Copyright © 2015 Tech Science Press CMES, vol.107, no.4, pp.321-343, 2015

An Advanced ACA/BEM for Solving 2D Large-Scale Elastic Problems with Multi-Connected Domains

T. Gortsas[1], S.V. Tsinopoulos[2], D. Polyzos[1,3]

Abstract: An advanced Boundary Element method (BEM) accelerated via Adaptive Cross Approximation (ACA) and Hierarchical Matrices (HM) techniques is presented for the solution of large-scale elastostatic problems with multi-connected domains like in fiber reinforced composite materials. Although the proposed ACA/ BEM is demonstrated for two-dimensional (2D) problems, it is quite general and it can be used for 3D problems. Different forms of ACA technique are employed for exploring their efficiency when they combined with a BEM code. More precisely, the fully and partially pivoted ACA with and without recompression are utilized, while the solution of the final linear system of equations is accelerated via an iterative GMRES solver. The proposed methodology is demonstrated with the solution of large scale, plane strain elastic problems dealing with the bending of unidirectional fiber composite plates with large numbers of periodically or randomly distributed cylindrical elastic fibers embedded in a matrix medium.

Keywords: Boundary Element Method; Hierarchical Matrices; Adaptive Cross Approximation; ACA; Large Scale Elastic Problems; Composite Materials.

1 Introduction

The Boundary Element Method (BEM) is a well-known and robust numerical tool successfully used for the solution of elastostatic and elastodynamic problems [Beskos (1997); Bonnet (1999); Aliabadi (2002); Beer, Smith, and Duenser (2008); Manolis and Polyzos (2009)]. Two remarkable advantages it offers as compared to other numerical methods is the reduction of the dimensionality of the problem by one and its high solution accuracy. Despite the advantages the brutal application of BEM to large-scale problems suffers from very time consuming computations and

[1] Department of Mechanical Engineering & Aeronautics, University of Patras, Patras, Greece
[2] Department of Mechanical Engineering, Technical Research Institute, Patras, Greece
[3] Department of Mechanical Engineering and Aeronautics, University of Patras, 26504, Patras, Greece, +0030 2610969442, Email: polyzos@mech.upatras.gr

high demands for computer memory capacity. Both problems come from the generation of the non-symmetric coefficient matrix and the solution of the final system of algebraic equations. More precisely, the fully populated matrices produced by BE-M require $O(N^2)$ operations for its buildup and $O(N^3)$ operations for the solution of the final matrix system through Gaussian elimination or typical LU-decomposition solvers. The use of iterative solvers decreases the operation requirements from $O(N^3)$ to $O(M \times N^2)$, with M being the number of iterations, but still remains inefficient for large scale problems. To the same conclusion we reach when parallel computing methods are exploited for the solution of the problem. In order to solve the aforementioned drawbacks of BEM and extend its application to real industrial problems, many techniques have been proposed so far in the literature. One can mention the Fast Multipole Boundary Element Method (FM/BEM) [Liu (2009)], the Fast Wavelet/BEM [Bucher, Wrobel, Mansur, and Magluta (2003); Ntalaperas, Tsinopoulos, and Polyzos (2010)], the Precorrected Fast Fourier Transform (FFT) Accelerated BEM [Xiao, Ye, Cai, and Zhang (2012)] and the Hierarchical Matrices-Adaptive Cross Approximation (ACA)/BEM [Rjasanow and Steinbach (2007)].

The Fast Wavelet/BEM relies on the fast wavelet transform so that to compress the dense and fully populated final matrices of BEM and change them to semi-banded and sparse ones. Although its use can be considered as a black box in a BEM code, appears the disadvantage of requiring the knowledge of the final system of algebraic equations the construction of which is in general computational expensive. Details one can find in the representative works of [Bucher, Wrobel, Mansur, and Magluta (2003); Ravnik, Škerget, and Zunic (2009); Ebrahimnejad and Attarnejad (2010); Wang, Ma, Meng, and Huang (2013)].

A very efficient method that accelerates drastically the solution process of a BEM code is that of the FM/BEM. The FM/BEM accelerates the computation of matrix $[A]$ of the final system of algebraic equations $[A]\{x\} = \{b\}$ to $O(N)$ operations and reduces the memory requirements to $O(N)$ as well. This is accomplished, first, by analytically expanding the fundamental solutions of the differential operator of the problem around the centers of a hierarchical cell structure through fast multipole expansions, second, by performing analytical integrations utilizing constant elements and third, by using an iterative solver (e.g. GMRES) instead of a direct one. The main disadvantage of FM/BEM is its dependence on the fundamental solution of the problem and the use of constant elements in order to perform analytical integrations on flight without storage memory requirements. Representative works are those of [Bapat and Liu (2010); Frangi and Bonnet (2010); Wang, Miao, and Zheng (2010); Wang and Yao (2011); Yusa and Yoshimura (2013)], while the application of FM/BEM to the solution of composite material problems is demonstrated in the works of [Hu, Wang, Tan, Yao, and Yuan (2000); Kong, Yao, and Zheng (20002);

Yao, Kong, Wang, and Wang (2004); Liu, Nishimura, Otani, Takahashi, Chen, and Munakata (2005); Lei, Yao, Wang, and Wang (2006); Wang and Yao (2008)].

The precorrected FFT/BEM is a method that accelerates less than the FM/BEM but utilizes the same idea of cells as the FM/BEM does. The idea is to project the boundary fields to a uniform grid of points, then to compute the grid fields by using FFT and finally to interpolate the grid fields to boundary nodes. The nearby interactions are evaluated directly as in conventional BEM. The result is a reduction of the total operations to $O(N \log N)$. Representative works are those of [Phillips and White (1997); Xiao, Ye, Cai, and Zhang (2012)].

An alternative approach to accelerate BEM is the recently proposed adaptive cross approximation algorithm (ACA) along with hierarchical matrices [Bebendorf (2000); Bebendorf and Rjasanow (2003); Rjasanow and Steinbach (2007); Borm, Grasedyck, and Hackbusch (2003, 2003a); Bebendorf (2008); Brancati, Aliabadi, and Benedetti (2009)]. The acceleration here is achieved because only a small number of elements of the collocation matrix $[A]$ are calculated, while the rest ones are approximated via the already calculated values. In a BEM collocation matrix, this is possible and effective due to the nature of the fundamental solutions, which are functions of the distance between the source and field points. According to A-CA/BEM, the matrix $[A]$ is organized into a hierarchical structure of blocks based on the problem's geometry. Applying a geometrical criterion the blocks are characterized either as non-admissible, where the ACA algorithm is not efficient and thus, the conventional BEM is used or admissible, where ACA is effective and is used to approximate them by only calculating a small number of their rows and columns. Each admissible block is represented in a low rank matrix format via the product of two matrices formed by the previously calculated rows and columns, respectively. This low rank format, in conjunction with a usage of an iterative solver, results both significant reductions in memory requirements and a CPU time due to the acceleration of the matrix vector multiplication. The ACA/BEM has already been successfully used for solving static and dynamic linear elastic problems, as it is explained in [Bebendorf and Grzhibovskis (2006); Benedetti, Alliabadi, and Davi (2008); Benedetti, Milazzo, and Alliabadi (2009); Zechner (2012)] for elastostatics and [Benedetti and Alliabadi (2009); Messner and Schanz (2010); Millazzo, Benedetti, and Alliabadi (2012)] for elastodynamics.

Although the FMM/BEM seems to be faster than ACA/BEM, it appears some disadvantages as it is compared to the ACA/BEM, namely, it requires the knowledge of the multipole expansions of the fundamental solution of the problem and significant and complex modifications in a conventional BEM code in order to be implemented. On the other hand, ACA/BEM is a black box algorithm applied upon the collocation matrix $[A]$ and thus its implementation is the same regardless of the differential operator of the problem. The main structure of a conventional BEM code, including the significant integration library, remains the same and the only changes needed in the code are restricted to the assembly procedures. A comparison between FM/BEM and ACA/BEM can be found in the work of [Brunner, Junge, Rapp, Bebendorf, and Gaul (2010)].

