Linear-time CUR approximation of BEM matrices

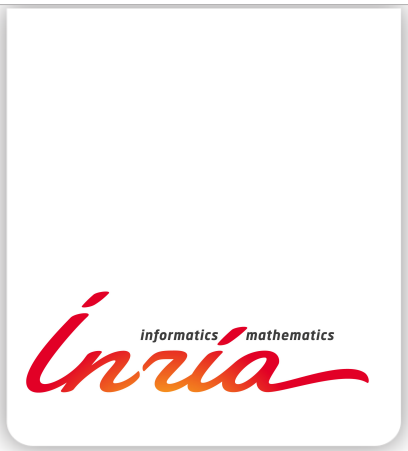
Alan Ayala, Xavier Claeys, Laura Grigori

Linear-time CUR approximation of BEM matrices

Alan Ayala , Xavier Claeys , Laura Grigori

RESEARCH REPORT
N° 9208
October 2018
Project-Team Alpines

ISSN 0249-6399 ISRN INRIA/RR--9208--FR+ENG



Linear-time CUR approximation of BEM matrices

Alan Ayala [*†], Xavier Claeys [‡], Laura Grigori[*]

Project-Team Alpines

Research Report n° 9208 — October 2018 — 29 pages

Abstract: In this paper we propose linear-time CUR approximation algorithms for admissible matrices obtained from the hierarchical form of Boundary Element matrices. We propose a new approach called geometric sampling to obtain indices of most significant rows and columns using information from the domains where the problem is posed. Our strategy is tailored to Boundary Element Methods (BEM) since it uses directly and explicitly the cluster tree containing information from the problem geometry. Our CUR algorithm has precision comparable with low-rank approximations created with the truncated QR factorization with column pivoting (QRCP) and the Adaptive Cross Approximation (ACA) with full pivoting, which are quadratic-cost methods. When compared to the well-known linear-time algorithm ACA with partial pivoting, we show that our algorithm improves, in general, the convergence error and overcomes some cases where ACA fails. We provide a general relative error bound for CUR approximations created with geometrical sampling. Finally, we evaluate the performance of our algorithms on traditional BEM problems defined over different geometries.

Key-words: CUR approximation, Linear time algorithms, BEM matrices

* INRIA Paris, Sorbonne Université, Univ Paris-Diderot SPC, CNRS, Laboratoire Jacques-Louis Lions, équipe ALPINES, France

† corresponding author: alan.ayala-obregon@inria.fr

‡ Sorbonne Université, Univ Paris Diderot SPC, CNRS, INRIA, Laboratoire Jacques-Louis Lions, équipe ALPINES, F-75005 Paris

RESEARCH CENTRE
PARIS

2 rue Simone Iff
CS 42112 - 75589 Paris Cedex 12

Approximation CUR en temps linéaire des matrices de type BEM

Résumé : Dans cet article, nous présentons des algorithmes pour créer une approximation de rang faible de type CUR pour des matrices résultant de la discrétisation des équations intégrales par la méthode des éléments de frontière (BEM). Notre approche consiste à utiliser l'information sur la géométrie du problème pour choisir des colonnes et des lignes les plus représentatives de la matrice. Nous montrons que notre algorithme principal, dont le coût est linéaire, a la même précision que des méthodes, ayant coût quadratique, comme QRCP et Approximation Adaptative Croisée (ACA) avec pivotage complet. Nous présentons des expériences numériques sur des domaines complexes en utilisant des noyaux intégrales fréquemment utilisés dans la littérature.

Mots-clés : Approximation CUR, Algorithmes en temps linéaire, Matrices BEM

1 Introduction

In this paper, we are interested in accelerating the matrix-vector products for matrices arising from the discretization of boundary integral operators, usually referred to as BEM matrices. A BEM matrix has entries of type $\mathscr{G}(x_i, y_j)$, where $\mathscr{G} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$, is a kernel integral operator and $X := [x_1, \cdots, x_m]$ and $Y := [y_1, \cdots, y_n]$ are interaction domains known as source and target domains respectively. For the scope of this work we consider $d = 3$ by default, however the theory straightforwardly holds for higher dimensions. The classical approach to accelerate the matrix-vector products for BEM matrices, is to separate the kernel evaluation into far-field (tailored to low-rank approximation) and near-field (direct evaluation). One of the most prominent methods to approximate the far field interactions is the Fast Multipole Method FMM [19, 26]; however, it has important drawbacks such as the kernel-dependency, high cost for problems with multiple right-hand sides and its difficult implementation. Remedies to these drawbacks have been and are currently being developed, such as Kernel independent FMM methods [13, 30].

Our approach consists in using the hierarchical form of the BEM matrix to obtain submatrices corresponding to the far-field interaction, which are known as *admissible blocks* and are constructed in a tree-fashion structure using a geometric admissibility criterion for clustering [3, 4, 22]. Typically, hierarchical matrices are constructed such that most of its blocks are admissible and hence the cost for compression and matrix-vector product is dominated by the cost of low-rank approximation of admissible blocks.

Let $A \in \mathbb{C}^{m\times n}$ denote one admissible block. A popular algorithm for approximating A is the Adaptive Cross Approximation (ACA), which has $\mathcal{O}(m+n)$ cost and its accuracy is good enough for many practical applications. The methodology performed by ACA can be seen as a CUR (or skeleton) approximation, this is,

$$A \approx \xi_k = CUR,$$

where $C := A(:, J)$, $R := A(I, :)$ and $U := A^{-1}(I, J) \in \mathbb{C}^{k\times k}$, I and J are sets of indices with cardinality k and must ensure that $A(I, J)$ is invertible. For the case of ACA, I and J are selected adaptively based on a greedy approach to make $A(I, J)$ have maximum absolute determinant among all $k \times k$ submatrices of A. Our approach consists in finding such indices using information from the problem geometry, we call our methodology geometric sampling and provide a general bound for the approximation error $\|A - \xi_k\|$. We analyze different methods to select I and J such as the Nearest-Neighbors (NN) criterion, which have recently been evaluated on multiple kernels in high dimensions showing good accuracy [29]. We propose a novel criterion called Gravity Centers Sampling (GCS) which, having asymptotic complexity of $\mathcal{O}((m + n)k)$, in most cases overcomes the accuracy of ACA and the NN criterion.

Skeleton approximations are mainly important when structure in the data must be preserved [28]. For BEM matrices, preserving data structure is not a priori relevant, our interest on CUR approximations is to achieve linear complexity. Note that preserving data approximations are tailored for the development of machine learning algorithms that can be trained to predict the most representative source and target points by simply analyzing properties of the domains where the problem is posed.

There exist randomized approaches to select indices I and J that can achieve linear-time complexity algorithms, *e.g.* via uniformly random selection [29], and even sublinear cost algorithms such as the one presented in [6], however their accuracy is not always guaranteed, see *e.g.* [29]. The methodologies presented in this paper are purely algebraic (no kernel dependency), deterministic and can be obtained in linear time. Related works are the IE-QR algorithm [33], which constructs a low-rank QR approximation using the modified Gram-Schmidt algorithm and costs $\mathcal{O}(N^{3/2})$, with $N = \max(m, n)$; the IES3 algorithm [25], a kernel independent method for electromagnetic simulations which costs $\mathcal{O}(N \log(N))$; and Interpolative Decompositions [5, 38], which rely on rank-revealing QR factorizations [21] and cost

$\mathcal{O}(mnk)$. Refer to [27] for a survey on the different approaches for low-rank approximation.

The paper is organized as follows. Section 2 presents classical methods to compute low-rank approximations based on QR factorizations and CUR decompositions. Section 3 presents the notion of geometric sampling to create a CUR approximation. We provide an algorithm and prove a relative error bound that can be used for any geometric sampling method. Section 4 presents and discusses several numerical experiments to validate our algorithm by using different types of geometries and integral kernels. Finally, Section 5 concludes our paper.

2 Definitions and basic background

2.1 Notations

Let us first state notational conventions that we shall use through this article. In the sequel, $A \in \mathbb{C}^{m\times n}$ will refer to a (not necessarily square $m \neq n$) matrix having entries $A(i,j) = \mathscr{G}(x_i, y_j)$, where $\mathscr{G}$ is kernel function which satisfies regularity properties such that the singular values of A exponentially decrease. We denote $\|A\|_2$ and $\|A\|_F$ the spectral and Frobenius norms respectively. Also, $\|A\|_{\max} := \max_{i,j} |A(i,j)|$ is the Chebyshev (or maximum) norm. We denote A^* the conjugate transpose of A, and I_p is the $p \times p$ identity matrix. Row and column indices vectors are denoted as I and J respectively. We use MATLAB notation.

2.2 Best low rank approximation

Definition 2.1. The rank of a matrix $A \in \mathbb{C}^{m\times n}$ is defined as the maximal number of linearly independent rows or columns of A, we denote it as $r := \operatorname{rank}(A)$.

Definition 2.2. We denote

$$\mathbb{C}_k^{m\times n} := \{B \in \mathbb{C}^{m\times n} : \operatorname{rank}(B) \leq k\}, \tag{2.1}$$

the set of complex matrices having at most rank-k.

The SVD decomposition states that A can be decomposed into a sum of rank-one matrices, see *e.g.* [24, Thm. 3.1.1], this is

$$A = \sum_{i=1}^{r} u_i \sigma_i v_i^T \equiv U_r \Sigma_r V_r^T, \tag{2.2}$$

where the matrices $U_r = [u_1, \cdots, u_r] \in \mathbb{C}^{m\times r}$ and $V_r = [v_1, \cdots, v_r] \in \mathbb{C}^{n\times r}$ are unitary, and their columns are the left and right singular vectors respectively. For matrix $\Sigma_r = \operatorname{diag}(\sigma_1, \cdots, \sigma_r) \in \mathbb{R}^{r\times r}$, we assume $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$ so that Σ is uniquely determined by A. The values σ_i are the singular values of A. Next, consider $k \in \mathbb{N}$, assuming that $\operatorname{rank}(A) \geq k$, the k-truncated singular value approximation of A is given as

$$A_k = \sum_{i=1}^{k} u_i \sigma_i v_i^T \equiv U_k \Sigma_k V_k^T. \tag{2.3}$$

Theorem 2.3. *(Mirsky, [31, Thm. 2]) Consider the matrix $A \in \mathbb{C}^{m\times n}$, with singular triplets (u_i, σ_i, v_i) for $i = 1, \cdots, \min(m,n)$. Then, $A_k = \sum_{i=1}^{k} u_i \sigma_i v_i^T$ is a solution of the following problem*

$$\begin{cases} \text{Find } B \in \mathbb{C}_k^{m\times n} \text{ such that} \\ \qquad \|A - B\| \leq \|A - C\|, \quad \forall\, C \in \mathbb{C}_k^{m\times n}, \end{cases} \tag{2.4}$$

where $\|\cdot\|$ stands for any unitarily invariant norm.

Note that for the spectral and Frobenius norms it holds

$$\|A_k - A\|_2 = \sigma_{k+1}, \qquad \|A_k - A\|_F = \sqrt{\sigma_{k+1}^2 + \cdots + \sigma_r^2}. \tag{2.5}$$

Next theorem presents some useful inequalities that will be helpful in the following sections.

