Adaptive cross approximation based solver for boundary element method with single domain in 3D

T. Grytsenko & A. Peratta
Wessex Institute of Technology, Southampton, UK

Abstract

The approach presented in this paper is based on the Adaptive Cross Approximation (ACA) applied to BEM-derived matrices. The ACA method is purely algebraic and does not require information on the kernel. In this way, it offers an interesting alternative to the well established Fast Multipole Method. The algorithm uses a hierarchical approach for the matrix storage and manipulation. This paper describes the assembly approach for the usual H and G matrices coming from BEM equations into a hierarchical coefficient matrix A_H. The efficiency of the developed scheme is tested for a potential problem involving a 3D unitary cube, and then the method is applied to an electrostatic problem involving an anatomic model of the human body. This 3D example consisting of 80 000 degrees of freedom could easily be solved in a low specification PC.
Keywords: boundary elements, adaptive cross approximation, hierarchical matrices, linear systems of equations.

1 Introduction

This paper considers some technical aspects related to implementation of the Adaptive Cross Approximation (ACA) algorithm [1] to general potential problems solved by the collocation Boundary Element Method (BEM). The great advantage of the ACA algorithm is that it is purely algebraic and does not need to operate with the kernel. The algorithm uses a hierarchical matrix storage approach [2] where the matrix is split into many blocks classified into two categories, weakly and strongly coupled. The former are off-diagonal blocks which represent remote interactions between source points and field elements, and therefore can be approximated by low-rank matrices. For this purpose the ACA algorithm is used [2]. These blocks are stored in a special Rk-format [2]. The latter blocks describing close interactions between source points and field elements are stored without any changes in a full-matrix format [2].

The novelty of this paper consists in a formulation of the approach allowing to assemble both H and G matrices from BEM formulation (see Section 2) within only one hierarchical matrix A_H. Two different ways to implement the approach for existing BEM formulations are considered. This paper demonstrates the efficiency of the approach with more than 80 000 degrees of freedom (DOF) and reports its numerical properties. This paper is organised as follows: in Section 2, the BEM method formulation for a general electrostatic problem is presented; in Section 3, the assembly scheme applied to a simple example is described; Section 4 suggests different ways to implement the approach within existing BEM formulations; Section 5 presents the numerical results of the application to electrostatic problem; Section 6 elaborates the concluding remarks.

2 Boundary element method formulation

Consider solving the Laplace equation for the unknown scalar field $u(\mathbf{x})$ given by:

$$\nabla^2 u(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega \subseteq R^3 \tag{1}$$

where Ω is the integration domain with boundary $\Gamma = \partial(\Omega)$ of outward unit normal $\hat{\mathbf{n}}$, and proper boundary conditions are applied to Γ, i.e. Dirichlet or Neumann type. Then, the boundary integral formulation for eq. (1) can be expressed in the following way [3]:

$$c_i u(\mathbf{x}_i) + \int_\Gamma q^*(\mathbf{x}, \mathbf{x}_i) u(\mathbf{x}) d\Gamma - \int_\Gamma u^*(\mathbf{x}, \mathbf{x}_i) q(\mathbf{x}) d\Gamma = 0, \tag{2}$$

where q is the normal derivative of u in $\hat{\mathbf{n}}$ direction, u^* is the Green's function of Laplace equation such that $\nabla^2 u^* = 0$, q^* its normal derivative in $\hat{\mathbf{n}}$ direction, and c_i is the self-interaction coefficient. In 3D problems u^* and q^* become: $u^* = 1/(4\pi r)$ and $q^* = -\mathbf{r} \cdot \hat{\mathbf{n}}/(4\pi r^3)$, respectively, where $\mathbf{r} = \mathbf{x} - \mathbf{x}_i$ and $r = |\mathbf{r}|$ is the distance between the field ($\mathbf{x}$) and source ($\mathbf{x}_i$) points.

