Universität des Saarlandes

Fachrichtung 6.1 – Mathematik

Preprint

Adaptive Low-Rank Approximation of Collocation Matrices

M. Bebendorf and S. Rjasanow

Preprint No. 39

Saarbrücken 2001

Universität des Saarlandes

Fachrichtung 6.1 – Mathematik

Adaptive Low-Rank Approximation of Collocation Matrices

M. Bebendorf

Max Planck Institute for
Mathematics in the Sciences
Inselstraße 22–26
04103 Leipzig
Germany
E-Mail: bebendorf@mis.mpg.de

S. Rjasanow

Fachrichtung 6.1 – Mathematik
Universität des Saarlandes
Postfach 151150
66041 Saarbrücken
Germany
E-Mail: rjasanow@num.uni-sb.de

submitted: September 24, 2001

Preprint No. 39

Saarbrücken 2001

Edited by
FR 6.1 – Mathematik
Im Stadtwald
D–66041 Saarbrücken
Germany

Fax:	+ 49 681 302 4443
e-mail:	preprint@math.uni-sb.de
WWW:	http://www.math.uni-sb.de/

Abstract

This article deals with the solution of integral equations using collocation methods with almost linear complexity. This is done by generating a blockwise low-rank approximation to the system matrix. In contrast to fast multipole and panel clustering the proposed algorithm is based on only few entries from the original matrix. In this article the results concerning matrix approximation from [1] are generalized to collocation matrices and improved. Furthermore, we present a new algorithm for matrix partitioning that dramatically reduces the number of blocks generated.

AMS Subject Classification: 41A63, 41A80, 65D05, 65D15, 65F05, 65F30
Keywords: integral equations, hierarchical matrices, low-rank approximation, fast solvers

1 Introduction

This article is concerned with the efficient solution of integral equations

$$\int_D \kappa(x,y)\, u(x)\,\mathrm{d}\mu_x = f(y), \quad y \in D \tag{1}$$

with given right hand side f. The domain of integration D is dim-dimensional, i.e. there are two constants $c_1, c_2 > 0$ so that for $z \in D$

$$c_1 r^{dim} \le \mu(D \cap K_r(z)) \le c_2 r^{dim} \quad \text{for all } r > 0. \tag{2}$$

Depending on dim the functional μ denotes the surface or the volume measure and $K_r(z)$ is the Euclidian ball with center z and radius r.
The kernel $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ in (1) is assumed to be *asymptotically smooth*, i.e. $\kappa(x,\cdot) \in C^\infty(\mathbb{R}^d \setminus \{x\})$ for all $x \in \mathbb{R}^d$ and for all multiindices $\alpha \in \mathbb{N}_0^d$ it holds that

$$|D_y^\alpha \kappa(x,y)| \le c_p\, |x-y|^{g-p}, \quad p = |\alpha|,$$

where c_p depends only on p. As usual we denote by D_y^α the partial derivative

$$D_y^\alpha = \left(\frac{\partial}{\partial y_1}\right)^{\alpha_1} \cdots \left(\frac{\partial}{\partial y_d}\right)^{\alpha_d}.$$

In order to solve equation (1) the domain of integration D is divided into smaller, possibly overlapping pieces X_i, $i \in I := \{1, \ldots, N\}$, which in turn are assembled from sets π_i, i.e. for each X_i there is an index set $\mathcal{J}_i$, $\#\mathcal{J}_i = \nu_1$ so that

$$X_i = \bigcup_{j\in\mathcal{J}_i} \pi_j \quad \text{and} \quad \operatorname{diam} X_i \le \nu_1 \max_{j\in\mathcal{J}_i} \operatorname{diam} \pi_j. \tag{3}$$

Moreover, each $\pi \in P$ is in at most ν_2 of the sets X_i, $i \in I$.
Besides the *Galerkin method* the *collocation method* is often used to solve (1) numerically. The solution u is approximated from a finite dimensional ansatz space, i.e. the approximation u_h is sought of the form $u_h = \sum_{j=1}^N x_j\varphi_j$. The set X_j may be looked at as the support of the ansatz function φ_j. In the case of collocation methods we are led to the following system of linear equations

$$Ax = b, \quad a_{ij} = \int_D \kappa(x, y_i)\, \varphi_j(x)\, \mathrm{d}x, \; b_i = f(y_i), \quad i, j \in I, \tag{4}$$

where $y_i \in X_i$ are the so called *collocation points*. Generally speaking the matrix entry a_{ij} arises from the interaction of the sets X_i and X_j:

$$a_{ij} = (\mathcal{L}_j\kappa)(y_i), \quad i, j \in I,$$

where for each of the operators $\mathcal{L}_j$ there is a continuous linear functional $\tilde{l}_j \in B'(X_j)$, the dual to the space of bounded functions on X_j, such that $(\mathcal{L}_j\kappa)(y) = \langle \kappa_y, \tilde{l}_j\rangle$ for all $y \in \mathbb{R}^d \setminus X_j$. Here the function κ_y is defined by $\kappa_y(x) = \kappa(x, y)$, $x \in D$. This includes the operators investigated in [1]

$$(\mathcal{L}_j\kappa)(y) = \kappa(x_j, y),$$

where $x_j \in X_j$, $j \in I$.
A naive strategy for the solution of (4) would need at least $\mathcal{O}(N^2)$ operations and memory. Methods such as fast multipole [3, 7] and panel clustering [6] provide an approximation to the solution vector x in almost linear complexity. These methods are based on explicitly given kernel approximations by degenerate kernels, i.e. a finite sum of separable functions, which may be seen as a blockwise low-rank approximation of the system matrix. The blockwise approximant permits a fast matrix-vector multiplication, which can be exploited in iterative solvers, and can be stored efficiently. In contrast to the methods mentioned we will generate the low-rank approximant from the matrix itself using only few entries and without using any explicit a priori known degenerate kernel approximation.
In the case of boundary integral equations for π_i so called panels are often used. For volume integrals π_i are usually tetrahedra. We assume this set $P = \{\pi_i, i = 1, \ldots, m\}$ to be regular and quasi-uniform, i.e. there is a constant c_u so that

$$c_u\mu(\pi) > h^{dim} \quad \text{for all } \pi \in P, \tag{5}$$

where $h = \max_{\pi \in P} \operatorname{diam} \pi$. This guarantees in particular that we are able to find $c_3 > 0$ such that $\operatorname{diam} X_i \leq c_3 \operatorname{diam} X_j$, $i, j \in I$.