Theorem 2.4. *(Horn and Jonhson, [24, Thm. 3.3.16]) Let $A, B \in \mathbb{R}^{m\times n}$ and $q = \min(m, n)$. Then, for $1 \le i, j$ and $i + j \le q + 1$, the following inequalities hold,*

$$\sigma_{i+j-1}(AB^T) \le \sigma_i(A)\sigma_j(B), \tag{2.6}$$
$$\sigma_{i+j-1}(A + B) \le \sigma_i(A) + \sigma_j(B). \tag{2.7}$$

2.3 Approximation via Pivoted QR Factorization

The strategy we wish to present uses QR factorization as an intermediate tool. A pivoted truncated QR factorization of a matrix A takes the following form

$$AP = \begin{array}{c} \\ m \end{array} \begin{array}{c} \begin{array}{cc} k & m-k \end{array} \\ \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \end{array} \begin{array}{c} \\ \begin{array}{c} k \\ m-k \end{array} \end{array} \begin{array}{c} \begin{array}{cc} k & n-k \end{array} \\ \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} \end{array}, \tag{2.8}$$

where P is a permutation matrix, $Q = [Q_1, Q_2]$ is an unitary matrix, R_{11} is an upper triangular matrix, and R_{22} is not necessarily upper triangular. The rank-k approximation is naturally obtained as

$$\xi_k := Q_1 \begin{bmatrix} R_{11} & R_{12} \end{bmatrix} P^T, \tag{2.9}$$

and the approximation error in the spectral norm (note that it holds for any other unitarily invariant norm) is given as

$$\|A - \xi_k\|_2 = \|Q_2[0 \quad R_{22}]P^T\|_2 = \|[0 \quad R_{22}]\|_2 = \|R_{22}\|_2. \tag{2.10}$$

Bounds for the error in (2.10) depend on the technique used to select P. The most well established pivoting techniques, see *e.g.* [9, 21, 34], lead to bounds of the form

$$\|R_{22}\|_2 \le f(k, n)\, \sigma_{k+1}, \tag{2.11}$$
$$\|R_{11}^{-1} R_{12}\|_{\max} \le g(k), \tag{2.12}$$
$$\sigma_k(A) \le f(k, n)\sigma_k(R_{11})\,, \tag{2.13}$$

where $f(k, n)$ and $g(k)$ are explicitly known functions of k and n. For a compilation of some of the different algorithms of this kind and their computational complexity see [5, Tbl. 2]. The following table presents two classical deterministic QR algorithms which we henceforth use to construct an approximation of type (2.9).

Table 1: Error bound for classical QR algorithms for a matrix $A \in \mathbb{C}^{m\times n}$, where $k < \min(m, n)$ is the truncation rank and ν is a constant.

Algorithm	Reference	$f(k, n)$	$g(k)$	Time
Column Pivoting QRCP	[14, Alg. 5.4.1]	$2^k\sqrt{n-k}$	2^{k-1}	$\mathcal{O}(mnk)$
Strong RRQR	[21, Alg. 4]	$\sqrt{1+\nu^2 k(n-k)}$	ν	$\mathcal{O}\left((m + n\log_\nu n)n^2\right)$

Lemma 2.5 presents a new bound for QRCP which is 2/3 of the bound presented in Table 1, its proof is given in the supplementary material and it helps to understand the origin of the exponential factor.

Lemma 2.5. Consider the truncated QRCP factorization (2.8). Then,

$$\|R_{22}\|_2 \leq \sqrt{1 + 2k + \sum_{j=1}^{k-1} 4^j (k-j)\sqrt{n-k}}\, \sigma_{k+1}, \tag{2.14}$$

To conclude this section, note that by a simple algebraic calculation it can be shown that the error (2.10) can also be written as

$$\|R_{22}\|_2 = \|(I_m - Q_1 Q_1^*)A\|_2, \tag{2.15}$$

and using this representation, a bound for the error when using a general pivoting technique can be obtained, see *e.g.* [1, Lem. 2.9].

2.4 CUR approximations

Consider a matrix $A \in \mathbb{C}^{m\times n}$. Let row and column indices $I = \{i_1, \cdots, i_k\}$ and $J = \{j_1, \cdots, j_k\}$ be chosen such that $A(I,J) \in \mathbb{C}^{k\times k}$ is non-singular. The CUR approximation of A has the form

$$A \approx CUR, \tag{2.16}$$

where $C := A(:,J) \in \mathbb{C}^{m\times k}$, $R := A(I,:) \in \mathbb{C}^{k\times n}$, and $U := A(I,J)^{-1}$. Equation (2.16) is also known as skeleton approximation [15, 16]. The search of I and J is known as *sampling*. Note that if $\operatorname{rank}(A) = k$, then its skeleton approximation is exact, this is $A = CUR$.

Error of CUR approximation

Let us consider the indices $\tilde{I} := [I, \{1, \cdots, m\}\backslash I]$, and $\tilde{J} := [J, \{1, \cdots, n\}\backslash J]$, such that

$$A(\tilde{I}, \tilde{J}) := \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \tag{2.17}$$

where $A_{11} = A(I,J) \in \mathbb{C}^{k\times k}$. A simple decomposition of A follows as

$$A(\tilde{I}, \tilde{J}) = \underbrace{\begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}}_{=:C(\tilde{I},:)} \underbrace{A_{11}^{-1}}_{=:U} \underbrace{\begin{bmatrix} A_{11} & A_{12} \end{bmatrix}}_{=:R(:,\tilde{J})} + \begin{bmatrix} 0 & 0 \\ 0 & S(A_{11}) \end{bmatrix} \equiv C(\tilde{I},:)UR(:,\tilde{J}) + \begin{bmatrix} 0 & 0 \\ 0 & S(A_{11}) \end{bmatrix}, \tag{2.18}$$

where $S(A_{11}) := A_{22} - A_{21}A_{11}^{-1}A_{12}$ is known as the Schur complement of A_{11}. Hence, the approximation error is given as

$$\|A - CUR\| = \|A(\tilde{I}, \tilde{J}) - C(\tilde{I},:)UR(:,\tilde{J})\| = \|S(A_{11})\|, \tag{2.19}$$

where $\|\cdot\|$ stands for any unitarily invariant norm and the maximum norm. Hence, to get a good rank-k CUR approximation we need to sample I and J such that the norm of the Schur complement of $A(I,J)$ is small. Next, we consider a sub-optimal sampling technique consisting in finding I and J to make $A(I,J)$ have *maximal volume*, *i.e.* maximal absolute determinant among all $k \times k$ submatrices of A.

Theorem 2.6. *Consider $A \in \mathbb{C}^{m\times n}$, and row and column indices I and J respectively, with $|I| = |J| = k$. Define $G := A(I, J) \in \mathbb{C}^{k\times k}$. If G is non-singular and has maximal volume among all $k \times k$ submatrices of A, then*

$$\|A - CUR\|_{\max} \leq (1 + k)\, \sigma_{k+1}, \tag{2.20}$$

$$\|A - CUR\|_{\max} \leq (1 + k)^2 \cdot \min_{B\in\mathbb{C}_k^{m\times n}} \|A - B\|_{\max}, \tag{2.21}$$

where $C := A(:, J)$, $R := A(I, :)$, and $U := G^{-1}$.

Proof. Inequality (2.20) is proved in [16, Thm. 2.1], and (2.21) in [17, Thm. 1]. □

Although the sampling from Theorem 2.6 is nearly optimal, finding submatrices of maximal volume is NP-hard [7].

Algorithm 1, which is adapted from [3, Alg. 3.1], computes $A(:, J) \cdot A^{-1}(I, J) \cdot A(I, :) \equiv CUR$ as sum of rank-one matrices. The advantages of this form is that we can update the choice of the selected rows and columns adaptively, and it also allows to monitor the evolution of the determinant of the submatrix formed by the selected indices at a given rank of approximation.

Data: An integral kernel $\mathscr{G} : \mathbb{R}^{d\times d} \to \mathbb{C}$, Indices: I and J, each of size k,
Source and target points: $X = [x_1, \cdots, x_m]$ and $Y = [y_1, \cdots, y_n]$
Result: A matrix ξ_k of rank at most k and given as sum of rank-one matrices

```
for h = 1 → k do
    Set i = I(h) and j = J(h) ;
    ṽ_h := [𝒢(x_i, y_1), ··· , 𝒢(x_i, y_n)];
    u_h := [𝒢(x_1, y_j), ··· , 𝒢(x_m, y_j)]^T;
    for l = 1 → h − 1 do
        ṽ_h := ṽ_h − u_l(i) v_l
    end
    if ṽ_h(j) vanishes then
        Update column index j = argmax_{s=1,···,n} |ṽ_h(s)|
    end
    Set δ(h) = ṽ_h(j) ;
    Normalize v_h := ṽ_h/δ(h) ;
    for l = 1 → h − 1 do
        u_h := u_h − v_l(j) u_l
    end
end
```

Algorithm 1: Skeleton approximation with fixed pivots

Algorithm 1 requires $(m+n)k$ evaluations of kernel function $\mathscr{G}$ and $\mathcal{O}\left((m + n)k^2\right)$ complex operations. When it halts, we get a rank-k matrix

$$\xi_k := \sum_{h=1}^{k} u_h v_h \equiv CUR. \tag{2.22}$$

This approximation only requires $(m + n)k$ units of storage. Defining $M_k := A(I, J)$, we can also obtain the volume of the submatrix obtained by our choice of row and column indices, it is given as [2, Lem. 2],

$$|\det(M_k)| = \left| \prod_{i=1}^{k} \delta(i) \right|. \tag{2.23}$$

In Section 4 we plot the value $|\det(M_k)|$ for different sampling techniques, to analyze its impact on increasing the approximation accuracy.

3 Linear-time CUR approximation via Geometric Sampling

In this section, we present the concept of *geometric sampling* to select row and column indices I and J by using information from the geometry of the source and target points. Then, a CUR approximation directly follows by using the theory from the previous section.

3.1 Geometrical sampling

Algorithm 2 shows our sampling technique. We select $t > k$ (oversampling) points from the target domain and store them into an index vector $\tilde{J}$ which defines a matrix $\tilde{C} := A(:, \tilde{J}) \in \mathbb{C}^{m \times t}$ of sampled columns. Then, we work on the m-dimensional space, selecting a set of k column indices J corresponding to the *most significant* columns of $\tilde{C}$, we do this by computing the pivoted QR factorization $\tilde{C}(:, p_c) = \hat{Q}\hat{R}$. Then, we set $Q = \hat{Q}(:, 1:k)$ and perform a truncated pivoted QR factorization on matrix Q^T to obtain a permutation vector p_r. Finally we return $I = p_r(1:k)$ and $J = p_c(1:k)$ from which a CUR approximation directly follows as done in section 2.4, *c.f.* Algorithm 1.

Data: Approximation Rank k; Source and target points: $X = [x_1, \cdots, x_m]$ and $Y = [y_1, \cdots, y_n]$
Result: Low-rank CUR approximation of A
1 Set oversampling: $t = 2^l$ such that $2^l > k > 2^{l-1}$;
2 Decompose Y into t subdomains, see Appendix A.2.2 ;
3 Form J with the t indices of target points closest to the gravity centers of subdomains;
4 Set $C = A(:, J)$ and compute its pivoted QR factorization $\tilde{C}(:, p_c) = \hat{Q}\hat{R}$;
5 Set $Q = \hat{Q}(:, 1:k)$ and compute the truncated QR factorization of Q^T to get permutation p_r ;
6 Set $J = p_c(1:k)$ and $I = p_r(1:k)$.
7 Return $CUR = A(:, J) \cdot A^{-1}(I, J) \cdot A(I, :)$, which can be computed via Algorithm 1.