In order to solve eq. (2), Γ is discretised into N_e constant triangular boundary elements Γ_e. Thus, the discretised boundary integral equation becomes:

$$c_i u_i + \sum_{j=1}^{N_e} h^{(j)} u_j - \sum_{j=1}^{N_e} g^{(j)} q_j = 0, \tag{3}$$

where u_j is the average potential in the j-th element, q_j is the mean normal flux in the centroid of j-th element, and $q^{(j)}$ and $h^{(j)}$ are the following BEM integrals:

$$g^{(j)} = \frac{1}{4\pi} \int_{\Gamma_j} \frac{1}{r} d\Gamma_j \tag{4}$$

$$h^{(j)} = \frac{1}{4\pi} \int_{\Gamma_j} \frac{\mathbf{r} \cdot \mathbf{n}}{r^3} d\Gamma_j \tag{5}$$

The assembly scheme consists in appending one equation (2) per each selected source point $\mathbf{x}_i$, to the global system of equations ($\mathbf{A}\,\mathbf{x} = \mathbf{b}$), where $\mathbf{A} \in R^{n \times m}$ contains the coefficients $h^{(j)}$ and $g^{(j)}$, $\mathbf{x}$ is a 1-column array with the unknown u and q, $\mathbf{b}$ is the right hand side 1-column array formed by the boundary conditions. The matrix A will be presented in hierarchical format using the techniques described in [2]. Each admissible block [2] is approximated using the ACA algorithm [1].

3 The assembly scheme

In order to take advantage of the efficiency provided by H-matrices in conjunction with ACA approximation it is not only necessary to build the structure of the matrix A_H from the equation $Ax = b$ but also to compute all the entries and assemble them in an appropriate way. There are two radically different ways to perform this operation: for the case when all the entries of matrix A have to be computed in advance and for the case when only several rows and columns of the initial matrix have to be calculated. The former approach has been referred to as fully-pivoted ACA algorithm in [1], while the latter as the partially-pivoted version of ACA algorithm [1].

Both approaches are based on a hierarchy of blocks in the coefficient matrix A_H called supermatrix [2]. Each block represents one of three cases: Full-block, Rk-block or a supermatrix of a lower level. The Full-blocks can not be approximated and therefore are computed and stored without any changes, while the Rk-blocks are approximated an stored in a factorized form according to the following expression:

$$B_{RK} = \sum_{i=1}^{k} UV^T \tag{6}$$

where k is the rank of the approximated block, $U \in R^{nxk}$ and $V \in R^{mxk}$, n and m are number of rows and columns in an Rk-block.

If the block under analysis is a supermatrix, it has to be further divided into either Full- or Rk-block. Due to the structure of A_H, the assembly approach is a recursive process.

The only difference with the partially pivoted ACA approach [1] is that the Rk-blocks can be generated on the fly during the assembly according to the algorithm of partially pivoted ACA [1]. The assembly scheme for the Full-blocks will remain the same: the matrix entry A_{ij} corresponds to the contribution of boundary element j to i-th BEM equation. Therefore A_{ij} element involves the source point x_i and the DOF associated with j BE.

It is worth mentioning that the coefficients of H and G matrices can be computed using different numerical algorithms. The computation of H and G is performed by an external function that gives as an output the values of corresponding h and g coefficients. Hence, the whole algorithm for the computation of sub-block entries can be represented in the following way:

$$Block[i][j] = \begin{cases} h_{ij} & \forall i \neq j \text{ and } bc[j] = 1 \\ g_{ij} & \forall bc[j] = 0 \\ 1/2 & \forall i = j \text{ and } bc[j] = 1 \end{cases} \tag{7}$$

where $i = 1 : rows$, $j = 1 : cols$, vector bc contains the information about boundary condition (BC), i.e. it contains 0 or 1 for the corresponding degree of freedom if appropriately the potential or normal flux are known:

$$bc[j] = \begin{cases} 0 & \text{if DOF } j \text{ has Dirichlet-type BC} \\ 1 & \text{if DOF } j \text{ has Neumann-type BC} \end{cases} \tag{8}$$

