DOF can be grouped in a cluster. This clustering is mainly obtained by a recursive bisection of a bounding box containing the computational domain; it then leads to define a binary tree, see Fig. 1a. The FMM are based on an oct-tree, thus the clustering is different.

Once the hierarchical clustering is performed, we compress the interactions between distant clusters of DOF; see green blocks in Fig. 1b. Let us denote by $\hat{t}$ and $\hat{s}$ the set of DOF indices in two such distant clusters t and s. The compression can be done by replacing the kernel g by a degenerate $\tilde{g}$ or by approximating directly $G|_{\hat{t}\times\hat{s}}$, the submatrix of G for the indices in $\hat{t}\times\hat{s}$, by a low-rank matrix with a prescribed error ϵ. Both approaches lead to a low-rank approximation

$$G|_{\hat{t}\times\hat{s}} \approx AB^T \tag{2}$$

with $G|_{\hat{t}\times\hat{s}} \in \mathbb{R}^{m\times n}$, $A \in \mathbb{R}^{m\times k}$, $B \in \mathbb{R}^{n\times k}$ and k the rank of the approximation. This format requires a storage of $k(m+n)$ instead of mn and the complexity of the matrix-vector product is proportional to $k(m+n)$ instead of mn. The rank k weakly



(a) (b)

Fig. 1. (a) Domain bisection and corresponding binary tree. (b) Two levels of the matrix partition induced. Green and red blocks are "far" and "near" interaction. Clusters in interaction are identified in each block.

depends on ϵ usually as $k \approx -\log\epsilon$, thus $k \ll m, n$, and storage and complexity of arithmetic operations are reduced significantly.

B. Adaptive cross approximation

The ACA only uses the knowledge of some matrix entries to build a low-rank approximation of a matrix M. Briefly, the ACA is an iterative method where at each iteration an algorithm chooses a "well-suited" couple of pivot indices (t^*, s^*) and computes the row t^* and the column s^* of M in order to generate a rank-one update which is added to the previous approximation. At the end of each iteration, the error is estimated and the algorithm is stopped when the required accuracy is reached. Details can be found in [4].

A problem is that the heuristics of the ACA may fail for domain with edges and kernels with differential operators [2]. This problem can be avoided using the HCA.

C. Hybrid cross approximation

Let us assume that

$$g(\mathbf{x},\mathbf{y}) = D_{\mathbf{x}} D_{\mathbf{y}} \gamma(\mathbf{x},\mathbf{y}), \qquad \forall (\mathbf{x},\mathbf{y}) \in \Omega_t \times \Omega_s \tag{3}$$

with $D_{\mathbf{x}}$, $D_{\mathbf{y}}$ two differential operators, $\Omega_t = \cup_{i\in\hat{t}} \mathrm{supp}(\varphi_i)$ and $\Omega_s = \cup_{j\in\hat{s}} \mathrm{supp}(\psi_j)$, the supports of the distant clusters t and s and γ a smooth generator function on $\Omega_t \times \Omega_s$. The SLP and DLP can easily be expressed using (3). We also define two bounding boxes B_t and B_s such that $\Omega_t \subset B_t$ and $\Omega_s \subset B_s$.

The HCA is an algorithm based on the cross approximation of the function γ contrary to the ACA where the cross approximation is applied at the matrix level. For this purpose, we define a grid of points in B_t (resp. B_s) where γ can be evaluated. Here, we consider a tensorization of order p Chebyshev's interpolation point $(\mathbf{x}_i)_{i=1}^K$ (resp. $(\mathbf{y}_j)_{j=1}^K$) with $K = (p+1)^d$ but other choices could be considered. Then we perform the ACA on the matrix S whose entries are

$$S_{i,j} = \gamma(\mathbf{x}_i, \mathbf{y}_j), \qquad \forall i,j \in \{1,\dots,K\} \tag{4}$$

to find k pivots. The computation cost of this ACA is low, because the rows and columns of S are easily assembled.