2 Matrix Partitioning

The aim of this section is to construct a partitioning of the coefficient matrix. We will divide the index set $I \times I$ into pairwise distinct subsets $t_1^j \times t_2^j$, $j = 1, \dots, n$ so that

$$I \times I = \bigcup_{j=1,\dots,n} t_1^j \times t_2^j$$

and for each pair (t_1, t_2) one of the following conditions holds:

(i) $\min\{t_1, t_2\} = 1$

(ii) $\operatorname{diam} X_{t_2} \leq \eta \operatorname{dist}(X_{t_1}, X_{t_2})$,

where for $t \subset I$ we set $X_t = \bigcup_{i \in t} X_i$. The parameter η is the so called relative distance. In contrast to the pairs (t_1, t_2) generated in [1, 5] we admit pairs that cannot be represented by a single block. Thus, these pairs are referred to as generalized blocks.
If a pair (t_1, t_2) fulfills condition (i) then each element from the corresponding block will be generated and stored. For all other pairs condition (ii) is valid and in section 3 we will investigate an algorithm for the approximation by matrices of low rank. Both storage and multiplication of the resulting matrix with a vector can be done blockwise, taking advantage of the efficient representation of low-rank matrices.
Instead of going through all possible partitionings the construction will be based on the so called cluster tree, which is frequently used in this field of research, cf. [6, 5], and contains a hierarchy of partitionings of I. Given a mapping S which associates to an index set $t \subset I$ a set of pairwise distinct subsets t_j, $j = 1, \dots, s_t$, so that

$$t = \bigcup_{j=1,\dots,s_t} t_j, \qquad 2 \leq s_t \leq q$$

and $S(t) = \varnothing \iff \#t = 1$ the cluster tree is constructed by recursively applying S to the index set I. Here $q \in \mathbb{N}$ denotes the maximum degree of nodes in the cluster tree. If we assume the ratios $\#t/\#t_j$ to be bounded by $R \in \mathbb{R}$ from below, the maximum depth of the cluster tree is of order $L = \log_R N$. Furthermore, we assume that $S(t)$ can be evaluated in $\mathcal{O}(\#t)$ operations. An example for S can be found in [2].

The complexity of an algorithm generating a partition of $I \times I$ with the properties described above is mainly determined by the cost of evaluating

$$\operatorname{dist}(X, Y) = \inf_{x \in X, y \in Y} |x - y|,$$

which depend on the product of the complexities of X and Y. Thus instead of calculating the far field $F(t) = \{i \in I : \operatorname{diam} X_t \leq \eta \operatorname{dist}(X_i, X_t)\}$ of $t \subset I$ we would do better to determine the following set

$$\tilde{F}(t) = \{i \in I : \operatorname{diam} X_t \leq \frac{\eta}{1+\eta} \operatorname{dist}(X_i, z_t)\},$$

where $z_t \in \mathbb{R}^d$ denotes some point in X_t. Consequently, the near field is replaced by $\tilde{N}(t) = I \setminus \tilde{F}(t)$. It can easily be seen that $\tilde{F}(t) \subseteq F(t)$, and the calculation of $\tilde{F}(t_1) \cap t_2$ can be performed with at most $c_4(\#t_1 + \#t_2)$ operations.

Algorithm 2.1. *Partitioning*(t_1, t_2)
If $\#t_1 = 1$ then *add* (t_1, t_2) *to the list of blocks.*
else begin
Subdivide t_1 *into* $S(t_1) = \{t_1^1, \ldots, t_1^{s_{t_1}}\}$.
for $j = 1, \ldots, s_{t_1}$ begin
if $t_2 \cap \tilde{F}(t_1^j) \neq \varnothing$
The block $(t_1^j, t_2 \cap \tilde{F}(t_1^j))$ *fulfills condition* (ii). *Add it to the list of blocks.*
call *Partitioning*$(t_1^j, t_2 \setminus \tilde{F}(t_1^j))$.
end
end

A partitioning with the desired properties is obtained by applying *Partitioning* to (I, I).

2.1 Computational complexity

In this subsection we will estimate the computational cost of the algorithm above. Quasi-uniformity of P immediately leads to

Lemma 2.2. *There is a constant* c_5 *such that* $\#t/c_5 \leq \mu(X_t)\, h^{-dim} \leq c_5 \#t$ *for all* $t \subset I$.

Proof. Using (2) and (3) we see that

$$\mu(X_t) \leq \sum_{i \in t} \sum_{j \in \mathcal{J}_i} \mu(\pi_j) \leq \nu_1 \sum_{i \in t} \max_{j \in \mathcal{J}_i} \mu(\pi_j) \leq c_2 \nu_1 \#t \max_{\pi \in P} \operatorname{diam}^{dim} \pi = c_2 \nu_1 \#t\, h^{dim}.$$

On the other hand
$$\nu_2\mu(X_t) \geq \sum_{i\in t}\mu(X_i).$$
Since P is quasi-uniform $\mu(X_i) \geq \nu_1 \min_{j\in\mathcal{J}_i}\mu(\pi_j) \geq \nu_1 h^{dim}/c_u$. $\square$

Lemma 2.3. *Let $t\subset I$. Then $\#\tilde{N}(t) \leq c(\eta)\#t$, $c(\eta) = \tilde{c}(1+1/\eta)^{dim}$ holds.*

Proof. From (5) follows $\operatorname{diam} X_i \leq c_3 \operatorname{diam} X_t$, $i\in I$. Since $\operatorname{dist}(X_i,z_t) < (1+1/\eta)\operatorname{diam} X_t$ for $i\in\tilde{N}(t)$ the ball $K_r(z_t)$ with $r=(c_3+1+1/\eta)\operatorname{diam} X_t$ covers $X_{\tilde{N}(t)} \supset X_t$. Now
$$\mu(X_{\tilde{N}(t)}) \leq c_2 r^{dim} = c_2(c_3+1+1/\eta)^{dim}\operatorname{diam}^{dim} X_t.$$
Using (2) and $\operatorname{diam} X_t \leq r$ we obtain
$$\mu(X_{\tilde{N}(t)}) \leq \frac{c_2}{c_1}(c_3+1+1/\eta)^{dim}\mu(X_t).$$
Lemma 2.2 leads to the desired estimate. $\square$

Theorem 2.4. *The number of operations and the amount of storage to perform the recursion Partitioning(I,I) is of order $\eta^{-dim}N\log_R N$. The number of blocks generated is of order N.*

Proof. The recursion *Partitioning*(I,I) descending the cluster tree generates at most two blocks in each node. The number of nodes is limited by qN, so the number of generated blocks is of order N.
Let N_{t_1} denote the number of operations needed to carry out a part *Partitioning*(t_1,t_2) of the whole recursion *Partitioning*(I,I). We will show that $N_{t_1} \leq \bar{c}(\eta)\#t_1\log_R\#t_1$, $\bar{c}(\eta) = c_4(1+qc(\eta))$. According to the remark preceding Algorithm 2.1 for the number of operations it holds that
$$N_{t_1} \leq \sum_{j=1}^{s_{t_1}} c_4(\#t_1^j + \#t_2) + N_{t_1^j} \leq c_4\#t_1(1+qc(\eta)) + \sum_{j=1}^{s_{t_1}} N_{t_1^j}.$$
For the last estimate we used Lemma 2.3 because *Partitioning* is only applied to pairs (t_1,t_2) for which $t_2\subset\tilde{N}(t_1)$ is valid. Thus
$$\begin{aligned}
N_{t_1} &\leq \sum_{j=1}^{s_{t_1}} \bar{c}(\eta)\#t_1^j\log_R\#t_1^j + \bar{c}(\eta)\#t_1\\
&= \bar{c}(\eta)\#t_1\sum_{j=1}^{s_{t_1}}\frac{\#t_1^j}{\#t_1}\left(\log_R\#t_1 - \log_R\frac{\#t_1}{\#t_1^j}\right) + \bar{c}(\eta)\#t_1\\
&= \bar{c}(\eta)\#t_1\log_R\#t_1 + \bar{c}(\eta)\#t_1\left(1-\sum_{j=1}^{s_{t_1}}\frac{\#t_1^j}{\#t_1}\log_R\frac{\#t_1}{\#t_1^j}\right)\\
&\leq \bar{c}(\eta)\#t_1\log_R\#t_1,
\end{aligned}$$

because we have assumed that $\#t_1 \geq R\,\#t_1^j$. $\square$

In Section 3 it will be shown that it is possible to generate a rank-k approximant of a block (t_1, t_2) in $\mathcal{O}(k^2(\#t_1 + \#t_2))$ operations. Thus for constant k the numerical effort for approximating, storing and multiplying a single block with a vector is of the same order as its generation. Therefore, for the whole matrix the cost for all these operations is $\mathcal{O}(\eta^{-dim} N \log_R N)$.