The i-th row of the right hand side vector RHS can be computed as follows:

$$RHS[i] = \sum_{j=1/bc[j]=1}^{cols} g_{ij}q[j] - \sum_{j=1//bc[j]=0}^{cols} h_{ij}u[j] \tag{9}$$

where $i = 1 : rows$, vector u contains appropriate values for known potentials while vector q stores known normal fluxes. If $bc[i] = 0$, then the $RHS[i]$ has to be updated as follows:

$$RHS[i] = RHS[i] - u[i]/2 \tag{10}$$

3.1 An example

This section illustrates the assembly scheme with a worked example. The problem represents a Laplacian equation in a 3D unitary cube (ie. of dimensions $1 \times 1 \times 1$) with conductivity $k = 1$ (see Figure 1). The model has mixed boundary conditions: Dirichlet for the potentials and Neumann for the normal fluxes. The boundary has been meshed with constant triangular elements yielding 12 discontinuous freedom nodes which are located in the centroid of each element (see Figure 2). Taking into account the boundary conditions, the model yields $N = 12$ unknowns: 4 normal fluxes and 8 potentials. A cluster tree [2] for this model is obtained with the help of bisection clustering technique [4] using the maximal number of indices per node $n_{\min} = 4$ [2]. Its clusters and the corresponding H-matrix A_H are shown in Figure 2. The number in each block represents its rank, shaded blocks represent the Full-blocks while the white ones are Rk-blocks. A_H a square matrix $A_H = [12 \times 12]$ which has 16 blocks where only 2 of them satisfy the admissibility condition [2]. Those two blocks are admissible and stored in Rk-format [2]. For the sake of simplicity, the assembly scheme of only two blocks, i.e. $B_F = [(1, 2) \times (1, 2)]$ and $B_{RK} = [(1, 2) \times (11, 12)]$ is considered. The block B_F is represented by

$vertex/el_{id}$	el_1	el_2	el_{11}	el_{12}
x_1	(0,1,0)	(0,0,0)	(1,1,0)	(1,1,0)
x_2	(0,0,0)	(1,0,0)	(1,1,1)	(0,1,0)
x_2	(0,0,1)	(0,0,1)	(1,0,1)	(1,1,1)

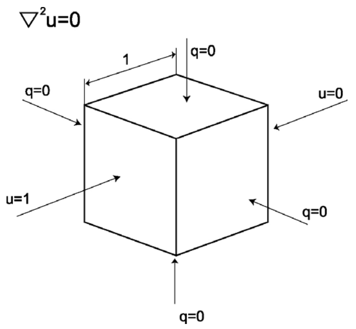

Figure 1: Model "cube".

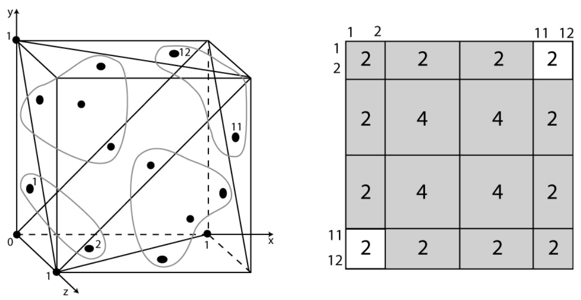

Figure 2: A collection of clusters for a test problem and the corresponding H-matrix.

Full-matrix format while the block B_{RK} is an Rk-block. There are several vectors storing the auxiliary information about the model. For example the vector

$$x_s = [(0, 0.33, 0.33); (0.33, 0, 0.33)], \tag{11}$$

consists of the 3D coordinates of the source points corresponding to the first and second row in the super-matrix A_H, i.e. row number 1 corresponds to the source point with 3D coordinates $(x, y, z) = (0, 0.33, 0.33)$ whereas the row number 2 corresponds to $(0.33, 0, 0.33)$; three vectors x_1, x_2 and x_3 that consist of the 3D coordinates of the vertices of triangular field elements corresponding to each column in the super-matrix A_H (see Table 3.1). In Table 3.1 the elements el_1, el_2, el_{11} and el_{12} correspond to the elements indicated in Figure 2 (the id of the field element coincide with the id of the DOF). The bc vector storing boundary conditions looks as follows:

