

Solving a large dense linear system by adaptive cross approximation

Katrijn Frederix *Marc Van Barel*

Report TW 513, January 2008



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Solving a large dense linear system by adaptive cross approximation

Katrijn Frederix *Marc Van Barel*

Report TW 513, January 2008

Department of Computer Science, K.U.Leuven

Abstract

An efficient algorithm for the direct solution of a linear system associated with the discretization of boundary integral equations with oscillatory kernels (in two dimensions) is described without having to compute the complete matrix of the linear system. This algorithm is based on the unitary-weight representation, for which a new construction based on adaptive cross approximation is proposed. This low rank approximation uses only a small part of the entries to construct the adaptive cross representation, and therefore the linear system can be solved efficiently.

Keywords : structured matrices; integral equations; cross approximation; rank structured matrix; unitary-weight representation.

MSC : Primary : 65F05; Secondary : 65R20.

Solving a large dense linear system by adaptive cross approximation.

Katrijn Frederix, Marc Van Barel

September 7, 2007

Abstract

An efficient algorithm for the direct solution of a linear system associated with the discretization of boundary integral equations with oscillatory kernels (in two dimensions) is described without having to compute the complete matrix of the linear system. This algorithm is based on the unitary-weight representation, for which a new construction based on adaptive cross approximation is proposed. This low rank approximation uses only a small part of the entries to construct the adaptive cross representation, and therefore the linear system can be solved efficiently.

1 Introduction

Solving integral equations by use of the Galerkin, Nyström or collocation method results in solving a linear system, $Ax = b$, where the matrix entries of A consist of time-consuming evaluation of integrals. The goal of this research is to solve efficiently the linear system without having to compute all entries of the matrix A . Matrices coming from integral equations are large and dense, and have no explicit structure in general but they may be well approximated by a rank structured matrix. This is a set of nested submatrices at some distance from the diagonal, starting in the left bottom matrix corner which can be well approximated by matrices of low rank.

In [1], two compact representations are constructed for a rank structured matrix, a unitary-weight representation and a Givens-weight representation. The last representation is used in [2] as a basis for the QR-based solver for rank structured matrices which solves efficiently the linear system. This solver can be updated for a unitary-weight representation, which will be used throughout this paper. When having a compact representation of the rank structure, the QR-based solver from [2] can be used to solve the linear system efficiently. In fact, the construction of the unitary-weight representation is based on the fact that all the entries of the rank structure are known. This is not preferable, therefore another method to construct a unitary-weight representation using only a few entries of the rank structure, is elaborated. The new construction uses adaptive cross approximation [3, 4, 5]. This purely algebraic method approximates a rank-deficient matrix¹ M by only using a part of the matrix entries, by successively computing skeletons or rank one matrices. Already several methods, for instance, the panel clustering method, the multipole method [6, 7, 8], the hierarchical (\mathcal{H}) matrix method [9, 10] and the mosaic-skeleton method [11, 12, 13, 14, 15] exist to solve efficiently large dense linear systems coming from integral equations. The methods provide an approximation to the matrix in almost linear complexity and solve a perturbed linear system. The first three methods are based on explicitly given kernel approximations by degenerate kernels

¹Throughout the paper a distinction is made between the matrix A and the matrix M . Matrix A denotes a dense, unstructured coefficient matrix and matrix M denotes an arbitrary rectangular and rank-deficient matrix.

(finite sum of separable functions) which may be seen as a block-wise low-rank approximation of the system. The other method is an algebraic method because it works on the matrix level. It partitions the matrix into mosaic blocks (no common elements) mostly based on the \mathcal{H} -format and determines also a block-wise low-rank approximation of the system. The block-wise approximation in all the methods permits a fast matrix-vector multiplication which can be exploited in iterative solvers and can be stored efficiently.

A way to determine the low rank approximation of the blocks in the mosaic-skeleton method is based on Tyrtyshnikov's observation [3], to construct a low rank approximation by only using a few original matrix entries without explicitly dealing with the kernel function. This has as a result that the existing computer code can be adjusted easily, whereas the other non-algebraic methods require a complete recoding of the matrix-vector multiplication including coefficients.

The method proposed in this paper, does not deal explicitly with the kernel function because adaptive cross approximation is used. The method differs in two ways from the previously discussed methods. The first difference is according to the partitioning of the blocks. It is assumed that the approximation of A consist of a rank structured matrix instead of a mosaic partitioning with no common elements. The second difference is that a direct solver is applied instead of an iterative solver.

Section 2 explains the idea behind cross approximation. Section 3 contains information about the construction of the unitary-weight representation. Section 4 shows how to solve efficiently the linear system with a QR-based solver which is based on a unitary-weight representation. Section 5 gives the numerical results for a scattering wave problem. Section 6 states the conclusion.

2 Cross approximation

In general, the best low rank approximation \tilde{M} of a matrix M up to an accuracy τ can be found by the singular value decomposition of the matrix M . But computing the singular value decomposition of a matrix, is unattractive for large-scale computations because of the computational complexity it cannot lead to fast algorithms. The aim of cross or skeleton approximation is to find an approximation using less computational effort and using especially only few entries from the original matrix. Given a matrix $M \in \mathbb{C}^{m \times n}$, the idea is to choose a small set of indices $\hat{n} \subset \{1, \dots, n\}$ and indices $\hat{m} \subset \{1, \dots, m\}$ so that for a matrix $S \in \mathbb{C}^{\hat{n} \times \hat{m}}$ there holds

$$\|M - \tilde{M}\|_2 \leq \tau \quad \text{with} \quad \tilde{M} = M|_{m \times \hat{n}} \cdot S \cdot M|_{\hat{m} \times n} \in \text{Rk } \min\{\#\hat{n}, \#\hat{m}\}. \quad (1)$$

In [3] it is proven that if a sufficiently good low rank approximation exists, then also a cross approximation with almost the same approximation quality exists. It is not stated how to determine the suitable index sets \hat{n} and \hat{m} , and to construct the matrix S . This is done in [4] where it is stated that the matrix S has to be chosen such that it is the submatrix of M with maximal volume (i.e. determinant in modulus). Different methods exist to construct the matrix S they are all based on the same principle, computing successively rank one approximations or skeletons.

The first method is cross approximation with full pivoting. In this case all the entries of the matrix M have to be computed in advance. In fact a low rank approximation of a matrix, is a sum of skeletons ($\tilde{M} = \sum_{l=1}^r a^l (b^l)^T$). In general, to determine a skeleton the following procedure is followed:

- Step 1: Determine the index pivot pair (i^*, j^*) as the indices of the maximal in modulus entry of M and set $\delta := M_{i^*, j^*}$.
- Step 2: Compute the entries

$$a_i := \frac{M_{i, j^*}}{\delta}, \quad i \in I = \{1, \dots, m\},$$

$$b_j := M_{i^*, j}, \quad j \in J = \{1, \dots, n\}.$$

The product of the vector a and b^T is the first skeleton. To compute the other skeletons, the same procedure is applied $(r - 1)$ -times to the remainder ($p = 2, \dots, r$)

$$R_{p-1} = M - \sum_{l=1}^{p-1} a^l (b^l)^T.$$

When the r skeletons are computed, the approximation is

$$\tilde{M} = \sum_{l=1}^r a^l (b^l)^T.$$

In the case that the value $\delta = 0$, the method stops and an approximation of rank $p - 1$ is determined. The two main drawbacks of this method are the determination of the pivot pairs and the computation and updating of the complete matrix M .

The second method introduces a method to determine the pivot pair without having to update the complete matrix M , and is called cross approximation with partial pivoting. The idea behind partial pivoting is to determine the indices of the rows and columns by looking for the maximum in modulus entry in one particular column or row. To start this method an arbitrary row index i_1^* has to be chosen. Then the following procedure will be followed until the r skeletons are determined ($p = 1, \dots, r$). Compute the row

$$(b^p)_j = M_{i_p^*, j} - \sum_{l=1}^{p-1} (a^l)_{i_p^*} (b^l)_j \quad (2)$$

and determine the maximum in modulus element of it, the location of the maximum will be the column pivot j_p^* . Let $\delta = (b^p)_{j_p^*}$, when it is not zero compute the column

$$(a^p)_i = \frac{1}{\delta} \left(M_{i, j_p^*} - \sum_{l=1}^{p-1} (a^l)_i (b^l)_{j_p^*} \right). \quad (3)$$

Set $p = p + 1$ and compute the pivot index i_p^* as the index of the maximum element in modulus of column a^{p-1} . Then proceed as before.