Algorithm 2: CUR with gravity centers sampling, CUR_GCS

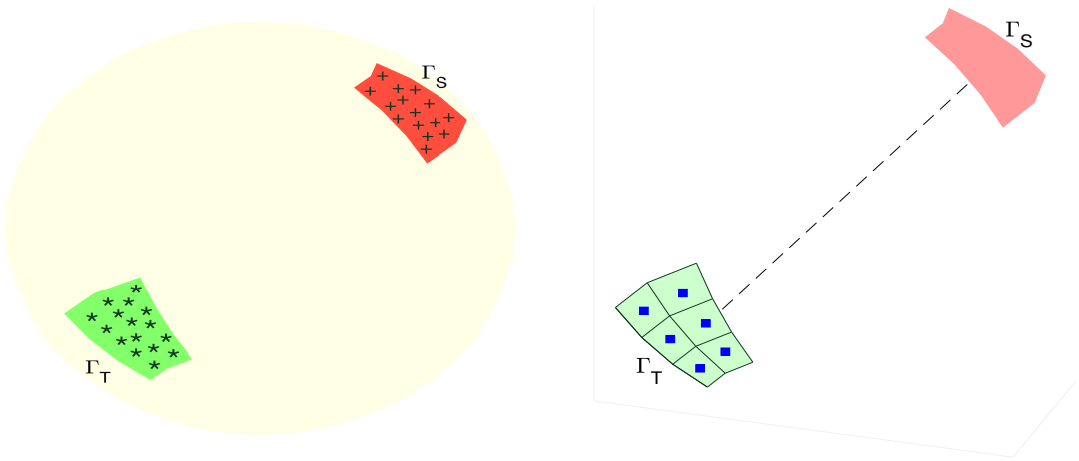
Remark 3.1. Note that the permutation vector p_r from Algorithm CUR_GCS, would be the same if we instead perform the QR factorization of the conjugate transpose Q^*. This is true since a simple algebraic effort shows that for any $M \in \mathbb{C}^{m \times n}$ with QR factorization $MP = QR$, it holds that $\overline{M}P = \overline{Q}\,\overline{R}$ is the pivoted QR factorization of $\overline{M}$ (matrix with complex conjugated entries).

The computational cost of CUR_GCS is given as: $\mathcal{O}(n \log_2(t))$ floating point operations to obtain J (see Appendix A.2.2), $\mathcal{O}(mt^2)$ complex operations to perform a truncated QR factorization on C, $\mathcal{O}(nk^2)$ complex operations to perform a truncated QR factorization on Q^T and $\mathcal{O}\left((m+n)k^2\right)$ complex operations to get the CUR approximation. Thus, the total cost is $\mathcal{O}(mt^2 + nk^2)$. Also, note that we do not need to form the whole matrix A, we only need $mt + nk$ evaluations of the kernel function.

Figure 1 illustrates the procedure of algorithm CUR_GCS, it displays a spherical domain with source and target distant subdomains. We use the *geometrically balanced* partition technique, *c.f.* [3, Sec. 1.4.1] and [4, Alg.2], to decompose the target domain into $t = 6$ subdomains, and then six target points (blue squares) are selected as the ones closest to the gravity centers of the subdomains. In appendix A.2.1 we provide a MATLAB code for algorithm CUR_GCS, and in appendix A.2.2 we provide the code for the

gravity centers sampling technique.

Note that we can easily modify Algorithm 2 to get alternative CUR approximations, by changing the partition technique. For instance, if using the Nearest-Neighbors criterion instead of the gravity centers criterion in line 2 of Algorithm 2, we obtain a new algorithm to which we refer to as CUR_NNS, *c.f.* Appendix A.2.3. The Nearest-Neighbors technique selects t target points as the ones closest to the source domain, this has been recently studied in [29]. In next subsection, we prove a bound on the CUR approximation error for an arbitrary domain partitioning technique, and in Section 3.3 we discuss the advantages of the partitioning technique of our algorithm CUR_GCS over CUR_NNS.



(a) Interaction of source (Γ_S) and target (Γ_T) points, discretized by $X = \{x_1, \cdots, x_n\}$ (marked with $+$) and $Y = \{y_1, \cdots, y_m\}$ (marked with $*$) respectively.

(b) Selection of representative targets points (squares), as points closest to the gravity centers of subdomains of Γ_T.

Figure 1: Interaction of distant subdomains on a sphere, and selection of representative target points.

3.2 Bound on the error of CUR approximation with geometric sampling

Consider that geometric sampling has been performed selecting indices I and J, with $|I| = |J| = k$, such that $A(I, J)$ is non-singular. Let $\tilde{I} := [I, \{1, \cdots, m\} \backslash I]$, and $\tilde{J} := [J, \{1, \cdots, n\} \backslash J]$ and let us apply a truncated-QR factorization in the permuted matrix $A(\tilde{I}, \tilde{J})$, this is

$$A(\tilde{I}, \tilde{J}) := \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \check{Q}\check{R} \equiv \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}. \tag{3.1}$$

Next, we use an idea presented in a previous paper [20] for the case of real matrices, where the authors observed that $S(A_{11}) = S(Q_{11})R_{22}$, with

$$S(Q_{11}) := Q_{22} - Q_{21}Q_{11}^{-1}Q_{12} = Q_{22}^{*-1}, \tag{3.2}$$

where Q_{22}^* is the conjugate transpose of Q_{22}, and the last equality can be verified by computing $Q_{22}^* S(Q_{11})$ and using the fact that $\check{Q}\check{Q}^* = I_m$. Then, Equation (2.19) is rewritten

$$\|A - CUR\|_2 = \|S(A_{11})\|_2 \leq \|Q_{22}^{*-1}\|_2 \|S(R_{22})\|_2, \tag{3.3}$$

where $C = A(:, J)$, $R = A(I, :)$ and $U = A^{-1}(I, J)$ and by using the CS decomposition [14, Thm.2.6.3], which tells us that $\sigma_{\min}(Q_{11}) = \sigma_{\min}(Q_{22})$, finally we get the bound

$$\|A - CUR\|_2 \leq \frac{1}{\sigma_{\min}(Q_{22})}\|R_{22}\|_2 = \frac{1}{\sigma_k(Q_{11})}\|R_{22}\|_2. \tag{3.4}$$

Error of column sampling

We first state a theorem to bound the error of column sampling. This bound involves C and its QR factorization. From this we then derive a bound for CUR approximation.

Theorem 3.2. *Consider $A \in \mathbb{C}^{m\times n}$ and a set of indices J, with $|J| = t$. Let $C := A(:, J)$ be at least rank-k and consider its QRCP factorization,*

$$C(:, p_c) = \hat{Q}\hat{R} \equiv \hat{Q} \begin{array}{c} \\ k \\ m-k \end{array} \begin{array}{c} \begin{array}{cc} k & t-k \end{array} \\ \begin{bmatrix} \hat{R}_{11} & \hat{R}_{12} \\ 0 & \hat{R}_{22} \end{bmatrix} \end{array} \tag{3.5}$$

where $\hat{Q} \in \mathbb{C}^{m\times m}$ is unitary, $\hat{R} \in \mathbb{C}^{m\times t}$, and p_c is a permutation vector of size t. Define $Q = \hat{Q}(:, 1:k)$, then

$$E := \|(I_m - QQ^*)A\|_2 \leq \sqrt{f^2(k,t) + k\left(\frac{2^{k-1}\|A\|_F}{\mu}\right)^2} \cdot \sigma_{k+1}(A). \tag{3.6}$$

where $\mu = |\hat{R}(k,k)|$ and $f(k,t)$ is defined in Table 1.

Note that $\|R_{22}\|_2 = \|(I_m - QQ^*)A\|_2$ according to (2.15), hence one of the factors of the bound on the CUR approximation error (3.4) follows from the proof of Theorem 3.2. This bound can also be interpreted as the error of a rank-k truncated QR approximation with geometric sampling as pivoting technique, *c.f.* section 2.3.

Proof. Let us consider $\hat{J} = J(p_c)$ and define $p = [\hat{J}, \{1, \cdots, n\}\backslash\hat{J}]$, we get

$$\hat{Q}^T A(:, p) = \underbrace{\begin{array}{c} \\ k \\ m-k \end{array} \begin{array}{c} \begin{array}{ccc} k & t-k & n-t \end{array} \\ \begin{bmatrix} \hat{R}_{11} & \hat{R}_{12} & \hat{B}_1 \\ 0 & \hat{R}_{22} & \hat{B}_2 \end{bmatrix} \end{array}}_{=:\tilde{R}}, \tag{3.7}$$

where $\hat{B}_1 \in \mathbb{C}^{k\times(n-t)}$ and $\hat{B}_2 \in \mathbb{C}^{(m-k)\times(n-t)}$. Note that approximating A by $QQ^*A = Q[\hat{R}_{11}, \hat{R}_{12}, \hat{B}_1]$, we get (*c.f.* (2.15))

$$\|(I_m - QQ^*)A\|_2 = \|[\hat{R}_{22}, \hat{B}_2]\|_2,$$

and bounding the right hand side of the previous equation would give us the desired bound of the theorem. However, we do not want to compute $\hat{B}_2$ and also, this form does not allow to directly get a bound as in (3.6). Hence, we proceed to use a technique developed by Gu and Eisenstat [21, Thm. 3.2]. Let us define the following block diagonal matrix,

$$Z := \begin{bmatrix} \alpha\hat{R}_{11} & \\ & [\hat{R}_{22}, \hat{B}_2] \end{bmatrix} = \begin{bmatrix} \hat{R}_{11} & \hat{R}_{12} & \hat{B}_1 \\ 0 & \hat{R}_{22} & \hat{B}_2 \end{bmatrix} \underbrace{\begin{bmatrix} \alpha I_k & -\hat{R}_{11}^{-1}[\hat{R}_{12}, \hat{B}_1] \\ & I_{n-k} \end{bmatrix}}_{=:W} \equiv \tilde{R}W$$

where $\alpha = \sigma_{\max}([\hat{R}_{22}, \hat{B}_2])/\sigma_{\min}(\hat{R}_{11})$. Note that this choice of α ensures that $\sigma_{k+1}(Z) = \sigma_1([\hat{R}_{22}, \hat{B}_2])$. Next, using Theorem 2.4 we get,

$$E = \|[\hat{R}_{22}, \hat{B}_2]\|_2 = \sigma_{k+1}(Z) \leq \sigma_{k+1}(\tilde{R})\|W\|_2 = \sigma_{k+1}(A)\|W\|_2, \tag{3.8}$$

where the last equality holds since $\hat{Q}$ is unitary. Then, to complete the proof, it remains to bound $\|W\|_2$. We proceed as follows,

$$\begin{aligned}
\|W\|_2^2 &\leq 1 + \|\hat{R}_{11}^{-1}[\hat{R}_{12}, \hat{B}_1]\|_2^2 + \alpha^2 && (3.9)\\
&= 1 + \|[\hat{R}_{11}^{-1}\hat{R}_{12}, \hat{R}_{11}^{-1}\hat{B}_1]\|_2^2 + \|\hat{R}_{11}^{-1}\|_2^2 \left(\|[\hat{R}_{22}, \hat{B}_2]\|_2^2\right) && (3.10)\\
&\leq 1 + \|\hat{R}_{11}^{-1}\hat{R}_{12}\|_F^2 + \|\hat{R}_{11}^{-1}\hat{B}_1\|_F^2 + \|\hat{R}_{11}^{-1}\|_F^2 \left(\|\hat{R}_{22}\|_F^2 + \|\hat{B}_2\|_F^2\right) && (3.11)\\
&\leq \left(1 + \|\hat{R}_{11}^{-1}\hat{R}_{12}\|_F^2 + \|\hat{R}_{11}^{-1}\|_F^2\|\hat{R}_{22}\|_F^2\right) + \|\hat{R}_{11}^{-1}\|_F^2 \left(\|\hat{B}_1\|_F^2 + \|\hat{B}_2\|_F^2\right). && (3.12)
\end{aligned}$$

From the QRCP factorization (3.5), we get that (*c.f.* proof of [21, Thm. 7.2]),

$$1 + \|\hat{R}_{11}^{-1}\hat{R}_{12}\|_F^2 + \|\hat{R}_{11}^{-1}\|_F^2\,\|\hat{R}_{22}\|_F^2 \leq f^2(k,t). \tag{3.13}$$

Hence,

$$\|W\|_2^2 \leq f^2(k,t) + \|\hat{R}_{11}^{-1}\|_F^2 \left(\|\hat{B}_1\|_F^2 + \|\hat{B}_2\|_F^2\right). \tag{3.14}$$

Next, we observe that

$$\|\hat{R}_{11}^{-1}\|_F \leq \sqrt{k}\|\hat{R}_{11}^{-1}\|_2 \leq \sqrt{k}\frac{2^{k-1}}{\mu},$$

where for the first inequality we use a classic relationship between the spectral and Frobenius norms, and for the last inequality we use a theorem from [23, Thm. 8.14].