Once the pivots $\tilde{\mathbf{x}}_{i_l}$ and $\tilde{\mathbf{y}}_{j_l}$ are chosen ($\tilde{\mathbf{x}}_{i_l} \subset \mathbf{x}_i$), the functional cross approximation we consider is

$$\tilde{\gamma}(\mathbf{x},\mathbf{y}) = \begin{pmatrix} \gamma(\mathbf{x},\tilde{\mathbf{y}}_{i_1}) \\ \vdots \\ \gamma(\mathbf{x},\tilde{\mathbf{y}}_{i_k}) \end{pmatrix}^T \Gamma_k^{-1} \begin{pmatrix} \gamma(\tilde{\mathbf{x}}_{j_1},\mathbf{y}) \\ \vdots \\ \gamma(\tilde{\mathbf{x}}_{j_k},\mathbf{y}) \end{pmatrix}, \tag{5}$$

where Γ_k a $k\times k$-matrix whose coefficient (q,l) is $\gamma(\tilde{\mathbf{x}}_{i_q}, \tilde{\mathbf{y}}_{j_l})$. Γ_k^{-1} is not practically used, but the LU-factorization of Γ_k can be computed in the ACA subroutine [5]. Thus we replace Γ_k^{-1} by the solutions of systems with an upper U_k and a lower L_k triangular matrices.

Finally, the degenerate kernel is

$$\tilde{g} = D_{\mathbf{x}} D_{\mathbf{y}} \tilde{\gamma}, \tag{6}$$

thus the approximation $\tilde{G}|_{\hat{t}\times\hat{s}}$ of $G|_{\hat{t}\times\hat{s}}$ is given by

$$\tilde{G}|_{\hat{t}\times\hat{s}} = AB^T = (\tilde{A}U_k^{-1})(L_k^{-1}\tilde{B}^T), \tag{7}$$

with $\tilde{A} \in \mathbb{R}^{m\times k}$ and $\tilde{B} \in \mathbb{R}^{n\times k}$ define as follow

$$\tilde{A}_{iq} = \int_{\Omega_t} \varphi_i(\mathbf{x}) D_{\mathbf{x}} \gamma(\mathbf{x}, \tilde{\mathbf{y}}_{j_q}) \mathrm{d}\mathbf{x}, \tag{8}$$

$$\tilde{B}_{il} = \int_{\Omega_s} \psi_j(\mathbf{y}) D_{\mathbf{y}} \gamma(\tilde{\mathbf{x}}_{i_l}, \mathbf{y}) \mathrm{d}\mathbf{y}. \tag{9}$$

Note that we have to compute two simple integral instead of a two-fold integral, leading to a lower complexity. It is also easier to implement compared to the ACA: the ACA needs to assemble a row/column at each iteration but it is not directly provided by the usual mesh-element-loop approach.

III. Electrostatic example

Let us consider an electrostatic equation on a sphere with a standard Galerkin discretization of the SLP

$$\begin{aligned} \frac{1}{4\pi} \sum_j \left(\int_S \int_S \frac{\varphi_i(\mathbf{x})\varphi_j(\mathbf{y})}{\|\mathbf{x}-\mathbf{y}\|} \mathrm{d}\mathbf{y}\mathrm{d}\mathbf{x} \right) q_j \\ = \int_S \varphi_i(\mathbf{x}) V_0(\mathbf{x}) \mathrm{d}\mathbf{x}, \end{aligned} \tag{10}$$

where φ_j are piece-wise constant functions, q_j are the charge density unknowns and V_0 a real constant. To compare the FMM and the $\mathcal{H}$-matrices, in the assembly step we set ϵ_{ACA}, the relative error in Froebenius norm for each block, to 10^{-4} so that both methods gives an equivalent accuracy. The relative norm of the residual in the iterative solver GMRes has to be lower than 10^{-6}. The histogram in Fig. 2 shows the assembly and solution times for both methods.

We can see that the $\mathcal{H}$-matrix is faster than the FMM, even if it is slower to be assembled. In return, the FMM's iterations are about 10 times slower than the $\mathcal{H}$-matrix's, because the chosen FMM variant consists in storing only the near-interactions. This variant allows us to use little memory as shown in Fig. 3. Note also that the $\mathcal{H}$-matrix storage (here in kb/DOF) is close to the theoretical logarithmic complexity.