The pivot pair (i^*, j^*) , which is computed in each step p , is not the maximum in modulus element of the whole remaining matrix, but at least for one row. The complete matrix M need no update because only single rows and columns are needed in the approximation and these can be computed using the previous skeletons (see Equation (2)-(3)).

The two methods normally stop when r skeletons are determined. It means that the rank r of the matrix M has to be known in advance, which is in general not the case. Therefore adaptive cross approximation (ACA) is introduced (see Algorithm 1). This method determines the rank adaptively by using a stopping criterion. In the literature, several possible stopping criteria are proposed [5]. These stopping criteria are not based on the complete approximation but on the single already computed skeletons. For instance, let

$$\begin{aligned} \epsilon_{\text{abs}}(l) &= \| a^l \|_2 \| b^l \|_2, \\ \epsilon_{\text{rel}}(l) &= \| a^l \|_2 \| b^l \|_2 / \| a^1 \|_2 \| b^1 \|_2, \end{aligned}$$

then the stopping criterion could be

$$\epsilon_{\text{abs}}(p - 1) \leq \tau \quad \text{or} \quad \epsilon_{\text{rel}}(p - 1) \leq \tau. \quad (4)$$

A drawback of the ACA is that it is not robust, for instance let the matrix M have the following form

$$M = \begin{pmatrix} M_{11} & 0 \\ 0 & M_{22} \end{pmatrix} \quad (5)$$

with $M_{11} \in \mathbb{C}^{m_1 \times n_1}$, $M_{22} \in \mathbb{C}^{m_2 \times n_2}$, $m = m_1 + m_2$ and $n = n_1 + n_2$. When a start row index i^* is chosen, it can be in the index set $I_1 = \{1, \dots, m_1\}$ or $I_2 = \{m_1 + 1, \dots, m_1 + m_2\}$, and as a result the next chosen pivot indices will also be elements of the same index set. So, only one of the two blocks (M_{11} or M_{22}) will be approximated correctly. To avoid this problem an improved adaptive cross approximation (ACA+) is proposed. In this method two reference vectors are chosen, which will tell where to start with the pivot search. Such that both blocks of the matrix will be approximated. This improved method works well for the above mentioned problem but convergence is not yet proven. For more information about these methods and improvements to these methods the reader is referred to [5].

Another method to avoid that only one of the blocks (M_{11} or M_{22}) is well approximated, is to adjust the stopping criterion. Stopping criterion (4) uses only the computed rows and columns, and does not take the rest of the matrix into account. The new stopping criterion which is used in our numerical tests is based on a fixed amount t of arbitrarily taken matrix entries $M_{\hat{i}_l \hat{j}_l}$ for $l = 1, \dots, t$ with $\hat{i}_l \in I \setminus Z_i^p$ and $\hat{j}_l \in J \setminus Z_j^p$ from the part of the matrix which lies outside the area spanned by the already computed skeletons. Here, $Z_i^p = \{i_1^*, \dots, i_p^*\}$ and $Z_j^p = \{j_1^*, \dots, j_p^*\}$ denote the set of used pivot row and column indices. When a new skeleton is determined the value of these entries are updated by subtraction of the new skeleton $(R_p)_{\hat{i}_l \hat{j}_l} = (R_{p-1})_{\hat{i}_l \hat{j}_l} - a_{\hat{i}_l}^p b_{\hat{j}_l}^p$ and compared with the original entries of the matrix M : $|M_{\hat{i}_l \hat{j}_l} - (R_p)_{\hat{i}_l \hat{j}_l}|$. If the following condition holds

$$\frac{|M_{\hat{i}_l \hat{j}_l} - (R_p)_{\hat{i}_l \hat{j}_l}|}{M_{\max}} \leq \tau, \quad \forall l = 1, \dots, t, \quad (6)$$

the algorithm will stop (M_{\max} is the maximum in modulus of the computed entries of the matrix M). The accuracy of the approximation will depend on the amount of matrix entries k that are taken into account in the stopping criterion. For instance, if only one matrix entry of a 100×100 matrix is taken into account the algorithm stops when that matrix entry fulfills the stopping criterion, but it is possible that the matrix is not well approximated. The determination of the amount of matrix entries t to derive an approximation of accuracy τ , is related to the probability that all the chosen matrix entries are in the set of the already well approximated matrix entries. After each iteration, the indices of the matrix elements of the remainder R_p can be divided into three disjunct subsets of indices:

$$R_s = \{(i, j) | i \in Z_i^p, j \in Z_j^p\}, \quad (7)$$

$$R_g = \{(i, j) | \frac{|M_{ij} - (R_p)_{ij}|}{M_{\max}} \leq \tau, i \in I \setminus Z_i^p, j \in J \setminus Z_j^p\}, \quad (8)$$

$$R_b = \{(i, j) | \frac{|M_{ij} - (R_p)_{ij}|}{M_{\max}} > \tau, i \in I \setminus Z_i^p, j \in J \setminus Z_j^p\}, \quad (9)$$

with R_s the set of indices coming from the skeletons, R_g the set of indices outside the area spanned by the skeletons which fulfill criterion (6) and R_b the set of indices outside the area spanned by the skeletons which violate criterion (6). In every iteration a the indices of the new row and column elements will be added to the sets R_s . Over several iterations, the set R_g will obtain more and more indices. For a specific iteration step, this is not always the case because it is possible, when adding a new skeleton, that some indices from the set R_g will move to set R_b . It is assumed that this case does not occur very often. When choosing t indices from the set $R_g \cup R_b$, it is not preferable that all these indices after some iteration will be in the set R_g when the matrix does

not have a good approximation. Let $d_g = \#R_g$ and $d_b = \#R_b$. The probability that an entry is in the set R_g is $q_g = \frac{d_g}{d_g+d_b}$ and for the set R_b the probability is $q_b = \frac{d_b}{d_g+d_b}$, with $q_g + q_b = 1$. The probability that the t elements (taking only t elements) are all in the set R_g is

$$P_t = \frac{d_g(d_g - 1) \dots (d_g - t + 1)}{(d_g + d_b)(d_g + d_b - 1) \dots (d_g + d_b - t + 1)}.$$

If t is relatively small with respect to d_g , then the following holds

$$P_t \approx \left(\frac{d_g}{d_g + d_b} \right)^t = q_g^t.$$

Figure 1 (consider a considerably large matrix ($m, n \geq 100$)) shows for different values of q_g , the probability that all the t indices are in the set R_g . When t increases the probability decreases, for $q_g = 0.8$ and $q_g = 0.7$ the decrease is faster than for $q_g = 0.95$. Normally over several iterations, the set of elements R_g increases because more elements will fulfill criterion (6). If the chosen amount of matrix entries t is small, the probability that all the indices of the t elements are in the set R_g is big, this can be seen in Figure 1. Meaning that there is a great probability that the algorithm will stop before reaching the wanted accuracy τ . In this paper, an approximation of a low rank matrix is wanted without computing all the matrix entries, especially computing as less as possible elements. Therefore it is not possible to know the probability q_g . The only thing that is known about q_g , is that in the beginning it is close to zero, when skeletons are added, it slowly increases and when p (iteration step) approaches the rank (for that given accuracy τ) it becomes close to one. This concludes that the amount of matrix entries t , which has to be checked in the stopping criterion, has to be considerably large ($t \geq 100$), but not too large because as less as possible elements have to be computed. The matrices which are tested are low rank submatrices taken from matrices coming from an integral equation (see Section 5). Figure 2 (a)-(c)-(e) shows, for different matrix sizes, when taking an amount of matrix entries (noted in percentage P : $k = Pnm/100$) what relative error $\|M - \tilde{M}\|/\|M\|$ is obtained (these results are obtained with the availability of all the matrix entries, which is normally not the case). Figure 2 (b)-(d)-(e) shows the average number (sample size is ten) of iteration steps. These figures show that taking more matrix entries, will not increase the accuracy or the amount of iterations. In our future numerical experiments, one percent of the total matrix entries is taken as the number of checkpoints.