From (3.7) we get that $\|\hat{B}_1\|_F^2 + \|\hat{B}_2\|_F^2 \leq \|A\|_F^2$. Hence,

$$\|W\|_2^2 \leq f^2(k,t) + k\left(\frac{2^{k-1}\|A\|_F}{\mu}\right)^2, \tag{3.15}$$

and the result follows by replacing (3.15) in (3.8).

□

Remark 3.3. The value μ in the previous theorem depends on k and its inverse can be bounded. Consider $D := \mathrm{diag}(\mathrm{diag}(\hat{R}))$ and define Y such that $R = DY$. Then, using Theorem 2.4 we get $\sigma_i(\hat{R}) \leq \sigma_i(D)\|Y\|_2 = \hat{R}(i,i)\|Y\|_2$. Also, since $\sigma_i(C) = \sigma_i(\hat{R})$ we get

$$\frac{1}{\mu} = \frac{1}{\hat{R}(k,k)} \leq \frac{\|Y\|_2}{\sigma_k(\hat{R})} = \frac{\|Y\|_2}{\sigma_k(C)} \leq \sqrt{\frac{t(t+1)}{2\sigma_k(C)}},$$

where the last inequality holds since all entries of Y are smaller than 1.

Remark 3.4. A bound can also be obtained when the strong rank-revealing factorization is used to factor C. From (2.13) we get $\sigma_i(\hat{R}_{11}) \leq \sigma_i(C) \leq f(k,t)\sigma_i(\hat{R}_{11})$, then $\|\hat{R}_{11}^{-1}\|_F \leq \sqrt{k}\|\hat{R}_{11}^{-1}\|_2 \leq \sqrt{k}\frac{f(k,t)}{\sigma_k(C)}$. Hence, by using (3.8) and (3.14) as done in the proof of the theorem, we obtain

$$E \leq f(k,t)\sqrt{1 + k\left(\frac{\|A\|_F}{\sigma_k(C)}\right)^2} \cdot \sigma_{k+1}(A), \tag{3.16}$$

where $f(k,t) = \sqrt{1 + \nu^2 k(t-k)}$ and ν is a parameter of the strong rank revealing QR factorization.

Error of row sampling

Next, let us complete the bound (3.4). We use a simple technique found in [20], which is described as follows. Once the set of column indices J, with $|J| = k$, is obtained, define $C := A(:, J)$ with QR factorization $C = QR_{11}$, where $Q \in \mathbb{C}^{m\times k}$ and $R_{11} \in \mathbb{C}^{k\times k}$. According to Algorithm 2, we apply row sampling by performing a truncated pivoted QR on Q^*, *c.f.* Remark 3.1, this is

$$Q^*(:, \tilde{I}) \equiv [Q_{11}^*, Q_{21}^*] = \tilde{Q}[\tilde{R}_1, \tilde{R}_2]. \tag{3.17}$$

where $\tilde{Q} \in \mathbb{C}^{k\times k}$ is unitary and $\tilde{R}_1 \in \mathbb{C}^{k\times k}$ is upper triangular. Using (2.13), we get

$$1 = \sigma_k(Q) \leq f(k, m)\sigma_k(\tilde{R}_1) \leq f(k, m)\sigma_k(Q_{11}), \tag{3.18}$$

where $f(k, m)$ can be obtained from Table 1 depending on the algorithm chosen. Hence,

$$\frac{1}{\sigma_k(Q_{11})} \leq f(k, m). \tag{3.19}$$

CUR error bound

Finally, let $C := A(:, J(1 : k))$, $R := A(I, :)$ and $U = A^{-1}(I, J) \in \mathbb{C}^{k\times k}$. The final bound on the error $\|A - CUR\|_2$ is obtained by replacing the bound from Theorem 3.2 and (3.19) in (3.4).

3.3 Discussion on geometric sampling technique

As presented in Section 2.4, the accuracy of a rank-k CUR approximation greatly depends on the choice of row and column indices I, J, with $|I| = |J| = k$. We need to ensure that matrix $A(I, J)$ is as well conditioned as possible; and we know that if it has maximal volume, then we get a suboptimal approximation.

By construction, our CUR approximation first computes J and then using $C = A(:, J)$ it finds I. Hence, finding I can be performed suboptimally in linear-time by using routines such as the Strong RRQR (see Table 1) or *maxvol* [15] to find k most representative rows of C. Therefore, it is most important to select a good set of column indices and geometric sampling allows to find them in linear-time.

To show why the gravity center criterion from Algorithm CUR_GCS is a good choice, let c_j be the j-th column of C, and by seeing the columns of C as points in $\mathbb{C}^m$, let us compute the volume of the simplex formed by these points. For this, we use the Cayley-Menger determinant [36, Pag. 24], the volume of such simplex is given as

$$\mathcal{V}_k := \mu \left| \begin{bmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & d_{12}^2 & \cdots & d_{1k}^2 \\ 1 & d_{21}^2 & 0 & \cdots & d_{2k}^2 \\ \vdots & & & \ddots & \vdots \\ 1 & d_{k1}^2 & d_{k2}^2 & \cdots & 0 \end{bmatrix} \right|,$$

where $\mu = \frac{(-1)^k}{2^{k-1}(k-1)!^2}$ and $d_{jl} = \|c_l - c_j\|_2$ for $j, l = 1, \cdots, k$.

First, note that $\mathcal{V}_k = 0$ if and only if there are at least two linearly dependent columns. Hence, our selection of J can be seen as an approach to obtain a value of $\mathcal{V}_k$ as large as possible while keeping d_{ij} of the same order of magnitude (this is different from the approach that finds maximal projective volume rectangular submatrices [32] and closely related to the approach of volume sampling [12]). For the sake of simplicity, let us consider a smooth kernel function $\mathscr{G} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ to construct A (and hence C). Then, by using the mean value theorem, we get

$$d_{jl}^2 = |y_j - y_l|^2 \sum_{i=1}^{m} |\partial_y \mathscr{G}(x_i, \psi_{lj})|^2,$$

where ψ_{lj} is a real number that lies between y_l and y_j. Hence, the values of d_{jl}^2 are directly related to the distance between the selected target points y. Then, if the selected target points are very close to each other (a behavior that is commonly observed for nearest-neighbors criterion) we get a small value $\mathcal{V}_k$, while the gravity centers criterion is an approach created to maintain d_{jl}^2 different from zero and to keep the rows linearly independent. According to our experiments, the value of $\mathcal{V}_k$ for matrix $C = A(:, J)$, when J is obtained by the gravity center criterion, is in general greater than the case when the nearest-neighbors or uniformly random selection are used, in some cases by one or two orders of magnitude.

4 Numerical Experiments

In this section we numerically show the benefits of our algorithms. We consider the following three kernels encountered in the discretization of elliptic partial differential equations by means of integral equations techniques, see *e.g.* [3, 37],

$$\mathscr{G}_{\mathrm{g}}(x,y) = \frac{1}{4\pi\|x-y\|_2}, \quad \mathscr{G}_{\mathrm{e}}(x,y) = \frac{\exp(\imath\|x-y\|_2)}{\|x-y\|_2}, \quad \mathscr{G}_l(x,y) = -\frac{1}{2\pi}\log(\|x-y\|_2), \tag{4.1}$$

where $\imath$ is the imaginary unit. We construct a matrix $A \in \mathbb{C}^{m\times n}$ by evaluating one of the above kernels on three-dimensional interaction points, *i.e.* $A(i,j) = \mathscr{G}(x_i, y_j)$, where $X := [x_1, \cdots, x_m]$ (red points) are known as sources and $Y := [y_1, \cdots, y_n]$ (green points) as targets. Domains X and Y hold an admissibility condition given as

$$\min(\mathrm{diam}(X), \mathrm{diam}(Y)) \leq \eta\, \mathrm{dist}(X, Y), \tag{4.2}$$

with $\eta < 1$, ensuring that singular values of A exponentially decrease [2, 3]. In our plots we report the value η that makes (4.2) an equality.

Comparing linear-time algorithms

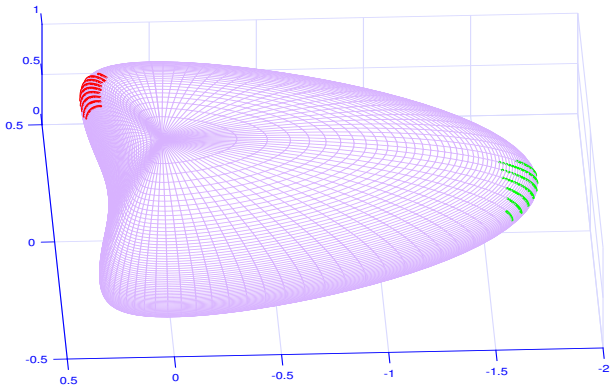
Our first experiments are performed on admissible submatrices taken from global hierarchical matrices. We compare algorithms CUR_GCS and CUR_NNS, introduced in Section 3.1 to ACA with partial pivoting (ACAp) [2, Alg. 2], for which we only modify the first row pivot by an efficient one proposed in [3, Sec. 3.4.3]. For all three methods we also plot values $\delta(k)$ and $\det(M_k)$ to show that not necessarily we need to approximate maxvol submatrices to get higher accuracy. In order to show that CUR_GCS produces a quasi-optimal approximation, we also display a line tagged Bound_MaxVol, corresponding to the value $(k+1)\sigma_{k+1}$ given in eq. (2.20). In all plots we also show the optimal error obtained by the truncated SVD as a reference curve. These plots are displayed in figures 4, 7, 10 and 13.

We also plot the performance of our main algorithm CUR_GCS, and compare it with quadratic cost algorithms QRCP and ACA with full pivoting (ACAf). The latter is a quadratic cost implementation of ACA consisting in iteratively sampling rows and columns using the maximum element of residual matrices, see *e.g.* [35]. We show that our linear-time CUR_GCS algorithm has accuracy comparable with these methods and in some cases even overcomes them. This can be seen in figures 5, 8, 11 and 14.

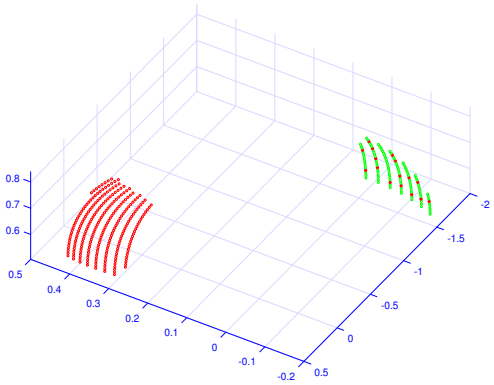
To conclude, we compare the performance of CUR_GCS against ACAp for approximating an entire hierarchical matrix, see Section 4.5.

