Institut für Numerische Simulation
Rheinische Friedrich-Wilhelms-Universität Bonn

Wegelerstraße 6 • 53115 Bonn • Germany
phone +49 228 73-3427 • fax +49 228 73-7527
www.ins.uni-bonn.de

M. Bebendorf[1] and S. Kunis[2]

Recompression techniques for adaptive cross approximation

INS Preprint No. 0708

Dec. 2007

[1]Institut für Angewandte Mathematik, Universität Bonn, D-53115 Bonn
[2]Fakultät für Mathematik, Technische Universität Chemnitz, D-09107 Chemnitz

Recompression techniques for adaptive cross approximation

M. Bebendorf* and S. Kunis†

The adaptive cross approximation method generates low-rank approximations to suitable $m \times n$ sub-blocks of discrete integral formulations of elliptic boundary value problems. A characteristic property is that the approximation, which requires $k(m+n)$, $k \sim |\log \varepsilon|^*$, units of storage, is generated in an adaptive and purely algebraic manner using only few of the matrix entries. In this article we present further recompression techniques which bring the required amount of storage down to sublinear order kk', where k' depends logarithmically on the accuracy of the approximation but is independent of the matrix size.

AMS Subject Classification: 65D05, 65D15, 65F05, 65F30.
Keywords: adaptive cross approximation, integral equations, hierarchical matrices.

1 Introduction

The finite element discretization of integral formulations of elliptic boundary value problems leads to fully populated matrices $K \in \mathbb{R}^{N\times N}$ of large dimension N. By the introduction of the fast multipole method [13], the panel clustering method [19], the wavelet Galerkin method [1], and hierarchical ($\mathcal{H}$-) matrices [15, 17] it has become possible to treat such matrices with almost linear complexity. While most of these methods can be used only to store and to multiply approximations by a vector, $\mathcal{H}$-matrices provide efficient approximations to the matrix entries. The latter property is useful because preconditioners can be constructed from the matrix approximant in a purely algebraic way; see [4].

There are two main techniques for the construction of $\mathcal{H}$-matrices in the context of integral operators

$$(\mathcal{K}u)(x) = \int_\Omega \kappa(x,y)u(y)\,\mathrm{d}y$$

with given domain $\Omega \subset \mathbb{R}^d$ and kernel function κ which consists of the singularity function S or its derivatives. The first technique constructs the approximants by polynomial approximation and requires the explicit knowledge of the function κ. The second is the adaptive cross approximation (ACA) method (see [2]), which approximates suitable sub-blocks $A \in \mathbb{R}^{m\times n}$ of the discretized operator K by

$$A\Pi_2(\Pi_1 A\Pi_2)^{-1}\Pi_1 A \;\approx\; A, \tag{1.1}$$

where $\Pi_1 \in \mathbb{R}^{k\times m}$ consists of the first $k \ll \min\{m,n\}$ rows of a permutation matrix and $\Pi_2 \in \mathbb{R}^{n\times k}$ are the first k columns of another permutation matrix. Hence, instead of computing and storing all entries of A, we compute and store their approximation with complexity $\mathcal{O}(k(m+n))$. This latter method is in general more efficient with respect to both the number of operations and the

*Institut für Angewandte Mathematik, Universität Bonn, D-53115 Bonn
†Fakultät für Mathematik, Technische Universität Chemnitz, D-09107 Chemnitz

quality of the approximant. Additionally, ACA is more convenient since it relies on few of the original matrix entries. Hence, existing "slow" codes can be accelerated with small changes while kernel approximation methods such as the fast multipole method and methods based on interpolation require a fundamental recoding.

Both techniques rely on the smoothness of S. It is shown in [6] that for integral formulations of elliptic boundary value problems the function S is asymptotically smooth, i.e., for $x \in D_1$ and $y \in D_2$, where $D_1, D_2 \subset \Omega$, there are constants $c, \gamma_1, \gamma_2 > 0$ such that

$$|\partial_x^\alpha \partial_y^\beta S(x,y)| \leq c\,|\alpha|!\,|\beta|!\,\gamma_1^{|\alpha|}\gamma_2^{|\beta|}|x-y|^{-|\alpha|-|\beta|}|S(x,y)| \tag{1.2}$$

for all $\alpha, \beta \in \mathbb{N}^d$. Using either method, an $\mathcal{H}$-matrix approximation of K is generated which has complexity $\mathcal{O}(kN\log N)$, where k depends logarithmically on the approximation accuracy ε.

Although ACA generates approximants of high quality, the amount of storage required for an approximant can still be reduced. The reason for this is visible from the special structure of the approximant. The representation (1.1) uses parts $\Pi_1 A$ and $A\Pi_2$ of the original matrix A for its approximation. Since $\Pi_1 A$ and $A\Pi_2$ have the same smoothness properties as the entire block A, they can be additionally approximated using polynomial approximation, for instance. Note that our construction will not be based on polynomial approximation of the kernel κ since we can afford more advanced methods due the fact that one of the dimensions of $\Pi_1 A$ and $A\Pi_2$ is k which can be considered to be small. Our aim is to devise a method which preserves both the adaptivity and the property that only the matrix entries are used. The way we will achieve this is based on projecting $\Pi_1 A$ and $A\Pi_2$ to explicitly given bases. In total, this recompression generates so-called uniform $\mathcal{H}$-matrices (see [15]) from few of the original matrix entries.

For uniform $\mathcal{H}$-matrices it is necessary to store the coefficients of the projection together with the bases. The amount of storage for the coefficients is of the order kN (cf. [16]), i.e., compared with $\mathcal{H}$-matrices the factor $\log N$ is saved. However, storing the bases still requires $\mathcal{O}(kN\log N)$ units of storage. A continuation of the development of $\mathcal{H}$-matrices has led to $\mathcal{H}^2$-matrices (see [18]) which allow to store the bases with $\mathcal{O}(kN)$ complexity. For $\mathcal{H}^2$-matrices the factor k in the asymptotic complexity can even be removed if variable order approximations (see [18, 8]) are employed. However, then the approximation accuracy is not arbitrarily small and will in general not improve with the quality of the discretization unless operators of order zero are considered. The construction of uniform $\mathcal{H}$- and $\mathcal{H}^2$-matrices is usually based on polynomial approximations of the kernel function.

Since we will use explicitly given row and column bases, only the coefficients of the projection of the ACA approximant on these bases are stored. Hence, we will improve the overall asymptotic complexity to $\mathcal{O}(kN)$. However, in this case the bases have to be constructed on-the-fly when multiplying the approximant by a vector. It will be seen that this additional effort does not change the asymptotic complexity of the matrix-vector multiplication. A slight increase of the actual run-time can be tolerated since the multiplication is computationally cheap while it is necessary to further reduce the storage requirements of ACA approximants.

In this article we will concentrate on a single sub-block $A \in \mathbb{R}^{m\times n}$ of an hierarchical matrix of size $N \times N$. We assume that A has the entries

$$a_{ij} = \int_\Omega \int_\Omega \kappa(x,y)\psi_i(x)\varphi_j(y)\,\mathrm{d}x\,\mathrm{d}y, \quad i = 1,\ldots,m,\ j = 1,\ldots,n,$$

with test and ansatz functions ψ_i and φ_j having supports in D_1 and D_2, respectively. This kind of matrices corresponds to a Galerkin discretization of integral operators. The sub-block A results from a matrix partitioning which guarantees that the domains

$$D_1 = \bigotimes_{\nu=1}^d [a_\nu, b_\nu] \quad \text{and} \quad D_2 = \bigotimes_{\nu=1}^d [a'_\nu, b'_\nu]$$

are in the far-field of each other, i.e.,

$$\max\{\operatorname{diam} D_1, \operatorname{diam} D_2\} \le \eta \operatorname{dist}(D_1, D_2) \tag{1.3}$$

with a given parameter $\eta \in \mathbb{R}$. For properties of the hierarchical structure (matrix partitioning, complexity estimates) the reader is referred to the literature on hierarchical matrices; cf. [12, 5].