For these results, we did not use the $\mathcal{H}$-LU preconditioning [4] of the $\mathcal{H}$-matrices because it is a well-preconditioned problem. However for a more general case, when we used the general, efficient and light $\mathcal{H}$-LU preconditioning we conclude

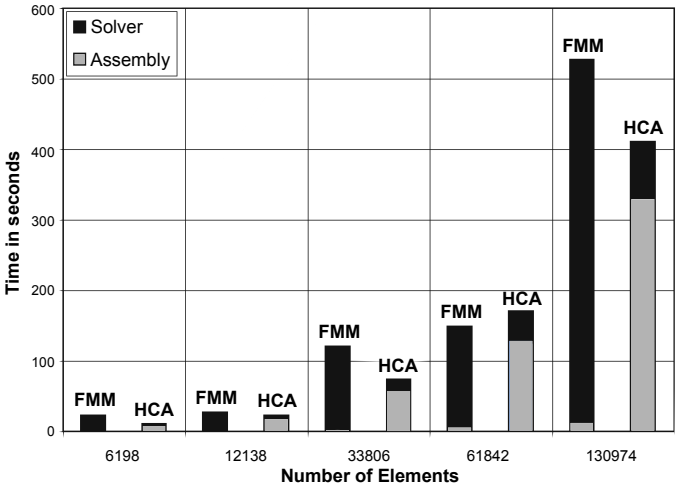


Fig. 2. Comparison of the FMM and $\mathcal{H}$-matrix times on a DELL PRECISION M4800 with an Intel(R) Core(TM) i7-4800MQ CPU @2.70GHz

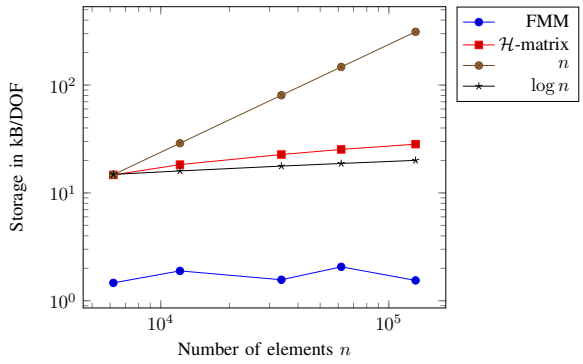


Fig. 3. Storage requirements.

that the $\mathcal{H}$-matrices are expected to be faster than the FMM in the solution step. It is especially relevant for problems with multiple right-hand sides.

IV. Magnetostatic Formulation

A. Formulation

Let us consider the ferromagnetic behavior law

$$\mathbf{M}(\mathbf{x}) = \chi(\mathbf{x}, \mathbf{H})\mathbf{H}(\mathbf{x}), \qquad \forall \mathbf{x} \in \Omega, \tag{11}$$

with $\mathbf{M}$ the magnetization, $\mathbf{H}$ the magnetic field, and Ω an isotropic ferromagnetic material with χ its magnetic susceptibility. We define χ as a constant for the next experiments. The magnetic field can be written as $\mathbf{H} = \mathbf{H}_{\text{red}} + \mathbf{H}_0$ with $\mathbf{H}_{\text{red}}$ the reduced magnetic field created by the ferromagnetic material and $\mathbf{H}_0$ the magnetic source field created by current flows.

We consider the following integral equation [6] using the total scalar potential Φ

$$\Phi(\mathbf{x}) + \frac{1}{4\pi}\int_\Omega \chi(\mathbf{y}, \mathbf{H}) \frac{\nabla\Phi(\mathbf{y}) \cdot (\mathbf{x}-\mathbf{y})}{\|\mathbf{x}-\mathbf{y}\|^3} \mathrm{d}\mathbf{y} = \Phi_0(\mathbf{x}) \tag{12}$$

with Φ_0 the scalar potential deriving from the magnetic field $\mathbf{H}_0$. Using a P_1 finite element approximation for Φ and the

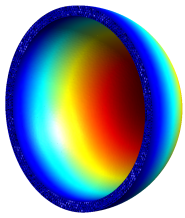


Fig. 4. Isovalues of the solution on a Spherical shell.

Galerkin discretization leads to the linear system

$$([M] + [G])\,\mathbf{\Phi} = \mathbf{D} \tag{13}$$

with the mass matrix and the right-hand-side vector

$$M_{ij} = \int_\Omega \varphi_i(\mathbf{x})\varphi_j(\mathbf{x})\mathrm{d}\mathbf{x}, \quad \mathbf{D}_i = \int_\Omega \varphi_i(\mathbf{x})\Phi_0(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{14}$$

and the volume integral

$$G_{ij} = \frac{1}{4\pi}\int_\Omega \varphi_i(\mathbf{x}) \int_\Omega \chi\nabla\varphi_j(\mathbf{y}) \cdot \nabla_\mathbf{y} \frac{1}{\|\mathbf{x}-\mathbf{y}\|}\mathrm{d}\mathbf{y}\mathrm{d}\mathbf{x}. \tag{15}$$

Solving (13) permits to compute the magnetic field $\mathbf{H}$ in the air [7]. The mass matrix is a finite element matrix with a sparse storage, so the limitation here is the dense matrix G.