4.1 BEM matrix from Laplacian kernel

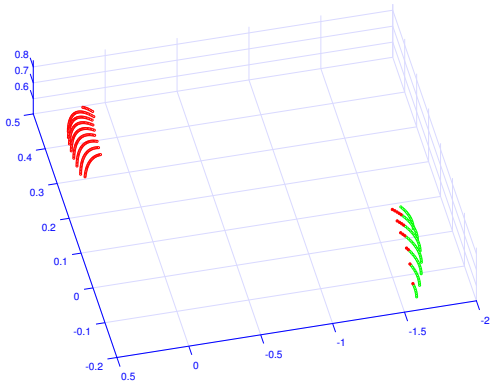
Using the kernel function $\mathscr{G}_l$, we construct matrix A which entries are obtained by evaluating $\mathscr{G}_l$ on source and target points located on a 3D surface proposed in [2], which is shown in Figure 2 together with target points sampled by the gravity centers and nearest-neighbors methodologies.



(a) Admissible subdomains interaction



(b) Sampling targets with the gravity centers criterion



(c) Sampling targets with the nearest-neighbors criterion

Figure 2: Surface from [2], with admissible subdomains created with $\eta = 0.15$.

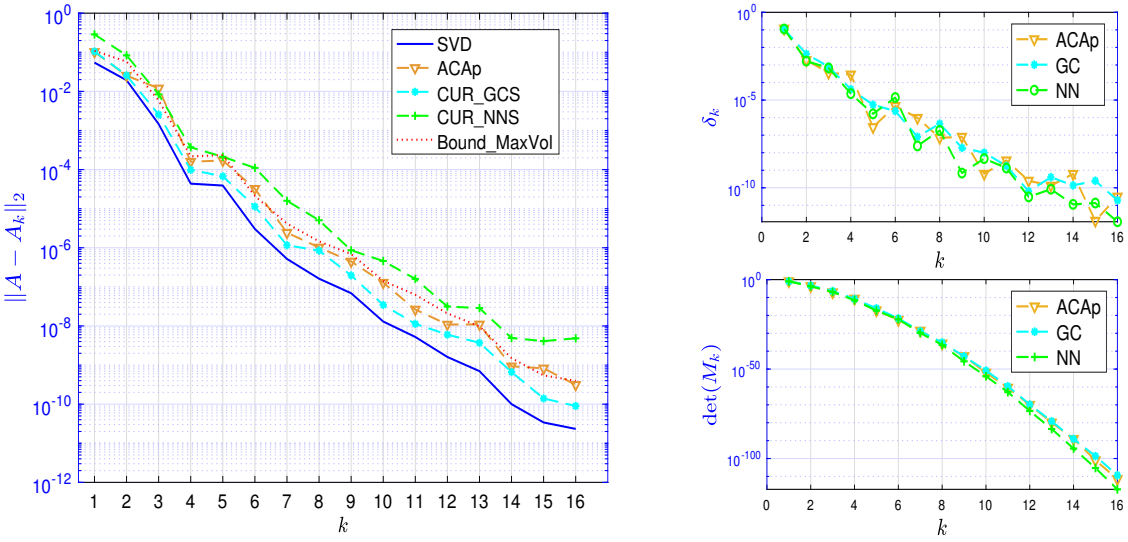


Figure 4: Error convergence of CUR approximation with geometric sampling. The values of $\delta(k)$ and $\det(M_k)$ allow to show the method that better approaches a maximal volume submatrix.

In Figure 4, we observe that even when the value of $\delta(k)$ corresponding to CUR_GCS is smaller than the other methods, we still get better accuracy. For reference, we also show the optimal error obtained by the truncated SVD. In fact, we observe that the accuracy of CUR_GCS is comparable to those of quadratic cost algorithms QRCP and ACAf, as it can be seen in Figure 5.

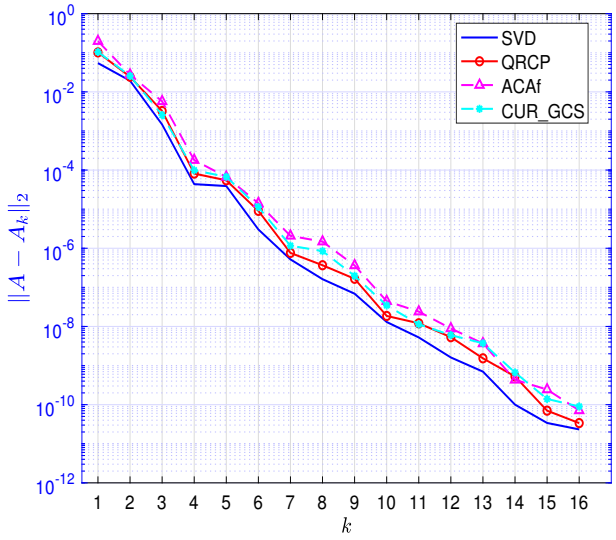


Figure 5: Comparison of our linear cost method CUR_GS versus $\mathcal{O}(mnk)$ cost methods QRCP and ACAf.

4.2 BEM matrix from Exponential kernel

We use kernel $\mathscr{G}_e$ to construct a complex BEM matrix A using a 3D airplane surface that we construct using MATLAB, see Figure 6. Analogously to previous subsection, we show the error convergence for CUR_GCS and compare it with classical methods, showing in all cases a clear improvement, see Figures 7 and 8 respectively.

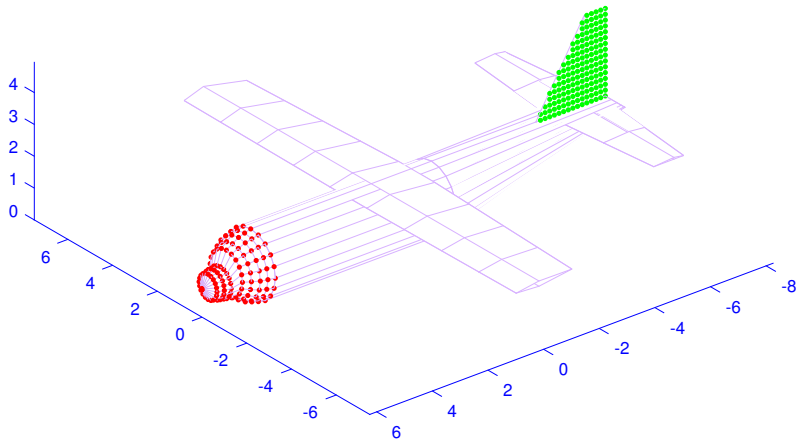


Figure 6: Airplane surface with admissible subdomains created with $\eta = 0.22$.

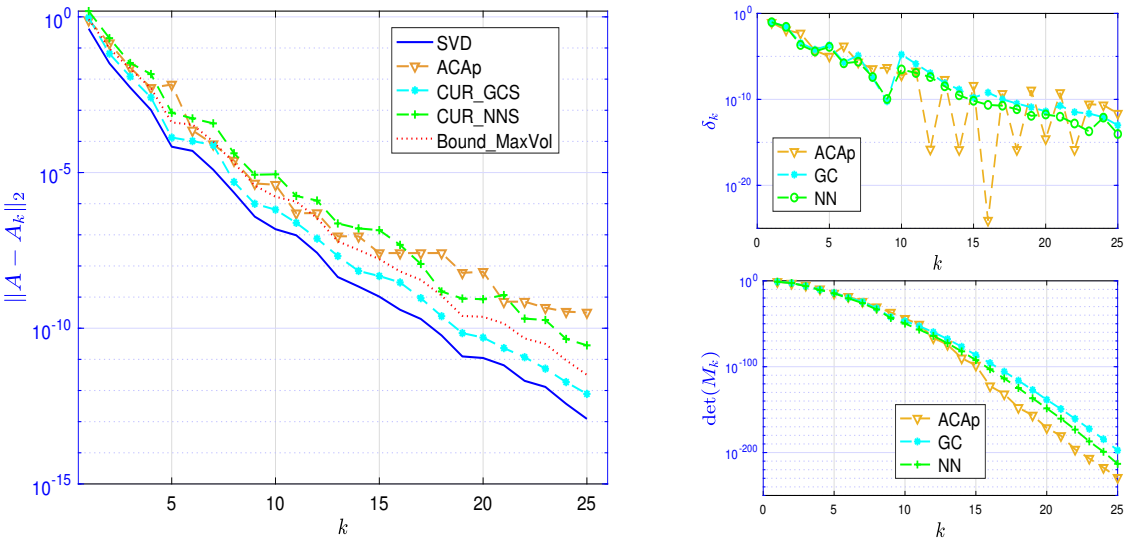


Figure 7: Error convergence of CUR approximation with geometric sampling. The values of $\delta(k)$ and $\det(M_k)$ allow to show the method that better approaches a maximal volume submatrix.

In Figure 8 we compare our linear cost method CUR_GCS versus quadratic cost methods QRCP and ACAf, showing comparable accuracy and even improving them in some cases.

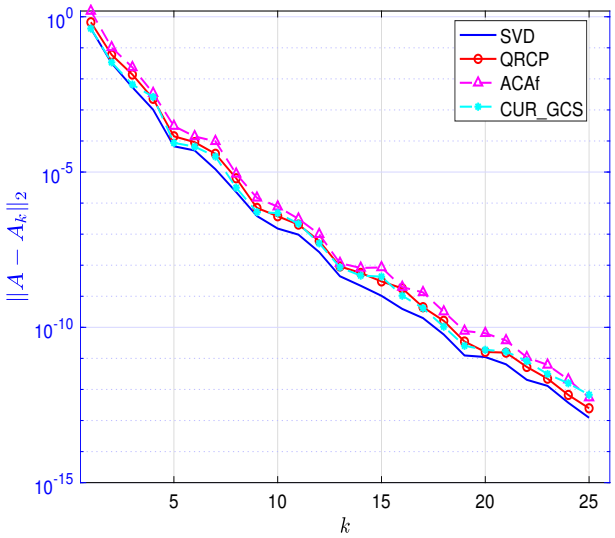


Figure 8: Comparison of our linear cost method CUR_GS versus $\mathcal{O}(mnk)$ cost methods QRCP and ACAf.

4.3 BEM matrix from Gravity kernel

We use kernel $\mathscr{G}_{\mathrm{g}}$ to construct matrix A using a toroid surface that we construct using MATLAB, see Figure 9.

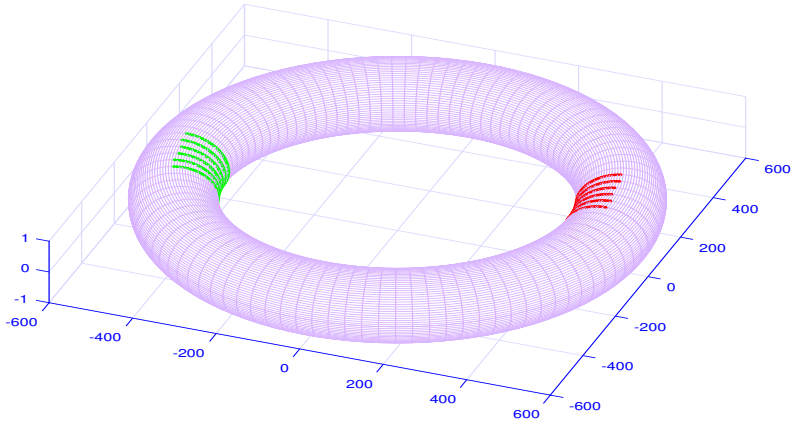


Figure 9: Toroid surface with admissible subdomains created with $\eta = 0.22$.

In Figure 10 we plot convergence curves for linear-time algorithms, showing that CUR_GCS has better accuracy than ACAp (about one order of magnitude). In fact, for this case study, CUR_GCS has practically the same accuracy as quadratic cost algorithms, see Figure 11.