In the present work, a two dimensional hierarchical ACA/BEM formulation is proposed for the solution of elastostatic problems dealing with multi-connected domains like those appearing in the perpendicular to the fibers cross-section of a fiber reinforced composite plate. Although many BEM formulations accelerated by hierarchical matrices and ACA methodologies have appeared in the literature, to our best knowledge, an ACA/BEM methodology for multi-connected domains appears for the first time. According to the proposed formulation the matrices $\tilde{G}$ and $\tilde{H}$ of each region, formed from the integrals of the elastostatic fundamental displacement and traction, respectively, are approximated by hierarchical matrices, constructed by means of the ACA algorithm. Thus, the global collocation matrix $\tilde{A}$ is not constructed explicitly, saving significant amount of memory. The matrix approximation is accomplished by the implementation of four ACA formulations, the fully and partially pivoted ACA with and without recompression in order to study their efficiency. For the solution of the final linear system of equations an iterative solver (GMRES) is employed. The boundary conditions of the problem as well as the interface continuity conditions between the inclusions and the matrix are applied during the matrix vector multiplication procedure. From the approximated operators $\tilde{G}$ and $\tilde{H}$ a block diagonal preconditioner is constructed for the faster convergence of the solution. The efficiency of the proposed formulation is demonstrated by solving a large-scale elastostatic problem dealing with the bending of a fiber composite plate. For a given accuracy of approximation the CPU time as well as the memory demands are compared with the corresponding ones of the classical BEM. The largest problem solved corresponds to 1.2 million degrees of freedom utilizing a desktop pc with 64 GB RAM.

2 Conventional boundary element method

As it has been already mentioned, the boundary element methodology proposed in the present work concerns multi-connected domains and it is demonstrated through the problem described below. Consider a $2D$ rectangular fibrous composite plate of length L and width D, as shown in Fig. 1. The plate occupies a region Ω_0 of boundary S_0, is made of a matrix material with Young's modulus E_M and Poisson's ratio v_M and is reinforced with N_f identical circular fibers of radius a with material properties E_F and v_F, respectively. The volume fraction of the fibers with respect to the plate is V_f. The fibers are either randomly distributed (Fig. 1a) or periodically arranged in a square pattern (Fig. 1b). Each fiber occupies a region Ω_i of boundary S_i, where $i = 1, N_f$. The plate is fixed at its one end and is subjected to a bending load P applied at its free end.

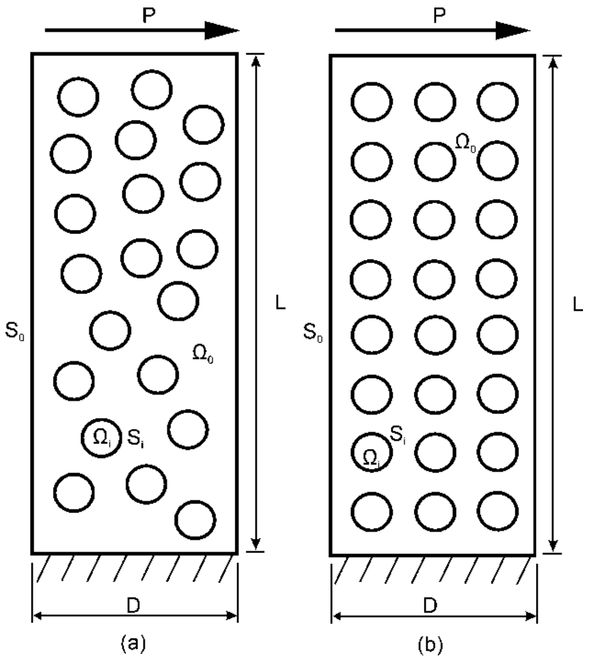


Figure 1: 2D rectangular fiber reinforced composite plates with (a) randomly distributed and (b) periodically arranged fibers.

In the present section, the just described 2D elastostatic problem is solved numerically using first the conventional boundary element method for various numbers of fibers N_f.

The problem solution can be obtained by solving a combined system of boundary integral equations written for the matrix and each of the N_f fibers. The boundary integral equations for the matrix and for the i^{th} fiber are written as:

$$\tilde{\mathbf{c}}(\mathbf{x}) \cdot \mathbf{u}(\mathbf{x}) + \int_S \tilde{\mathbf{t}}^*(\mathbf{x},\mathbf{y}) \cdot \mathbf{u}(\mathbf{y})\, dS_{\mathbf{y}} = \int_S \tilde{\mathbf{u}}^*(\mathbf{x},\mathbf{y}) \cdot \mathbf{t}(\mathbf{y})\, dS_{\mathbf{y}} \tag{1}$$

$$\tilde{\mathbf{c}}(\mathbf{x}) \cdot \mathbf{u}(\mathbf{x}) + \int_{S_i} \tilde{\mathbf{t}}^*(\mathbf{x},\mathbf{y}) \cdot \mathbf{u}(\mathbf{y})\, dS_{\mathbf{y}} = \int_{S_i} \tilde{\mathbf{u}}^*(\mathbf{x},\mathbf{y}) \cdot \mathbf{t}(\mathbf{y})\, dS_{\mathbf{y}} \tag{2}$$

where $S = S_0 + \sum_{i=1}^{N_f} S_i$, $\mathbf{x}$ and $\mathbf{y}$ are points on the boundary, $\mathbf{u}$ and $\mathbf{t}$ are the displacement and traction vectors, respectively, $\tilde{\mathbf{c}}$ is a free term tensor depended on local geometry at point $\mathbf{x}$ (for a smooth boundary $\tilde{\mathbf{c}} = 1/2\tilde{\mathbf{I}}$, with $\tilde{\mathbf{I}}$ being the unity tensor) and $\tilde{\mathbf{u}}^*(\mathbf{x},\mathbf{y})$ and $\tilde{\mathbf{t}}^*(\mathbf{x},\mathbf{y})$ are the 2D free space elastostatic fundamental displacement and traction, respectively, given as [Polyzos, Tsinopoulos and Beskos (1998)].

$$u_{kj}^*(\mathbf{x},\mathbf{y}) = \frac{(1+v)}{4\pi E(1-v)}\left((3-4v)\,\delta_{kj}\ln\left(\frac{1}{r}\right)+r_{,k}r_{,j}\right)\text{with}k,j=1,2 \tag{3}$$

$$t_{kj}^*(\mathbf{x},\mathbf{y}) = \frac{1}{4\pi(1-v)\,r}\left[\left((1-2v)\,\delta_{kj}+2r_{,k}r_{,j}\right)\frac{\partial r}{\partial n}+(1-2v)\left(r_{,k}n_j-r_{,j}n_k\right)\right] \tag{4}$$

where r is the distance between the points $\mathbf{x}$ and $\mathbf{y}$, n_j the unit vector normal to the boundary at point $\mathbf{y}$, $r_{,j}$ denotes spatial derivatives of r and $\partial r/\partial n$ is the directional derivative with respect to the normal vector at $\mathbf{y}$.

According to conventional BEM formulation, the boundary S is discretized into E three-noded quadratic or two-noded linear isoparametric line boundary elements. For a node k, belonging to the boundary S the discretized integral equation (1) takes the form

$$\begin{aligned}&\frac{1}{2}\mathbf{u}\left(\mathbf{x}^k\right)+\sum_{e=1}^{E}\sum_{a=1}^{A(e)}\int_{-1}^{1}\tilde{\mathbf{t}}^*\left(\mathbf{x}^k,\mathbf{y}^e(\xi)\right)N^a(\xi)\,J^e(\xi)\,\mathrm{d}\xi\cdot\mathbf{u}_a^e\\&=\sum_{e=1}^{E}\sum_{a=1}^{A(e)}\int_{-1}^{1}\tilde{\mathbf{u}}^*\left(\mathbf{x}^k,\mathbf{y}^e(\xi)\right)N^a(\xi)\,J^e(\xi)\,\mathrm{d}\xi\cdot\mathbf{t}_a^e\end{aligned} \tag{5}$$

where $A(e)$ is the number of nodes of the element e ($A=3$ for a quadratic and $A=2$ for a linear element), N^a stand for the element shape functions, J^e the Jacobian of the transformation from the global (X_1, X_2) to the local co-ordinate system ξ and $\mathbf{u}_a^e$ and $\mathbf{t}_a^e$ represent the nodal values of the displacement and traction vectors, respectively. Adopting a global numbering for the nodes, each pair (e,a) is associated to a number β and the equation (5) is written as

$$\frac{1}{2}\mathbf{u}^k+\sum_{\beta=1}^{L}\tilde{\mathbf{h}}_\beta^k\cdot\mathbf{u}^\beta=\sum_{\beta=1}^{L}\tilde{\mathbf{g}}_\beta^k\cdot\mathbf{t}^\beta \tag{6}$$

where L is the total number of nodes that S has been discretized into and

$$\tilde{\mathbf{h}}_\beta^k=\left.\int_{-1}^{1}\tilde{\mathbf{t}}^*\left(\mathbf{x}^k,\mathbf{y}^e(\xi)\right)N^a(\xi)\,J^e(\xi)\,\mathrm{d}\xi\right|_{(e,a)\to\beta} \tag{7}$$

$$\tilde{\mathbf{g}}_\beta^k=\left.\int_{-1}^{1}\tilde{\mathbf{u}}^*\left(\mathbf{x}^k,\mathbf{y}^e(\xi)\right)N^a(\xi)\,J^e(\xi)\,\mathrm{d}\xi\right|_{(e,a)\to\beta} \tag{8}$$