3 Low-rank approximation

In the preceding section we explained how to partition a matrix into blocks (t_1, t_2) such that for the corresponding domains D_1 and D_2

$$\operatorname{diam} D_2 \leq \eta \operatorname{dist}(D_1, D_2) \tag{6}$$

holds or the block degenerates to a vector. In this section we may therefore concentrate on a single block A satisfying (6).

The aim of this section is to decompose the functions $\mathcal{L}_j\kappa$ in the following way:

$$(\mathcal{L}_j\kappa)(y) = ([\mathcal{L}]_k\kappa)(y)^T G_k\,(\mathcal{L}_j\kappa)([\zeta]_k) + R_k(y), \quad y \in D_2. \tag{7}$$

Here G_k is a $k \times k$ matrix and R_k the approximation error. For the sake of simplicity we use the abbreviations

$$f([\zeta]_k) = \begin{bmatrix} f(\zeta_{i_1}) \\ \vdots \\ f(\zeta_{i_k}) \end{bmatrix} \quad \text{and} \quad ([\mathcal{L}]_k\kappa)(y) = \begin{bmatrix} (\mathcal{L}_{j_1}\kappa)(y) \\ \vdots \\ (\mathcal{L}_{j_k}\kappa)(y) \end{bmatrix}$$

with points $\zeta_i \in D_2$, $i = 1, \ldots, m_p$. Provided $\{\zeta_{i_1}, \ldots, \zeta_{i_k}\} \subset \{y_1, \ldots, y_m\}$ the decomposition (7) constitutes the analytic form of a pseudo skeleton decomposition, cf. [4]. In contrast with multipole [3, 7] and panel clustering [6] methods we will not approximate the functions $\mathcal{L}_j\kappa$ by using appropriate approximations to the kernel κ. Instead we will employ a small number of functions chosen from $\mathcal{L}_1\kappa, \ldots, \mathcal{L}_n\kappa$ to approximate $\mathcal{L}_j\kappa$.

3.1 Incomplete low-rank approximation

For the sake of simplicity we define for $i = 1, \ldots, m + m_p$

$$z_i = \begin{cases} \zeta_i, & 1 \leq i \leq m_p \\ y_{i-m_p}, & m_p < i \leq m_p + m \end{cases}$$

and extend A to $\hat{A}$ by setting $\hat{a}_{ij} = (\mathcal{L}_j\kappa)(z_i)$, $i = 1, \ldots, m+m_p$, $j = 1, \ldots, n$.

Algorithm 3.1.
Let $i_0 = 0$.
For $k = 1, 2, \ldots$
let i_k *be the smallest integer so that* $i_{k-1} < i_k \leq m_p$ *and*

$$(\tilde{v}_k)_j = (\mathcal{L}_j \kappa)(\zeta_{i_k}) - \sum_{l=1}^{k-1} (\hat{u}_l)_{i_k} (v_l)_j, \quad j = 1, \ldots, n$$

is non-zero.
If *no such* i_k *can be found* then *the algorithm terminates.*
else begin
choose $j_k = \mathrm{argmax}_{j=1,\ldots,n} |(\tilde{v}_k)_j|$ *and set* $\gamma_k = (\tilde{v}_k)_{j_k}^{-1}$.
let $v_k = \gamma_k \, \tilde{v}_k$ *and*

$$(\hat{u}_k)_i = (\mathcal{L}_{j_k} \kappa)(z_i) - \sum_{l=1}^{k-1} (\hat{u}_l)_i (v_l)_{j_k}, \quad i = 1, \ldots, m_p + m.$$

end

It is worth remarking that the algorithm differs for different types of matrices only with respect to the initial matrix. Define $\hat{S}_k = \sum_{l=1}^{k} \hat{u}_l v_l^T$ and $\hat{R}_k = \hat{A} - \hat{S}_k$. It is easy to see that $(\hat{u}_k)_i = (\hat{R}_k)_{ij_k}$, $i = 1, \ldots, m + m_p$ and $(\tilde{v}_k)_j = (\hat{R}_k)_{i_k j}$, $j = 1, \ldots, n$. The rank of $\hat{S}_k$ is bounded by k. For the calculation of $\hat{S}_k$ we need at most $k(m + m_p + n)$ units of memory and

$i_k n + k(m_p + m)$	evaluations of $\mathcal{L}_j \kappa$,
$2 k i_k n + k^2 (m_p + m)$	additions and multiplications,
kn	divisions.

Under the assumption that the evaluation of $(\mathcal{L}_j \kappa)(y)$ can be done in $\mathcal{O}(1)$ operations the cost for the generation of the approximant $\hat{S}_k$ sum up to $\mathcal{O}(i_k^2 (m + m_p + n))$ operations.
We have already pointed out in [1] that each step of Algorithm 3.1 may be understood as the generation of an approximant using the column-pivoted LU decomposition. To see this let us assume that $i_l = j_l = l$, $l = 1, \ldots, k$. In this case we have

$$\hat{R}_k = (I - \gamma_k \hat{R}_{k-1} e_k e_k^T) \hat{R}_{k-1} = L_k \hat{R}_{k-1}.$$

The $(m_p+m)\times(m_p+m)$ matrix L_k

$$L_k = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & 0 & & & \\ & & & -\frac{(\hat{R}_{k-1})_{k+1,k}}{(\hat{R}_{k-1})_{k,k}} & 1 & & \\ & & & \vdots & & \ddots & \\ & & & -\frac{(\hat{R}_{k-1})_{m+m_p,k}}{(\hat{R}_{k-1})_{k,k}} & & & 1 \end{bmatrix}$$

differs from a *Gaussian matrix* only in position $(k,\, k)$.
In the rest of this article the entries of $\hat{R}_k$ will be estimated. To this end we will relate $\hat{R}_k$ with functions r_k constructed by the following rule. Let $r_0(x,y)=\kappa(x,y)$, $s_0(x,y)=0$ and for $k=1,2,\ldots$

$$r_k(x,y) = r_{k-1}(x,y) - \gamma_k\,(\mathcal{L}_{j_k} r_{k-1})(y)\, r_{k-1}(x,\zeta_{i_k}), \tag{8a}$$

$$s_k(x,y) = s_{k-1}(x,y) + \gamma_k\,(\mathcal{L}_{j_k} r_{k-1})(y)\, r_{k-1}(x,\zeta_{i_k}). \tag{8b}$$

The following relation between $\hat{R}_k$ and r_k is obvious.

Lemma 3.2. *For $1\le i\le m+m_p$ and $1\le j\le n$ it holds that $(\hat{R}_k)_{ij} = (\mathcal{L}_j r_{k-1})(z_i)$.*

The number of zeros of $\mathcal{L}_j r_k$ increases for increasing k.