Remark: This value 1% will not work in all cases. It depends on what you want to achieve and what kind of matrix it is. In our case, with dense matrices coming from oscillatory integrals it satisfies our requirements.

3 Unitary-weight representation

A unitary-weight or Givens-weight representation is a compact representation for a rank structure of a matrix. The method to construct these representations for rank structured matrices was introduced in [1]. A rank structure \mathcal{R} on $\mathbb{C}^{m \times n}$ is defined in [1] as a collection of so-called structure blocks $\mathcal{R} = \{\mathcal{B}_l\}_l$. Each block \mathcal{B}_l is characterized as a 3-tuple $\mathcal{B}_l = (i_l, j_l, r_l)$ with i_l the row index, j_l the column index and r_l the rank upper bound. A matrix $A \in \mathbb{C}^{m \times n}$ satisfies the rank structure \mathcal{R} if for each l ,

$$\text{rank } A(i_l : m, 1 : j_l)^2 \leq r_l.$$

²The notation is a MATLAB like notation. The colon notation has to be interpreted as follows: $i : m = [i, i + 1, i + 2, \dots, m]$ and $A(i : m, j : n) = A_{i:m, j:n}$.

Algorithm 1 Adaptive cross approximation (ACA)

Choose $i_1^* \in I = \{1, \dots, m\}$, $Z_i := \{\}$, $p = 1$, $J = \{1, \dots, n\}$, stopcrit:=false.

while Not stopcrit **do**

found := false

while Not found **do**

Compute the maximal entry in modulus of the row

$$\begin{aligned}(b^p)_j &:= M_{i_p^*, j} - \sum_{l=1}^{p-1} (a^l)_{i_p^*} (b^l)_j, & j \in J, \\ j_p^* &:= \operatorname{argmax}_{j \in J} |(b^p)_j|, \\ \delta &:= (b^p)_{j_p^*}.\end{aligned}$$

Determine logical: found := $|\delta| > \tau$

Set $Z_i := Z_i \cup \{i_p^*\}$

if Not found **then**

if ($\#Z_i \neq m$) **then**

Choose $i_p^* \in I \setminus Z_i$.

else

stopcrit:= true

end if

end if

end while

Compute the entries of the vectors a^p :

$$(a^p)_i := \frac{1}{\delta} \left(M_{i, j_p^*} - \sum_{l=1}^{p-1} (a^l)_i (b^l)_{j_p^*} \right), \quad i \in I.$$

Set $p = p + 1$.

Compute the maximal entry in modulus of the column

$$i_p^* := \operatorname{argmax}_{i \in I \setminus Z_i} |(a^{p-1})_i|.$$

Determine logical stopcrit by checking the stopping criterion.

end while

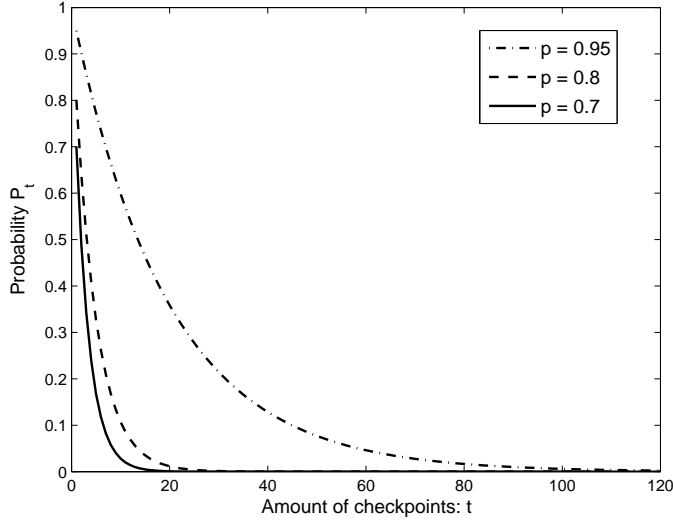


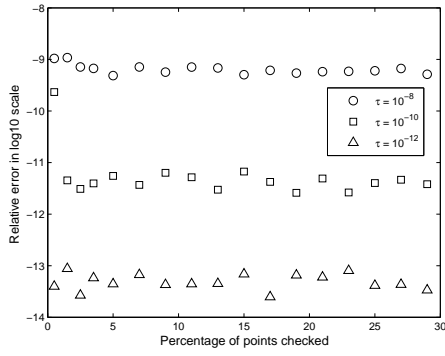
Figure 1: Probability that all the elements are in R_g .

The unitary-weight or Givens-weight representation consists of only a small number of parameters, in fact a pair $(\{Q_l\}_{l=1}^L, \{W_l\}_{l=1}^L)$ where Q_l are unitary transformations and W_l weight matrices ($l = 1, \dots, L$). The representation is internal and works strictly on the area spanned by the structure blocks. Each unitary transformation has a certain action radius, in the sense that it acts only on a limited number of columns. By definition, it is assumed that all structure blocks start from the lower left matrix corner. Two representations are possible, a unitary-weight representation and a Givens-weight representation. The last one is in fact a special case of a unitary-weight representation because each unitary transformation Q_l is then decomposed into a product of Givens arrows. For more details, the reader is referred to [1]. This paper will only focus on the unitary-weight representation.

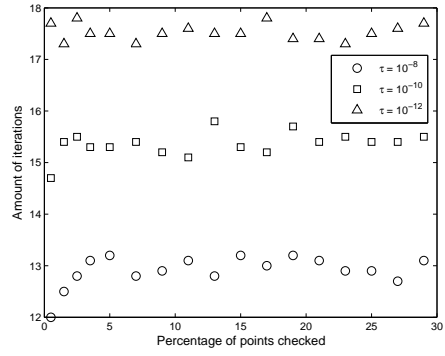
In general, the process to construct a unitary-weight representation of a rank structure works from the bottom to the top of the structure. Consider a rank structure which consists of L blocks as shown in Figure 3. Each block ($l = 1, \dots, L$) is characterized by $\mathcal{B}_l = (i_l, j_l, r_l)$ as defined before. Let $j_0 = 0$. Consider the bottom block L . This block is represented by $M_L = M(i_L : m, 1 : j_L)$. Apply a unitary transformation on block M_L with the intention to create zeros in all these rows except the r_L top rows. The block M_L can now be represented by $M_L = Q_L \begin{bmatrix} \bar{W}_L \\ \mathbf{0} \end{bmatrix}$, where $Q_L \in \mathbb{C}^{(m-i_L+1) \times (m-i_L+1)}$ and $\bar{W}_L \in \mathbb{C}^{r_L \times j_L}$. Save the unitary transformation Q_L and that part of the matrix \bar{W}_L that lies outside the action radius of the next structure block \mathcal{B}_{L-1} as the weight matrix $W_L = \bar{W}_L(1 : r_L, j_{L-1} + 1 : j_L)$.

For all the other blocks $l = L - 1, \dots, 1$ the method proceeds as follows. Construct a new block M_l , by concatenating vertically two matrices. The first matrix contains the elements of the new structure block and the second matrix contains the weights of the previously handled block $l + 1$ which lie inside the action radius of the structure block \mathcal{B}_l . Block M_l becomes

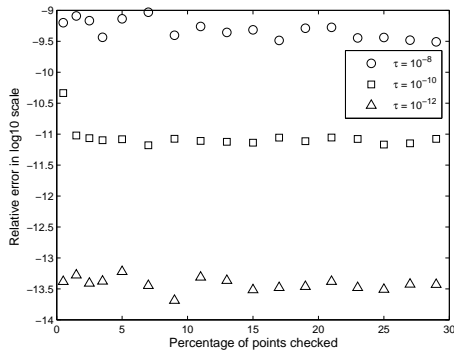
$$M_l = \begin{bmatrix} M(i_l : i_{l+1} - 1, 1 : j_l) \\ \bar{W}_{l+1}(1 : r_{l+1}, 1 : j_l) \end{bmatrix}.$$



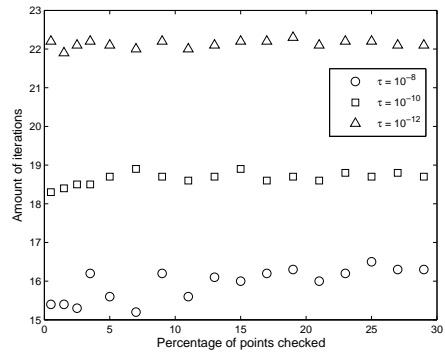
(a) $n = 100$.