Collocating Eq (6) at all nodes L, the following linear system of algebraic equations is obtained

$$\begin{bmatrix} {}^{M}\tilde{\mathbf{H}}_0^0 & {}^{M}\tilde{\mathbf{H}}_1^0 & \dots & {}^{M}\tilde{\mathbf{H}}_{N_f}^0 \\ {}^{M}\tilde{\mathbf{H}}_0^1 & {}^{M}\tilde{\mathbf{H}}_1^1 & \dots & {}^{M}\tilde{\mathbf{H}}_{N_f}^1 \\ \vdots & \vdots & \vdots & \vdots \\ {}^{M}\tilde{\mathbf{H}}_0^{N_f} & {}^{M}\tilde{\mathbf{H}}_1^{N_f} & \dots & {}^{M}\tilde{\mathbf{H}}_{N_f}^{N_f} \end{bmatrix} \cdot \begin{Bmatrix} \mathbf{u}^0 \\ \mathbf{u}^1 \\ \vdots \\ \mathbf{u}^{N_f} \end{Bmatrix} = \begin{bmatrix} {}^{M}\tilde{\mathbf{G}}_0^0 & {}^{M}\tilde{\mathbf{G}}_1^0 & \dots & {}^{M}\tilde{\mathbf{G}}_{N_f}^0 \\ {}^{M}\tilde{\mathbf{G}}_0^1 & {}^{M}\tilde{\mathbf{G}}_1^1 & \dots & {}^{M}\tilde{\mathbf{G}}_{N_f}^1 \\ \vdots & \vdots & \vdots & \vdots \\ {}^{M}\tilde{\mathbf{G}}_0^{N_f} & {}^{M}\tilde{\mathbf{G}}_1^{N_f} & \dots & {}^{M}\tilde{\mathbf{G}}_{N_f}^{N_f} \end{bmatrix} \cdot \begin{Bmatrix} \mathbf{t}^0 \\ \mathbf{t}^1 \\ \vdots \\ \mathbf{t}^{N_f} \end{Bmatrix} \tag{9}$$

or

$${}^{M}\tilde{\mathbf{H}} \cdot \mathbf{u}^M = {}^{M}\tilde{\mathbf{G}} \cdot \mathbf{t}^M \tag{10}$$

where the superscript M indicates matrix medium, ${}^{M}\tilde{\mathbf{H}}_{\gamma}^{m}$ and ${}^{M}\tilde{\mathbf{G}}_{\gamma}^{m}$ are matrices formed by the integrals (7) and (8), respectively, with the term 1/2 added at the diagonal elements of ${}^{M}\tilde{\mathbf{H}}_{m}^{m}$, while the indices m and γ take values $0, 1, 2, \dots N_f$, corresponding to the total number of nodes $L_0, L_1, \dots, L_{Nf}$ ($L = L_0 + \sum_{i=1}^{N_f} L_i$) that the boundaries S_0, $S_1, \dots, S_{Nf}$ have been discretized into, respectively. The vectors $\mathbf{u}^m$ and $\mathbf{t}^m$ contain the nodal displacement and traction vectors of the nodes L_m, respectively.

Similarly, collocating the discretized version of the integral equation (2) for all the nodes of the fibers, the following system of algebraic equations can be obtained

$${}^{F}\tilde{\mathbf{H}} \cdot \mathbf{u}^F = {}^{F}\tilde{\mathbf{G}} \cdot \mathbf{t}^F \tag{11}$$

Taking into account the continuity conditions at the matrix-fiber interfaces ($\mathbf{u}^M = \mathbf{u}^F, \mathbf{t}^M = -\mathbf{t}^F$), the system of equations (10) and (11) contains all the unknown displacement and traction interfacial nodal values, while the half nodal values of $\mathbf{u}^0$ and $\mathbf{t}^0$ are known through the boundary conditions at the boundary S_0 and the other half ones unknown.

When $\beta \neq k$, integrals (7) and (8) are non-singular and can be easily computed numerically by Gauss quadrature. In case where $\beta = k$, the integrals (7) and (8) become singular with singularities $O(\ln r)$ and $O(1/r)$, respectively. Singular integrals are evaluated with high accuracy via direct integration techniques [Frangi and Guiggiani (2000)].

Combing equations (10) and (11) and rearranging according to the boundary conditions at the boundary S_0, one obtains a linear system of algebraic equations of the form

$$\tilde{\mathbf{A}} \cdot \mathbf{x} = \mathbf{b} \tag{12}$$

where the vectors $\mathbf{x}$ and $\mathbf{b}$ contain all the unknown and known nodal components of the boundary fields, respectively, while the solution of system (12) is usually accomplished through a typical LU decomposition algorithm.

3 Hierarchical and ACA accelerated BEM

In conventional BEM, the matrix $\tilde{\mathbf{A}}$ of the system (12) is full populated requiring $O(N^2)$ operations for its solution with N being the DOFs of the problem. Taking also into account that the condition number of the system becomes worse as N increases, it is apparent that conventional BEM is not able to treat realistic problems with hundreds of thousands DOFs. In order to overcome that difficulty and solve the problem described in the previous section for a large number of fibers, a hierarchical ACA accelerated BEM is proposed. Furthermore, a significant reduction of the problem solution time is also accomplished with the aid of a GMRES iterative solver.

The departure point of the proposed method is that both matrices $\tilde{\mathbf{H}}$ and $\tilde{\mathbf{G}}$, appearing in equations (10) and (11), are represented hierarchically using a block tree structure. By means of simple geometric considerations the blocks corresponding to large distances r are characterized as far-field blocks (or admissible) and compressed through low rank matrices found by ACA algorithm, while the rest blocks of the tree, which are dominated by the singular behavior of the kernels (3) and (4), are characterized as near field blocks (or non-admissible) and are calculated explicitly as in conventional BEM.

The ACA/BEM methodology proposed here can be analyzed in steps as follows:

Step 1: The nodes of each region Ω_i are partitioned in a tree structure containing clusters that include only closely located nodes. Starting from a cluster containing all the nodes of the region, two sub-clusters are created. This procedure is repeated recursively for the created sub-clusters and is ended when the number of nodes inside a cluster is equal or less than a predefined number L_t, which is named the cardinality of the tree. The nodes belonging to the same cluster are renumbered so that the corresponding integrals of $\tilde{\mathbf{H}}$ and $\tilde{\mathbf{G}}$ matrices to be placed at neighbor rows and columns. The splitting of the clusters can be accomplished with the aid of various bisection techniques like the principal component analysis [Bebendorf (2008)] adopted in the present work. According to that technique, the centroid of each already created cluster of nodes is calculated by using the following equation:

$$x_{c_i} = \frac{x_i^1 + x_i^2 + \ldots + x_i^{L_c}}{L_c} \tag{13}$$

where, L_c is the number of nodes of the cluster and $x_i^1, x_i^2, \ldots x_i^{L_c}$ denote the coordinates of the cluster nodes, and $i = 1,2$. Then, the corresponding cluster's data matrix of dimensions $d \times L_c$ is formulated in the following way:

$$\mathbf{D} = \begin{bmatrix} x_1^1 - x_{c_1} & x_1^2 - x_{c_1} & \ldots & x_1^{L_c} - x_{c_1} \\ x_2^1 - x_{c_2} & x_2^1 - x_{c_2} & \ldots & x_2^{L_c} - x_{c_2} \end{bmatrix} \tag{14}$$

where d is the number of spatial dimensions ($d = 2$ for a 2D problem).