Lemma 3.3. *For $k\ge 0$ and $1\le l\le k$ it holds that*

$$(\mathcal{L}_{j_l} r_k)(y) = 0 \quad \text{for all } y\in\mathbb{R}^d\setminus D_1.$$

The preceding lemma can be prooven by inductively applying (8).
We define the matrix $\hat{A}_k\in\mathbb{R}^{k\times k}$ as

$$\hat{A}_k = \begin{bmatrix} (\mathcal{L}_{j_1}\kappa)(\zeta_{i_1}) & \ldots & (\mathcal{L}_{j_k}\kappa)(\zeta_{i_1}) \\ \vdots & & \vdots \\ (\mathcal{L}_{j_1}\kappa)(\zeta_{i_k}) & \ldots & (\mathcal{L}_{j_k}\kappa)(\zeta_{i_k}) \end{bmatrix}. \tag{9}$$

Furthermore, let $\hat{A}_k(l,j)\in\mathbb{R}^{k\times k}$ be the matrix resulting from $\hat{A}$ by replacing the lth column with the vector $(\mathcal{L}_j\kappa)([\zeta]_k)$ and $M_k(l,y)\in\mathbb{R}^{k\times k}$ the matrix that forms after replacing the lth row with the vector $([\mathcal{L}]_k\kappa)(y)$. Especially, $\hat{A}_k=\hat{A}_k(l,j_l)=M_k(l,\zeta_{i_l})$ holds.

For the determinants of the matrices $\hat{A}_k(l,j)$ it is possible to obtain a recurrence relation.

Lemma 3.4. *For $1 \leq l < k$ we have*

$$\det \hat{A}_k(l,j) = \gamma_k^{-1} \det \hat{A}_{k-1}(l,j) - (\mathcal{L}_j r_{k-1})(\zeta_{i_k}) \det \hat{A}_{k-1}(l,j_k),$$

$\det \hat{A}_1(1,j) = (\mathcal{L}_j \kappa)(\zeta_{i_1})$ *and* $\det \hat{A}_k(k,j) = (\mathcal{L}_j r_{k-1})(\zeta_{i_k}) \det \hat{A}_{k-1}$ *for* $k > 1$. *Especially,*

$$\det \hat{A}_k = \prod_{l=1}^{k} \gamma_l^{-1}.$$

The proof uses the same ideas as the proof for the analogous assertion in [1] and is therefore omitted.
Not only does the last lemma guarantee that $\hat{A}_k$ is non-singular, we also notice that the larger the product of the values γ_k^{-1} the larger the determinant of $\hat{A}_k$.
We are now in a position to show that the decomposition of κ into s_k and r_k is of type (7).

Lemma 3.5. *The sequences* $\{s_k\}_k$ *and* $\{r_k\}_k$ *generated in* (8) *satisfy*

$$s_k(x,y) + r_k(x,y) = \kappa(x,y),$$

where for $k \geq 1$

$$s_k(x,y) = ([\mathcal{L}]_k \kappa)(y)^T \hat{A}_k^{-1} \kappa(x, [\zeta]_k).$$

Proof. The lemma is obviously true for $k = 1$. We continue by induction. From the definition of r_k and s_k we can see that

$$s_k(x,y) + r_k(x,y) = s_{k-1}(x,y) + r_{k-1}(x,y),$$

which according to the assumption coincides with $\kappa(x,y)$.
For the sake of simplicity we set

$$a_k = \hat{A}_{k-1}^{-1} \, (\mathcal{L}_{j_k} \kappa)([\zeta]_{k-1}) \quad \text{and} \quad b_k = \hat{A}_{k-1}^{-T} \, ([\mathcal{L}]_{k-1} \kappa)(\zeta_{i_k}).$$

Since

$$\begin{aligned} s_k(x,y) &= s_{k-1}(x,y) + \gamma_k \, (\mathcal{L}_{j_k} r_{k-1})(y) \, r_{k-1}(x, \zeta_{i_k}) \\ &= \begin{bmatrix} ([\mathcal{L}]_{k-1} \kappa)(y) \\ (\mathcal{L}_{j_k} \kappa)(y) \end{bmatrix}^T \begin{bmatrix} \hat{A}_{k-1}^{-1} + \gamma_k a_k b_k^T & -\gamma_k a_k \\ -\gamma_k b_k^T & \gamma_k \end{bmatrix} \begin{bmatrix} \kappa(x, [\zeta]_{k-1}) \\ \kappa(x, \zeta_{i_k}) \end{bmatrix} \end{aligned}$$

and

$$\hat{A}_k \begin{bmatrix} \hat{A}_{k-1}^{-1} + \gamma_k a_k b_k^T & -\gamma_k a_k \\ -\gamma_k b_k^T & \gamma_k \end{bmatrix} = I_k$$

together with s_{k-1} also s_k has the form

$$s_k(x,y) = ([\mathcal{L}]_k\kappa)(y)^T \hat{A}_k^{-1}\kappa(x,[\zeta]_k).$$

$\square$

Using Cramer's rule we see that

$$\Big(([\mathcal{L}]_k\kappa)(y)^T\hat{A}_k^{-1}\Big)_l = \frac{\det M_k(l,y)}{\det \hat{A}_k},$$

where $M_k(l,y)$ is the matrix defined from (9). Thus

$$(\mathcal{L}_j s_k)(y) = \sum_{l=1}^{k} \frac{\det M_k(l,y)}{\det \hat{A}_k}\,(\mathcal{L}_j\kappa)(\zeta_{i_l}). \tag{10}$$

The representation (10) shows that $\mathcal{L}_j s_k$ is the uniquely defined interpolant of $\mathcal{L}_j\kappa$ in the nodes ζ_{i_l} in the linear hull of the functions $\mathcal{L}_{j_l}\kappa$, $l=1,\dots,k$. Let $\phi_1,\dots,\phi_k$ be a basis of the function space Φ, $x_i \in \mathbb{R}^d$ and $f_i \in \mathbb{R}$, $i=1,\dots,k$. Define

$$M_l(x) = \begin{bmatrix} \phi_1(x_1) & \dots & \phi_k(x_1) \\ \vdots & & \vdots \\ \phi_1(x) & \dots & \phi_k(x) \\ \vdots & & \vdots \\ \phi_1(x_k) & \dots & \phi_k(x_k) \end{bmatrix} \begin{matrix} \\ \\ \leftarrow l. \\ \\ \\ \end{matrix}$$

Furthermore, let $M = M_l(x_l) \in \mathbb{R}^{k\times k}$. It is obvious that provided M is non-singular the Lagrange functions

$$\chi_l(x) = \frac{\det M_l(x)}{\det M}$$

are in Φ and $\chi_l(x_i) = \delta_{il}$, $1 \le i,l \le k$, where δ denotes Kronecker's symbol. The function

$$L_k^\phi(x) = f_1\chi_1(x) + \dots + f_k\chi_k(x) \in \Phi$$

solves the interpolation problem

$$L_k^\phi(x_i) = f_i, \quad 1 \le i \le k \tag{11}$$

and is uniquely defined.

We need an expression for the interpolation error. To this end, we will relate the error $\mathcal{L}_j r_k$ to the error in another system of functions. Let $\psi_1, \ldots, \psi_{m_p}$ be a system of functions with

$$\det[\psi_j(\zeta_i)]_{ij} \neq 0,$$

spanning the space Ψ. For this system let the error

$$E_p^{\psi}[\mathcal{L}_j \kappa] = \mathcal{L}_j \kappa - L_{m_p}^{\psi}[\mathcal{L}_j \kappa]$$

be known.