The structure of the article is as follows. In Sect. 2 we will review the adaptive cross approximation method. In order to exploit the smoothness of $\Pi_1 A$ and $A\Pi_2$, we present an alternative formulation of ACA. In Sect. 3 we will show that the matrices $\Pi_1 A$ and $A\Pi_2$ can be approximated using Chebyshev polynomials. This approximation requires the evaluation of the kernel function at transformed Chebyshev nodes, which has to be avoided if we want to use the original matrix entries. One solution to this problem is to find a least squares approximation. In Sect. 3.4 we show how this can be done in a purely algebraic and adaptive way. Another possibility is to replace the additional nodes by original ones which are close to Chebyshev nodes. Error estimates for this kind of approximation will be presented in Sect. 3.5.2. Finally, in Sect. 3.5.3 we will investigate an approximation that relies on the discrete cosine transform (DCT) – at least if the discretization nodes are close to transformed Chebyshev nodes, numerical evidence is given that the number of coefficients to store depends logarithmically on the accuracy but not on the matrix size of A. Numerical results support the derived estimates.

2 Adaptive Cross Approximation

In contrast to other methods like fast multipole, panel clustering, etc., the low-rank approximant resulting from the adaptive cross approximation is not generated by replacing the kernel function of the integral operator. The algorithm uses few of the original matrix entries to compute the low-rank matrix. Note that this does not require to build the whole matrix beforehand. The algorithm will specify which entries have to be computed.

The singular value decomposition would find the lowest rank that is required for a given accuracy. However, its computational complexity makes it unattractive for large-scale computations. ACA can be regarded as an efficient replacement which is tailored to asymptotically smooth kernels. Note that not the kernel function itself but only the information that the kernel is in this class of functions is required. This enables the design of a black-box algorithm for discrete integral operators with asymptotically smooth kernels.

Assume condition (1.3) holds for $A \in \mathbb{R}^{m\times n}$. Then the rows and columns of the matrix approximant UV^T, $U \in \mathbb{R}^{m\times k}$ and $V \in \mathbb{R}^{n\times k}$, are computed for $k = 1, 2, \ldots$ as

$$\begin{aligned}
\hat{u}_k &:= A_{1:m,j_k} - \sum_{\ell=1}^{k-1} u_\ell (v_\ell)_{j_k},\\
u_k &:= (\hat{u}_k)_{i_k}^{-1} \hat{u}_k, \quad \text{where } i_k \text{ is found from } |(\hat{u}_k)_{i_k}| = \|\hat{u}_k\|_\infty,\\
v_k &:= A^T_{i_k,1:n} - \sum_{\ell=1}^{k-1} (u_\ell)_{i_k} v_\ell.
\end{aligned}$$

The choice of the row index j_k is detailed in [6]. The iteration stops if a prescribed accuracy is reached, which can be checked by inspecting the norms of the last vectors u_k and v_k. The exponential convergence of ACA for the Nyström, the collocation, and the Galerkin method was proved in [2, 7, 6].

2.1 An alternative formulation of ACA

In [6] we have pointed out that for the computed approximant it holds that

$$UV^T = A_{1:m,j_{1:k}} A_k^{-1} A_{i_{1:k},1:n}, \tag{2.1}$$

where $A_k := A_{i_{1:k},j_{1:k}}$. The last expression is known as a pseudo-skeleton; see [10]. Since the methods of this section will be based on the pseudo-skeleton representation of the ACA approximant, we should construct and store $A_{1:m,j_{1:k}}$, $A_{i_{1:k},1:n}$, and A_k^{-1} instead of UV^T. In order to generate and apply A_k^{-1} in an efficient way, we use the LU decomposition of A_k.

Assume that pairs (i_ℓ, j_ℓ), $\ell = 1, \ldots, k$, have been found and assume that the normalized LU decomposition of the $k \times k$ matrix $A_k = L_k R_k$ has been computed. We find the new pivotal row i_{k+1} and column j_{k+1} as explained above. With the decomposition

$$A_{k+1} = \begin{bmatrix} A_k & b_k \\ a_k^T & c_k \end{bmatrix},$$

where $a_k := A_{i_{k+1},j_{1:k}}$, $b_k := A_{i_{1:k},j_{k+1}}$, and $c_k := A_{i_{k+1},j_{k+1}}$, the LU decomposition of A_{k+1} is given by

$$A_{k+1} = \begin{bmatrix} L_k & 0 \\ x_k^T & 1 \end{bmatrix} \begin{bmatrix} U_k & y_k \\ 0 & \alpha_k \end{bmatrix},$$

where x_k solves $R_k^T x_k = a_k$, y_k solves $L_k y_k = b_k$, and $\alpha_k = c_k - x_k^T y_k$. It is easy to see that

$$x_k^T = U_{i_{k+1},1:k}, \quad y_k^T = V_{j_{k+1},1:k}, \quad \text{and} \quad \alpha_k = (v_{k+1})_{j_{k+1}}.$$

This formulation of ACA has the same complexity $\mathcal{O}(k^2(m+n))$ as the original formulation. Due to the exponential convergence of ACA, the number of required steps k will be of the order $|\log \varepsilon|^d$, where $\varepsilon > 0$ is the prescribed approximation accuracy.

2.2 Recompression using the QR factorization

Since the columns of the matrices U and V generated by ACA are usually not orthogonal, they may contain redundancies, which can be removed by the following algebraic recompression technique; see [3]. This method may be regarded as the singular value decomposition optimized for rank-k matrices.

Assume we have computed the QR decompositions

$$U = Q_U R_U \quad \text{and} \quad V = Q_V R_V$$

of $U \in \mathbb{R}^{m\times k}$ and $V \in \mathbb{R}^{n\times k}$, respectively. Note that this can be done with $\mathcal{O}(k^2(m+n))$ operations. The outer-product of the two $k \times k$ upper triangular matrices R_U and R_V is then decomposed using the SVD of $R_U R_V^T$:

$$R_U R_V^T = \hat{U}\hat{\Sigma}\hat{V}^T.$$

Computing $R_U R_V^T$ and its SVD needs $\mathcal{O}(k^3)$ operations. Since $Q_U\hat{U}$ and $Q_V\hat{V}$ both are unitary,

$$A = UV^T = Q_U\hat{U}\hat{\Sigma}(Q_V\hat{V})^T$$

is an SVD of A. Together with the products $Q_U\hat{U}$ and $Q_V\hat{V}$, which require $\mathcal{O}(k^2(m+n))$ operations, the number of arithmetic operations of the SVD of a rank-k matrix sum up to $\mathcal{O}(k^2(m+n+k))$ operations. In addition to improving the blockwise approximation, one may also try to improve the block structure of the hierarchical matrix by agglomerating blocks; see [11] for a coarsening procedure. Although these techniques may reduce the required amount of storage, the asymptotic complexity of the approximation remains the same.

3 Approximation using Chebyshev polynomials

The matrices $A_{1:m,j_{1:k}}$ and $A_{i_{1:k},1:n}$ are submatrices of the original matrix block A. Hence, their matrix entries have the same smoothness properties as the original matrix A. The smoothness of the latter matrix was used by ACA. However, the smoothness of the former matrices has not been exploited so far. Our aim in this section is to approximate them using Chebyshev polynomials, i.e., we will construct coefficient matrices $X_1, X_2 \in \mathbb{R}^{k' \times k}$, $k' \approx k$, such that

$$\|A_{1:m,j_{1:k}} - B_1X_1\|_F \leq \|A_{1:m,j_{1:k}}\|_F \quad \text{and} \quad \|A^T_{i_{1:k},1:n} - B_2X_2\|_F \leq \|A_{i_{1:k},1:n}\|_F,$$

where $B_1 \in \mathbb{R}^{m\times k'}$ and $B_2 \in \mathbb{R}^{n\times k'}$ are explicitly given matrices which are generated from evaluating Chebyshev polynomials and which do not depend on the matrix entries of A. Combining the two approximations leads to

$$A \approx A_{1:m,j_{1:k}} A^{-1}_{i_{1:k},j_{1:k}} A_{i_{1:k},1:n} \approx B_1CB_2^T,$$

where $C = X_1A_k^{-1}X_2^T$. If $k' \leq 2k$, then one should store the entries of $C \in \mathbb{R}^{k'\times k'}$. Otherwise, it is more efficient to store C in outer product form

$$C = (X_1R_k^{-1})(X_2L_k^{-T})^T,$$

where L_k and R_k are the triangular matrices from Sect. 2.1.