$$bc = [0, 1, 0, 1], \tag{12}$$

i.e. the potential is known for the first element corresponding to column number 1. The vector of known potentials looks as follows:

$$u = [1, und, 0, und], \tag{13}$$

i.e. known potential for the first element is equal to 1, however, the potential for the element number 2 is not known and therefore represented by *und* which means undefined value. The vector of known normal fluxes stores the following data:

$$q = [und, 0, und, 0], \tag{14}$$

i.e. known normal fluxes for second and twelfth elements are both equal to 0. Having vectors (11–14) in mind and corresponding to eq. (7), the entries of block B_F and B_{RK} have to be computed as follows:

$$B_F = \begin{bmatrix} 0.1915 & -0.1148 \\ 0.0768 & 0.5 \end{bmatrix}, \quad B_{RK} = \begin{bmatrix} 0.0346 & -0.0324 \\ 0.0385 & -0.0304 \end{bmatrix} \tag{15}$$

Once the B_{RK} block is computed, it can be approximated within two iterations following the algorithm of fully pivoted ACA presented in [1]. As it has been mentioned in eq. (6), the Rk-blocks are presented in factorised form using the vectors V and U. At the end of first iteration, they look as follows:

$$V = [0.0346\ 0.0385], \quad U = [1\ -0.7901] \tag{16}$$

The next iteration leads to the final approximation as it reaches the maximal rank of block B_{RK}:

$$V = [0.0346\ 0.0385\ -0.00503\ 0], \quad U = [1\ -0.7901\ 0\ 1] \tag{17}$$

As a result of second iteration, the matrix approximated without any error, i.e. the initial block and the block after approximation are absolutely the same. It is possible to check this fact using the expression (6). In this particular example, the rank of the approximated block is equal to rank of the initial matrix. However, when the rank is fairly high the difference may reach hundreds of times economizing the memory consumption and reducing the complexity of matrix-vector multiplications.

4 Different ways to implement an ACA-based solver for existing BEM formulations

There are two ways to implement ACA-based solver for the solution of LSE coming from BEM method:

(i) As a “grey-box” solver, in which the solver takes as an input: the coefficient matrix A, the right hand side of the equation $Ax = b$ and the information about geometry, which could be provided by an existing BEM code;

(ii) or integrated within the BEM strategy where the assembly scheme is embedded within the BEM code. In this case A matrix has to be assembled according to its hierarchical representation during the evaluation of H and G matrix coefficients in the BEM formulation.

The first approach needs to store two copies of the matrix A at the same time, i.e. one original coefficient matrix A and its hierarchical representation A_H, i.e. the number of entries N_{ST} can be estimated as follows:

$$N_{ST} = NM + \sum_{i=1}^{N_{RK}} k_i(n_i + m_i) + \sum_{i=1}^{N_F} n_i m_i \tag{18}$$

where $N = rows$ and $M = cols$ are the number of rows and columns in the original matrix A, N_{RK} and N_F are the number of Rk and Full blocks appropriately, n_i and m_i are appropriately the number of rows and columns in block i of the coefficient H-matrix A_H. The term *NM* from the eq. (18) is reduced with the help of hierarchical matrix in conjunction with ACA approach. As a practical rule of thumbs for current PC systems (in this case Pentium iii, 1 GHz, 2 Gb RAM), if the model under analysis is fairly complex, the only way to solve it within reasonable memory space is to read the corresponding entries from the initial matrix A block by block. In this case, the memory capacity can be reduced up to:

$$N_{ST} = \sum_{i=1}^{N_{RK}} k_i(n_i + m_i) + \sum_{i=1}^{N_F} n_i m_i \tag{19}$$