Applying the Singular Value Decomposition (SVD) algorithm for the matrix (14), one obtains $\mathbf{D} = \mathbf{U} \cdot \mathbf{S} \cdot \mathbf{V}^T$ with $\tilde{\mathbf{U}}, \tilde{\mathbf{S}}, \tilde{\mathbf{V}}$ being the matrices provided by SVD, $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ are orthogonal matrices and $\tilde{\mathbf{S}}$ is a diagonal matrix. It must be noted that the matrix $\tilde{\mathbf{V}}$ does not need to be calculated explicitly. We define as principal direction the vector represented by the first column of matrix $\tilde{\mathbf{U}}$ and symbolized as $\mathbf{w}$. Considering the centroid (13) as the base of any position vector, the nodes of the cluster are separated by considering the positive or negative value of their projection along the aforementioned principal vector $\mathbf{w}$. If C denotes the parent cluster, the two sub-cluster C_1 and C_2 containing L_{c1} and L_{c2} nodes respectively are defined as

$$C_1 = \left\{ L_i \in L_c : \mathbf{w}^T (\mathbf{x} - \mathbf{x_c}) \geq 0 \right\}, \quad C_2 = C - C_1 \tag{15}$$

The first four levels of the just described procedure for the region Ω_0 is depicted in Fig. 2.

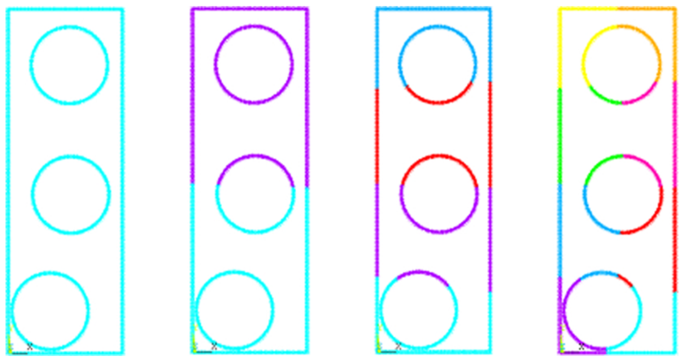


Figure 2: Clustering of initial domain Ω_0 to 2, 4 and 8 clusters.

Step 2: Matrices $\tilde{\mathbf{H}}$ and $\tilde{\mathbf{G}}$, appearing in (10) and (11), can be represented hierarchically by a block tree, which can be set up by using the previously described node clustering constructed cluster tree and a geometric admissibility criterion. Let's consider the matrix ${}^{M}\tilde{\mathbf{H}}_0^0$ in (9) corresponding to the region Ω_0, which contains L_0 nodes. Applying a geometric admissibility criterion the matrix ${}^{M}\tilde{\mathbf{H}}_0^0$ can be partitioned by the following procedure: (i) The root (level 0) of the cluster tree represents the total number of rows/columns. Thus, the combination of rows and columns at this level produces the entire matrix ${}^{M}\tilde{\mathbf{H}}_0^0$. By default, the admissible criterion for the entire matrix is false and the algorithm continues to the level 1. (ii) In level 1, both rows and columns, are partitioned into two parts, the entire matrix is divided into four blocks and the admissibility criterion is applied to each of them. In case where a block is admissible, it is characterized as a leaf and the recursive procedure is terminated. In case of a non-admissible block, step (ii) is repeated until all sub-blocks to be characterized either as non-admissible or as leaves. (iii) Finally, the last level non-admissible blocks in the block tree, are characterized as leaves.

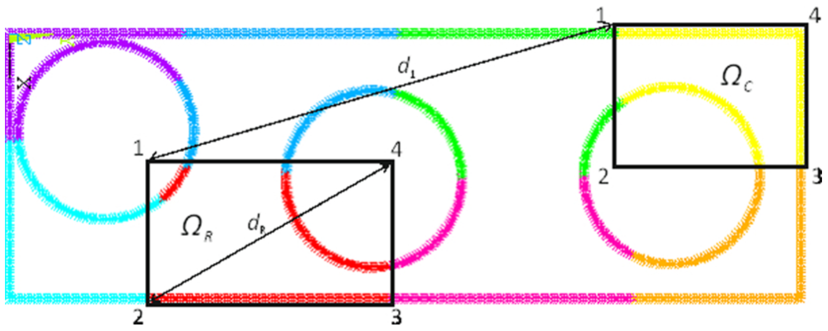


Figure 3: Boxes Ω_R and Ω_C that enclose the elements belonging to the clusters C_R and C_C, respectively, used in the admissibility criterion of Eq. 1.

The admissibility criterion adopted in the present work can be described as follows: consider a block of the entire matrix the rows and columns of which are represented by the clusters C_c and C_R, respectively. The elements associated with the nodes, contained in C_R and C_C, are enclosed in two boxes, Ω_R and Ω_C, respectively. Each box has its sides parallel to the Cartesian axes and is the smallest possible, with respect to the location of its nodes (Fig. 3). The corners are numbered anti-clockwise with the 1st and 3nd corner corresponding to the minimum and maximum coordinates of the box, respectively, as shown in Fig. 3. The block is admissible when the following geometric condition is true:

$$\min\left(d_R, d_C\right) \leq a \cdot dist\left(\Omega_R, \Omega_C\right) \tag{16}$$

where d_R, d_C are the diagonals of the boxes Ω_R and Ω_C, respectively, $dist\left(\Omega_R, \Omega_C\right)$ is the distance of the corresponding bounding boxes [Borm, Grasedyck, and Hackbusch (2003)] and a is a positive parameter. In the present work a has been chosen to be 1.2.

Step 3: The admissible blocks of the tree are approximated by low rank matrices with respect to a prescribed accuracy ε, by using the ACA algorithm. More precisely, considering an admissible block sub-matrix $\tilde{\mathbf{M}}$ of matrices $\tilde{\mathbf{H}}$ or $\tilde{\mathbf{G}}$, with dimensions $N \times L$ and a full rank $R = \min\{N, L\}$, the block $\tilde{\mathbf{M}}$ can be written as:

$$\tilde{\mathbf{M}} = \tilde{\mathbf{M}}^{(K)} + \tilde{\mathbf{R}}^{(K)} \tag{17}$$

where $\tilde{\mathbf{M}}^{(K)}$ is a K-rank approximation of $\tilde{\mathbf{M}}$, with K being less equal than R and $\mathbf{R}^{(K)}$ is the residual of the approximation.

The matrix $\tilde{\mathbf{M}}^{(K)}$ is constructed by two matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$, of dimensions $N \times K$ and $L \times K$, respectively, formed by K vectors $\mathbf{a}^i$ and $\mathbf{b}^i$ of dimensions N and L, respectively, as follows:

$$\tilde{\mathbf{M}}^{(K)} = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{B}}^T \tag{18}$$

where

$$\begin{aligned} \tilde{\mathbf{A}}_{N\times K} &= \left\{\mathbf{a}^1\mathbf{a}^2\ldots\mathbf{a}^K\right\} \\ \tilde{\mathbf{B}}_{L\times K} &= \left\{\mathbf{b}^1\mathbf{b}^2\ldots\mathbf{b}^K\right\} \end{aligned} \tag{19}$$

Both vectors $\mathbf{a}^i$ and $\mathbf{b}^i$ satisfy the relation

$$\left\|\mathbf{R}^{(K)}\right\|_F \le \varepsilon \left\|\tilde{\mathbf{M}}\right\|_F \tag{20}$$

where $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\left|a_{ij}\right|^2}$ denotes the Frobenius norm.

The memory requirements and matrix multiplication CPU cost of a low rank block are $O(K(N+L))$, while for the corresponding full rank representation are $O(N \cdot L)$. It is obvious that the low rank approximation is efficient when the condition $K(N+L) \ll N \cdot L$ holds.