The approximations B_1X_1 and B_2X_2 have the special property that the matrices B_1 and B_2 do not have to be stored. Only $X_1 \in \mathbb{R}^{k'\times k}$ and $X_2 \in \mathbb{R}^{k'\times k}$ will be stored and the matrices B_1 and B_2 will be recomputed every time they are used. Although for the "basis matrices" B_1 and B_2 any suitable matrices could be used, we favor matrices which correspond to Chebyshev polynomials due to their attractive numerical properties. Later it will be seen that the construction of B_1 and B_2 can be done with mk' and nk' operations, respectively. The matrix-vector multiplication with B_1X_1 takes $\mathcal{O}(k'(m + k))$ floating point operations.

Our aim in this section is to approximate the matrix $A_{1:m,j_{1:k}}$. The matrix $A_{i_{1:k},1:n}$ has a similar structure and its approximation can be done analogously. For notational convenience, we denote the restricted matrix $A_{1:m,j_{1:k}} \in \mathbb{R}^{m\times k}$, $m \gg k$, again by the symbol A.

3.1 Chebyshev approximation

We first consider one-dimensional interpolation in Chebyshev nodes

$$t_j := \frac{a+b}{2} + \frac{b-a}{2} \cdot \cos\frac{2j+1}{2p}\pi, \quad j = 0,\dots,p-1.$$

For $[a, b] = [-1, 1]$, the polynomial with the zeros t_j, $j = 0,\dots,p-1$, is the Chebyshev polynomial $T_p(t) := \cos(p \arccos(t))$ which satisfies the three term recurrence relation

$$T_0(x) = 1, \quad T_1(t) = t, \quad \text{and} \quad T_{p+1}(t) = 2tT_p(t) - T_{p-1}(t), \quad p = 1, 2, \dots. \tag{3.1}$$

Lemma 3.1. *Let $f \in C^p[a, b]$. The polynomial interpolation $\mathfrak{I}_p : C[a, b] \to \Pi_{p-1}$, $f \mapsto q$ such that $q(t_j) = f(t_j)$, $j = 0,\dots,p-1$, is uniquely solvable and $\mathfrak{I}_pf$ obeys*

$$\|f - \mathfrak{I}_pf\|_{C[a,b]} \leq \frac{2(b-a)^p}{4^p\,p!}\|f^{(p)}\|_{C[a,b]}.$$

The operator norm $\|\mathfrak{I}_p\| := \max\{\|\mathfrak{I}_p f\|_{C[a,b]} : f \in C[a,b]$ *satisfying* $\|f\|_{C[a,b]} = 1\}$ *is the Lebesgue constant which depends only logarithmically on* p*, i.e.,*

$$\|\mathfrak{I}_p\| \leq 1 + \frac{2}{\pi} \log p. \tag{3.2}$$

Moreover, the interpolation can be rewritten in the form $\mathfrak{I}_p f(t) = \sum_{i=0}^{p-1} c_i T_i \left(2\frac{t-a}{b-a} - 1\right)$*, where*

$$c_0 := \frac{1}{p} \sum_{j=0}^{p-1} f(t_j) \quad \text{and} \quad c_i := \frac{2}{p} \sum_{j=0}^{p-1} f(t_j) \cos i \frac{2j+1}{2p}\pi, \quad i = 1, \ldots, p-1. \tag{3.3}$$

Proof. For $[a,b] = [-1,1]$ see e.g. [20]. □

The previous properties of univariate interpolation at Chebyshev nodes can be exploited for interpolating multivariate functions $f : D \to \mathbb{R}$ given on a domain $D := \bigotimes_{\nu=1}^{d} [a_\nu, b_\nu]$.

Corollary 3.2. *Let the tensor product Chebyshev nodes (addressed by a multi-index) be given by*

$$\boldsymbol{t_j} := \bigotimes_{\nu=1}^{d} t_{j_\nu}, \qquad \boldsymbol{j} = (j_1, \ldots, j_d) \in \mathbb{N}_0^d,\ 0 \leq j_\nu < p.$$

We define the interpolation operator $\boldsymbol{\mathfrak{I}}_p : C(D) \to \Pi_{p-1}^d$, $\boldsymbol{\mathfrak{I}}_p f := \mathfrak{I}_p^{(1)} \cdots \mathfrak{I}_p^{(d)} f$ *where* $\mathfrak{I}_p^{(i)} f$ *denotes the univariate interpolation operator applied to the* i*-th argument of* f*.*

The interpolation error is bounded by

$$\|f - \boldsymbol{\mathfrak{I}}_p f\|_{C(D)} \leq \left(1 + \frac{2}{\pi} \log p\right)^{d-1} \sum_{\nu=1}^{d} \|f - \mathfrak{I}_p^{(\nu)} f\|_{C(D)}. \tag{3.4}$$

Proof. By the triangle inequality and due to (3.2), we obtain

$$\begin{aligned}
\|f - \boldsymbol{\mathfrak{I}}_p f\|_{C(D)} &\leq \|f - \mathfrak{I}_p^{(1)} f\|_{C(D)} + \|\mathfrak{I}_p^{(1)} (f - \mathfrak{I}_p^{(2)} \cdots \mathfrak{I}_p^{(d)}) f\|_{C(D)} \\
&\leq \|f - \mathfrak{I}_p^{(1)} f\|_{C(D)} + \|\mathfrak{I}_p^{(1)} (f - \mathfrak{I}_p^{(2)} f)\|_{C(D)} + \ldots + \|\mathfrak{I}_p^{(1)} \cdots \mathfrak{I}_p^{(d-1)} (f - \mathfrak{I}_p^{(d)} f)\|_{C(D)} \\
&\leq \sum_{\nu=1}^{d} \|f - \mathfrak{I}_p^{(\nu)} f\|_{C(D)} \prod_{j=1}^{\nu-1} \|\mathfrak{I}_p^{(j)}\| \\
&\leq \left(1 + \frac{2}{\pi} \log p\right)^{d-1} \sum_{\nu=1}^{d} \|f - \mathfrak{I}_p^{(\nu)} f\|_{C(D)}.
\end{aligned}$$

□

3.2 Construction of Approximations

The discretization method by which the sub-block A was obtained from the integral operator $\mathcal{K}$ is either the Nyström, the collocation, or the Galerkin method. We consider integral operators $\mathcal{K}$ of the form

$$(\mathcal{K}u)(x) = \int_\Omega S(x,y) u(y) \, \mathrm{d}y,$$

where S is a positive singularity function. Hence, each block A of the stiffness matrix takes the form

$$a_{ij} = \int_\Omega \int_\Omega S(x,y)\psi_i(x)\varphi_j(y)\,\mathrm{d}x\,\mathrm{d}y, \quad i=1,\dots,m,\ j=1,\dots,k, \tag{3.5}$$

where ψ_i and φ_j are non-negative finite element test and ansatz functions satisfying $\operatorname{supp}\psi_i \subset D_1$ and $\operatorname{supp}\varphi_j \subset D_2$. Note that Galerkin matrices formally include collocation and Nystöm matrices if one sets $\psi_i = \delta(\cdot - \tilde{x}_i)$ and $\varphi_j = \delta(\cdot - \tilde{y}_j)$ with Dirac's δ for some points $\tilde{x}_i$ and $\tilde{y}_j$.