It is now possible to relate the remainder $\mathcal{L}_j r_k$ of our approximation to the remainder E_p^{ψ} of the interpolation in the ψ-system. Notice that the points ζ_i, $i_k < i < i_{k+1}$, which were omitted during the construction of the sequences $\{r_k\}_k$ and $\{s_k\}_k$, are not lost for the approximation error.

Lemma 3.6. *Let $\{i_1, \ldots, i_k\} \subset \{1, \ldots, m_p\}$ be the indices used during the construction of $\{r_k\}_k$ and $\{s_k\}_k$. Then the functions $\mathcal{L}_j r_k$ satisfy*

$$(\mathcal{L}_j r_k)(y) = E_p^{\psi}[\mathcal{L}_j \kappa](y) - \sum_{l=1}^{k} \frac{\det \hat{A}_k(l,j)}{\det \hat{A}_k}\, E_p^{\psi}[\mathcal{L}_{j_l} \kappa](y), \quad y \in \mathbb{R}^d \setminus D_1. \quad (12)$$

Proof. Let χ_i be the ith Lagrange function in Ψ, i.e. for $1 \leq i, l \leq m_p$ we have $\chi_i(\zeta_l) = \delta_{il}$. Set

$$\chi(y) = \begin{bmatrix} \chi_1(y) \\ \vdots \\ \chi_{m_p}(y) \end{bmatrix}.$$

For the interpolant $L_{m_p}^{\psi}[\mathcal{L}_{j'} \kappa](y) = (\mathcal{L}_{j'} \kappa)([\zeta]_{m_p})^T \chi(y)$, $1 \leq j' \leq n$, we obtain

$$L_{m_p}^{\psi}[\mathcal{L}_{j'} \kappa](y) = \sum_{i=1}^{i_1 - 1} (\mathcal{L}_{j'} \kappa)(\zeta_i)\, \chi_i(y) + \sum_{l=1}^{k} \left\{ (\mathcal{L}_{j'} \kappa)(\zeta_{i_l})\, \chi_{i_l}(y) + \sum_{i=i_l+1}^{i_{l+1}-1} (\mathcal{L}_{j'} \kappa)(\zeta_i)\, \chi_i(y) \right\},$$

where we set $i_{k+1} = m_p + 1$. Since $(\mathcal{L}_{j'} r_l)(\zeta_i) = 0$ for all $i_l < i < i_{l+1}$ we obtain with the aid of Lemma 3.5

$$0 = (\mathcal{L}_{j'} r_l)(\zeta_i) = (\mathcal{L}_{j'} \kappa)(\zeta_i) - ([\mathcal{L}]_l \kappa)(\zeta_i)^T \hat{A}_l^{-1} (\mathcal{L}_{j'} \kappa)([\zeta]_l).$$

Thus

$$(\mathcal{L}_{j'} \kappa)(\zeta_i) = \sum_{\nu=1}^{l} \left(([\mathcal{L}]_l \kappa)(\zeta_i)^T \hat{A}_l^{-1} \right)_{\nu} (\mathcal{L}_{j'} \kappa)(\zeta_{i_\nu}).$$

From this it follows that

$$\begin{aligned}\sum_{i=i_l+1}^{i_{l+1}-1} (\mathcal{L}_{j'}\kappa)(\zeta_i)\,\chi_i(y) &= \sum_{i=i_l+1}^{i_{l+1}-1} \sum_{\nu=1}^{l} \left(([\mathcal{L}]_l\kappa)(\zeta_i)^T \hat{A}_l^{-1}\right)_\nu (\mathcal{L}_{j'}\kappa)(\zeta_{i_\nu})\,\chi_i(y) \\ &= \sum_{\nu=1}^{l} (\mathcal{L}_{j'}\kappa)(\zeta_{i_\nu})\,\alpha_\nu^{(l)}(y),\end{aligned}$$

where $\alpha_\nu^{(l)}(y) = \sum_{i=i_l+1}^{i_{l+1}-1} \chi_i(y) \left(([\mathcal{L}]_l\kappa)(\zeta_i)^T \hat{A}_l^{-1}\right)_\nu$.
From $(\mathcal{L}_{j'}\kappa)(\zeta_i) = 0$, $1 \le i < i_1$ we see that

$$\begin{aligned} L_{m_p}^{\psi}[\mathcal{L}_{j'}\kappa](y) &= \sum_{l=1}^{k} \left\{ (\mathcal{L}_{j'}\kappa)(\zeta_{i_l})\,\chi_{i_l}(y) + \sum_{\nu=1}^{l} (\mathcal{L}_{j'}\kappa)(\zeta_{i_\nu})\,\alpha_\nu^{(l)}(y) \right\} \\ &= \sum_{l=1}^{k} (\mathcal{L}_{j'}\kappa)(\zeta_{i_l})\,\chi_{i_l}(y) + \sum_{\nu=1}^{k}\sum_{l=\nu}^{k} (\mathcal{L}_{j'}\kappa)(\zeta_{i_\nu})\,\alpha_\nu^{(l)}(y) \\ &= \sum_{l=1}^{k} (\mathcal{L}_{j'}\kappa)(\zeta_{i_l})\,q_l(y) = (\mathcal{L}_{j'}\kappa)([\zeta]_k)^T q(y), \end{aligned}$$

where $q \in \mathbb{R}^k$ is the vector with components $q_l(y) = \chi_{i_l}(y) + \sum_{\nu=l}^{k} \alpha_l^{(\nu)}(y)$.
Using Lemma 3.5 we obtain

$$\begin{aligned} (\mathcal{L}_j r_k)(y) &= (\mathcal{L}_j\kappa)(y) - ([\mathcal{L}]_k\kappa)(y)^T \hat{A}_k^{-1} (\mathcal{L}_j\kappa)([\zeta]_k) \\ &= E_p^{\psi}[\mathcal{L}_j\kappa](y) + (\mathcal{L}_j\kappa)([\zeta]_k)^T q(y) - ([\mathcal{L}]_k\kappa)(y)^T \hat{A}_k^{-1} (\mathcal{L}_j\kappa)([\zeta]_k) \\ &= E_p^{\psi}[\mathcal{L}_j\kappa](y) - \left(([\mathcal{L}]_k\kappa)(y) - \hat{A}_k^T q(y)\right)^T \hat{A}_k^{-1} (\mathcal{L}_j\kappa)([\zeta]_k) \\ &= E_p^{\psi}[\mathcal{L}_j\kappa](y) - \sum_{l=1}^{k} E_p^{\psi}[\mathcal{L}_{j_l}\kappa](y) \left(\hat{A}_k^{-1} (\mathcal{L}_j\kappa)([\zeta]_k)\right)_l . \end{aligned}$$

According to Cramer's rule

$$\left(\hat{A}_k^{-1} (\mathcal{L}_j\kappa)([\zeta]_k)\right)_l = \frac{\det \hat{A}_k(l,j)}{\det \hat{A}_k}.$$

The assertion follows. □

We are now able to control the approximation error by estimating the coefficients in (12). In general the choice of j_k in Algorithm 3.1 does not produce a submatrix of maximum determinant in modulus. However, we can see from Lemma 3.4 that this maximum element strategy is the best choice with respect to maximum determinants if we keep all previously chosen indices $j_1, \ldots, j_{k-1}$ fixed.