However, the CPU time T_{CPU} increases according to:

$$T_{CPU} = N_B(T_S + T_R + T_P) \tag{20}$$

where N_B is the number of blocks to be read, T_S is the time required to find the data in a file, T_R is a time needed to access the data and T_P is a CPU time to process the entry. If the number of blocks N_B is too high, the total CPU time T_{CPU} increases dramatically due to the fact that the time to find a corresponding entry in the original matrix increases non-linearly even for advanced searching techniques. On the other hand, if N_B is small, the CPU time is reasonable but the algorithm risks to go beyond the memory limits as the storage requirement for each block needed to be read is $O(N^2)$. As an example, if the whole previously assembled matrix has to be read, the model consisting of nearly 12 000 degrees of freedom needs about 1.3 Gb of RAM according to (18) and about 1 hour to be assembled. However, if it is read block by block, the total memory capacity needed for storing the A_H matrix is just about 200 Mb and takes about 3 hours to be assembled.

Another way to overcome the problem with the memory requirement mentioned above is to use the ACA within the computation of the coefficients of H-matrix A_H. In this case, the method does not need to store any additional information and can compute the coefficients on the fly. The storage requirement for this approach is as shown in the eq. (19). The total CPU time can be evaluated as follows:

$$T_{CPU} = N_B(T_C + T_P) \tag{21}$$

where T_C is a CPU time to compute the entries which is much less than the time to find the needed entry in a file when the large scale models are involved. For example, the model with nearly 12 000 degrees of freedom took about 3 hours to be assembled (on a Pentium iii, 1 Ghz, 2 Gb RAM) reading the coefficients block by block, while for the case when the entries are computed on the fly, it takes just about 20 minutes on the same PC. The memory requirement is same as for the reading the matrix block by block and for this particular example is about 200 Mb.

The partially-pivoted ACA allows to generate the Rk-blocks on the fly without storing the intermediate block [1]. Moreover, it does not need to compute all the entries. Although the partially pivoted ACA is slightly more difficult to implement in comparison with the fully pivoted version, it works faster as it does not need to compute all entries of the H-matrix A_H beforehand. Instead, it computes directly the vectors V and U without building the whole intermediate block. As a conclusion, the last method is the most stable and robust one among the analysed in this section, although less straightforward to adapt in existing BEM codes.

5 Application to low frequency electric fields induced in human tissues

The developed application has been applied to the problem involving the Laplace equation with high number of DOF in three dimensions. As an illustrative example, this section presents the calculation of the electrostatic field surrounding a grounded anatomic model of the human body (see Figure 3). In particular, current densities and electric fields induced in realistic models of the human body standing in power control room. The model shown in Figure 4 with single domain is meshed with constant triangular elements yielding nearly 50 000 and 80 000 degrees of freedom. The outcome of simulation is represented in Figure 5 which shows the current density distribution over the human body. The results obtained compare very well with similar models solved in previous work [5]. Table 1 represents

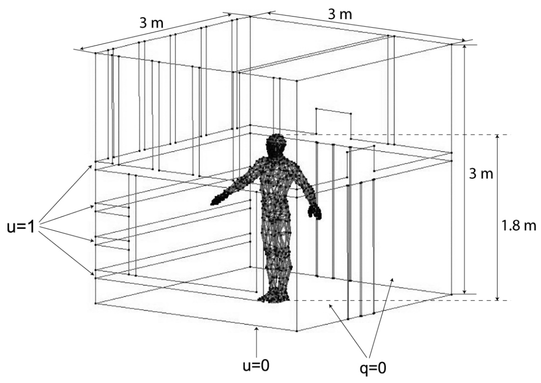

Figure 3: The BEM model under analysis: the potential $u = 1$ on each cable, the bottom has $u = 0$ as it is grounded and the normal flux $q = 0$ everywhere else as the model is isolated.