The best low rank approximation with respect to the Frobenius norm, for a given accuracy ε, can be found by means of the SVD (Bebendorf, (2000)). More precisely, according to SVD the matrix $\tilde{\mathbf{M}}^{(K)}$ reads

$$\tilde{\mathbf{M}}^{(K)} = \tilde{\mathbf{U}} \cdot \tilde{\mathbf{S}}^{(K)} \cdot \tilde{\mathbf{V}}^T \tag{21}$$

where $\tilde{\mathbf{U}}, \tilde{\mathbf{S}}^{(K)}, \tilde{\mathbf{V}}$ are the matrices provided by SVD, $\tilde{\mathbf{S}}^{(K)}$ is a diagonal matrix of dimensions $N \times L$, represented as $\mathbf{S}^{(K)} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_K, 0, \ldots, 0)$, with $\sigma_i(i =$

$1,\ldots,K)$ being the singular values arranged in a descending order $(\sigma_1 > \sigma_2 > \ldots > \sigma_K)$. The rank K is defined by the condition

$$K = \min\left\{ K \in \mathbb{N} : \sqrt{\sum_{l=k+1}^{R} \sigma_l^2} < \varepsilon \sqrt{\sum_{l=1}^{R} \sigma_l^2} \right\} \tag{22}$$

and ignoring the singular values σ_i, with $i = K+1,\ldots,R$.

Although the best low rank approximation is provided by the SVD, its cubic CPU cost renders that technique prohibitive for real world applications. Thus, instead of the SVD the Adaptive Cross Approximation (ACA), which is a special low rank approximation technique, is employed. The main idea of ACA is to construct a representation of $\tilde{\mathbf{M}}^{(K)}$ (Eq. 18) by suitably choosing a small subset of the rows and columns of a matrix $\tilde{\mathbf{M}}$. Based on this idea two algorithms have been developed (Bebendorf (2000), Bebendorf and Rjasanow (2003)); the ACA with full pivoting, which is an $O(K^2 \cdot N \cdot L)$ algorithm and requires as starting point the calculation of the entire matrix $\tilde{\mathbf{M}}$, and the partially pivoted ACA, which is an $O\left(K^2\left(N+L\right)\right)$ algorithm and requires the calculation of only a small part of $\tilde{\mathbf{M}}$. Apparently the partially pivoted ACA is faster and consumes less memory than the fully pivoted one, but the approximation accuracy ε is not guaranteed because its stopping criterion is heuristic since the $\|\tilde{\mathbf{M}}\|_F$ in Eq. (19) cannot be calculated exactly. In the present work the above mentioned drawback is cancelled applying extra convergence checks according to the methodology proposed by Bebendorf (2008).

Both the fully and partially pivoted ACA algorithms, as they are implemented in the present work, are illustrated in the Appendix.

Step 4: Although the ACA provides an efficient approximation, it can be improved further by optimizing the compression of the admissible blocks through recompression of matrix sub-blocks. A number of techniques have been proposed to that purpose and significant reduction of storage memory can be accomplished (Grasedyck (2005)). In the present work the recompression is accomplished by means of the SVD. Each block is recompressed immediately after its approximation by ACA, thus reducing the memory requirements in the process of assembling of hierarchical matrices.

More specifically, in the SVD algorithm described in the previous step the matrices $\tilde{\mathbf{A}}_{N\times K}$ and $\tilde{\mathbf{B}}_{L\times K}$ are further decomposed according to QR decompositions $\tilde{\mathbf{A}}_{N\times K} = \tilde{\mathbf{Q}}^A_{N\times K} \cdot \tilde{\mathbf{R}}^A_{K\times K}$ and $\tilde{\mathbf{B}}_{L\times K} = \tilde{\mathbf{Q}}^B_{L\times K} \cdot \tilde{\mathbf{R}}^B_{K\times K}$, respectively. Next, the SVD of the matrix $\tilde{\mathbf{R}}^A \left(\tilde{\mathbf{R}}^B\right)^T = \tilde{\mathbf{U}} \cdot \tilde{\mathbf{S}} \cdot \tilde{\mathbf{V}}^T$ is computed, where $\tilde{\mathbf{U}}, \tilde{\mathbf{S}}, \tilde{\mathbf{V}}$ are $K \times K$ matrices and the optimum rank k_{SVD} $(\leq k)$ is found so that the condition (22) to be fulfilled for prescribed accuracy ε. Finally, the optimum compressed matrices $\tilde{\mathbf{A}}_{N\times K}$

and $\tilde{\mathbf{B}}_{N\times K}$ are given by the relations $\tilde{\mathbf{A}}_{N\times K_{\text{opt}}} = \tilde{\mathbf{Q}}^{A}_{N\times K_{\text{opt}}} \cdot \tilde{\mathbf{U}}_{K_{\text{opt}}\times K_{\text{opt}}} \cdot \tilde{\mathbf{S}}_{K_{\text{opt}}\times K_{\text{opt}}}$ and $\tilde{\mathbf{B}}_{L\times K_{\text{opt}}} = \tilde{\mathbf{Q}}^{B}_{L\times K_{\text{opt}}} \cdot \tilde{\mathbf{V}}_{K_{\text{opt}}\times K_{\text{opt}}}$, respectively.

The non-admissible leaf blocks can also be compressed by using the standard SVD procedure explained in Step 3.

However, for non-admissible leaf blocks positioned exactly on the diagonal of the hierarchical matrices $\tilde{\mathbf{H}}$ or $\tilde{\mathbf{G}}$ the factorization is not efficient. These blocks are full rank blocks and the dimensions of $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$, provided by SVD, are of dimensions $N \times R$ and $L \times R$, respectively, which implies that storing the full matrix $\tilde{\mathbf{M}}$ of dimensions $N \times L$, consumes less memory.

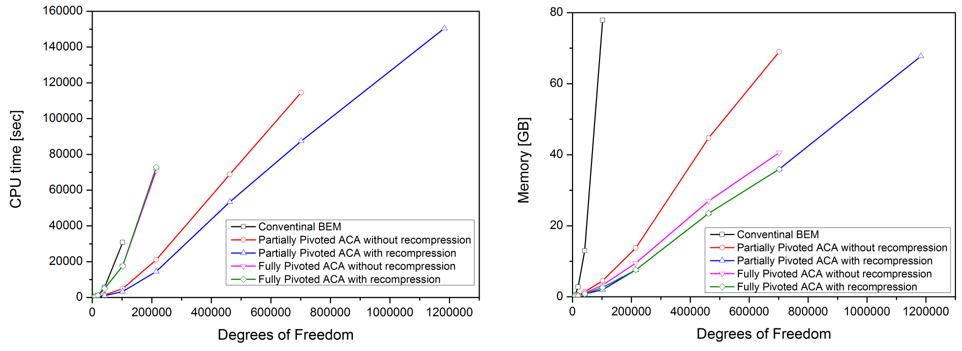
4 Numerical results

In order to demonstrate the efficiency of the proposed hierarchical ACA/BEM formulations, the 2D elastostatic bending problem, described in section 2, is solved using both ACA/BEM and conventional BEM. To compare similar things, a GMRES iterative solver is also employed for the solution of the system (11) in conventional BEM. Taking into account that fiber regions are uncoupled to each other, the matrix $\tilde{\mathbf{A}}$ of the final system of algebraic equations is never formed explicitly, thus saving significant amount of memory. The GMRES multiplications are performed directly on Equations (10) and (11), while a block left diagonal preconditioner is used to accelerate the convergence. The dimensions of each block in the preconditioner are chosen to be approximately equal to the number of nodes that each fiber is discretized into. The inversion of each block is accomplished by using the LU decomposition algorithm. The four ACA techniques described in the previous section and illustrated in Appendix, namely the fully and partially pivoted ACA with and without recompression, are used in the ACA/BEM that solves the aforementioned problem. The cardinality of the cluster tree L_t and the compression matrix accuracy ε, appeared in equations (19), (22) and used in the algorithms of partially and fully pivoted ACA explained in the appendix, are chosen $L_t = 32$ and $\varepsilon = 10^{-5}$, respectively. The GMRES accuracy for solving the algebraic system of equations (12) is chosen $\varepsilon_{GMRES} = 10^{-6}$, noting that the Iterative Solvers Package (ITSOL) free source routines were used to this end. For the computation of SVD and QR decompositions the free source CLAPACK routines were used.

In table 1, the maximum deflections of the composite plate with fibers periodically arranged in a square pattern are listed for various DOFs N and number of fibers N_f. The results were calculated using the above mentioned ACA/BEM formulations and the conventional BEM. The problem parameters, as defined in section 2, are the following: $L = 9$m, $D = 3$m, $E_M = 66$GPa, $v_M = 0.31$, $E_F = 360$GPa and $v_F = 0.25$, $u_f = 0.35$ and $P = 30$MN/m.

Table 1: Accuracy in maximum deflection calculation obtained by the partially and fully ACA with and without recompression formulations and the conventional BEM.