Lemma 3.7. *Assume that in each step we choose j_k so that*

$$|(\mathcal{L}_{j_k} r_{k-1})(\zeta_{i_k})| \geq |(\mathcal{L}_j r_{k-1})(\zeta_{i_k})| \quad \textit{for all } 1 \leq j \leq n.$$

Then for $1 \leq l \leq k$ and $j = 1, \ldots, n$ it holds that

$$|\det \hat{A}_k(l, j)| \leq 2^{k-l} |\det \hat{A}_k|. \tag{13}$$

Proof. See [1]. □

Thus we obtain

$$|(\mathcal{L}_j r_k)(y)| \leq (1 + 2^k) \sup_{y \in D_2} |E_p^{\psi}(\mathcal{L}_j \kappa)(y)|. \tag{14}$$

It is well known that elements may grow during a column-pivoted LU decomposition. The term 2^{m_p} in (14) is therefore not a consequence of overestimation.
In order to estimate the error of our interpolation and, according to Lemma 3.2, also the error of matrix approximation, we have to specify the system of functions $\psi_1, \ldots, \psi_{m_p}$ so that the interpolation error for it is known. The easiest choice are the monomials

$$\psi_{\boldsymbol{i}}(x) = x^{\boldsymbol{i}} = x_1^{i_1} \cdots x_d^{i_d}, \quad \boldsymbol{i} \in \mathbb{N}_0^d \text{ with } \|\boldsymbol{i}\|_\infty \leq p - 1$$

Accordingly we choose $m_p = p^d$ as the dimension of the spanned space Ψ. The set of points $\{\zeta_1, \ldots, \zeta_{m_p}\}$ from the construction of $\{s_k\}_k$ is chosen to be the tensor-product

$$\zeta_{\boldsymbol{i}} = (\xi_{i_1}, \ldots, \xi_{i_d}), \quad \boldsymbol{i} \in \mathbb{N}^d \text{ with } \|\boldsymbol{i}\|_\infty \leq p$$

of the zeros of Tschebyscheff polynomials on $[-a, a]$

$$\xi_\nu = a \cdot \cos\left(\frac{\pi}{2} \frac{2\nu - 1}{p}\right), \quad \nu = 1, \ldots, p.$$

The uniqueness of interpolation is inherited from one-dimensional interpolation, so the condition $\det[\psi_j(\zeta_i)]_{ij} \neq 0$ is fulfilled. The polynomial $L_p f \in \Pi_{p-1}[-a, a]$ interpolating a function $f \in C^p[-a, a]$ in the points ξ_ν, $\nu = 1, \ldots, p$ satisfies, cf. [9],

$$\|f - L_p f\|_\infty \leq \frac{a^p}{2^{p-1}} \frac{\|f^{(p)}\|_\infty}{p!} \quad \text{and} \quad \|L_p f\|_\infty \leq c \log p \, \|f\|_\infty. \tag{15}$$

For multivariate functions $f : [-a, a]^d \to \mathbb{R}$ we use the tensor-product interpolation polynomial $\boldsymbol{L}_p[f]$

$$\boldsymbol{L}_p[f] = L_p^{(1)} \cdots L_p^{(d)} f \in \Psi.$$

Then using standard tensor-product arguments we obtain with (15)

$$\|f - \boldsymbol{L}_p[f]\|_\infty \le \tilde{c}_p \, a^p \max_{l=1,\dots,d} \|\partial_l^p f\|_\infty, \quad \tilde{c}_p = \frac{1 + d(c \log p)^{d-1}}{2^{p-1} p!}. \tag{16}$$

We are now ready to estimate the remainder $\hat{R}_k$ in Algorithm 3.1. To this end we remove the virtual points ζ_i by letting $S_k \in \mathbb{R}^{m \times n}$ be the last m rows of $\hat{S}_k$, i.e.

$$S_k = \sum_{l=1}^{k} u_l v_l^T,$$

where $u_l \in \mathbb{R}^m$ are the last m entries of $\hat{u}_l$.

Theorem 3.8. *Let κ be an asymptotically smooth kernel and the pair (D_1, D_2) fulfill condition (6). Then for $p \ge g$, $i = 1, \dots, m$ and $j = 1, \dots, n$ it holds that*

$$|(R_k)_{ij}| \le C_p \operatorname{dist}^g(D_1, D_2) \, \eta^p \max_{j=1,\dots,n} \|\tilde{l}_j\|, \quad 0 < \eta < \frac{1}{4\sqrt{d}},$$

where $R_k = A - S_k$.

Proof. We can find a cube Q_a having sidelength $a = 2 \operatorname{diam} D_2$ such that $D_2 \subset Q_a$ for which we may assume that $Q_a = \{x \in \mathbb{R}^d : \|x\|_\infty \le a\}$. It is easy to check that

$$2 \operatorname{dist}(D_1, Q_a) \ge \operatorname{dist}(D_1, D_2) \quad \text{for } \eta < \frac{1}{4\sqrt{d}}.$$

From this follows

$$\operatorname{diam} Q_a \le \tilde{\eta} \operatorname{dist}(D_1, Q_a), \quad \tilde{\eta} = 4\sqrt{d}\,\eta.$$

By assumption the derivatives of κ are bounded on $D_1 \times Q_a$:

$$\sup_{y \in Q_a} \|(\partial_y^\alpha \kappa)_y\|_{D_1} \le c_p \operatorname{dist}^{g-p}(D_1, Q_a), \quad |\alpha| = p.$$

Here, we use $\|f\|_{D_1} = \sup_{x \in D_1} |f(x)|$. According to (16) we have

$$\|E_p[\mathcal{L}_j \kappa]\|_{Q_a} \le \tilde{c}_p \, a^p \max_{l=1,\dots,d} \|\partial_l^p \mathcal{L}_j \kappa\|_{Q_a}.$$

From the continuity of $\tilde{l}_j$ corresponding to $\mathcal{L}_j$ we obtain

$$\partial_{y_l}^p(\mathcal{L}_j\kappa)(y) = \partial_{y_l}^p\langle\kappa_y, \tilde{l}_j\rangle = \langle\partial_{y_l}^p\kappa_y, \tilde{l}_j\rangle = (\mathcal{L}_j\partial_{y_l}^p\kappa)(y), \quad y \in Q_a.$$

This leads to

$$\begin{aligned}\|\partial_{y_l}^p\mathcal{L}_j\kappa\|_{Q_a} = \|\mathcal{L}_j\partial_{y_l}^p\kappa\|_{Q_a} &= \sup_{y\in Q_a}|\langle(\partial_{y_l}^p\kappa)_y, \tilde{l}_j\rangle| \\ &\leq \sup_{y\in Q_a}\|(\partial_{y_l}^p\kappa)_y\|_{D_1}\|\tilde{l}_j\| \leq c_p\,\|\tilde{l}_j\|\,\mathrm{dist}^{g-p}(D_1, Q_a)\end{aligned}$$

and thus

$$\|E_p[\mathcal{L}_j\kappa]\|_{Q_a} \leq \tilde{c}_p\,c_p\,\mathrm{dist}^g(D_1, Q_a)\,\|\tilde{l}_j\|\,\left(\frac{\tilde{\eta}}{\sqrt{d}}\right)^p. \tag{17}$$

Using (12) and (14) we are finally led to

$$|(R_k)_{ij}| \leq C_p\,\mathrm{dist}^g(D_1, D_2)\,\eta^p \max_{j=1,\ldots,n}\|\tilde{l}_j\|,$$

where $C_p = \tilde{c}_p\,c_p\,2^{|g|}4^p(1+2^{m_p})$. $\square$

Corollary. *Let (D_1, D_2) fulfill condition (6) and κ be an asymptotically smooth kernel. In the case of matrices*

$$a_{ij} = \kappa(x_j, y_i), \quad i = 1, \ldots, m,\ j = 1, \ldots, n$$

with $x_j \in D_1$, $y_i \in D_2$ for $p \geq g$ it holds that

$$|(R_k)_{ij}| \leq C_p\,\mathrm{dist}^g(D_1, D_2)\,\eta^p, \quad 0 < \eta < \frac{1}{4\sqrt{d}} \tag{18}$$

where $R_k = A - S_k$.

