EFFICIENT WAYS TO CONSTRUCT BOUNDARY-ELEMENT CODES: CASE OF 3D PROBLEMS

Francisco Célio de Arajo

dearaujofc@gmail.com

Department of Civil Engineering, Federal University Ouro Preto

35400-000, Ouro Preto, MG, Brazil

Abstract. *Compared to domain methods, applying boundary-element formulations to solve 3D complex solids (e.g. general composites) may be quite advantageous as domain discretizations are avoided. This statement has been recurrently mentioned in the boundary-element literature. Indeed, the Finite Element Method (FEM) has been the tool of choice for analyzing structural systems by means of simplified theories as beam or plate ones. On the other hand, the modeling of 3D domains with finite elements may pose special difficulties, e.g. in case of describing complex (microstructural) geometrical details or large (infinite) regions. And in fact, discretizing only the boundary of the domain, as typically happens in applying the Boundary Element Method (BEM), alleviates considerably the modeling of geometrically complex solids. Besides, as boundary-element formulations start from the analytical integral representations of sought-after responses, they provide high-accurate solutions, fulfill radiation conditions, and allow for easily deriving unified processes for dealing with thick-walled and thin-walled domains. The BEM also does not require interelement compatibility. And all this together conspires to make the BEM convenient to deal with continuum mechanics problems. This paper discusses efficient ways to construct boundary-element codes, taking into account the relevance of the method to solve real-life problems. In this aspect, three points are considered: (1) subregioning techniques to model heterogeneous problems, (2) special numerical quadratures for singular and quasi-singular integrals, and (3) the solution of the non-hermitian system of equations via direct and iterative solvers. Details of the involved formulations are presented. The performance of the technique is verified by solving complex 3D solids and composites.*

Keywords: *First keyword, Second keyword, Third keyword (up to 5 keywords)*

1 INTRODUCTION

It has been recurrently mentioned in the boundary-element literature that boundary-element formulations are advantageous in comparison domain methods because domain discretizations are avoided [7]. In fact, by considering simplified theories as beam or plate ones to analyze structural systems, the Finite Element Method (FEM) has been the tool of choice. On the other hand, by dealing with 3D contiunuum problems, finite element formulations may pose special difficulties, e.g. in case of describing complex (microstructural) geometrical details or large (infinite) regions.

Indeed, discretizing only the boundary of the domain considerably simplifies the modeling of geometrically complex solids. Besides, boundary-element formulations start from the analytical integral representations of sought-after responses, and therefore they may provide high-accurate solutions as long as the available integrals are accurately evaluated [2]. In addition, they do not require any interelement compatibility for convergence. Thus, the BEM may be quite convenient to model complex solids as e.g. the grain-based microstructural composite materials considered in [6]. However, to deal with heterogeneous solids, BEM formulations demand substructuring (subregioning) techniques as the integral representations are essentially constructed based for homogeneous domains. Efficient integration algorithms are also required for evaluating the singular and nearly-singular integrals involved [2].

To cope with general heterogeneous domains, the boundary-element subregion-by-subregion (BE-SBS) technique, described in Araujo and Gray [2], and in Araujo, d'Azevedo and Gray [5], is recommended. This technique essentially consists of a robust BE-BE coupling technique based on the domain decomposition method (DDM) that allows for modeling the coupling of a generic number of subdomains. In this technique, no explicit assembling of the subdomain submatrices into a global system of equations is carried out. Instead, iterative Krylov solvers are employed, which allow for the independent treatment of the many subdomains availble in the model. Consequently, huge memory saving is brought about as the zero blocks in the highly spare coupled system are excluded. The coupling conditions are imposed only during the system solution.

In this paper, the robustness and relevance of the SBS algorithm are discussed. Besides, one attempts to show details of the mathematical formulations of the Krylov solvers employed. The whole computational implementation of the SBS algorithm, including the use of the SBS matrices to generate an incomplete LU preconditioner, is also presented. Complex problems are solved to shown the robustness and performance of the algorithm.

THE GENERIC SUBREGIONING ALGORITHM

This paper mainly focuses on discussing efficient ways to create a robust algorithm for coupling any number of boundary-element (BE) models. For practical purposes, we are talking of an algorithm that (1) will allow the easy modeling of complex heterogeneous solids, and (2) that will provide good computational performance by optimizing memory storage room and CPU time. In the SBS (subegion-by-subregion) algorithm discussed here, robustness will be established

by the following characteristics: (1) independent generation of a desired number of BE models, (2) automated search strategy for identifying coupling nodes, (3) correct simulation of traction discontinuity, and (4) optimal use of memory and CPU time reduction by excluding all zero blocks of the sparse coupled system of equations.