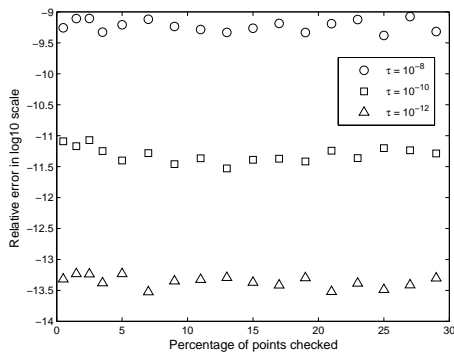
(b) $n = 100$.



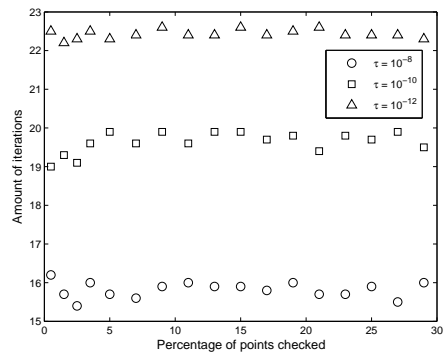
(c) $n = 200$.



(d) $n = 200$.



(e) $n = 400$.



(f) $n = 400$.

Figure 2: Figures (a)-(c)-(e) show the relative error (\log_{10} scale) for different percentage of the matrix elements for three different accuracies $\tau = 10^{-8}, 10^{-10}, 10^{-12}$. Figures (b)-(d)-(f) show the amount of iterations obtained for the different percentages and the three different accuracies. n is size of square matrix.

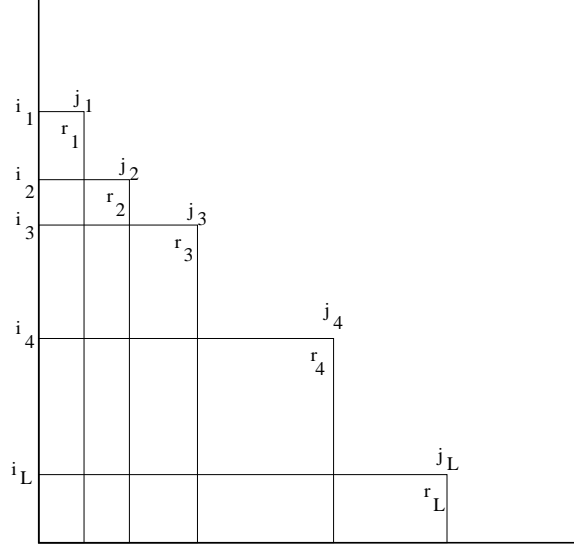


Figure 3: Example of a rank structure with L different structure blocks.

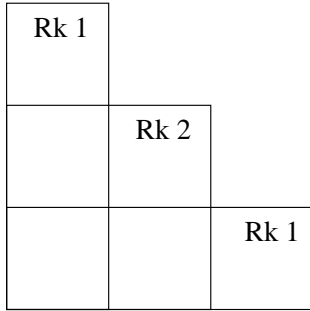
Apply a unitary transformation on block M_l with the intention to create zeros in all these rows except the r_l top rows. The block M_l can now be represented by $M_l = Q_l \begin{bmatrix} \bar{W}_l \\ \mathbf{0} \end{bmatrix}$, where $Q_l \in \mathbb{C}^{(i_{l+1}-i_l+r_{l+1}) \times (i_{l+1}-i_l+r_{l+1})}$ and $\bar{W}_l \in \mathbb{C}^{r_l \times j_l}$. Save the unitary transformation Q_l and that part of the matrix \bar{W}_l that lies outside the action radius of the next structure block \mathcal{B}_{l-1} as the weight matrix $W_l = \bar{W}_l(1 : r_l, j_{l-1} + 1 : j_l)$ (if $l = 1$, $W_1 = \bar{W}_1$).

A simple example consisting of three blocks is elaborated in Figure 4. Figure 4 (a), shows the rank structure with three blocks \mathcal{B}_1 , \mathcal{B}_2 and \mathcal{B}_3 with their ranks. Figure 4 (b), shows the first unitary transformation Q_3 which is applied to the last block. Two rows of zeros and one row of weights are created. The elements which lie outside the action radius of the next structure block are saved in the weight matrix. Figure 4 (c), shows the second unitary transformation which is applied to the original block and a part of the weights of the previous block. Figure 4 (d) gives the schematic picture of the unitary-weight representation.

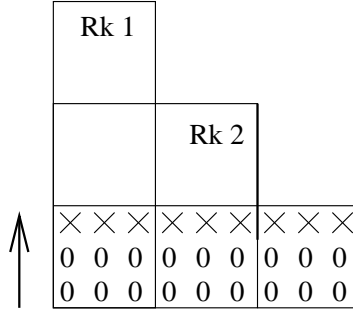
The above construction of the unitary-weight representation, assumes that all entries of the rank structure are computed. As mentioned before, this is time-consuming and therefore another construction of the unitary-weight representation is proposed. This one is based on the fact that only a small part of the original entries of each structure block of the rank structure have to be computed. This new construction uses adaptive cross approximation to construct a low rank approximation for each of the structure blocks.

Consider, as before, a rank structure consisting of L blocks and a construction starting at the bottom of the structure. Let $j_0 = 0$ and start at the bottom block L .

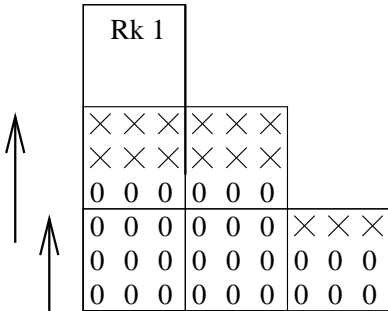
- Step 1 : Construct the low rank approximation of the block $M_L = M(i_L : n, 1 : j_L) \approx A_L B_L$ with aid of the ACA ($M_L \in \mathbb{C}^{(m-i_L+1) \times j_L}$, $A_L \in \mathbb{C}^{(m-i_L+1) \times s_L}$ and $B_L \in \mathbb{C}^{s_L \times j_L}$).
- Step 2 : In fact, now a compact representation is available but the matrix A_L is not a unitary matrix and this is necessary to obtain a unitary-weight representation. Therefore, a QR-factorization is applied to the matrix A_L , such that $A_L = Q_L \begin{bmatrix} R_L \\ \mathbf{0} \end{bmatrix}$, with a unitary



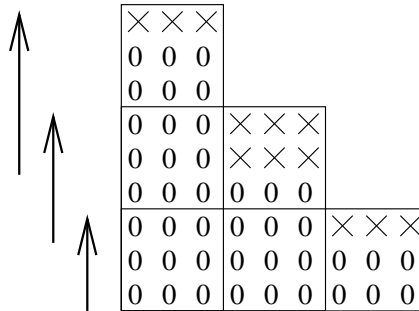
(a) Rank structure with three blocks \mathcal{B}_1 , \mathcal{B}_2 and \mathcal{B}_3 . The surrounding box will not be shown anymore.



(b) The first unitary transformation is applied on the bottom block, and created zeros in the two bottom rows. The top row contains compressed information of the whole block.



(c) The next unitary transformation is applied, and the new block of weights is stored.



(d) Schematic picture of the unitary-weight representation for the rank structure.

Figure 4: Construction of a unitary-weight representation.

matrix $Q_L \in \mathbb{C}^{(m-i_L+1) \times (m-i_L+1)}$ and an upper triangular matrix $R_L \in \mathbb{C}^{s_L \times s_L}$.

- Step 3 : Save the unitary transformation Q_L and the weight matrix $W_L = R_L B_L|_{J_L}$ with $J_L = j_{L-1} + 1, \dots, j_L$ ($W_L \in \mathbb{C}^{s_L \times \#J_L}$). Put $\hat{A}_L = A_L$ and $r_L = s_L$.