B. Degenerate expansion

We use the HCA to compress the matrix G. In (15) the kernel function $g(\mathbf{x}, \mathbf{y}) = \nabla_\mathbf{y} \dfrac{1}{\|\mathbf{x}-\mathbf{y}\|}$ enables to identify

$$\gamma(\mathbf{x}, \mathbf{y}) = \frac{1}{\|\mathbf{x}-\mathbf{y}\|}, \qquad D_\mathbf{x} = \mathrm{Id}, \qquad D_\mathbf{y} = \nabla_\mathbf{y}. \tag{16}$$

The two single integrals to compute are then

$$\int_\Omega \frac{\varphi_i(\mathbf{x})}{\|\mathbf{x}-\tilde{\mathbf{y}}_{j_l}\|}\mathrm{d}\mathbf{x} \text{ and } \int_\Omega \chi\nabla\varphi_j(\mathbf{y}) \cdot \nabla_\mathbf{y} \frac{1}{\|\tilde{\mathbf{x}}_{i_q}-\mathbf{y}\|}\mathrm{d}\mathbf{y}. \tag{17}$$

The computational cost of these integrals is proportional to the number of quadrature points per element instead of the quadratic cost of the twofold integral, allowing us to have a lower complexity than the ACA for the same rank.

C. Numerical results

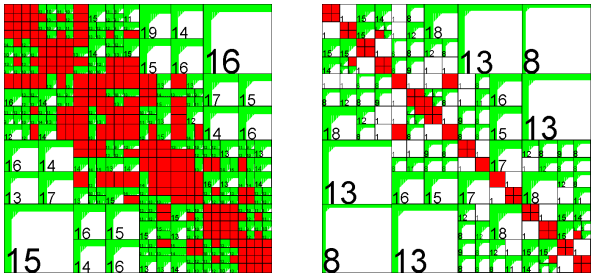
Two cases are considered in this section, the first consists in solving (13) on a spherical shell meshed with tetrahedra; the solution is shown in Fig. 4. The second consists of a numerical study of the ACA and the HCA for the assembly of (15) on the TEAM Problem 13 geometry [8].

For the first results, we keep the same parameters as for the electrostatic example and we can compare the results with those obtained with the full-storage. We can see in Table I that the HCA is faster than the ACA to assemble an $\mathcal{H}$-matrix, but unfortunately the HCA requires more memory compared to the ACA. These results match to these found in the literature, which also treat spherical geometries for other integral equations.

These memory requirements can be reduced using two algebraic re-compression techniques, the SVD of the low-rank matrices and the Agglomeration. Both methods are explicitly

TABLE I
COMPUTING TIMES, STORAGE AND ERROR WITHOUT AGGLOMERATION.

	NbDOF	Assembly (s)	Solver (s)	Storage (kb/DOF)	Error
A	3,846	718	0.5	15.4	1.53%
C	12,938	7321	10	24.3	0.85%
A	48,154	56,112	89	37.6	0.37%
H	3,846	278	0.6	18.3	5.23%
C	12,938	2,333	12	29.4	1.33%
A	48,154	18,154	109	44.9	1.42%



(a) (b)

Fig. 5. Example of a $\mathcal{H}$-matrix agglomeration. (a) Initial $\mathcal{H}$-matrix. (b) Agglomerated $\mathcal{H}$-matrix. The number in the green blocks indicates the rank.

TABLE II
COMPUTING TIMES, STORAGE AND ERROR WITH AGGLOMERATION.

	NbDOF	Assembly (s)	Solver (s)	Storage (kB/DOF)	Error
A	3,846	952	0.5	7.0	1.66%
C	12,938	9,774	10	10.3	0.27%
A	48,154	74,604	95	14.9	0.45%
H	3,846	287	0.5	7.0	1.63%
C	12,938	2,586	12	10.4	0.54%
A	48,154	18,876	98	14.9	0.72%

described in [4]. These methods do not change the asymptotic complexity of the assembly step, and they allow us to reduce the complexity of the arithmetic operations with the matrices. We do not detail these methods but we can see in Fig. 5 the simplification of the structure and the reduced storage.