Using the results stated in the previous section, for each $y \in \Omega$ we can define an approximating polynomial

$$\mathfrak{I}_{x,p} S(x,y) \approx S(x,y)$$

and a matrix $\tilde{A}^{\mathrm{CH}} \in \mathbb{R}^{m\times k}$ having the entries

$$\tilde{a}_{ij}^{\mathrm{CH}} := \int_\Omega \int_\Omega \mathfrak{I}_{x,p} S(x,y)\psi_i(x)\varphi_j(y)\,\mathrm{d}x\,\mathrm{d}y, \quad i=1,\dots,m,\ j=1,\dots,k. \tag{3.6}$$

Theorem 3.3. *Let D_1 be convex. If $c\,\gamma_1\eta < 1$ (cf.* (1.2), (1.3)*), then the following error estimate is fulfilled*

$$\|A - \tilde{A}^{\mathrm{CH}}\|_F \le \bar{c}\left(1 + \frac{2}{\pi}\log p\right)^{d-1} \left(\frac{\gamma_1\eta}{4}\right)^p \|A\|_F.$$

Proof. Due to Lemma 3.1 together with the asymptotic smoothness (1.2) of S we have

$$\begin{aligned}\|S(\cdot,y) - \mathfrak{I}_{x,p}^{(\nu)} S(\cdot,y)\|_{C(D_1)} &\le \frac{2(b_\nu - a_\nu)^p}{4^p\,p!}\|\partial_{x_\nu}^p S(\cdot,y)\|_{C(D_1)} \\ &\le 2c\left(\frac{\gamma_1}{4}\right)^p \left(\frac{\operatorname{diam} D_1}{\operatorname{dist}(D_1,D_2)}\right)^p \|S(\cdot,y)\|_{C(D_1)}\end{aligned}$$

Let $x^* \in D_1$ be chosen such that $|S(x^*,y)| = \|S(\cdot,y)\|_{C(D_1)}$. Then for some $\tilde{x} \in D_1$ we have

$$|S(x,y) - S(x^*,y)| = |(x-x^*)\nabla S(\tilde{x},y)| \le c\,\gamma_1 \frac{\operatorname{diam} D_1}{\operatorname{dist}(D_1,D_2)}\|S(\cdot,y)\|_{C(D_1)} \le c\,\gamma_1\eta \|S(\cdot,y)\|_{C(D_1)}$$

and thus $\|S(\cdot,y)\|_{C(D_1)} \le (1 - c\,\gamma_1\eta)^{-1}|S(x,y)|$. From (3.4) and the far-field condition (1.3) it follows that

$$\begin{aligned}|S(x,y) - \mathfrak{I}_{x,p} S(x,y)| &\le \left(1 + \frac{2}{\pi}\log p\right)^{d-1} \sum_{\nu=1}^{d} \|S(\cdot,y) - \mathfrak{I}_{p,x}^{(\nu)} S(\cdot,y)\|_{C(D_1)} \\ &\le \bar{c}\left(1 + \frac{2}{\pi}\log p\right)^{d-1} \left(\frac{\gamma_1\eta}{4}\right)^p |S(x,y)| \\ &= \bar{c}\left(1 + \frac{2}{\pi}\log p\right)^{d-1} \left(\frac{\gamma_1\eta}{4}\right)^p S(x,y).\end{aligned}$$

The last equality follows from the positivity of S. From

$$\begin{aligned}\|A - \tilde{A}^{\mathrm{CH}}\|_F^2 &= \sum_{i=1}^{m}\sum_{j=1}^{k} |a_{ij} - \tilde{a}_{ij}^{\mathrm{CH}}|^2 = \sum_{i=1}^{m}\sum_{j=1}^{k} \left(\int_{\mathbb{R}^d}\int_{\mathbb{R}^d} |S(x,y) - \mathfrak{I}_{x,p} S(x,y)|\psi_i(x)\varphi_j(y)\,\mathrm{d}x\,\mathrm{d}y\right)^2 \\ &\le \bar{c}^2\left(1 + \frac{2}{\pi}\log p\right)^{2(d-1)} \left(\frac{\gamma_1\eta}{4}\right)^{2p} \sum_{i=1}^{m}\sum_{j=1}^{k} |a_{ij}|^2\end{aligned}$$

one obtains the assertion. The Nyström case and the collocation case follow from the same arguments. □

The previous theorem shows the exponential convergence of $\tilde{A}^{\mathrm{CH}}$ provided $\gamma_1\eta < 4$. Instead of the singularity function S the kernel function of $\mathcal{K}$ may also contain normal derivatives of S. An example is the double-layer potential operator in boundary element methods. In this case

$$(\mathcal{K}u)(x) = \int_\Omega \zeta(x,y)S(x,y)u(y)\,\mathrm{d}y,$$

where the function $\zeta(\cdot,y)$ is polynomial for fixed $y \in \Omega$. We set

$$\tilde{a}_{ij}^{\mathrm{CH}} := \int_\Omega \int_\Omega \zeta(x,y)\mathfrak{I}_{x,p}S(x,y)\psi_i(x)\varphi_j(y)\,\mathrm{d}x\,\mathrm{d}y, \quad i=1,\dots,m,\ j=1,\dots,k. \tag{3.7}$$

It is obvious that a similar error estimate as presented in Theorem 3.3 can be obtained also for this kind of operators.

3.3 Evaluation of the approximation

The polynomial approximation (3.6) of the matrix entries (3.5) takes the form

$$\tilde{a}_{ij}^{\mathrm{CH}} = \sum_{\substack{\boldsymbol{\alpha}\in\mathbb{N}_0^d\\ \alpha_\nu<p}} \underbrace{\int_{\mathbb{R}^d} \prod_{\nu=1}^d T_{\alpha_\nu}(\xi^{(\nu)})\psi_i(x)\,\mathrm{d}x}_{=:b_{i\boldsymbol{\alpha}}} \cdot \underbrace{\int_{\mathbb{R}^d} c_{\boldsymbol{\alpha}}(y)\varphi_j(y)\,\mathrm{d}y}_{=:X_{\boldsymbol{\alpha}j}}, \qquad \xi^{(\nu)} = 2\frac{x^{(\nu)}-a_\nu}{b_\nu - a_\nu} - 1$$

with coefficient functions $c_{\boldsymbol{\alpha}}$. Hence, the matrix $\tilde{A}^{\mathrm{CH}}$ has the factorization $\tilde{A}^{\mathrm{CH}} = BX^{\mathrm{CH}}$, where $X^{\mathrm{CH}} \in \mathbb{R}^{k'\times k}$, $k' := p^d$, and $B \in \mathbb{R}^{m\times k'}$ has the entries

$$b_{i\boldsymbol{\alpha}} = \int_{\mathbb{R}^d} \prod_{\nu=1}^d T_{\alpha_\nu}(\xi^{(\nu)})\psi_i(x)\,\mathrm{d}x. \tag{3.8}$$

Let us consider the collocation and the Nyström case, i.e., we first assume that $\psi_i = \delta(\cdot - x_i)$. Then B has the entries

$$b_{i\boldsymbol{\alpha}} = \prod_{\nu=1}^d T_{\alpha_\nu}(\xi_i^{(\nu)}).$$

This matrix can be computed efficiently using the recurrence relation (3.1). For all $\boldsymbol{\alpha} \in \mathbb{N}^d$ satisfying $0 \le \alpha_\nu \le 1$, $\nu = 1,\dots,d$, let $b_{i\boldsymbol{\alpha}} = \prod_{\nu=1}^d (\xi_i^{(\nu)})^{\alpha_\nu}$. If there is $1 \le j \le d$ such that $\alpha_j \ge 2$, then

$$\begin{aligned} b_{i\boldsymbol{\alpha}} &= T_{\alpha_j}(\xi_i^{(j)}) \prod_{\nu\ne j} T_{\alpha_\nu}(\xi_i^{(\nu)}) \\ &= 2\xi_i^{(j)} T_{\alpha_j-1}(\xi_i^{(j)}) \prod_{\nu\ne j} T_{\alpha_\nu}(\xi_i^{(\nu)}) - T_{\alpha_j-2}(\xi_i^{(j)}) \prod_{\nu\ne j} T_{\alpha_\nu}(\xi_i^{(\nu)}) \\ &= 2\xi_i^{(j)} b_{i\boldsymbol{\alpha}-e_j} - b_{i\boldsymbol{\alpha}-2e_j}, \end{aligned}$$

where e_j is the j-th canonical vector. Hence, the matrix B can be set up with $\mathcal{O}(mk')$ arithmetic operations. For computing the matrix-vector product $w := BXv$ without explicitly constructing B one can use the Clenshaw-like algorithm shown in Alg. 1. This algorithm has complexity $\mathcal{O}(k'(k+m))$ and we state it for sake of simplicity for spatial dimension $d = 2$.