Num. of fibers	Degrees of Freedom	Conven-tional BEM	Partially pivoted ACA without recompression	Partially pivoted ACA with recompression	Fully pivoted ACA without recompr-ession	Fully pivoted ACA with recompr-ession
12	4048	0.0323697	0.0323697	0.0323700	0.0323698	0.0323701
48	8904	0.0305282	0.0305282	0.0305285	0.0305279	0.0305282
108	19112	0.0301607	0.0301607	0.0301617	0.0301600	0.0301608
243	41623	0.0300014	0.0300014	0.0300024	0.0300244	0.0300044
507	102104	0.0299572	0.0299572	0.0299576	0.0299575	0.0299580
1083	214896		0.0299250	0.0299227	0.0299259	0.0299234
2352	461840		0.0299784	0.0299784		
3072	701800		0.0299448	0.0299446		
6075	1.183e6			0.0299567		



(a) (b)

Figure 4: Total CPU time (a) and the memory demand (b), using conventional BEM and fully and partially pivoted ACA with and without recompression formulations.

Observing the results depicted in Table 1, one can say that the first general remark is that for a predefined compression matrix accuracy $\varepsilon = 10^{-5}$ we obtain results with an error being less than to 10^{-4} in the maximum deflection of the composite plate. Another remark is that the partially pivoted ACA without recompression is the most accurate, while the partially pivoted ACA with recompression is the most efficient.

In Figs 4(a) and 4(b) the normalized total CPU time and memory requirements as function of DOFs N are depicted, respectively. In the total CPU time, both the time required for the evaluation of the matrices $\tilde{\mathbf{H}}$, $\tilde{\mathbf{G}}$ and the system solution time have been considered. The total CPU time is normalized by the corresponding time required for the solution of the problem for 100000 DOFs by means of the conventional BEM. Figs 4(a) and 4(b) reveal the well known statement that the total CPU time and the memory demand using conventional BEM is of order $O(N^2)$. Among the four ACA formulations the partially pivoted ACA with recompression is the most efficient. The CPU time and memory requirements for solving a 2D elastic problem with 10^5 DOFs using conventional BEM are about the same with the corresponding ones needed for the same problem with 10^6 DOFs when the partially pivoted ACA is utilized, i.e., an order of magnitude reduction is achieved.

In the sequel, the displacement fields and the corresponding stresses are evaluated for a composite plate with different number of elastic fibers. Figs 8 and 9 depict the contours of displacement u_x and normal stress σ_y, respectively, for the plates with periodically arranged and randomly distributed fibers with $N_f = 48$ and $V_f = 0.35$. In the same figures, the corresponding displacements and stresses for a composite plate homogenized according to Christensen and Lo (1979) self-consistent homogenization model are also provided. Similarly, in Figs 10 and 11 the corresponding contours for $N_f = 1000$ and $V_f = 0.35$, are depicted. As it is apparent from all contours, the Christensen model becomes accurate only for large number fibers and when they are randomly distributed.

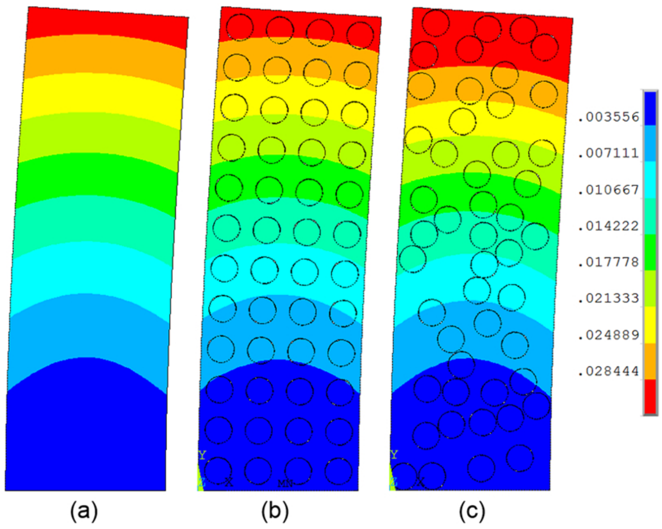


Figure 5: Contours of the displacement magnitude, for $N_f = 48$ and $u_f = 0.35$; (a) homogenized plate (b) periodically arranged fibers and (c) randomly distributed fibers.

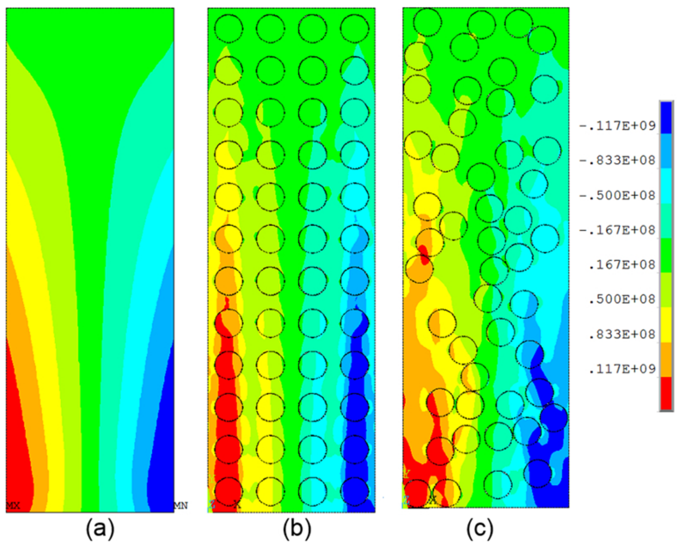


Figure 6: Contours of normal bending stress, for $N_f = 48$ and $u_f = 0.35$; (a) homogenized plate (b) periodically arranged fibers and (c) randomly distributed fibers.

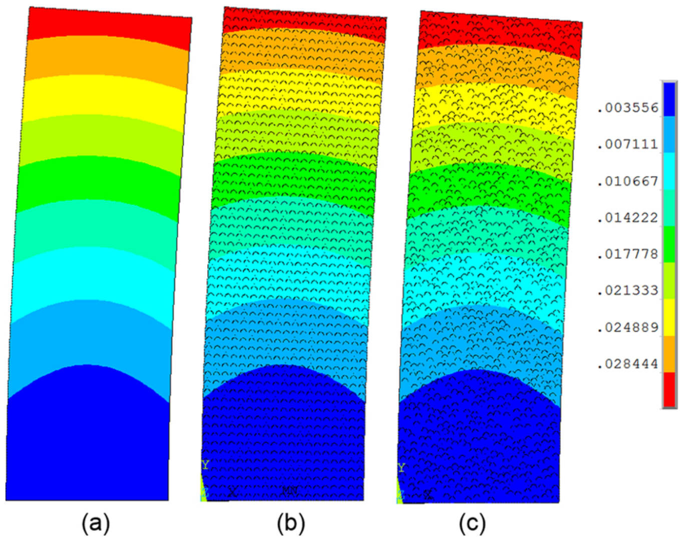


Figure 7: Contours of the displacement magnitude, for $N_f = 1083$ and $u_f = 0.35$; (a) homogenized plate (b) periodically arranged fibers and (c) randomly distributed fibers.

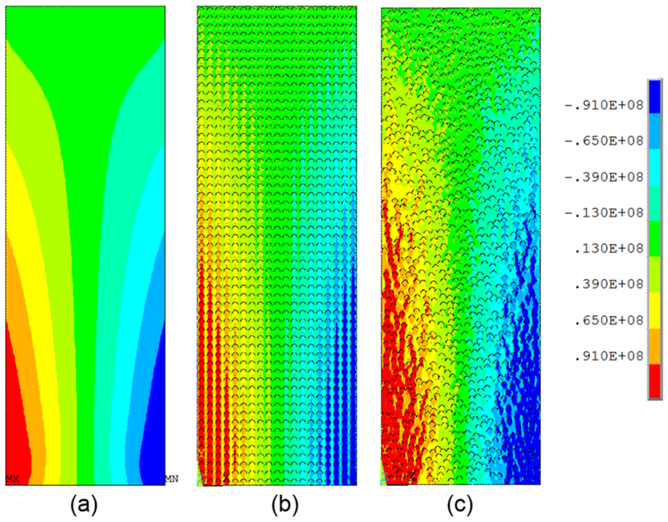


Figure 8: Contours of normal bending stress, for $N_f = 1083$ and $u_f = 0.35$; (a) homogenized plate (b) periodically arranged fibers and (c) randomly distributed fibers.