Table II is obtained using the agglomeration after the assembly of the $\mathcal{H}$-matrix. The assembly times have only slightly increased compared to Table I, but the major improvement is that both methods have an equivalent and reduced storage.

Finally, we compare the accuracy of both methods (without using the agglomeration) applied to the TEAM Problem 13 geometry (see Fig. 6), meshed with $29,362$ tetrahedra. For a matter of ease, the "Error" column is computed comparing the matrix-vector products $\|Gx - \tilde{G}x\|/\|Gx\|$, where x is a random vector. It is not the conventional way to do, but since we are using an iterative solver this information is relevant for the solution quality.

The results of the assembly step are presented in Table III, where we see that the ACA needs an unusually higher storage than the HCA. However the HCA is less accurate than the ACA, but still of the same order as ϵ_{ACA}. On this realistic problem, the HCA is more efficient comparing the storage requirements and the assembly times.

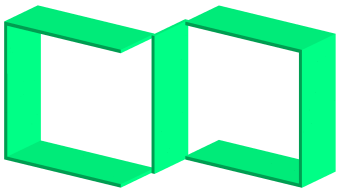


Fig. 6. TEAM Problem 13 geometry.

TABLE III
ASSEMBLY TIME, STORAGE AND ERROR. TEAM PROBLEM 13.

	ϵ_{ACA}	Assembly (s)	Storage (kb/DOF)	Error
A	1E-02	8,574	12.4	1.950E-3
C	1E-03	11,961	15.5	1.643E-4
A	1E-04	15,690	19.2	1.346E-5
H	1E-02	2,607	10.4	4.070E-2
C	1E-03	2,773	13.9	3.300E-3
A	1E-04	2,857	18.6	8.890E-4

V. CONCLUSION

This paper first demonstrates the efficiency of the $\mathcal{H}$-matrix to solve an electrostatic problem compared to the FMM. Thanks to the general, efficient and light $\mathcal{H}$-LU preconditioning, we note that the $\mathcal{H}$-matrices are generally more efficient for other considered problems. Then we focus on the construction of the low-rank approximations, and we compare ACA and HCA considering two cases. The ACA does not fail for this formulation and the considered examples. Comparing ACA and HCA, we show that the HCA is faster but requires a larger storage. Adding an agglomeration step, storage is equivalent for both methods. At the end the HCA, seems more efficient and reliable than the ACA for this formulation.

ACKNOWLEDGMENTS

We thank the GDR SEEDS for its financial support.

REFERENCES

[1] M. Bebendorf, "Approximation of boundary element matrices," *Numer. Math.*, vol. 86, no. 4, pp. 565–589, 2000.

[2] S. Börm, L. Grasedyck, and W. Hackbusch, *Hierarchical Matrices*, Max-Planck-Institute for Mathematics in the Sciences, Leipzig, Germany, 2003. Revised June 2006, no. 21.

[3] S. Börm and L. Grasedyck, "Hybrid cross approximation of integral operators," *Numerische Mathematik*, vol. 101, no. 2, pp. 221–249, Jun. 2005.

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

[5] M. Bebendorf and S. Kunis, "Recompression techniques for adaptive cross approximation," *Journal of Integral Equations and Applications*, vol. 21, no. 3, pp. 331–357, 09 2009.

[6] A. Kalimov, "Application of a hybrid integrodifferential method for analysis of thin magnetic shields," *Magnetics, IEEE Transactions on (Volume:34 , Issue: 5)*, pp. 2453–2456, Sep 1998.

[7] A. Carpentier, O. Chadebec, N. Galopin, G. Meunier, and B. Bannwarth, "Resolution of Nonlinear Magnetostatic Problems With a Volume Integral Method Using the Magnetic Scalar Potential," *IEEE Transactions on Magnetics*, vol. 49, no. 5, pp. 1685–1688, May 2013.

[8] T. NAKATA, N. TAKAHASHI, and K. FUJIWARA, "Summary of results for team workshop problem 13 (3-d nonlinear magnetostatic model)," *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 14, pp. 91 – 101, 1995.