Now consider Galerkin matrices, i.e., ψ_i is a piecewise polynomial function on each of the M polyhedrons defining the computational domain. We assume that the support of each function ψ_i, Input: $k, k', m \in \mathbb{N}$, nodes $x_i, i = 1, \ldots, m$, coefficient matrix $X \in \mathbb{R}^{k' \times k}$, vector $v \in \mathbb{R}^k$.

Compute the vector $\tilde{v} = Xv$ and backup $\bar{v} = \tilde{v}$.
for $i = 1, \ldots, m$ **do**
  Restore $\tilde{v} = \bar{v}$.
  for $\alpha_2 = p, p-1, \ldots, 2$ **do**
    for $\alpha_1 = p, p-1, \ldots, 2$ **do**
      $\tilde{v}_{\alpha_1-1,\alpha_2} += 2\xi_i^{(1)} \tilde{v}_{\alpha_1,\alpha_2}$
      $\tilde{v}_{\alpha_1-2,\alpha_2} += -\tilde{v}_{\alpha_1,\alpha_2}$
    end for
    $\tilde{v}_{0,\alpha_2} += \xi_i^{(1)} \tilde{v}_{1,\alpha_2}$
    $\tilde{v}_{0,\alpha_2-1} += 2\xi_i^{(2)} \tilde{v}_{0,\alpha_2}$
    $\tilde{v}_{0,\alpha_2-2} += -\tilde{v}_{0,\alpha_2}$
  end for
  $w_i = \tilde{v}_{0,0} + \xi_i^{(2)} \tilde{v}_{0,1}$
end for

Output: Vector $w \in \mathbb{R}^m$.

Algorithm 1: Clenshaw-like algorithm for matrix-vector multiplication $w = BXv$.

$i = 1, \ldots, m$, is the union of at most μ elements τ_j, $j \in \mathcal{I}_i$, with $|\mathcal{I}_i| \leq \mu$, i.e., $\operatorname{supp} \psi_i = \bigcup_{j \in \mathcal{I}_i} \tau_j$. Each element τ_j is the image of the reference element τ under a mapping F_j. The restriction of ψ_i to each polyhedron τ_j is a polynomial of degree q, and we apply a cubature formula

$$\int_\tau f(x)\, \mathrm{d}x \approx \sum_{\ell=1}^{P} w_\ell f(x_\ell)$$

with weights w_ℓ and points x_ℓ, $\ell = 1, \ldots, P$, on the reference element τ as suggested in [14], i.e.,

$$b_{i\alpha} = \sum_{j \in \mathcal{I}_i} \int_{\tau_j} \prod_{\nu=1}^{d} T_{\alpha_\nu}(\xi^{(\nu)}) \psi_i(x)\, \mathrm{d}x = \sum_{j \in \mathcal{I}_i} \sum_{\ell=1}^{P} w_\ell \psi_i(F_{\tau_j}(x_\ell)) \prod_{\nu=1}^{d} T_{\alpha_\nu}(F_{\tau_j}(\xi_\ell^{(\nu)})).$$

Let $\mathcal{I} := \bigcup_{i=1}^m \mathcal{I}_i$. The computation of the matrix B can therefore be done by first computing the matrix

$$b'_{(j,\ell),\alpha} := \prod_{\nu=1}^{d} T_{\alpha_\nu}(F_{\tau_j}(\xi_\ell^{(\nu)})), \quad \alpha \in \mathbb{N}^d,\ \alpha_\nu < p,\ j \in \mathcal{I},\ \ell = 1, \ldots, P,$$

having at most $\mu m P$ rows. The matrix B' has the same structure as B in (3.8) and the number of cubature nodes is bounded by $P = \mathcal{O}(k')$. In a second step one computes the matrix

$$c_{i,(j,\ell)} := w_\ell \psi_i(F_{\tau_j}(x_\ell))$$

prior to computing the product

$$b_{i\alpha} = \sum_{j \in I_i} \sum_{\ell=1}^{P} c_{i,(j,\ell)} b'_{(j,\ell),\alpha}.$$

Note that the previous construction can also be applied to matrices (3.7).

As readily seen from (3.3), the computation of the coefficients X^{CH} requires additional evaluations of κ at the tensor Chebyshev nodes $\boldsymbol{t_j}$. Since our aim is a method that is based on the matrix entries and does not require the kernel function, in the following we investigate a least squares approximation.

3.4 Least Squares Approximation

Let $B \in \mathbb{R}^{m\times k'}$ be the matrix defined in (3.8). According to Theorem 3.3 there is $X^{\mathrm{CH}} \in \mathbb{R}^{k'\times k}$ such that

$$\|A - BX^{\mathrm{CH}}\|_F \leq \varepsilon \|A\|_F.$$

We have pointed out that the computation of X^{CH} is not desirable. Additionally, there may be a matrix $X^{\mathrm{LS}} \in \mathbb{R}^{k'\times k}$ which provides a better approximation than X^{CH}. Hence, we aim at solving the least squares problem

$$\text{find } X \in \mathbb{R}^{k'\times k} \text{ such that } \|A - BX\|_F \text{ is minimized.}$$

Let $B = U_B \Sigma V_B^T$, $\Sigma \in \mathbb{R}^{k'\times k'}$, be a singular value decomposition of B, which can be computed with complexity $\mathcal{O}((k')^2 m)$. Then $X^{\mathrm{LS}} := V_B \Sigma^+ U_B^T A$, where

$$(\Sigma^+)_{ij} = \begin{cases} \sigma_i^{-1}, & i = j \text{ and } \sigma_i \neq 0, \\ 0, & \text{else,} \end{cases}$$

is the best approximation with minimum Frobenius norm. The following errors estimate for $\tilde{A}^{\mathrm{LS}} := BX^{\mathrm{LS}}$ is an obvious yet important consequence.

Lemma 3.4. *For the approximation $\tilde{A}^{\mathrm{LS}}$ we obtain*

$$\|A - \tilde{A}^{\mathrm{LS}}\|_F \leq \bar{c}\left(1 + \frac{2}{\pi}\log p\right)^{d-1} \left(\frac{\gamma_1 \eta}{4}\right)^p \|A\|_F.$$

Proof. Since X^{LS} minimizes $\|A - BX\|_F$, we compare with the solution $\tilde{A}^{\mathrm{CH}} = BX^{\mathrm{CH}}$ obtained by interpolation at the Chebyshev nodes. □

In what follows we will devise an efficient adaptive strategy for the solution of the least squares problem. According to the previous lemma, we may assume that

$$\|A - BX^{\mathrm{LS}}\|_F \leq \varepsilon \|A\|_F$$

with arbitrary $\varepsilon > 0$. Depending on, for instance, the geometry, the columns of B can be close to linear dependent. Hence, the number of required columns of B may be significantly smaller than k'. Using the singular value decomposition of B, it is possible to construct a minimum orthonormal basis $U \in \mathbb{R}^{n\times k''}$ and coefficients $C \in \mathbb{R}^{k''\times k}$ such that

$$\|B - UC\|_F \leq \varepsilon \|B\|_F. \tag{3.9}$$

In this case we would have to store the matrix U for later computations. Since our aim is to generate the basis of approximation on the fly every time it appears in the computations, we have to find appropriate columns of B which are sufficient to represent the remaining columns. To this end, we construct a rank-revealing QR decomposition of B

$$B\Pi = QR = Q \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix},$$

where $Q \in \mathbb{R}^{m\times m}$ is unitary, $\Pi \in \mathbb{R}^{k'\times k'}$ is a permutation matrix, and $R \in \mathbb{R}^{m\times k'}$ is upper triangular. We determine $0 \le r_B \le k'$ such that $R_{11} \in \mathbb{R}^{r_B\times r_B}$ is non-singular and

$$\| \begin{bmatrix} 0 & R_{22} \end{bmatrix} X^{\mathrm{LS}} \|_F \le \varepsilon \|A\|_F.$$