For all the other blocks $l = L - 1, \dots, 1$:

- Step 1 : Construct the low rank approximation of the block $M_l = M(i_l : i_l - 1, 1 : j_l) \approx A_l B_l$ with aid of the ACA ($M_l \in \mathbb{C}^{(i_{l+1}-i_l) \times j_l}$, $A_l \in \mathbb{C}^{(i_{l+1}-i_l) \times s_l}$ and $B_l \in \mathbb{C}^{s_l \times j_l}$).
- Step 2 : Construct the block of interest based on previous information. The block of interest consists of the new block M_l and weights containing compressed information about the previous block M_{l+1} . The block of interest can be represented in the following way

$$\bar{A}_l \bar{B}_l = \begin{bmatrix} A_l & 0 \\ 0 & Q_{l+1}^H \hat{A}_{l+1} \end{bmatrix} \begin{bmatrix} B_l \\ B_{l+1}|_{1, \dots, j_l} \end{bmatrix} \approx \begin{bmatrix} M_l \\ R_{l+1} B_{l+1}|_{1, \dots, j_l} \end{bmatrix}, \quad (10)$$

with $\bar{A}_l \in \mathbb{C}^{(i_{l+1}-i_l+r_{l+1}) \times (s_l+r_{l+1})}$ and $\bar{B}_l \in \mathbb{C}^{(s_l+r_{l+1}) \times j_l}$.

- Step 3 : Matrix \bar{B}_l is not of full rank, therefor apply a truncated singular value decomposition (consider only the singular values greater than τ) on \bar{B}_l such that $\bar{B}_l \approx U_l \Sigma_l V_l^*$ with $U_l \in \mathbb{C}^{(s_l+r_{l+1}) \times r_l}$, $\Sigma_l \in \mathbb{C}^{r_l \times r_l}$ and $V_l^* \in \mathbb{C}^{r_l \times j_l}$.
- Step 4 : Construct $\hat{A}_l = \bar{A}_l U_l \Sigma_l$ ($\hat{A}_l \in \mathbb{C}^{(i_{l+1}-i_l+r_{l+1}) \times r_l}$), and apply a QR-factorization on \hat{A}_l such that $\hat{A}_l = Q_l \begin{bmatrix} R_l \\ \mathbf{0} \end{bmatrix}$ with $Q_l \in \mathbb{C}^{(i_{l+1}-i_l+r_{l+1}) \times (i_{l+1}-i_l+r_{l+1})}$ and $R_l \in \mathbb{C}^{r_l \times r_l}$.
- Step 5 : Save the unitary transformation Q_l and the weight matrix $W_l = R_l V_l^*|_{J_l}$ with $J_l = j_{l-1} + 1, \dots, j_l$ ($W_l \in \mathbb{C}^{r_l \times \#J_l}$).

At the end, a unitary-weight representation is obtained for a rank structure without having to compute all the entries of the rank structured matrix.

Remark: Notice the difference between the rank s_l and r_l . With s_l the rank of the block $M(i_l : i_{l+1} - 1, 1 : j_l)$ is indicated (for $l = L$: $i_{l+1} - 1 = n$) and with r_l the rank of the block $M(i_l : n, 1 : j_l)$ is indicated.

4 Solving a rank structured linear system

In [2], it is shown how to compute efficiently the QR-factorization of a rank structured matrix, using the Givens-weight representation which was introduced in [1]. It is also shown how this QR-factorization can be used as a preprocessing step for the solution of linear systems. In this section, the efficient computation of the QR-factorization of a rank structured matrix, using a unitary-weight representation, and the method to use this QR-factorization as a preprocessing step for solving the linear system, are explained. In the previous section only the rank structure of the matrix was considered, in this section the complete matrix is considered to solve the linear system $Ax = b$. It is also considered that A and A^T are rank structured matrices.

The output of the algorithm, to construct a QR-factorization of a rank structured matrix, assuming that there is a given unitary-weight representation for this matrix, consists of the Q and R -factors of the QR-factorization. The Q -factor is decomposed as a product of unitary transformations and the R -factor has the form of a unitary-weight representation.

In [2], the inheritance of the structure by the upper triangular matrix $R = Q^H A$ obtained at the end of the algorithm is investigated. The following result is achieved if A is a square nonsingular matrix. Let $A \in \mathbb{C}^{m \times m}$ be a matrix having two low rank submatrices $A(I_1, J_1) = \text{Rk } r$ and $A(I_2, J_2) = \text{Rk } s$, where

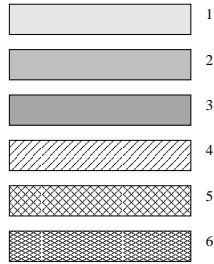


Figure 5: Explanation of coloring.

- I_1 and I_2 form a partition of the index set $\{1, 2, \dots, m\}$,
- $J_1 = \{1, 2, \dots, j_1\}$, for certain j_1 ,
- J_2 is arbitrary.

Then for the QR-factorization $A = QR$ it holds that $R(J_1, J_2) = \text{Rk}(r + s)$. For the proof of this result and for the case that A is a general rectangular matrix the interested reader is referred to [2]. This result is used in the algorithm to construct efficiently a QR-factorization of a rank structured matrix. The algorithm consists of a preparative and a residual phase. In [2], the algorithm is elaborated for a general rank structure. This means that A and A^T are rank structured matrices and that there is no restriction on the size of blocks and the distance between the blocks and the diagonal. In this paper, the following restrictions will be valid for the rank structure. Blocks of the same size are considered, and the distance between the blocks and the diagonal has to be the same for all blocks and this value is a multiple of the size of each block. It is also assumed that a unitary-weight representation is available for the rank structure. Figure 6 (a) shows an example. In Figure 5, the different types of coloring are shown which are used in Figure 6 - 7. Color 1 indicates unstructured elements, color 2 indicates weight elements from the lower part of the matrix, color 3 indicates weight elements from the upper part of the matrix, color 4 indicates elements after applying the precomputed unitary transformations to the elements lying to the right of the current action radius, color 5 indicates elements after applying the precomputed unitary transformations to the column weights (color 2) and color 6 indicates elements after applying the precomputed unitary transformations to the column weights (color 3).

The first phase of the algorithm is the preparative phase. The unitary-weight representation is in fact an internal representation and the precomputed unitary transformations are only applied to the rank structure. In the preparative phase these precomputed unitary transformations are applied to the columns lying on the right of their current action radius, this means to the unstructured part as well as to the weights of the upper triangular representation, hereby updating the representation in the upper triangular part. This is shown in Figure 6 (b). When applying these unitary transformations, it is not allowed to mix real-size elements and weights. To avoid this problem, the column representation is enlarged, see Figure 6 (c)-(d). This means that the rows which lie below the column representation and where the unitary representation has effect on, will be brought into the column representation. After this, it is safe to apply the unitary transformation but it is better to bring the elements of these rows as much as possible to the right. Otherwise the upper triangular part will be completely filled in. The procedure to make the matrix completely upper triangular will be done in the residual phase and will proceed from top to bottom of the matrix. The basic flow of the residual phase is to apply unitary transformations to make the subsequent blocks upper triangular. But mixture of real-size elements and weights is not allowed.

Therefore, the action radius of the column representation has to be regressed. This means that the necessary unitary column representations are spread out on the corresponding rows. See Figure 7 (a)-(d). Note that in Figure 7 (d), the underlying structure corresponds to the predictions from the inheritance of structure by the upper triangular part.

At the end of the algorithm a QR-factorization of A is achieved, which is used to solve the linear system $Ax = b$. This is done by rewriting the system in the form $Rx = Q^H b$ which can be solved by backward substitution. For more information the reader is referred to [2]. The basic flow of the algorithm is determined by solving the subsequent blocks of the upper triangular matrix R by backward substitution, hereby obtaining the subsequent components of the vector x . When a new structure block in the upper triangular part is entered, the vector x has to be multiplied with the inverse of the precomputed unitary column operations associated to this structure block. An auxiliary vector is used for performing these operations, such that the already computed values of the vector x are not overwritten. At the end the full vector x is obtained, hereby solving the linear system.

5 Numerical results

In this section, the described method is tested for a specific problem which results in solving a linear system with a dense, unstructured matrix. First the problem is formulated, then the parameters of the problem are defined, and finally the results of the numerical experiments are discussed.