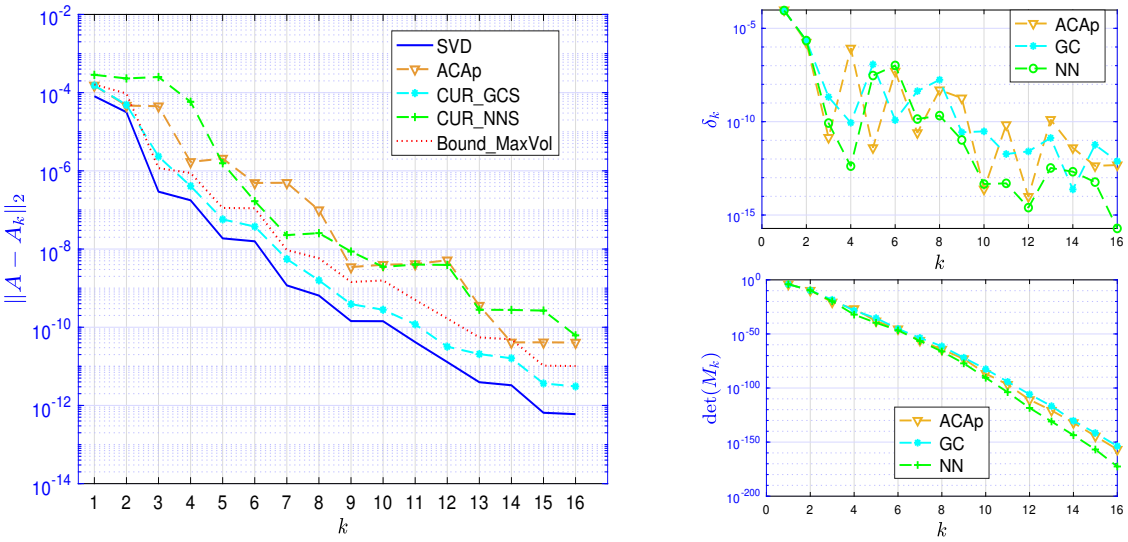


Figure 10: Error convergence of CUR approximation with geometric sampling. The values of $\delta(k)$ and $\det(M_k)$ allow to show the method that better approaches a maximal volume submatrix.

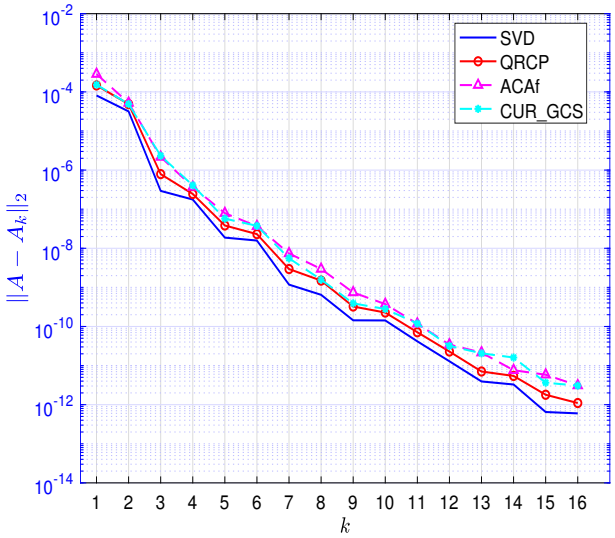


Figure 11: Comparison of our linear cost method CUR_GS versus $\mathcal{O}(mnk)$ cost methods QRCP and ACAf.

4.4 When ACA with partial pivoting fails

Next, we evaluate our algorithms on a challenging problem reported in [3, Sec. 3.4.3]. We build matrix A with a kernel given as

$$\mathscr{G}_{\mathrm{b}}(x,y) = \frac{(x-y)\cdot n_x}{4\pi\|x-y\|_2}, \tag{4.3}$$

where n_x is a unit vector normal to Γ_X at point x, and Γ_X is a surface from where the discretization points X are taken, see Figure 12.

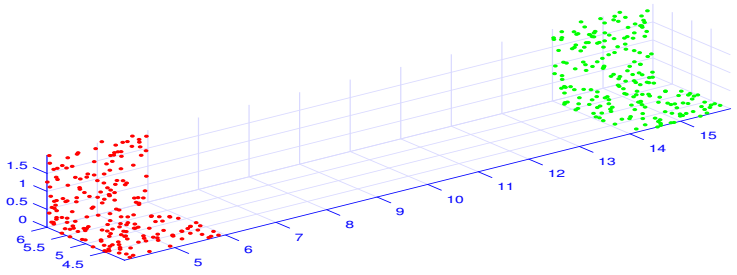


Figure 12: Two admissible subdomains, created with $\eta = 0.39$. By computing their interaction via the kernel function (4.3), they produce a matrix of type (4.4).

When such kernel is evaluated in domains from Figure 12, we can get a matrix A of type

$$\begin{bmatrix} 0 & A_{12} \\ A_{21} & 0 \end{bmatrix}, \tag{4.4}$$

and a simple analysis shows that under this configuration ACAp fails to converge. Even though there are improvements of ACA sampling to ensure convergence, see *e.g. [3, Sec. 3.4.3]*, our methodology is accurate and much simpler, see Figure 13.

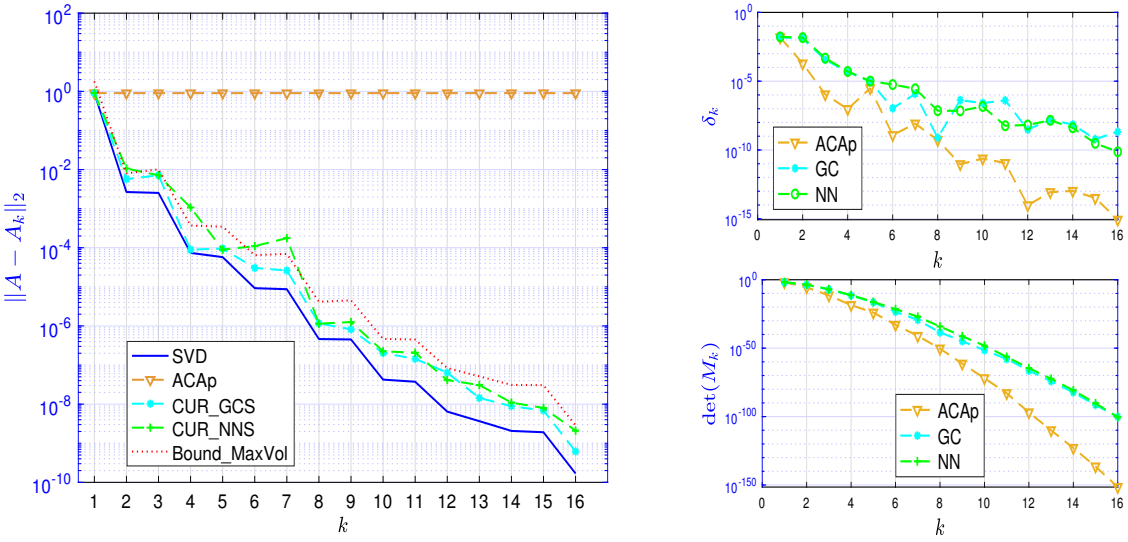


Figure 13: Error convergence of CUR approximation with geometric sampling. The values of $\delta(k)$ and $\det(M_k)$ allow to show the method that better approaches a maximal volume submatrix.

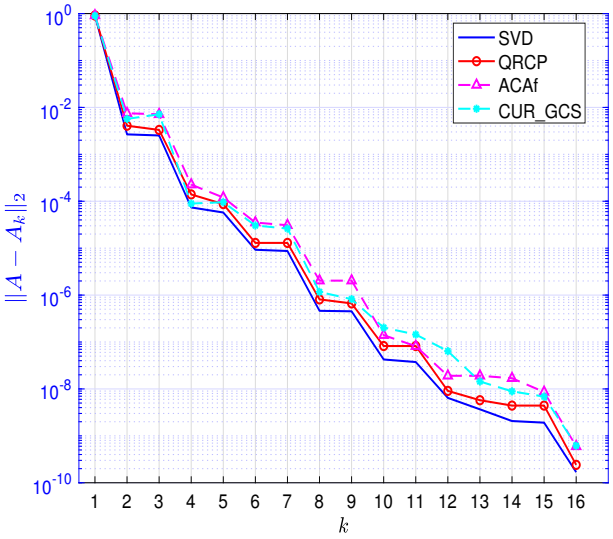


Figure 14: Comparison of our linear cost method CUR_GS versus $\mathcal{O}(mnk)$ cost methods QRCP and ACAf.

4.5 Approximating a Hierarchical matrix

To finalize our numerical experiments, we compare the performance of CUR_GCS and ACAp to approximate all the admissible blocks of a hierarchical matrix, obtained from the discretization of the integral

$$\frac{1}{4\pi}\int_\Gamma\int_\Gamma \frac{1}{\|x-y\|_2}\mathrm{dxdy},$$

where Γ is the surface of a cavity domain, see figure below.

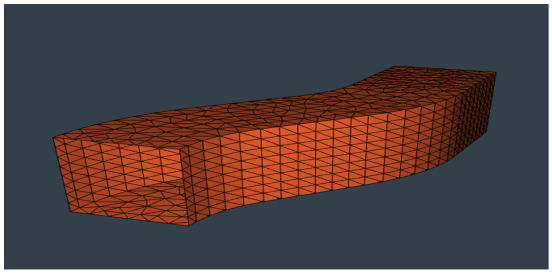


Figure 15: 3D cavity domain.

We use Galerkin discretization using a triangular mesh as in Figure 15, obtaining a square matrix $A \in \mathbb{R}^{N\times N}$, with entries given as

$$A(i,j) = \frac{1}{4\pi}\int_{\tau_i}\int_{\tau_j}\frac{1}{\|x-y\|_2}\varphi_i(x)\mathrm{d}x\varphi_j(y)\mathrm{d}y,$$

where φ_i and φ_j are polynomials of degree one and τ_i, τ_j are triangular elements from the discretization mesh.

The left and right figures below show the approximation error and execution time to form the hierarchical matrix corresponding to A, where the admissible blocks are approximated by low-rank matrices created, respectively, with ACAp and CUR_GCS. We can clearly see the tradeoff between amount of computation and accuracy. We can confirm the linear behavior of the computational cost of our algorithm CUR_GCS as presented in the theory. For the experiment, we have used C++ libraries `HTool`[1] and `BemTool`[2]. We have run the experiment using 4 MPI processes on a MacBook Pro with 4 cores and frequency of 2.5 GHz.

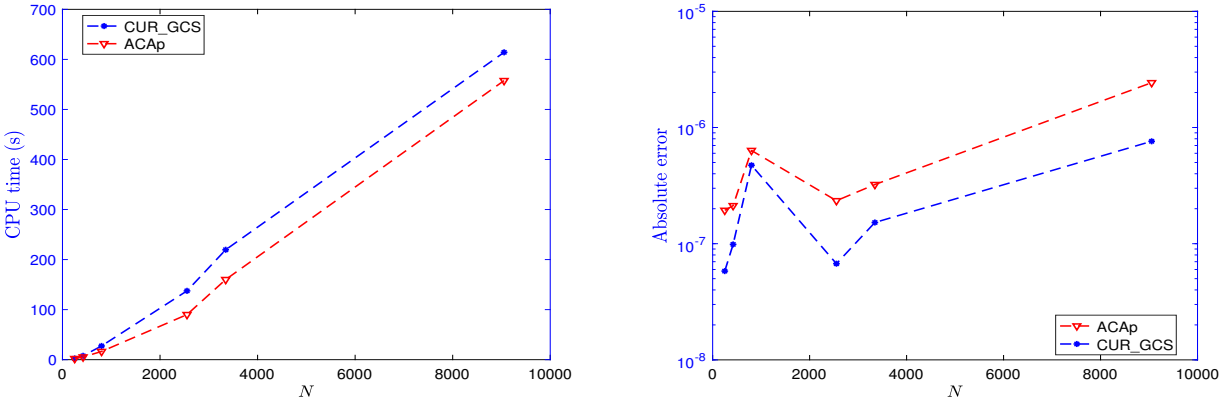


Figure 16: Comparison of the execution time and absolute approximation error between ACAp and CUR_GCS .