Denote by Π_{r_B} the first r_B columns of Π. Hence, setting $X_1 := [I, R_{11}^{-1}R_{12}]\Pi^{-1}X^{\mathrm{LS}}$, we have

$$\begin{aligned}
\|A - B\Pi_{r_B}X_1\|_F &\le \|A - BX^{\mathrm{LS}}\|_F + \|BX^{\mathrm{LS}} - B\Pi_{r_B}X_1\|_F \\
&\le \varepsilon\|A\|_F + \|B\Pi\Pi^{-1}X^{\mathrm{LS}} - B\Pi_{r_B}X_1\|_F \\
&= \varepsilon\|A\|_F + \| \left(\begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} - \begin{bmatrix} R_{11} \\ 0 \end{bmatrix} \begin{bmatrix} I & R_{11}^{-1}R_{12} \end{bmatrix} \right) \Pi^{-1}X^{\mathrm{LS}}\|_F \\
&= \varepsilon\|A\|_F + \| \begin{bmatrix} 0 & R_{22} \end{bmatrix} \Pi^{-1}X^{\mathrm{LS}}\|_F \\
&\le 2\varepsilon\|A\|_F.
\end{aligned}$$

Although we have reduced the basis B to $B\Pi_{r_B}$, it still holds that

$$\min_{X\in\mathbb{R}^{r_B\times k}} \|A - B\Pi_{r_B}X\|_F \le 2\varepsilon\|A\|_F.$$

In addition to redundancies in the basis vectors B, the columns of A may be close to linear dependent. An extreme case is $A = 0$. Then there is no need to store a coefficient matrix X of size $r_B \times k$. Therefore, our aim is to find $X \in \mathbb{R}^{r\times k}$ with minimum $0 \le r \le r_B$ such that

$$\|A - B\Pi_r X\|_F = \min_{Y\in\mathbb{R}^{r\times k}} \|A - B\Pi_r Y\|_F \le 2\varepsilon\|A\|_F.$$

Let $Q = [Q_1, Q_2]$, where $Q_1 \in \mathbb{R}^{m\times r}$. Since

$$\|A - B\Pi_r X\|_F = \|Q^T A - Q^T B\Pi_r X\|_F = \|Q^T A - \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix} X\|_F = \| \begin{bmatrix} Q_1^T A - \hat{R}X \\ Q_2^T A \end{bmatrix} \|_F,$$

where $\hat{R} \in \mathbb{R}^{r\times r}$ is the leading $r \times r$ submatrix in R_{11}, it follows that

$$\|A - B\Pi_r X\|_F = \|Q_2^T A\|_F$$

if X solves $\hat{R}X = Q_1^T A$. This $X \in \mathbb{R}^{r\times k}$ satisfies $\|A - B\Pi_r X\|_F = \min_{Y\in\mathbb{R}^{r\times k}} \|A - B\Pi_r Y\|_F$. The required r can thus be found from the condition

$$\|Q_2^T A\|_F \le 2\varepsilon\|A\|_F.$$

The computation of $Q^T A \in \mathbb{R}^{m\times k}$ can be done with $\mathcal{O}(kk'm)$ operations provided Q is represented by a product of k' Householder transforms.

In total, we have the following algorithm which requires $\mathcal{O}((k')^2 m)$ flops.

Finally, in Tab. 3.4 we compare the asymptotic complexities of ACA, the recompression technique from this section (labeled “RACA”), and the standard method without any approximation.

3.5 Further topics

Subsequently, we discuss a further reduction in memory usage when the kernel is translation invariant, an interpolation approach which uses a subset of the original nodes, and a heuristic technique based on the discrete cosine transform. For simplicity, we only consider the Nyström case, i.e., our matrix block $A \in \mathbb{R}^{m\times k}$ is given by

$$a_{ij} = \kappa(x_i, y_j), \quad i = 1, \dots, m, \ j = 1, \dots, k.$$

Input: Matrix $A \in \mathbb{R}^{m\times k}$, approximation accuracy $\varepsilon > 0$, and $k' \in \mathbb{N}$.

1: Set up the matrix $B \in \mathbb{R}^{m\times k'}$.
2: Compute an SVD $B = U_B \Sigma V_B^T$ and the least squares coefficients $X^{\mathrm{LS}} = V_B \Sigma^+ U_B^T A \in \mathbb{R}^{k'\times k}$.
3: Compute a rank-revealing QR decomposition $B\Pi = QR$.
4: Determine $0 \le r_B \le k'$ and partition

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}$$

such that $\| \begin{bmatrix} 0 & R_{22} \end{bmatrix} X^{\mathrm{LS}} \|_F \le \varepsilon \|A\|_F$.
5: Compute $Q^T A \in \mathbb{R}^{m\times k}$ by k' Householder transforms.
6: Find $0 \le r \le r_B$ such that $\|Q_2^T A\|_F \le 2\varepsilon \|A\|_F$.
7: Solve $\hat{R}X = Q_1^T A$ for $X \in \mathbb{R}^{r\times k}$.

Output: Reduced rank r, permutation Π_r, and coefficient matrix $X \in \mathbb{R}^{r\times k}$.

Algorithm 2: Reduction of B and least squares solver.

	memory usage	matrix-vector multiplication	setup time
standard	mn	mn	mn
ACA	$k(m+n)$	$k(m+n)$	$k^2(m+n)$
RACA	kk'	$k'(m+n+k)$	$(k^2+(k')^2)(m+n)$

Table 1: Asymptotic complexities.

3.5.1 Translation invariant kernels

In case the matrix entries obey

$$a_{ij} = \kappa(x_i - y_j), \quad i = 1,\dots,m,\ j = 1,\dots,k,$$

we simultaneously approximate all columns of A. The interpolation at tensor product Chebyshev nodes is given by

$$\bar{a}_{ij} = (\mathfrak{I}_p \kappa)(x_i - y_j),$$

and allows for the error estimate in Theorem 3.3. Analogously, error estimates for the least squares approximation and for the interpolation at perturbed Chebyshev nodes follow. The crucial point is the further reduction in storage, since we need only one vector of coefficients $\bar{X}^{\mathrm{CH}} \in \mathbb{R}^{k'}$, $k' = p^d$. In total, the matrix block $A \in \mathbb{R}^{m\times k}$ is compressed to k' coefficients – compared to kk' coefficients when no translation invariance is exploited.

3.5.2 Interpolation at perturbed Chebyshev nodes

Up to now, we have defined an interpolation operator $\mathfrak{I}_p$ which is based on the Chebyshev nodes. The computation of the coefficients X^{CH} requires the evaluation of the kernel function at additional nodes and hence, we have proposed a method which is based on a least squares problem.

In this section we will investigate the error if the interpolation is based on some of the original nodes x_i, $i = 0,\dots,p-1$, instead of the Chebyshev nodes. For simplicity we consider this problem in one spatial dimension, i.e., we consider the interpolation problem

$$\sum_{j=0}^{p-1} c_j^{\mathrm{IP}} T_j(x_i) = f(x_i), \quad i = 0, \ldots, p-1.$$

The following result states that a similar result as in Lemma 3.1 and Theorem 3.3 can be achieved if points are chosen that are close to the Chebyshev nodes. The matrix $\tilde{A}^{\mathrm{IP}}$ denotes the approximant resulting from this kind of interpolation. The generalization to dimensions $d > 1$ using (3.4) is straightforward for sampling points which lie on a perturbed tensor product grid $(t_j \pm \delta_j, t_\ell \pm \delta_{j,\ell}, \ldots)^T \in \mathbb{R}^d$, $j, \ell = 0, \ldots, p-1$ with perturbations $|\delta_j|, |\delta_{j,\ell}| \le \delta$.