5.1 Problem formulation

The problem that is considered in this report is the scattering of a time-harmonic wave by a scattering obstacle $\Omega \subset \mathbb{R}^2$ with boundary $\Gamma := \partial\Omega$ [16]. This can be modelled by the Helmholtz equation

$$\Delta u + k^2 u = 0, \quad (11)$$

with Δ the Laplacian and k the wavenumber that determines the frequency of the waves ($k \in \mathbb{R}$). Outside the obstacle the unknown function $u(x)$ is defined and at the boundary Γ a boundary condition $u(x) = f(x)$ is implied. This elliptic boundary value problem can be reformulated as a Fredholm integral equation of the first kind over the boundary by use of Green's identities [16],[17]

$$\int_{\Gamma} G(x, y) q(y) ds_y = f(x), \quad x \in \Gamma, \quad (12)$$

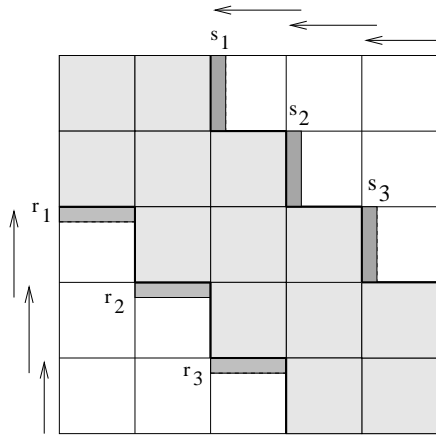
with $G(x, y) = \frac{i}{4} H_0^{(1)}(k|x - y|)$ the kernel function³, $H_0^{(1)}(z)$ the Hankel function of the first kind and order zero, and $i = \sqrt{-1}$. This formulation has two numerical advantages, the unknown function $q(y)$ is now defined on a finite domain Γ and the dimension of Γ is lower than that of \mathbb{R}^2 . With Galerkin discretization this integral equation results in a linear system $Aq = f$ with

$$A_{ij} = \int_{\Gamma} \int_{\Gamma} G(x, y) \varphi_j(y) \varphi_i(x) ds_x ds_y, \quad (13)$$

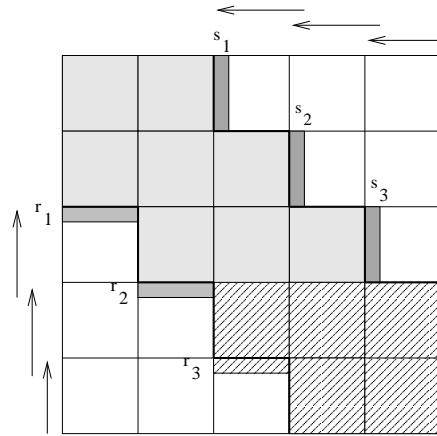
$$f_i = \int_{\Gamma} f(x) \varphi_i(x) ds_x, \quad (14)$$

with $\varphi_i(x)$ ($i = 1, \dots, N$) basis functions. In general, the matrix A is a dense, unstructured matrix with N^2 elements but it can be well approximated with a rank structured matrix. The problem

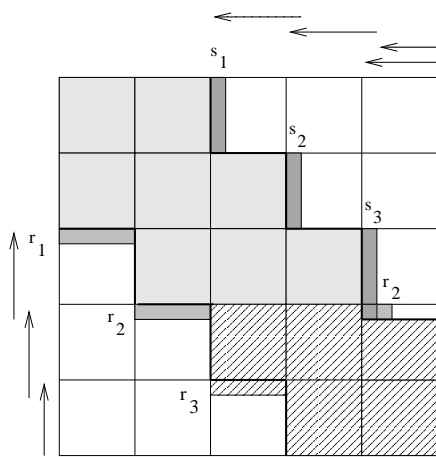
³In general, the function is referred to as Green's function, but in the context of integral equations it is called the kernel function.



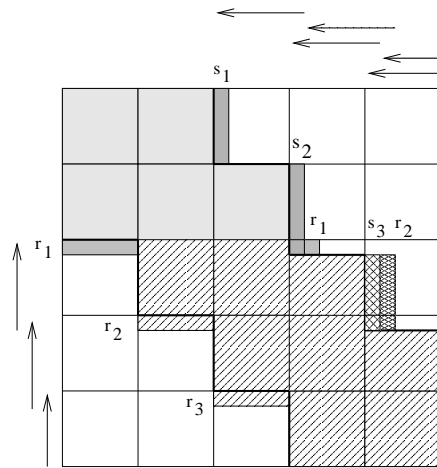
(a) Starting rank structure with unitary-weight representation.



(b) The first two unitary transformations are applied (shaded region).

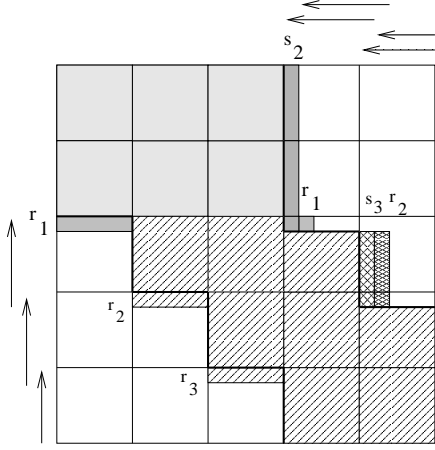


(c) Enlarging action radius of third structure block.

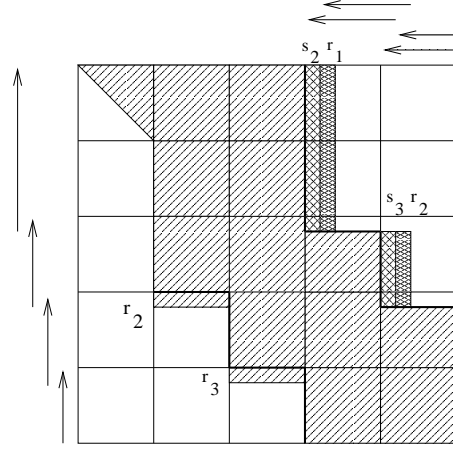


(d) All the precomputed unitary transformations are applied to the elements outside the action radius of the lower blocks. To be entirely correct the action radius of the second block above has to be enlarged before going to the next phase.

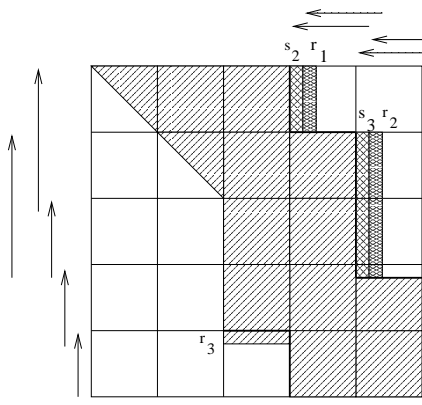
Figure 6: Preparative phase.



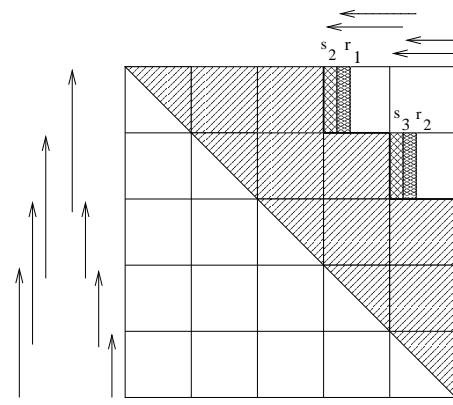
(a) Spreading out of column representation to avoid mixing of real-size elements and weights.



(b) Make first block column upper triangular, and apply the same unitary transformation to the other part of the matrix.



(c) Spreading out of column representation and making second block upper triangular.



(d) Unitary transformations at the left give the Q -factor, and the upper triangular matrix together with unitary-weight representation gives the R -factor.

Figure 7: Residual phase.

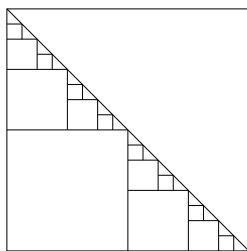


Figure 8: Matrix with rank structures with different block sizes in the lower part of the matrix.

related to solving this type of linear systems is the efficient solution for increasing values of the wavenumber k . At larger frequencies, the solution $q(y)$ is more oscillatory. Therefore the number of basis functions N needs to grow with k , in order to represent the solution with a fixed accuracy. Normally a fixed number (at least two) of basis functions is chosen per wavelength⁴. The values k and N have also an effect on the underlying rank structure of the approximation of the matrix A . At larger frequencies the rank of the structure blocks increases to capture the increasing amount of oscillations. Also when the discretization error of the matrix becomes smaller, for instance, when k is fixed and N increases, the rank becomes larger. In [16], the condition number of A for a Fredholm integral equation of the first kind is derived, $\kappa(A) = O(N)$, this means that the condition number increases linearly with the amount of basis functions.