5 Conclusions

An advanced Boundary Element method (BEM) accelerated via Adaptive Cross Approximation (ACA) and Hierarchical Matrices (HM) techniques for the solution of large-scale elastostatic problems with multi-connected domains has been proposed. Fully and partially pivoted ACA techniques with and without recompression have been testing with the most efficient being the partially pivoted ACA with recompression. The solution of the final linear system of equations has been accelerated via an iterative GMRES solver and the total gain in storage memory and solution time renders the proposed ACA/BEM a significant candidate for solving realistic problems. The largest problem solved in the framework of the present work corresponds to 1.2 million degrees of freedom utilizing a desktop PC with 64 GB RAM. The proposed methodology has been demonstrated with the solution of large scale, plane strain elastic problems dealing with the bending of unidirectional fiber composite plates with large numbers of periodically or randomly distributed cylindrical elastic fibers embedded in a matrix medium. The obtained results have been compared to corresponding ones taken for a plate homogenized according to Christensen and Lo (1979) self-consistent homogenization model. An interesting conclusion revealed by the solution of the aforementioned problems is that Christensen's model becomes accurate only for large number fibers and when they are randomly distributed.

Acknowledgement: This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Framework (NSRF) – Research Funding Program: ARCHIMEDES III.

Appendix

In this appendix the two algorithms dealing with the implementation of the fully and partially pivoted ACA are illustrated.

Algorithm 1: Fully pivoted ACA

Let's consider a matrix block $\tilde{\mathbf{M}}$ of dimensions $N \times L$ with $N \leq L$.

1) Assembly the entire matrix $\tilde{\mathbf{M}}$ by performing the corresponding integrations.

2) Set $K = 1$, $\tilde{\mathbf{R}}^{(1)} = \tilde{\mathbf{M}}$ and calculate the norm $\left\|\mathbf{R}^{(1)}\right\|_F$.

3) Find the pivot element (i_K, j_K) of the matrix $\tilde{\mathbf{R}}_{(K)}$ which corresponds to the maximum absolute value p_K, i.e. $p_K = \max|R_{ij}^{(K)}|$ $i = 1,\dots,N$ and $j = 1,\dots,L$.

4) Calculate the vectors $\mathbf{a}^K$ and $\mathbf{b}^K$, appeared in (17), by storing the column (i, j_K) and the row (i_K, j) containing the pivot element, respectively, as follows:

$$a_i^K = \frac{R_{i,j_K}^K}{p_K} \text{ and } b_j^K = \mathrm{R}_{i_K,j}^K.$$

5) Find the new residual:
$\tilde{\mathbf{R}}^{(K+1)} = \tilde{\mathbf{R}}^{(K)} - \mathbf{a}^K \left(\mathbf{b}^K\right)^T$ and calculate the norm $\left\|\mathbf{R}^{(K+1)}\right\|_F$.

6) Check the stopping criterion:

$$\|\mathbf{R}^{(K+1)}\|_F \leq \varepsilon \cdot \|\mathbf{R}^{(1)}\|_F.$$

7) If the stopping criterion does not hold set $K = K + 1$ and repeat the algorithm returning to the step 3.

8) If the stopping criterion is satisfied, the matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ are formed via Eq. (19) and utilizing all the above calculated vectors $\mathbf{a}^K$ and $\mathbf{b}^K$.

Algorithm 2: Partially Pivoted ACA

Let's consider a matrix block $\tilde{\mathbf{M}}$ of dimensions $N \times L$ with $N \leq L$.

1) Set $K = 1$ and form a list Z containing the indices of the block rows, i.e., $\mathrm{Z} = \{i_1, i_2, \ldots, i_N\}$, noting that the block, as already mentioned, is of dimensions $N \times L$. The indices i_K are sorted in an ascending order. In the first two positions are placed the indices (one for each DOF) corresponding to the node of the C_R cluster which is closer to the center of the Ω_C box and so on. By taking into to account the behavior of the kernels (3) and (4) with respect to the distance r, this sorting increases the probability that the maximum absolute value of the block is located at rows i_1 or i_2.

2) Subtract the index i_K form the list Z, i.e. $\mathrm{Z} = Z - \{i_K\}$.

3) Calculate the vector $\mathbf{b}^K$, appeared in (17), by storing the row (i_K, j) as follows: $b^K_j = \tilde{\mathbf{M}}^K_{i_K,j}$.

4) Compute the new vector $\mathbf{b}^K$ as follows: $\mathbf{b}^K = \mathbf{b}^K - \sum_{m=1}^{K-1} a^m_{i_K} \mathbf{b}^m$, noting for $K = 1$ the summation is trivial.

5) If $\left|\mathbf{b}^K\right| \leq eps$ and $\mathrm{Z} \neq \emptyset$ return and repeat the algorithm form Step 2 after choosing a new index i_K from the list Z, else stop the algorithm. In the present work $eps = 10^{-14}$.

6) Find the index j_K corresponding to the maximum absolute value p_K, i.e., $p_K = \max\left|\mathbf{b}^K\right|$.

7) Normalize $\mathbf{b}^k$ with p_K, i.e., $\mathbf{b}^K = \mathbf{b}^K / p_K$.

8) Compute the vector $\mathbf{a}^K$ as follows: $a^K_i = M_{i,j_K}$.

9) Compute new vector $\mathbf{a}^K$ as follows: $\mathbf{a}^K = \mathbf{a}^K - \sum_{m=1}^{K-1} b^m_{j_K} \mathbf{a}^m$, noting for $K = 1$ the summation is trivial.

10) Compute the Frobenius norm of the approximant matrix $\tilde{\mathbf{M}}^K$ by the following recursive formula:

$$(\|\tilde{\mathbf{M}}^{(K)}\|_F)^2 = (\|\tilde{\mathbf{M}}^{(K-1)}\|_F)^2 + 2\sum_{i=1}^{K-1} ((\mathbf{a}^K)^T \mathbf{a}^i)((\mathbf{b}^K)^T \mathbf{b}^i) + (\|\mathbf{a}^K\|_F)^2 (\|\mathbf{b}^K\|_F)^2,$$

noting for $K = 1$ the summation is trivial and $\left\|\tilde{\mathbf{M}}^{(0)}\right\|_F = 0$.

11) Check the stopping criterion:

$$\left\|\mathbf{a}^K\right\|_F \left\|\mathbf{b}^K\right\|_F \leq \varepsilon \left\|\tilde{\mathbf{M}}^K\right\|_F.$$

12) If the stopping criterion is not satisfied and $Z \neq \varnothing$ set $K = K + 1$ return and repeat the algorithm form Step 2.

13) If the stopping criterion is satisfied, the matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ are formed, via Eqs (19), using all the above calculated vectors $\mathbf{a}^K$ and $\mathbf{b}^K$.

It should be mentioned that (i) in the case of $N > L$ the above described partially pivoted ACA algorithm is modified by considering that the list Z contains the indices of the block columns i.e., $Z = \{j_1, j_2, \ldots, j_N\}$ and the followed algorithm is modified similarly. (ii) As it is already mentioned, the stopping criterion in the step 11 is heuristic, which implies that there are cases where although the algorithm terminates the calculated vectors $\mathbf{a}^K$ and $\mathbf{b}^K$ do not provide an approximant $\tilde{\mathbf{M}}^{(K)}$ with the given accuracy ε. In order to overcome this drawback extra convergence checks according to the methodologies introduced in [Bebendorf (2008)] are applied.

References

Aliabadi, M. H. (2002): The boundary element method: applications in solids and structures, vol. 2. John Wiley & Sons Ltd., England.

Bapat, M. S.; Liu, Y. J. (2010): A new adaptive algorithm for the fast multipole boundary element method. *CMES: Computer Modeling in Engineering & Sciences*, vol. 58, no. 2, pp. 161–184.

Bebendorf, M. (2000): Approximation of boundary element matrices. *Numerische Mathematik*, vol. 86, pp. 565–589.

Bebendorf, M. (2008): Hierarchical matrices. a means to efficiently solve elliptic boundary value problems. Berlin: Lecture Notes in Computational Science and Engineering, vol. 63, Springer Berlin, Heidelberg.