From Figure 16, we clearly see the improvement in the approximation error when using CUR_GCS with respect to ACAp, at the expense of performing more arithmetics. We believe that for large scale matrices, an optimized parallel implementation of CUR_GCS (or another CUR created with geometric sampling) would be faster and more accurate than current parallel implementations of ACAp, see *e.g.* [3,

[1] Developed by Pierre Marchand, https://github.com/PierreMarchand20/htool
[2] Developed by Xavier Claeys, https://github.com/xclaeys/BemTool

Sec. 3.4.6]. This is because CUR_GCS depends on QR truncated factorizations that can be computed with small communication cost, see *e.g.* [9]. And communication between processors is known to be a bottleneck for large scale problems running on computer clusters, and optimizing communication leads to considerable speed-ups [10, 11].

5 Conclusions

We have presented a technique called geometric sampling to construct linear-time CUR algorithms for admissible blocks of a hierarchical matrix coming from the discretization of a BEM problem. We have presented a relative error bound for geometric column sampling, which we then extended to a bound for a CUR approximation. Also, this bound can directly be used for truncated QR factorizations and interpolative decompositions. Numerical experiments showed good performance for different integral kernels evaluated on challenging domains. We compared two CUR algorithms created with geometric sampling against ACA with partial pivoting technique. The results showed that our main algorithm CUR_GCS is very efficient and even can handle convergence issues of ACA with partial pivoting, having accuracy comparable with quadratic cost algorithms QRCP and ACA with full pivoting.

As future work, it remains as an open research topic to evaluate the efficiency of geometrical for matrices that not necessarily have exponentially decreasing singular values, such as non-admissible blocks of hierarchical matrices. Moreover, the development of geometric sampling techniques to deal with highly oscillatory kernels is also an interesting research area, where more sophisticated geometric properties of the surfaces, containing source and target points, need to be explored.

Acknowledgement

The authors' work was supported by the NLAFET project as part of European Union's Horizon 2020 research and innovation program under grant 671633. X. Claeys also acknowledges support of the French National Research Agency (ANR) contract ANR-15-CE23-0017-01 (project NonlocalDD).

References

[1] A. Ayala, X. Claeys, and L. Grigori. ALORA: Affine Low-Rank Approximations. Research Report RR-9170, Inria Paris, 2018.

[2] M. Bebendorf. Approximation of boundary element matrices. *Numerische Mathematik*, 86(4):565–589, 2000.

[3] M. Bebendorf. *Hierarchical Matrices*. Springer, Leipzig, Germany, 2008.

[4] S. Börm. *Efficient Numerical Methods for Non-local Operators: H-2 matrix Compression, Algorithms and Analysis*. European Mathematical Society, 2010.

[5] C. Boutsidis, M. Mahoney, and P. Drineas. An improved approximation algorithm for the column subset selection problem. *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 968–977, 2009.

[6] J. Chiu and L. Demanet. Sublinear randomized algorithms for skeleton decompositions. *SIAM Journal on Matrix Analysis and Applications*, 34(3):1361–1383, 2013.

[7] A. Çivril and M. Magdon-Ismail. Exponential inapproximability of selecting a maximum volume sub-matrix. *Algorithmica*, 65(1):159–176, 2013.

[8] S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (STOC)*, pages 537–546, 2008.

[9] J. Demmel, L. Grigori, M. Gu, and H. Xiang. Communication avoiding rank revealing QR factorization with column pivoting. *SIAM J. Matrix Anal. Appl.*, 2015.

[10] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou. Communication-optimal parallel and sequential QR and LU factorizations. *Technical Report No. UCB/EECS-2008-89*, 2008.

[11] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou. Communication-optimal parallel and sequential QR and LU factorizations. *SIAM J. Sci. Comput.*, 34(1):A206–A239, 2012.

[12] A. Deshpande, L. Rademacher, S. Vempala, and G.Wang. Matrix approximation and projective clustering via volume sampling. *Theory of Computing*, 2:225–247, 2006.

[13] W. Fong and E. Darve. The black-box fast multipole method. *Journal of Computational Physics*, 228:8712–8725, 2009.

[14] G. Golub and C. Van Loan. *Matrix Computations*. Jonhs Hopkins University Press, Baltimore, 3rd edition, 1996.

[15] S. Goreinov, I. Oseledets, D. Savostyanov, E. Tyrtyshnikov, and N. Zamarashkin. How to find a good submatrix. *Research Report 08-10, ICM HKBU, Kowloon Tong, Hong Kong*, 2008.

[16] S. Goreinov and E. Tyrtyshnikov. The maximal-volume concept in approximation by low-rank matrices. *Contemporary Mathematics*, (280):47–51, 2001.

[17] S. Goreinov and E. Tyrtyshnikov. Quasioptimality of skeleton approximation of a matrix in the chebyshev norm. *Doklady Mathematics*, 83(3):374–375, 2011.

[18] A. Gray and M. A. N-body problems in statistical learning. *Adv. Neural Inf. Process. Syst.*, pages 521–527, 2001.

[19] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

[20] L. Grigori, S. Cayrols, and J. Demmel. Low rank approximation of a sparse matrix based on lu factorization with column and row tournament pivoting. *SIAM J. Sci. Comput.*, 40(2):C181–C209, 2018.

[21] M. Gu and S. Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM J. Matrix Anal. Appl.*, 17(4):848–869, 1996.

[22] W. Hackbusch. *Hierarchical Matrices: Algorithms and Analysis*. Springer Series in Computational Mathematics, Baltimore, 3rd edition, 2015.

[23] N. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, 2nd edition, 2002.

[24] R. Horn and C. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, New York, USA, 1991.

[25] S. Kapur and D. Long. N-body problems: IES3: Efficient electrostatic and electromagnetic simulation. *IEEE Computational Science and Engineering*, 5(4):60–67, 1998.

[26] P. Koumoutsakos. Fast multipole methods for three-dimensional N-body problems. *Center for Turbulence Research, NASA Ames/Stanford Univ.*, pages 377–390, 1995.

[27] N. Kumar and S. G. Literature survey on low rank approximation of matrices. *Linear and Multilinear Algebra*, 65(11):2212–2244, 2017.

[28] M. Mahoney and P. Drineas. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

[29] W. March and G. Biros. Far-field compression for fast kernel summation methods in high dimensions. *Applied and Computational Harmonic Analysis*, 43(1):39–75, 2017.

[30] P. G. Martinsson and V. Rokhlin. An accelerated kernel-independent fast multipole method in one dimension. *SIAM J. Sci. Comput.*, 29(3):1160–1178, 2007.

[31] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Quart. J. Math. Oxford Ser.*, 11(2):50–59, 1960.

[32] A. Osinsky and N. Zamarashkin. Pseudo-skeleton approximations with better accuracy estimates. *Linear Algebra and its Applications*, 537:221–249, 2018.

[33] N. A. Ozdemir and J.-F. Lee. A low-rank IE-QR algorithm for matrix compression in volume integral equations. *IEEE Transactions on Magnetics*, 40(2):1017–1020, 2004.

[34] C.-T. Pan and P. Tang. Bounds on singular values revealed by QR factorizations. *BIT Numerical Mathematics*, 39(4):740–756, 1999.

[35] S. Rjasanow. Adaptive cross approximation of dense matrices. *International Association for Boundary Element Methods Conference, IABEM*, 2002.

[36] D. Sommerville. *An Introduction to the Geometry of n Dimensions.* Dover, New York, 1958.

[37] O. Steinbach. *Numerical Approximation Methods for Elliptic Boundary Value Problems: Finite and Boundary Elements.* Springer, New York, USA, 2008.

[38] S. Voronin and P. G. Martinsson. Efficient algorithms for cur and interpolative matrix decompositions. *Advances in Computational Mathematics*, 43(3):495–516, 2017.

A Supplemental material

A.1 Proof of Lemma 2.5

For ease of notation, let us consider $X = R_{11}$ and $\tilde{X} = X^{-1}$. Next, let us rewrite the truncated QR from (2.8) as

$$AP_c = \begin{array}{c} \\ m \end{array} \begin{array}{c} \begin{array}{cc} k & m-k \end{array} \\ \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \end{array} \underbrace{\begin{array}{c} \\ \begin{array}{c} k \\ m-k \end{array} \end{array} \begin{array}{c} \begin{array}{cc} k & n-k \end{array} \\ \begin{bmatrix} X & R_{12} \\ 0 & R_{22} \end{bmatrix} \end{array}}_{=:\tilde{R}}, \tag{A.1}$$

where P_c is the permutation matrix obtained from QRCP. Then, we use Gu and Eisenstat's technique [21, Thm. 3.2] as done for the proof of Theorem 3.2, we define

$$Z := \begin{bmatrix} \alpha X & \\ & R_{22} \end{bmatrix} = \begin{bmatrix} X & R_{12} \\ 0 & R_{22} \end{bmatrix} \underbrace{\begin{bmatrix} \alpha I_k & -X^{-1}R_{12} \\ & I_{n-k} \end{bmatrix}}_{=:W} \equiv \tilde{R}W, \tag{A.2}$$

where $\alpha = \sigma_{\max}(R_{22})/\sigma_{\min}(X) = \|R_{22}\|\|\tilde{X}\|_2$. Note that this choice of α ensures that $\sigma_{k+1}(Z) = \sigma_1(R_{22})$. Next, using Theorem 2.4 in (A.2) we get $\sigma_{k+1}(Z) \leq \|W\|_2\sigma_{k+1}(\tilde{R})$. And since from (A.1) we have that the singular values of $\tilde{R}$ are equal to those of A, it follows that

$$\|R_{22}\|_2 \leq \|W\|_2\sigma_{k+1}(A). \tag{A.3}$$

Hence, it remains to bound $\|W\|_2$. Our proof goes as follows. We first show that $\|\tilde{X}\|_F \leq \rho \tilde{f}(k)$, where $\rho := \frac{1}{|X(k,k)|}$ and $\tilde{f}$ is a function to be defined later on. Then, we shall show that $\|W\|_2^2 \leq \tilde{f}(k+1)(n-k)$.

To bound $\|\tilde{X}\|_F$, we proceed to bound each of its entries. Let us compare the i-th row of the equality $X\tilde{X} = I_k$, we get $|\tilde{X}(i,i)| = |\frac{1}{X(i,i)}| \leq \rho$, where the right inequality holds by construction of QRCP. Also,

$$\sum_{l=i}^{h} X(i,l)\tilde{X}(l,h) = 0,$$

for $k \geq h > i$. By using the previous equality, replacing $h = i + j$, we get that

$$\tilde{X}(i,i+j) = \frac{1}{X(i,i)} \left(- \sum_{l=i+1}^{i+j} X(i,l)\tilde{X}(l,i+j) \right), \tag{A.4}$$

for $1 \leq j \leq k - i$.