5.2 Choosing the parameters of the problem

The choice of the problem parameters occurs in two steps. In the first step, the choices according to the discretization of the integral equation (13) are made. To obtain the matrix A and the right-hand side f , choices have to be made according to, for instance, the basis functions and the domain of the integral equation. A black box routine created by D. Huybrechs is used to construct the entries of the matrix A and the right-hand side f of the linear system, given the wavenumber k and the number of test functions N . The parameters of the black box routine are chosen as follows. Tent functions are used as basis functions, the domain of the integral equation is a circle with radius one, the integral formulation used is the Fredholm integral formulation of the first kind and the boundary condition is an incoming plane wave. These choices are the same for all the test problems.

In the second phase, the choices of the rank structure of the matrix for a given value k and N are made. These choices are different for every test problem. To solve the system, the size of the blocks of the rank structure, s_b , and the distance between the rank structure and the diagonal, D_b , have to be defined. These values have to be as small as possible because then the rank structure covers the most elements of the matrix and less elements have to be computed in the neighborhood of the diagonal. This is shown in Figure 8.

The optimal rank structure for each matrix is different because of the wavenumber k and the size of the matrix N . To give an indication for the size of the blocks (which is chosen equal to the rank of the blocks) and the distance to the diagonal, the number of singular values greater than the value $\frac{\|A\|}{\kappa(A)}\tau$ (τ is the chosen accuracy of approximation of the solution q of the linear system

⁴The wavelength is defined as $\lambda = 2\pi/k$. The amount of basis functions in one wavelength is defined as N/k (for a circle with radius one).

	$k = 4$ $N = 256$	$k = 4$ $N = 1024$	$k = 16$ $N = 256$	$k = 16$ $N = 1024$	$k = 64$ $N = 256$	$k = 64$ $N = 1024$
$\tau = 10^{-6}$	$D_b = 16$ $s_b = 16$	$D_b = 32$ $s_b = 32$	$D_b = 32$ $s_b = 32$	$D_b = 32$ $s_b = 32$	$D_b = 96$ $s_b = 32$	$D_b = 192$ $s_b = 64$
$\tau = 10^{-10}$	$D_b = 32$ $s_b = 32$	$D_b = 64$ $s_b = 32$	$D_b = 32$ $s_b = 32$	$D_b = 64$ $s_b = 64$	$D_b = 64$ $s_b = 64$	$D_b = 192$ $s_b = 64$
$\kappa(A)$	245.39	982.83	89.876	368.94	32.832	140.28
$\ A\ $	1.981	1.983	0.731	0.745	0.199	0.285

Table 1: Rank structure values for different matrices and accuracies. D_b is the distance between the rank structure and the diagonal, and s_b is the size of the blocks. Also the condition number $\kappa(A)$ and the norm $\|A\|$ of the different matrices is given.

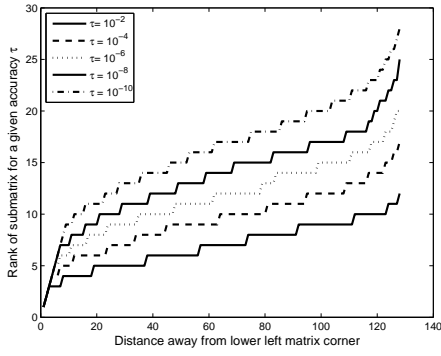
$Aq = f$) for different submatrices in the lower left corner of the matrix A is computed. The results are shown in Figure 9, for different values of $k = 4, 16, 64$ and $N = 256, 1024$. The values related to the point where the rank of the submatrices for a given accuracy increases dramatically, are a good indication for the distance between the rank structure and the diagonal and the size of the structure blocks. For small k , Figure 9 shows that the structure can be taken close to the diagonal and the size of the blocks has to increase as N increases. For larger k , the structure has to be further away from the diagonal and the size of the blocks also has to increase. Looking (in one figure) at the different accuracies, it follows that a higher accuracy needs a rank structure further away from the diagonal and a higher block size.

In section 4, the following restriction on the rank structure was assumed. The distance between the rank structure and the diagonal has to be a multiple of the size of the blocks: $D_b = n_b s_b$ with $n_b \in \mathbb{N}$ (in the example of section 4, $n_b = 1$). The size of the matrix A , m_A , divided by the size of the blocks is a positive integer: $\frac{m_A}{s_b} \in \mathbb{N}$. These restrictions are taken into account in the numerical experiments, where the values of Table 1 for the rank structure are used. Also the condition number and the norm of the different matrices are shown in this table.

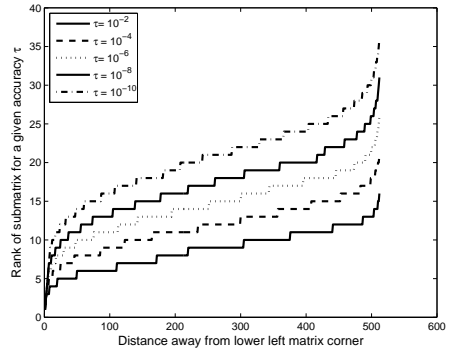
5.3 Results

The numerical results of this subsection are obtained by considering all the elements of the matrix A . This is done such that an estimate of the relative errors $\|q - \tilde{q}\|/\|q\|$ and $\|A\tilde{q} - f\|/\|q\|$ can be derived. Otherwise nothing can be said about the correctness of the proposed method. Values for the numerical experiments are $N = 256, 1024$ and $k = 4, 16, 64$. The method is tested ten times for each problem, and the average results are shown in Table 2 for an accuracy of $\tau = 10^{-6}$ and in Table 3 for an accuracy of $\tau = 10^{-10}$. The results are discussed column by column.

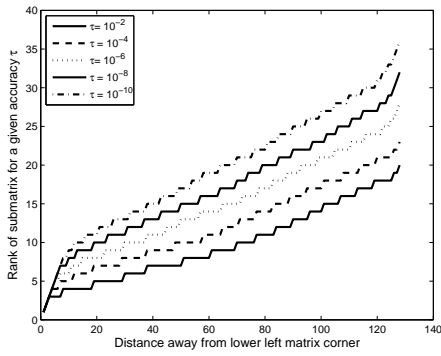
The first two columns of Tables 2 and 3 contain the relative error for the solution and the right-hand side. The relative error of the solution is between $[10^{-4}, 10^{-6}]$ for $\tau = 10^{-6}$ and between $[10^{-8}, 10^{-10}]$ for $\tau = 10^{-10}$, which shows that the approximated solution is very accurate. The relative error of the right-hand side is between $[10^{-5}, 10^{-6}]$ for $\tau = 10^{-6}$ and between $[10^{-9}, 10^{-11}]$ for $\tau = 10^{-10}$. In general, the following condition gives an indication about the accuracies on the



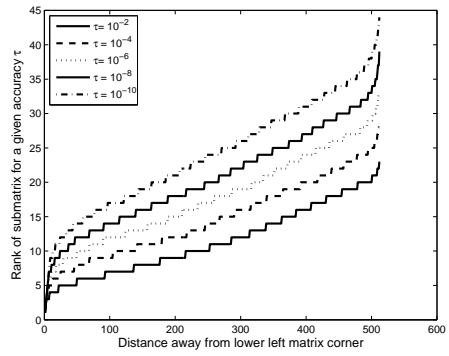
(a) $k = 4, N = 256$



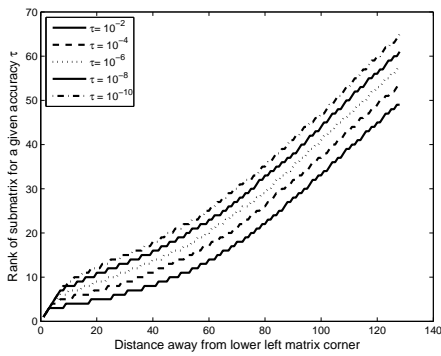
(b) $k = 4, N = 1024$



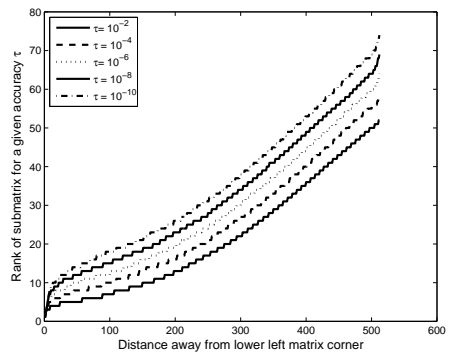
(c) $k = 16, N = 256$



(d) $k = 16, N = 1024$



(e) $k = 64, N = 256$



(f) $k = 64, N = 1024$

Figure 9: Amount of singular values greater than $\frac{\|A\|}{\kappa(A)}\tau$ for different sizes of submatrices of A .

solution:

$$\frac{\|q - \tilde{q}\|}{\|q\|} \leq \kappa(A) \min\left(\frac{\|A - \tilde{A}\|}{\|A\|}, \frac{\|A\tilde{q} - f\|}{\|f\|}\right).$$

Looking at the results together with the condition number it shows that all the test problems satisfy this condition for both the accuracies.