Corollary. *Let (D_1, D_2) fulfill condition (6) and κ be an asymptotically smooth kernel. In the case of collocation matrices*

$$a_{ij} = \int_D \kappa(x, y_i)\varphi_j(x)\,\mathrm{d}\mu_x, \quad i = 1, \ldots, m,\ j = 1, \ldots, n$$

with $\mathrm{supp}\,\varphi_j \subseteq D_1$, $\|\varphi_j\|_\infty = 1$ and $y_i \in D_2$ for $p \geq g$ it holds that

$$|(R_k)_{ij}| \leq C_p\,\mathrm{dist}^g(D_1, D_2)\,\mu(D_1)\,\eta^p, \quad 0 < \eta < \frac{1}{4\sqrt{d}}, \tag{19}$$

where $R_k = A - S_k$.

In what remains we will estimate how a prescribed accuracy $\varepsilon > 0$ for the approximation error $\|A - \tilde{A}\|_F < \varepsilon$ affects the cost of the algorithm. For an increasing p the term C_p in (18) and (19) grows faster than η^p. Hence, we have to keep p constant and control the error by η. Theorem 3.8 states that for each block $M \in \mathbb{R}^{m\times n}$ the approximant $\tilde{M}$ satisfies

$$\|M - \tilde{M}\|_F \le C_p \sqrt{mn}\, \mathrm{dist}^g(D_1, D_2)\, \eta^p$$

Since (D_1, D_2) fulfills (6) and $\operatorname{diam} \pi \ge h/c_u$ we obtain $\mathrm{dist}^g(D_1, D_2) \le cN^{-\frac{g}{dim}}$. Thus $\|A - \tilde{A}\|_F \le cC_p N^{1-\frac{g}{dim}} \eta^p$. Setting $\eta^p = \frac{\varepsilon}{cC_p} N^{\frac{g}{dim}-1}$ we get $\|A - \tilde{A}\|_F \le \varepsilon$. With this choice the overall complexity $\mathcal{O}(\eta^{-dim} N \log_R N)$ reads $\mathcal{O}(\varepsilon^{-\alpha} N^{1+\alpha} \log_R N)$.

4 Numerical experiments

Algorithm 3.1 may be stopped if the approximation reaches a certain accuracy. For this purpose the error estimator from [1] can be used. It is based on the idea that R_{m_p} is approximated by $\sum_{l=m_p}^{m_{p+1}} u_l v_l^T$ and that the latter can be evaluated efficiently.

4.1 Implementation aspects

In this section we discuss two possible implementations of the ACA (**A**daptive **C**ross **A**pproximation) method. Let $A \in \mathbb{R}^{m\times n}$ be a given matrix. Each of the following algorithms produces vectors $u_l \in \mathbb{R}^m$ and $v_l \in \mathbb{R}^n$, $l = 1, \ldots, k$ from which the approximant S_k can be formed

$$S_k = \sum_{l=1}^{k} u_l v_l^T.$$

Algorithm 4.1 (fully pivoted ACA). Set $R_0 = A$ and for $k = 0, 1, \ldots$

$$\begin{aligned}
(R_k)_{i_{k+1}, j_{k+1}} &= \max_{i,j} |(R_k)_{i,j}|, \\
u_{k+1} &= R_k e_{j_{k+1}}, \\
v_{k+1} &= R_k^T e_{i_{k+1}}, \\
R_{k+1} &= R_k - \gamma_{k+1} u_{k+1} v_{k+1}^T
\end{aligned}$$

with

$$\gamma_{k+1} = \left((R_k)_{i_{k+1}, j_{k+1}}\right)^{-1}.$$

We call this algorithm *fully pivoted ACA* since in each step the whole error matrix R_k is inspected for its maximal entry. In Algorithm 4.1 the following stopping criterion may be used

$$r: \ \|R_r\|_F \le \varepsilon \|A\|_F \, , \ \varepsilon > 0.$$

Hence the number of operations required to generate the approximant is $\mathcal{O}(r\,m\,n)$. Memory requirements for the algorithm are $\mathcal{O}(n\,m)$. Thus the algorithm is rather expensive and can not be used for real large matrices.
If the matrix A has not yet been generated but there is a possibility of generating its entries individually then the following *partially pivoted ACA method* can be used for the approximation.

Algorithm 4.2 (partially pivoted ACA). Set $\hat{\imath}_0 = 1$ and for $k = 1, 2, 3, \dots$ Let $i_k \in \{1, \dots, m\}$ be an index (set $i_k = \hat{\imath}_{k-1}$ if $\hat{\imath}_{k-1}$ is a possible choice) with nonzero

$$a_{i_k j} - \sum_{l=1}^{k-1} (u_l)_{i_k} (v_l)_j, \quad j = 1, \dots, n.$$

If no such i_k exists the algorithm terminates, otherwise set

$$(\tilde{v}_k)_j = a_{i_k j} - \sum_{l=1}^{k-1} (u_l)_{i_k} (v_l)_j, \quad j = 1, \dots, n.$$

Choose $j_k = \operatorname{argmax}_{j=1,\dots,n} |(\tilde{v}_k)_j|$ and set $v_k = (\tilde{v}_k)_{j_k}^{-1}\, \tilde{v}_k$ and

$$(u_k)_i = a_{i j_k} - \sum_{l=1}^{k-1} (u_l)_i (v_l)_{j_k}, \quad i = 1, \dots, m.$$

Set $\hat{\imath}_k = \operatorname{argmax}_{i \neq i_k} |(u_k)_i|$.

With regard to stopping criteria, the following considerations can be made. Since the matrix A will not be generated completely only the norm of the approximant S_k can be used instead. This can be recursively computed in the following way:

$$\|S_k\|_F^2 = \|S_{k-1}\|_F^2 + 2 \sum_{j=1}^{k-1} u_k^T u_j \, v_j^T v_k + \|u_k\|_F^2 \|v_k\|_F^2. \tag{20}$$

An appropriate stopping criterion is then

$$r: \ \|u_r\|_F \|v_r\|_F \le \varepsilon \|S_r\|_F. \tag{21}$$

The amount of numerical work required by Algorithm 4.2 is $\mathcal{O}(r^2(m+n))$.

4.2 Numerical results

We first apply the algorithm to a family of surfaces converging to the unit sphere. This sequence is generated by recursive refinement of the icosahedron dividing each of the surface triangles in four and projecting the new knots to the unit sphere as shown in Figure 1.