Bebendorf, M.; Grzhibovskis, R. (2006): Accelerating Galerkin Bem for linear elasticity using adaptive cross approximation, *Mathematical Methods in the Applied Sciences*, vol. 29, pp. 1721–1747.

Bebendorf, M.; Rjasanow, S. (2003): Adaptive low-rank approximation of collocation matrices. *Computing*, vol. 70, pp. 1–24.

Beer, G.; Smith, I.; Duenser, C. (2008): The boundary element method with programming. Springer.

Benedetti, I.; Alliabadi, M. H. (2009): A fast hierarchical dual boundary element method for three dimensional elastodynamic crack problems. *International Journal for Numerical Methods in Engineering*, vol. 84, no. 9, pp. 1356–1378.

Benedetti, I.; Alliabadi, M. H.; Davi, G. (2008): A fast 3D dual boundary element method based on hierarchical matrices. *International Journal of Solids and Structures*, vol. 45, pp. 2355–2376.

Benedetti, I.; Milazzo, A.; Alliabadi, M. H. (2009): A fast dual boundary element method for 3d anisotropic crack problems. *International Journal for Numerical Methods in Engineering*, vol. 80, no. 10, pp. 1356–1378.

Beskos, D. E. (1997): Boundary element methods in dynamic analysis: Part II (1986–1996). *Applied Mechanics Reviews*, vol. 50, pp. 149–197.

Borm, S.; Grasedyck, L.; Hackbusch, W. (2003): Introduction to hierarchical matrices with applications. *Eng Anal Boundary Elem*, vol. 27, pp. 405-422.

Borm, S.; Grasedyck, L.; Hackbusch, W. (2003a): Hierarchical matrices, Lecture notes no. 21, Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany.

Bucher, H. F.; Wrobel, L. C.; Mansur, W. J.; Magluta, C. (2003): Fast solution of problems with multiple load cases by using wavelet-compressed boundary element matrices. *Commun. Numer. Meth. Engng*, vol. 19, pp. 387–399.

Bonnet, M. (1999): Boundary Integral Equation Method for Solids and Fluids. John Wiley and Sons.

Brancati, A.; Aliabadi, M. H.; Benedetti, I. (2009): Hierarchical adaptive cross approximation GMRES technique for solution of acoustic problems using the boundary element method. *CMES: Computer Modeling in Engineering & Sciences*, vol. 43, no. 2, pp. 149–172.

Brunner, D.; Junge, M.; Rapp, P.; Bebendorf, M.; Gaul, L. (2010): Comparison of the fast multipole method with hierarchical matrices for the helmholtz-BEM. *CMES: Computer Modeling in Engineering & Sciences*, vol. 58, no. 2, pp. 131–158.

Christensen, R. M.; Lo, K. H. (1979): Solutions for effective shear properties in three phase sphere and cylinder models. *J Mech Phys Solids*, vol. 27, pp. 315–330.

Ebrahimnejad, L.; Attarnejad, R. (2010): Fast solution of BEM systems for elasticity problems using wavelet transforms, vol. 87, no. 1, pp. 77–93.

Frangi, A.; Bonnet, M. (2010): On the application of the fast multipole method to helmholtz-like problems with complex wavenumber. *CMES: Computer Modeling in Engineering & Sciences*, vol. 58, no. 3, pp. 271–296.

Frangi, A.; Guiggiani, M. (2000): A direct approach for boundary integral equations with high-order singularities. *International Journal for Numerical Methods in Engineering*, vol. 49, pp. 871–898.

Golub, G. H.; Van Loan, C. F. (1996): Matrix Computations. John Hopkins University Press, London.

Grasedyck, L. (2005): Adaptive recompression of H-matrices for BEM, *Computing*, vol. 74, pp. 205–223.

Hu, N.; Wang, B.; Tan, G.W.; Yao, Z. H.; Yuan, W. F. (2000): Effective elastic properties of 2-D solids with circular holes: numerical simulations. *Compos. Sci. Technol.*, vol. 60, no. 9, pp. 1811–1823.

Kong, F. Z.; Yao, Z. H.; Zheng, X. P. (2002): BEM for simulation of a 2D elastic body with randomly distributed circular inclusions. *Acta Mech. Solida Sinica*, vol. 15, no. 1, pp. 81–88.

Lei, T.; Yao, Z. H.; Wang, H. T.; Wang, P. B. (2006): A parallel fast multipole BEM and its applications to large-scale analysis of 3-D fiber-reinforced composites. *Acta Mech, Sinica*, vol. 22, no. 3, pp. 225–232.

Liu, Y. (2009): Fast multipole boundary element method. Campridge University Press, UK.

Liu, Y. J.; Nishimura, N.; Otani, Y.; Takahashi, T.; Chen, X. L.; Munakata, H. (2005): A fast boundary element method for the analysis of fiber-reinforced composites based on a rigid-inclusion model. *Journal of Applied Mechanics*, vol. 72, pp. 115–128.

Manolis, G. D.; Polyzos, D. (2009): Recent advances in boundary element methods. A Volume to Honor Professor Dimitri Beskos. Springer.

Messner, M.; Schanz, M. (2010): An accelerated symmetric time-domain boundary element formulation for elasticity. *Engineering Analysis with Boundary Elements*, vol. 34, pp. 944–955.

Millazzo, A.; Benedetti, I.; Alliabadi, M. H. (2012): Hierarchical fast BEM for anisotropic time-harmonic 3-D elastodynamics, *Computers and Structures*, vol. 96–97, pp. 9–24.

Ntalaperas, D.; Tsinopoulos, S. V.; Polyzos, D. (2010): A Fast Wavelet/BEM for Wave Scattering Solutions. In Mathematical Methods in Scattering Theory and Biomedical Engineering, A. Charalambopoulos, D. Fotiadis, D. Polyzos, Eds, World Scientific Publishing.

Phillips, J. R. and White, J. K. (1997): A precorrected-FFT method for electrostatic analysis of complicated 3-D structures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 10, pp. 1059–1072.

Polyzos, D.; Tsinopoulos, S. V.; Beskos, D. E. (1998): Static and dynamic boundary element analysis in incompressible linear elasticity. *European Journal of Mechanics, A/Solids*, vol. 17, pp. 515–536.

Ravnik, J.; Škerget, L.; Zunic, Z. (2009): Comparison between wavelet and fast multipole data sparse approximations for Poisson and kinematics boundary – domain integral equations. *Comput. Methods Appl. Mech. Engng*, vol. 198, pp. 1473–1485.

Rjasanow, S.; Steinbach, O. (2007): The Fast Solution of Boundary Integral Equations. Springer New York.

Wang, L.; Ma, Y.; Meng, Z.; Huang, J. (2013): Wavelet operational matrix method for solving fractional integral and differential equations of Bratu-type. *CMES: Computer Modeling in Engineering & Sciences*, vol. 92, no. 4, pp. 353–368.

Wang, Q.; Miao, Y.; Zheng, J. (2010): The hybrid boundary node method accelerated by fast multipole expansion technique for 3D elasticity. *CMES: Computer Modeling in Engineering & Sciences*, vol. 70, no. 2, pp. 123–152.

Wang, H. T.; Yao, Z. H. (2008): A rigid- fiber-based boundary element model for strength simulation of carbon nanotube reinforced composites. *Comput. Model. Eng. Sci.*, vol. 29, no. 1, pp. 1–13.

Wang, H. T.; Yao, Z. H. (2011): A fast multipole dual boundary element method for the three-dimensional crack problems. *CMES: Computer Modeling in Engineering & Sciences*, vol. 72, no. 2, pp. 115–148.

Xiao, J.; Ye, W.; Cai, Y.; Zhang, J. (2012): Precorrected FFT accelerated BEM for large-scale transient elastodynamic analysis using frequency-domain approach. *International Journal for Numerical Methods in Engineering*, vol. 90, no. 1, pp. 116–134.

Yao, Z. H.; Kong, F.; Wang, H.; Wang, P. (2004): 2D simulation of composite materials using BEM. *Engineering Analysis with Boundary Elements*, vol. 28, pp. 927–935.

Yusa, Y.; Yoshimura, S. (2013): A fast regularized boundary integral method for practical acoustic problems. *CMES: Computer Modeling in Engineering & Sciences*, vol. 91, no. 6, pp. 463–484.

Zechner, J. (2012): A fast boundary element method with hierarchical matrices for elastostatics and plasticity. http://www.ifb.tugraz.at/juergen/phd_zechner_2012.