The third column of Tables 2 and 3 discusses the percentage of the computed elements of the rank structure in the lower part of the matrix, i.e. the elements computed in the approximation of the rank structure in the lower part of the matrix. The fourth column of Tables 2 and 3 discusses the percentage of the computed elements, i.e. the elements computed in the approximation of the rank structure and the elements outside the rank structure necessary to solve the linear system. For the matrices of size $N = 256$, the column shows that more than 50% is computed and this percentage rises as the wavenumber increases because the rank structure is in this case further away from the diagonal. For the matrices with size $N = 1024$ and $k = 4, 16$ around 30% of the matrix elements are computed. These matrices have the same rank structure which lies close to the diagonal. But for $k = 16$ the rank of the blocks is bigger, therefore more elements have to be computed.

The last column of Tables 2 and 3 discusses the total average rank, i.e. the sum of the ranks of all the structure blocks, of the nested structure. If k is fixed and N increases, the rank of the structure blocks increases. If N is fixed, and k increases the rank of the structure blocks increases. When the number of basisfunctions in one wavelength is kept constant, this is the case when $\frac{N}{k} = 64, 16$ then also the rank increases. When comparing the two last columns of Tables 2 and 3, it shows that for a higher accuracy a higher rank is needed.

6 Conclusion

The main goal of this paper was to design an efficient algorithm to solve a linear system coming from an oscillatory integral equation without having to compute all the entries of the matrix A . The method proposed is based on adaptive cross approximation which is adapted such that for a given accuracy τ and one percent test points it derives a low-rank approximation of M with accuracy τ . This approximation is used to construct a unitary-weight representation. Exploiting this representation, the corresponding linear system can then be solved efficiently by a QR -based method designed by Delvaux and Van Barel [2].

From the numerical results it can be concluded that an accurate solution is obtained and that the total amount of computed elements is related to the chosen rank structure which is determined by the wave number k and the amount of basis functions N . For small k , the proposed method is efficient because only a small part of the elements have to be computed. For higher values of k , the proposed method becomes less efficient.

Acknowledgement

The authors wish to thank Daan Huybrechs for providing the black box method for computing the coefficient matrix and the right-hand side vector, and for the fruitful discussions.

References

- [1] S. Delvaux and M. Van Barel. A Givens-weight representation for rank structured matrices. Technical Report TW453, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, March 2006. (To appear in SIMAX).

	$\frac{\ q - \tilde{q}\ }{\ q\ }$	$\frac{\ A\tilde{q} - f\ }{\ f\ }$	Elements computed in rank structure	Elements computed in total matrix	Rank
$k = 4$ $N = 256$	$9.665 \cdot 10^{-6}$	$3.225 \cdot 10^{-7}$	48%	57%	16
$k = 4$ $N = 1024$	$4.758 \cdot 10^{-5}$	$4.495 \cdot 10^{-7}$	23%	30%	21
$k = 16$ $N = 256$	$1.983 \cdot 10^{-5}$	$2.099 \cdot 10^{-6}$	46%	65%	20
$k = 16$ $N = 1024$	$6.545 \cdot 10^{-5}$	$2.371 \cdot 10^{-6}$	28%	34%	28
$k = 64$ $N = 256$	$1.512 \cdot 10^{-5}$	$3.990 \cdot 10^{-6}$	70%	90%	24
$k = 64$ $N = 1024$	$7.099 \cdot 10^{-5}$	$9.192 \cdot 10^{-6}$	30%	57%	40

Table 2: Results for accuracy $\tau = 10^{-6}$.

	$\frac{\ q - \tilde{q}\ }{\ q\ }$	$\frac{\ A\tilde{q} - f\ }{\ f\ }$	Elements computed in rank structure	Elements computed in total matrix	Rank
$k = 4$ $N = 256$	$2.280 \cdot 10^{-9}$	$3.997 \cdot 10^{-11}$	46%	64%	20
$k = 4$ $N = 1024$	$7.052 \cdot 10^{-9}$	$5.187 \cdot 10^{-11}$	30%	41%	32
$k = 16$ $N = 256$	$3.294 \cdot 10^{-9}$	$2.017 \cdot 10^{-10}$	59%	73%	28
$k = 16$ $N = 1024$	$1.268 \cdot 10^{-8}$	$3.468 \cdot 10^{-10}$	25%	38%	37
$k = 64$ $N = 256$	$1.855 \cdot 10^{-9}$	$4.401 \cdot 10^{-10}$	75%	91%	40
$k = 64$ $N = 1024$	$1.274 \cdot 10^{-8}$	$1.037 \cdot 10^{-9}$	37%	62%	58

Table 3: Results for accuracy $\tau = 10^{-10}$.

- [2] S. Delvaux and M. Van Barel. A QR-based solver for rank structured matrices. Technical Report TW454, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, March 2006.
- [3] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. A theory of pseudoskeleton approximations. *Linear Algebra and its Applications*, 261:1–21, 1997.
- [4] S. A. Goreinov, N. L. Zamarashkin, and E. E. Tyrtyshnikov. Pseudo-skeleton approximations by matrices of maximal volume. *MathNotes*, 62(4):515–519, 1997.
- [5] S. Börm, L. Grasedyck, and W. Hackbusch. Hierarchical matrices. Technical Report 21, Max-Planck-Institute for Mathematics in the Sciences, Inselstrasse 22, 04103 Leipzig, Germany, 2003.
- [6] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, 1987.
- [7] V. Rokhlin. Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics*, 60:187–207, 1983.
- [8] H. Cheng, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm in three dimensions. *Journal of Computational Physics*, 155:468–498, 1999.
- [9] W. Hackbusch. A sparse matrix arithmetic based on \mathcal{H} -matrices. part I: Introduction to \mathcal{H} -matrices. *Computing*, 62:89–108, 1999.
- [10] W. Hackbusch and B. N. Khoromskij. A sparse \mathcal{H} -matrix arithmetic, part II: Application to multi-dimensional problems. *Computing*, 64(1):21–47, 2000.
- [11] E. E. Tyrtyshnikov. Mosaic-skeleton approximations. *Calcolo*, 33:47–58, 1996.
- [12] S. Kurz, O. Rain, and S. Rjasanow. Application of the adaptive cross approximation technique for the coupled BE-FE solution of symmetric electromagnetic problems. 32:423–429, 2003.
- [13] E. E. Tyrtyshnikov. Incomplete cross approximation in the mosaic-skeleton method. *Computing*, 64:367–380, 2000.
- [14] S. A. Goreinov, E. E. Tyrtyshnikov, and A. Y. Yeremin. Matrix-free iterative solution strategies for large linear systems. *Numerical Linear Algebra with Applications*, 4(4):273–294, 1997.
- [15] M. Bebendorf and S. Rjasanow. Adaptive low-rank approximation of collocation matrices. *Computing*, 70:1–24, 2003.
- [16] D. Huybrechs. *Multiscale and hybrid methods for the solution of oscillatory integral equations*. PhD thesis, Department of Computer Science, Katholieke Universiteit Leuven, May 2006.
- [17] R. P. Kanwal. *Linear Integral Equations*. Birkhäuser, second edition, 1997.