Lemma 3.5. *Let $p \in \mathbb{N}$, $\delta \le \min\{\frac{b-a}{4}, \frac{b-a}{2}(p-1)^{-2}\}$ and p perturbed Chebyshev nodes $x_i \in [a,b]$ with $|x_i - t_i| \le \delta$ for $i = 0, \ldots, p-1$ be given. Then, the polynomial interpolation at these nodes $x_i \in [a,b]$ obeys*

$$\|A - \tilde{A}^{\mathrm{IP}}\|_F \le \bar{c}(p+1) \left(\frac{\gamma_1 \eta}{4}\right)^p \|A\|_F.$$

Moreover, if $|x_\ell - x_i| \ge \zeta |t_\ell - t_i|$ for all $\ell, i = 0, \ldots, p-1$ and some $\zeta \ge p^{1/(1-p)}$, then the Lebesgue constant for these nodes, denoted by $\|\tilde{\mathfrak{I}}_p\|$, can be bounded by

$$\|\tilde{\mathfrak{I}}_p\| \le p^2 (1 + \frac{2}{\pi} \log p).$$

Proof. For simplicity assume $[a,b] = [-1,1]$ and hence, $\delta \le \min\{\frac{1}{2}, (p-1)^{-2}\}$. The standard error estimate for polynomial interpolation, i.e. $q(x_i) = f(x_i)$, $i = 0, \ldots, p-1$, $q \in \Pi_{p-1}$, reads

$$\|f - q\|_{C[-1,1]} \le \frac{\|\omega\|_{C[-1,1]}}{p!} \|f^{(p)}\|_{C[-1,1]}$$

with $\omega(x) := \prod_{i=0}^{p-1}(x - x_i)$. First, note that

$$\omega'(x) = \sum_{\ell=0}^{p-1} \prod_{\substack{i=0 \\ i \ne \ell}}^{p-1} (x - x_i), \quad \omega^{(2)}(x) = \sum_{k=0}^{p-1} \sum_{\substack{\ell=0 \\ \ell \ne k}}^{p-1} \prod_{\substack{i=0 \\ i \ne \ell,k}}^{p-1} (x - x_i), \quad \ldots, \quad \omega^{(p)}(x) = p!.$$

Moreover, Markov's inequality, see e.g. [9, p. 97], yields

$$\|\omega'\|_{C[-1,1]} \le (p-1)^2 \|\omega\|_{C[-1,1]}, \quad \|\omega^{(2)}\|_{C[-1,1]} \le (p-2)^2 (p-1)^2 \|\omega\|_{C[-1,1]}, \quad \ldots.$$

For notational convenience let $\bar{\omega}(x) = \prod_{i=0}^{p-1}(x - t_i) = T_p(x)/2^{p-1}$. We estimate $|\omega(x)|$ by

$$\begin{aligned}
|\omega(x)| &= |(x - t_0 + t_0 - x_0) \cdot \ldots \cdot (x - t_{p-1} + t_{p-1} - x_{p-1})| \\
&\le \prod_{i=0}^{p-1} |x - t_i| + \sum_{\ell=0}^{p-1} \delta \prod_{\substack{i=0 \\ i \ne \ell}}^{p-1} |x - t_i| + \sum_{k=0}^{p-1} \sum_{\ell=k+1}^{p-1} \delta^2 \prod_{\substack{i=0 \\ i \ne \ell,k}}^{p-1} |x - t_i| + \ldots \\
&\le |\bar{\omega}(x)| + \delta \left|\bar{\omega}'(x)\right| + \delta^2 \left|\bar{\omega}^{(2)}(x)\right| + \ldots + \delta^{p-1} \left|\bar{\omega}^{(p-1)}(x)\right| + \delta^p |\bar{\omega}^{(p)}(x)| \\
&\le (p+1) \|\bar{\omega}\|_{C[-1,1]} \le \frac{p+1}{2^{p-1}},
\end{aligned}$$

which proves

$$\|f - q\|_{C[-1,1]} \le 2 \frac{(p+1)(b-a)^p}{4^p \, p!} \|f^{(p)}\|_{C[-1,1]}.$$

Hence, compared with Lemma 3.1, which is the basis for Theorem 3.3, we obtain an additional factor $p+1$. The assertion follows from the same arguments as in the proof of Theorem 3.3.

In what follows, the Lebesgue is bounded by using the same technique for the numerator and the assumption $|x_\ell - x_i| \geq \zeta |t_\ell - t_i|$ for the denominator:

$$\|\tilde{\mathfrak{I}}_p\| = \max_{x\in[-1,1]} \sum_{\ell=0}^{p-1} \prod_{\substack{i=0\\ i\neq\ell}}^{p-1} \frac{|x-x_i|}{|x_\ell - x_i|} \leq \max_{x\in[-1,1]} \sum_{\ell=0}^{p-1} \prod_{\substack{i=0\\ i\neq\ell}}^{p-1} \frac{p}{\zeta^{p-1}} \frac{|x-t_i|}{|t_\ell - t_i|} \leq p^2 \|\mathfrak{I}_p\|.$$

□

3.5.3 Cosine transforms

Discrete Fourier transforms (DFTs) and similar methods are used in signal and image processing, especially for data compression, because under certain assumptions most of the signal information tends to be concentrated in a few low-frequency components. Our aim is to approximate the matrix $A \in \mathbb{R}^{m\times k}$ by removing small high-frequency components.

The (univariate) discrete cosine transform of type two (DCT-II) of a vector $a \in \mathbb{R}^m$ is defined as

$$c_k = \sum_{i=0}^{m-1} a_i \cos k \frac{2i+1}{2m}\pi, \quad k = 0, \ldots, m-1.$$

We note that the computation of the Chebyshev coefficients in (3.3) is (up to normalization) a DCT of length p. Now lets assume for the moment, that the nodes x_i, $i = 1, \ldots, m$, are the $m = \bar{m}^d$ Chebyshev nodes. Then a (multivariate) DCT applied to the j-th column of the matrix $A \in \mathbb{R}^{m\times k}$ computes exactly the coefficients of the interpolating polynomial $\mathfrak{I}_{\bar{m}}\kappa(\cdot, y_j)$. Due to the fact that the columns of A are samples of a smooth function, we have exponentially fast decay in the DCT-coefficients. Hence, we suggest to keep only the "lowest" $k' = p^d$ coefficients which again results in a storage reduction from $\mathcal{O}(km)$ to $\mathcal{O}(kk')$. The DCT-II and its inverse, which is (up to normalization) given by the so called DCT-III are equivalent to symmetric real valued DFTs of length $4m$ and can be computed with $\mathcal{O}(m \log m)$ arithmetic operations. This allows for a fast matrix-vector multiplication (of order $\mathcal{O}(kk' + m \log m)$) with the approximated matrix.

However note, that this technique completely neglects the given nodes and indeed fails for less regular nodes as shown in the following numerical results.

4 Numerical results

We start by a very simple example illustrating the exponential convergence of the proposed schemes. Consider the matrix block $A \in \mathbb{R}^{m\times k}$, $m = 1000$, $k = 20$,

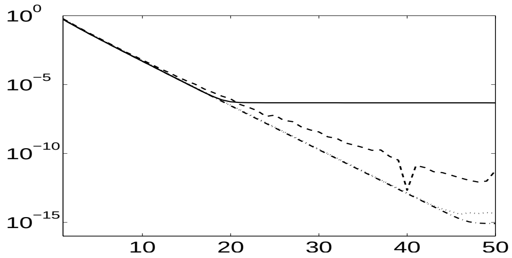
$$a_{ij} = \frac{1}{x_i - y_j}, \quad i = 1, \ldots, m,\ j = 1, \ldots, k,$$

with randomly chosen nodes $y_j \in [0, 2]$ and (a) perturbed Chebyshev nodes $x_i = 5 + 2.5 \cos \frac{2i-1}{2m}\pi + \delta_i$, $|\delta_i| \leq 10^{-6}$ and (b) randomly chosen nodes $x_i \in [2.5, 7.5]$. For each row of A, the approximations are obtained by

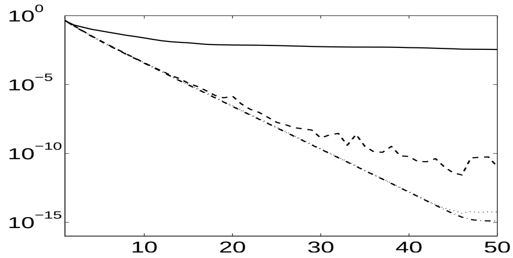
1. a truncated DCT (solid); see Sect. 3.5.3,
2. interpolation at p Chebyshev nodes, using additional evaluations (dotted); see Sect. 3.2,

3. the least squares procedure (dash-dot); see Sect. 3.4, and
4. the interpolation at p chosen original nodes (dashed); see Sect. 3.5.2.