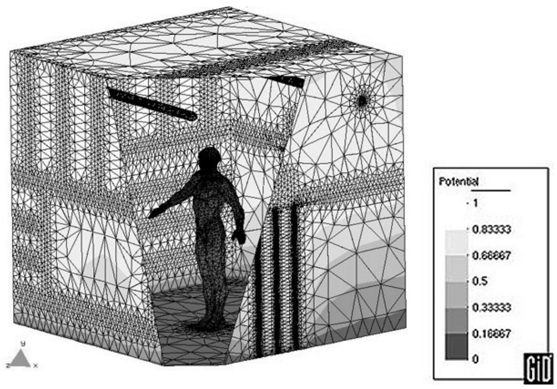

Figure 4: BEM model under analysis.

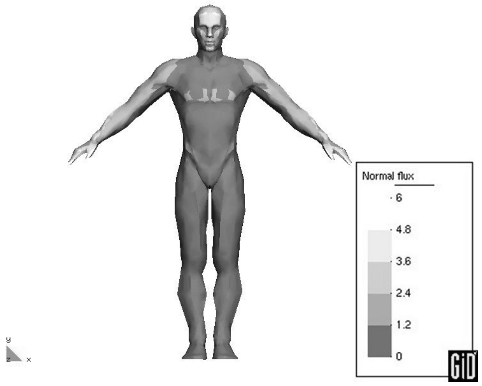

Figure 5: Normal flux distribution across the human body.

Table 1: Numerical results (Processor: Pentium iii, 1 GHz, 2 Gb RAM).

N	N_{Rk}	N_F	N_{it}	Eps	ε_{ACA}	Mem	T_p	T_s	T
47466	13124	7094	21	10^{-8}	10^{-4}	763	133	54	187
82946	91142	41012	17	10^{-8}	0.005	424	170	34	204

the numerical properties of the ACA-based solver. The following labels are used: N is the number of DOF, N_{Rk} is a number of low-rank blocks in the coefficient matrix, N_F is a number of blocks without approximation, N_{it} is a number of iterations, *Eps* in an error of iterative solver (GMRES), ε_{ACA} is an error of ACA approximation, *Mem* is a memory capacity in megabytes, T_p is a duration of the preconditioner generation in seconds, T_s is a solution time in seconds and T is a total time in seconds. The working space to allocate all necessary components can be estimated as follows: $(N*100*8)/(1024*1024)$ Mb for temporary array, the preconditioner needs about 10% of matrix A_H space, i.e. $N * 100 * 8/10^6 + 0.1 * Mem$ Mb. So, for the biggest example, the working space is about 105 Mb.

6 Conclusions

An ACA-based solver has been implemented in order to solve linear systems of equations coming from the collocation BEM. The main advantage of the approach explained in this paper over other approaches is that it assembles both H and G BEM matrices within the same coefficient H-matrix A_H. An illustrative example explaining the assembly scheme using a very simple model of the 3D unitary cube demonstrates the approach in detail. Two different ways of using the ACA-solver within existing BEM codes have been presented. The advantages and disadvantages of both of them as well as the appropriate estimates are given in the paper. An 3D example of electrostatic model with more complex geometry has been successfully solved using the developed approach.

References

[1] M. Bebendorf and R. Grzibovski. Accelerating galerkin bem for linear elasticity using adaptive cross approximation. May 2006.

[2] S. Borm, L. Grasedyck, and W. Hackbusch. Hierarchical matrices. April 2005.

[3] C. Brebbia, J. Telles, and L. Wrobel. *Boundary Elements Techniques*. Splinger-Verlag, Berlin, Heidelberg, New York and Tokio, 1984.

[4] J. E. Flaherty T. James D., K. D. Devine. Partitioning and dynamic load balancing for the numerical solution of partial differential equations. *Num. Sol. of Part. Diff. Equations on Parallel Computers*, August 2005.

[5] A. Peratta, C. Gonzalez, and D. Poljak. Current density induced in the human body due to power distributions lines using the boundary element method. *Journal of Communications Software and Systems*, 3:11–16, 2007.