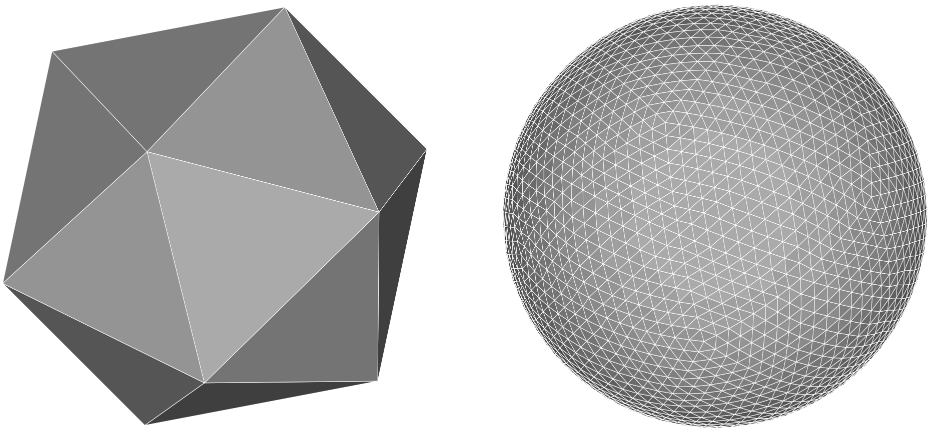

Figure 1. Icosahedron ($n = 20$) and its refinement ($n = 5120$).

The following numerical tests were performed for the boundary integral formulation

$$\mathcal{A}v = (\frac{1}{2}\mathcal{I} + \mathcal{B})f,$$

where

$$(\mathcal{A}\varphi)(x) = \int_{\partial\Omega} s(x,y)\varphi(y)\,\mathrm{d}s_y \quad \text{and} \quad (\mathcal{B}\varphi)(x) = \int_{\partial\Omega} \partial_{n_y} s(x,y)\varphi(y)\,\mathrm{d}s_y$$

of the Dirichlet problem for Laplace's equation

$$\Delta u = 0 \text{ in } \Omega, \tag{22a}$$

$$u = f \text{ on } \partial\Omega \tag{22b}$$

using collocation with piecewise constant ansatz functions. In the above the function s is the so called singularity function $s(x,y) = \frac{1}{4\pi}|x-y|^{-1}$.
The table below shows for different problem sizes the compression factors for the single layer and double potential matrix, the number of iterations when using unpreconditioned GMRES and the accuracy

$$\left(\sum_{\pi\in\Pi_h} \mu(\pi)\,|\partial_n s(x_0, m_\pi) - v_h(\pi)|^2\right)^{1/2}, \quad m_\pi \text{ center of } \pi$$

of the solution v_h. Since we chose $f = s(x_0, \cdot)|_{\partial\Omega}$, $x_0 \notin \overline{\Omega}$, the solution of (22) is known to be $u = s(x_0, \cdot)$.
For the approximation of the blocks we used Algorithm 4.2 and in the stopping criterion (21) ε was chosen as 10^{-6}, while the relative accuracy of GMRES was 10^{-8}.

n	single layer	double layer	# It	accuracy
80	100 %	100 %	14	0.791e-2
320	96 %	100 %	19	0.297e-2
1280	57 %	64 %	24	0.927e-3
5120	25 %	27 %	28	0.268e-3
20480	9 %	10 %	34	0.796e-4
81920	3 %	3 %	39	0.263e-4

An example of the partition of the BEM matrix is presented in Figure 2. Note that the numbering of the columns in the matrix corresponds to the permutation obtained during the construction of the cluster-tree, whereas the numbering of the rows is individual for each column due to Algorithm 2.1. However, it remains fixed within each block. The gray scale in Figure 2 indicates the quality of the approximation of the block as a percentage. Thus the big light blocks are very well approximated while the compression of the small dark blocks is either not possible or the compression rate is low.

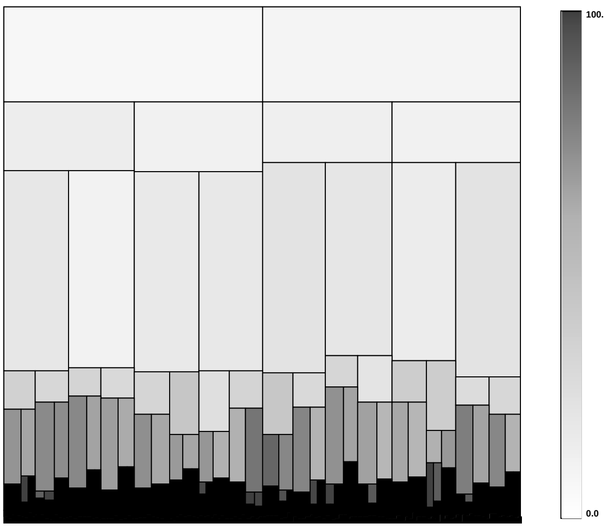

Figure 2. Block structure of the matrix for $n = 1280$.

In the remaining tests the aim is to compare different methods for the generation of low-rank approximants for the following mesh which consists of $n = 19712$ elements. This mesh comes from the TEAM 10 benchmark problem (see [8]) frequently used in the computational electrodynamic community.

The speciality of this multiply connected mesh is an extremely thin split in the middle and mesh refinement on the edges.

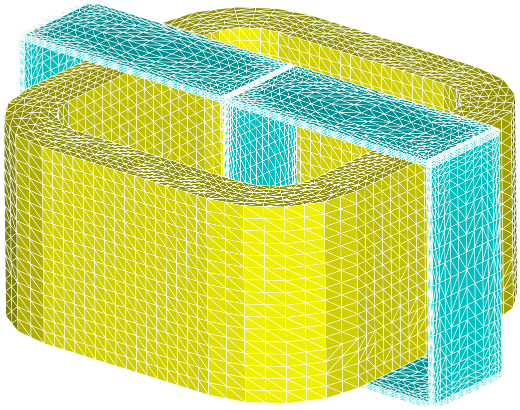

Figure 3. TEAM 10 problem ($n = 19712$).

For the relative accuracy in each case $\varepsilon = 10^{-4}$ was used. Generating the whole matrix without approximation would have led to 2964.5 MB of storage.

	single layer		double layer		CPU-time
SVD	199.11 MB	6.72 %	310.98 MB	10.49 %	100 h
ACA full	277.54 MB	9.36 %	410.29 MB	13.84 %	20.5 h
ACA partial	242.46 MB	8.18 %	376.19 MB	12.69 %	10 min

Acknowledgement

The authors wish to thank Dr. S. Kurz, Bosch GmbH, Stuttgart for providing the mesh for the second numerical example.

References

[1] M. Bebendorf. Approximation of boundary element matrices. *Numerische Mathematik*, (86):565–589, 2000. published online DOI 10.1007/s002110000192.

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

[3] H. Cheng, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm in three dimensions. *J. Comput. Phys.*, 155(2):468–498, 1999.

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

[5] W. Hackbusch and B. N. Khoromskij. A sparse $\mathcal{H}$-matrix arithmetic. II. Application to multi-dimensional problems. *Computing*, 64(1):21–47, 2000.

[6] W. Hackbusch and Z. P. Nowak. On the fast matrix multiplication in the boundary element method by panel clustering. *Numer. Math.*, 54(4):463–491, 1989.

[7] K. Nabors, J. Phillips, F. T. Korsmeyer, and J. White. Multipole and precorrected-FFT accelerated iterative methods for solving surface integral formulations of three-dimensional Laplace problems. In *Domain-based parallelism and problem decomposition methods in computational science and engineering*, pages 193–215. SIAM, Philadelphia, PA, 1995.

[8] T. Nakata, N. Takahashi, and K. Fujiwara. Summary of results for benchmark problem 10 (steel plates around a coil). *COMPEL*, 11:335–344, Sept. 1992.

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