Fig. 1 shows that methods 2–4 lead to exponential convergence for both perturbed Chebyshev nodes and randomly chosen nodes. Method 1, however, converges exponentially only up to the size of the perturbation δ in (a). For randomly chosen nodes method 1 did not converge at all.



(a) Perturbed Chebyshev nodes.



(b) Randomly chosen nodes.

Figure 1: Error of the matrix approximations in the Frobenius norm with respect to the polynomial degree $p = 1, \ldots, 50$.

We proceed with a more realistic example. The single layer potential operator $\mathcal{V} : H^{-1/2}(\Omega) \to H^{1/2}(\Omega)$ defined by

$$(\mathcal{V}u)(x) = \frac{1}{4\pi} \int_{\Omega} \frac{u(y)}{|x-y|} \, \mathrm{d}s_y$$

is used to test the proposed algorithm which is based on the least squares solution. In the following experiments Ω is the surface from Fig. 2.

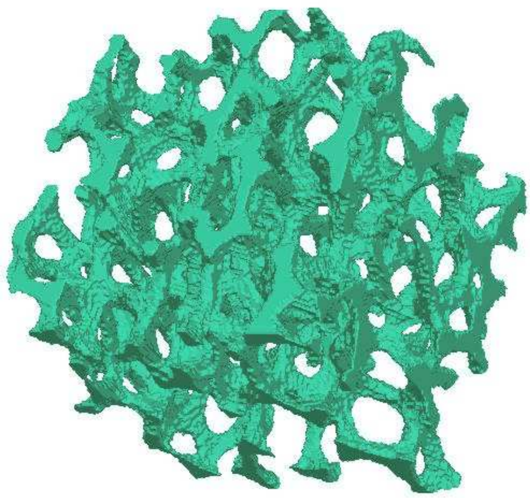


Figure 2: The computational surface.

A Galerkin discretization with piecewise constants $\varphi_i = \psi_i$, $i = 1, \ldots, N$, leads to the matrix $V \in \mathbb{R}^{N \times N}$ with entries

$$v_{ij} = (\mathcal{V}\varphi_i, \varphi_j), \quad i, j = 1, \ldots, N,$$

which is symmetric since $\mathcal{V}$ is self-adjoint with respect to $(\cdot,\cdot)_{L^2(\Omega)}$. Therefore, it is sufficient to approximate the upper triangular part of V by an hierarchical matrix.

Table 2 and 3 compare the hierarchical matrix approximation generated by ACA with and without coarsening (see Sect. 2.2) and by the (least squares based) method from this article, which is abbreviated with "RACA". We test these methods on three discretizations of the surface from Fig. 2. Columns two, five and eight show the memory consumption in MByte, columns three, six and nine contain the memory consumption per degree of freedom in KByte. The cpu-time required for the construction of the respective approximation can be found in the remaining columns four, seven and ten. The relative accuracy of the approximation in Frobenius norm is ε. All tests were done on a shared memory system with four Intel Xeon processors at 3 GHz.

	ACA			coarsened ACA			RACA		
N	MB	KB/N	time [s]	MB	KB/N	time [s]	MB	KB/N	time [s]
28 968	115.7	4.1	42.5	93.0	3.3	46.2	59.3	2.1	43.8
120 932	607.8	5.1	229.7	490.1	4.1	245.9	244.3	2.1	237.1
494 616	2836.6	5.9	1113.9	2342.3	4.8	1175.0	963.0	2.0	1155.2

Table 2: Approximation results for $\varepsilon = 1e-3$.

	ACA			coarsened ACA			RACA		
N	MB	KB/N	time [s]	MB	KB/N	time [s]	MB	KB/N	time [s]
28 968	186.2	6.6	62.7	150.8	5.6	69.5	127.4	4.5	66.4
120 932	992.7	8.4	342.2	809.8	6.9	371.9	536.5	4.5	359.4
494 616	4727.7	9.8	1673.7	3928.4	8.1	1795.4	2214.2	4.5	1714.3

Table 3: Approximation results for $\varepsilon = 1e-4$.

Apparently, RACA produces approximations with much lower memory consumption than obtained by the ACA method even after coarsening. The numbers in the column "KB/N" supports our complexity estimates: The asymptotic complexity of the storage behaves linearly if the approximation accuracy is kept constant. The time required for RACA is less than the time for coarsened ACA.

References

[1] B. K. Alpert, G. Beylkin, R. Coifman, and V. Rokhlin. Wavelet-like bases for the fast solution of second-kind integral equations. *SIAM J. Sci. Comput.*, 14:159–184, 1993.

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

[3] M. Bebendorf. *Effiziente numerische Lösung von Randintegralgleichungen unter Verwendung von Niedrigrang-Matrizen.* PhD thesis, Universität Saarbrücken, 2000. dissertation.de, Verlag im Internet, 2001. ISBN 3-89825-183-7.

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

[5] M. Bebendorf. *Hierarchical matrices: a means to efficiently solve elliptic boundary value problems.* Habilitation thesis, University of Leipzig, Lecture Notes in Computational Science and Engineering (LNCSE). Springer, 2007, to appear.

[6] M. Bebendorf and R. Grzhibovskis. Accelerating Galerkin BEM for Linear Elasticity using Adaptive Cross Approximation. *Mathematical Methods in the Applied Sciences*, 29:1721–1747, 2006.

[7] M. Bebendorf and S. Rjasanow. Adaptive low-rank approximation of collocation matrices. *Computing*, 70:1–24, 2003.

[8] S. Börm, M. Löhndorf, and J. M. Melenk. Approximation of integral operators by variable-order interpolation. *Numer. Math.*, 99:605–643, 2005.

[9] R. A. DeVore and G. G. Lorentz. *Constructive Approximation.* Springer–Verlag, Berlin, 1993.

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

[11] L. Grasedyck. Adaptive recompression of $\mathcal{H}$-matrices for BEM. *Computing*, 74:205–223, 2005.

[12] L. Grasedyck and W. Hackbusch. Construction and arithmetics of $\mathcal{H}$-matrices. *Computing*, 70:295–334, 2003.

[13] L. F. Greengard and V. Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. In *Acta numerica*, volume 6, pages 229–269. Cambridge Univ. Press, 1997.

[14] A. Grundmann and H. M. Möller. Invariant integration formulas for the n-simplex by combinatorial methods. *SIAM J. Numer. Anal.*, 15:282–290, 1978.

[15] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing*, 62:89–108, 1999.

[16] W. Hackbusch and S. Börm. Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices. *Computing*, 69:1–35, 2002.

[17] W. Hackbusch and B. N. Khoromskij. A sparse $\mathcal{H}$-matrix arithmetic. Part II: Application to multi-dimensional problems. *Computing*, 64:21–47, 2000.

[18] W. Hackbusch, B. N. Khoromskij, and S. A. Sauter. On $\mathcal{H}^2$-matrices. In H.-J. Bungartz, R. H. W. Hoppe, and Ch. Zenger, editors, *Lectures on Applied Mathematics*, pages 9–29. Springer-Verlag, Berlin, 2000.

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

[20] A. Schönhage. *Approximationstheorie.* de Gruyter, Berlin, 1971.