Next, since $|X(i,i+j)| \leq |X(i,i)|$ for all $j \geq 1$ (by construction of QRCP), we get

$$|\tilde{X}(i,i+j)| \leq \sum_{l=i+1}^{i+j} |\tilde{X}(l,i+j)|. \tag{A.5}$$

Next, by recursively applying the previous inequality, we obtain

$$\begin{aligned}
|\tilde{X}(i,i+1)| &\leq 1\rho, \\
|\tilde{X}(i,i+2)| &\leq 2\rho \\
|\tilde{X}(i,i+3)| &\leq 2^2\rho \\
&\vdots \\
|\tilde{X}(i,i+j)| &\leq 2^{j-1}\rho \\
&\vdots \\
|\tilde{X}(i,k)| &\leq 2^{k-i-1}\rho,
\end{aligned}$$

since the previous bounds hold for any $1 \leq i \leq k$, we set $i = 1$ and write

$$\|\tilde{X}\|_F^2 \leq k|\tilde{X}(1,1)|^2 + (k-1)|\tilde{X}(1,2)|^2 + \cdots + 2|\tilde{X}(1,k-1)|^2 + |\tilde{X}(1,k)|, \tag{A.6}$$

$$\frac{1}{\rho}\|\tilde{X}\|_F^2 \leq 2k - 1 + \sum_{j=2}^{k-1} 4^{j-1}(k-j) =: \tilde{f}(k). \tag{A.7}$$

As a second step, let c be the $j-$th column of matrix $Y := \tilde{X}R_{12}$, then $\forall i = 1, \cdots, k$, we get

$$|c(i)| \leq \sum_{l=i}^{k} |\tilde{X}(i,l)||R_{12}(l,j)| = 1 + \sum_{h=1}^{k-i} 2^{h-1}\rho\,|R_{12}(i+h,j)|, \tag{A.8}$$

since $|R_{12}(i+h,j)| \leq 1/\rho$ (by construction of QRCP). Then,

$$|Y(i,j)| = |c(i)| \leq 2^{k-i}, \quad \text{and} \quad \|Y(:,j)\|_F^2 = \sum_{i=1}^{k} 4^{k-i}, \tag{A.9}$$

this result coincides with a previous bound found in [21, Thm. 7.2].

Next, let us bound $\|W\|_2$, note that

$$\|W\|_2^2 \leq 1 + \|Y\|_F^2 + \|\tilde{X}\|_F^2 \|R_{22}\|_F^2 = 1 + \sum_{j=1}^{n-k} \left(\|Y(:,j)\|_2^2 + \rho \tilde{f}(k) \|R_{22}(:,j)\|_F^2 \right). \tag{A.10}$$

Since QRCP also ensures that $\|R_{22}(:,j)\|_F \leq 1/\rho$ and replacing (A.9) in (A.10), we get

$$\|W\|_2^2 \leq (n-k)\tilde{f}(k+1), \tag{A.11}$$

Finally, replacing (A.11) in (A.3), we get the bound

$$\|R_{22}\|_2 \leq \|W\|_2 \sigma_{k+1}(A) = \sqrt{\tilde{f}(k+1)(n-k)}\,\sigma_{k+1}(A). \tag{A.12}$$

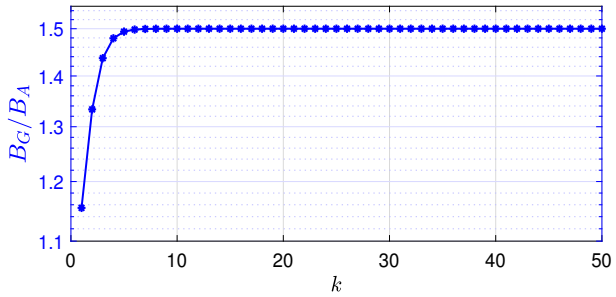


Figure 17: Ratio of classical bound B_G for QRCP (see Table 1) to the new bound B_A from Lemma 2.5.

A.2 Algorithms

A.2.1 CUR via Geometric sampling

We present a MATLAB code for Algorithm 2.

```
%% CUR approximation with Gravity points criterion
% Requires:
% X,Y: Source and target points, given as matrices of size (mxd) and (nxd)
% respectively, where d is the geometric dimension
% k: fixed approximation rank.
% fun: kernel function, e.g. Laplacian kernel: fun = @(x,y) -1/(2*pi)*log(norm(x-y);
% Exponential kernel: fun = @(x,y) exp(1i*norm(x-y))/norm(x-y);
% Gravitation kernel: fun = @(x,y) 1/(4*pi*norm(x-y));
% Returns:
% CUR: a rank-k approximation of matrix A(i,j)=fun(X(i,:),Y(j,:)).
% A \approx CxUxR, where C, R, U are complex matrices of size (mxk), (kxn), (kxk)

function [CUR] = CUR_GCS(fun,X,Y,k)

```

```
m = size(X,1); n = size(Y,1);

% Finding t: number of sampling columns
l = nextpow2(k);
% t = pow2(l);
if(k > pow2(l-1) && k>2 )
   t = pow2(l+1);
else
   t = pow2(l);
end
if(k==1); t = 1; end

C=zeros(m,t);
R=zeros(k,n);

% Decompose target domain into t subdomains
[J] = GC_Sampling(Y,t);
% [J] = NN_Sampling(Y,X,t); % Alternatively use Nearest-Neighbors sampling

% Form matrix C of sampling columns, C is of size mxt
for i=1:size(X,1)
   for j=1:t
      C(i,j) = fun(X(i,:),Y(J(j),:));
   end
end

[Q,~,p_c]=qr(C,'vector');
Q=Q(:,1:k);
C=C(:,p_c(1:k));

% Get column indices
[~,~,p_r]=qr(Q.','vector');
I=p_r(1:k);

% Form Matrix R
for i=1:k
   for j=1:size(Y,1)
      R(i,j) = fun(X(I(i),:),Y(j,:));
   end
end

% Construct the CUR rank-k approximation
G=C(I,:);
CUR=C*(G\R); % Use Algorithm 1 for computing this skeleton approximation in order to
    better handle and control the selected indices

return
```

A.2.2 Selecting columns using Gravity centers

The following algorithm presents a technique to decompose the target domain into t subdomains, in which we select a target point as the one closest to its gravity center, see Figure 2b. Partition is made by

calling function *BinaryPartition*, which is an approach known as *geometrically balanced* clustering, *c.f.* [4, Alg.2], [3, Sec.1.4.1]. Such partition is generated by using a binary tree in which every non-leaf node, $\mathcal{T} := \{y_1, \cdots, y_h\} \subset Y$ with gravity center $\mathrm{g} \in \mathbb{R}^3$, has two sons corresponding to disjoint sets of pints separated by the plane orthogonal to the line having direction given as the first left singular vector of matrix $T := [y_1, \cdots, y_h] - \mathrm{g} \in \mathbb{R}^{3\times r}$, and intersecting it at g. The following algorithm, based on a binary tree structure, costs $O(n \log_2(t))$ floating point operations.

```
%% Select target points using geometrically balanced partition
% Require:
% Y: set of n target points, Y is an mxd matrix, d: geometric dimension
% t: number of subclusters to obtain from Y
% Returns:
% J: indices of target points closest to the gravity centers of the t subclusters

function [J] = GC_Sampling(Y,t)

% Sanity check
l = log(t)/log(2);
if(floor(l) ~= l)
  error('t must be a power of 2!');
end

if(t==1); l=1; end

% Get 1st generation of sons
[G{1},S{1},GGC] = Geo_Bal_Partition(Y);
% GGC: index of target point closest to the gravity center of Y

% Sons of further generations
for i=2:l
S{i} = {};
G{i} = {};
   for j = 1:size(S{i-1},2) % number of clusters at previous generation
         [g,s] = Geo_Bal_Partition(S{i-1}{j});
         S{i} = cat(2,S{i},s);
         G{i} = cat(2,G{i},g);
   end
end

% Getting the indices of target points closest to gravity centers
for j=1:size(G{l},2)
   for i=1:size(Y,1)
   if(G{l}{j}'==Y(i,:))
   J(j)=i;
   end
   end
end

if(t==1)
J=GGC;
end

end
```

```
47
48
49  %% Function Geo_Bal_Partition
50  % Performs geometrically ballanced partition to divide a cluster into two clusters son
51  % Requires:
52  % S_y: cluster of points
53  % Returns:
54  % Son: list of two cluster sons
55  % G: contains the gravity centers of cluster sons
56  % GCC: index of target point closest to the gravity center of S_y
57
58  function [G,Sons,GGC] = Geo_Bal_Partition(S_y)
59
60  [n,~] = size(S_y);
61
62  g = S_y'*ones(n,1)/n;
63  Cov = S_y - g';
64  [~,~,v] = svd(Cov); v=v(:,1);
65
66  L = Cov*v;
67  b_1 = (L>0);
68  b_2 = (L<0);
69  Sons{1} = S_y(b_1,:);
70  Sons{2} = S_y(b_2,:);
71
72  % Getting the index of target point closest to the gravity center of S_y
73  v = (S_y - g.'); z = zeros(n,1);
74     for i=1:n
75        z(i)=norm(v(i,:));
76     end
77  [~,GGC]=min(z);
78
79  % Getting indices of target points closest to the gravity centers of clusters son
80  for j=1:2
81  n_s = size(Sons{j},1); G{j} = Sons{j}'*ones(n_s,1)/n_s;
82  v = (Sons{j} - G{j}.');
83  z = zeros(n_s,1);
84     for i=1:n_s
85        z(i)=norm(v(i,:));
86     end
87  [~,f]=min(z);
88  G{j}=Sons{j}(f,:)';
89  end
90  end
```

A.2.3 Selecting columns using Nearest-Neighbors approach

This approach consists in selecting t target points the closest to the set of source points, see Figure 2c. Then, indices corresponding to these points are the selected columns to be used to compute a CUR approximation and we call the resulting algorithm CUR_NNS as mentioned in section 3.1.

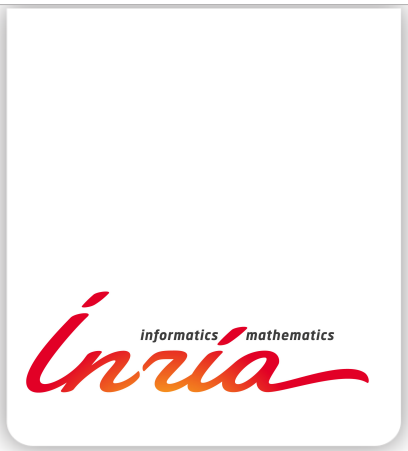
```
1  %% Select target points using Nearest-Neighbors
```

```
% Require:
% Y: set of n target points, Y is an mxd matrix, d: geometric dimension
% t: number of selected points Y
% Returns:
% J: indices of target points closest to the source domain
function [P] = NN_Sampling(Y,X,t)

% Finding the distance
   DX = bsxfun(@minus,Y(:,1),X(:,1)');
   DY = bsxfun(@minus,Y(:,2),X(:,2)');
   DZ = bsxfun(@minus,Y(:,3),X(:,3)');
   D = sqrt(DX.^2+DY.^2+DZ.^2); % The i-th line of D is the distance from
   % the i-th target point to X
   d = min(D(:));

for i=1:size(D,1)
   [dist(i),~] = min(D(i,:));
end
[~,P] = mink(dist,t); % Find t points on Y closest to X
end
```

Algorithm above costs $\mathcal{O}(mn)$ floating point operations. This non-linear complexity can be reduced by using efficient algorithms as the ones presented in [18]. In a recent work, March and Biros [29] showed that nearest-neighbors approach works well in practice for matrices created with kernels depending inversely on the distance of interaction points. However, they did not provide an explicit bound for the error, which we provide in Theorem 3.2. For higher dimension problems, nearest neighbors technique is still applicable by applying approximation techniques such as random trees [8].



RESEARCH CENTRE
PARIS

2 rue Simone Iff
CS 42112 - 75589 Paris Cedex 12

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399