As described below, the SBS algorithm proposed here with the premise to be robust may be subdivided in three generic parts: (1) the generation of the BE subregion models, (2) the automated search for coupled nodes, and (3) the establishment of the SBS data structure and solution of the coupled domain.

The Modeling of the BE Subregions

For a generic number of subregions, the BE models necessary to solve a certain problem are modeled separately. Here, a loop over the given number of subregions, ns, is carried out, and, for each subregion, the data are inputted, the corresponding matrices assembled, and boundary conditions imposed according to flowchart in Figure 1.

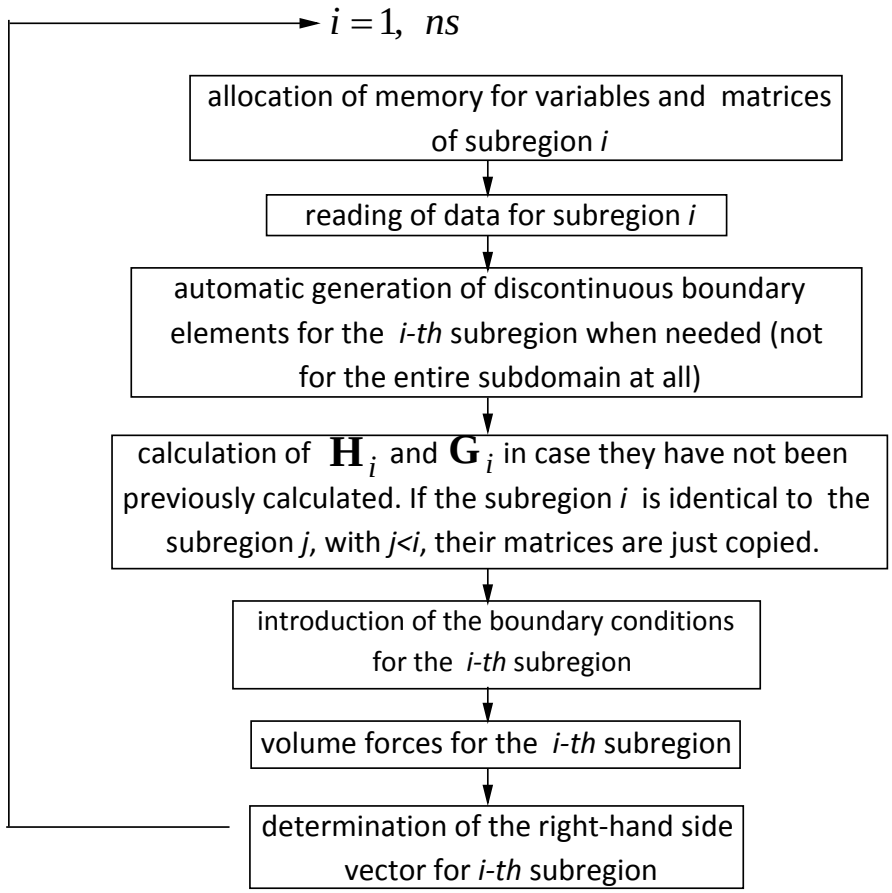


Figure 1. Generation of the BE models.

In this part of the algorithm, the associated matrices are calculated and assembled exactly in same way they are in standard BE formulations. One very important aspect of the generation of the models is that they demand the consideration of discontinuous boundary elements to cope with unavoidable traction discontinuities around corners and edges inside the solids (see Fig. 2). Notice that special care should be taken in evaluating the nearly singular integrals

over discontinuous boundary elements [2]. In fact, efficient evaluation of the boundary element integrals is a quite fundamental issue in BE formulations in general, which will reflect in the whole efficiency of the BE code. In [2], [14], [1], all details of the numerical integration techniques employed here are described. One mentions though, concerning the models generation, that discontinuous elements (generated by displacing the nodes inwards the element) are constructed by a fully automated process, and the analyst should just enter the information whether the element is discontinuous or not. In addition, the matrix-copy option for identical subregions avoids calculating repeatedly the same coefficient matrix (see flowchart in Figure 1).

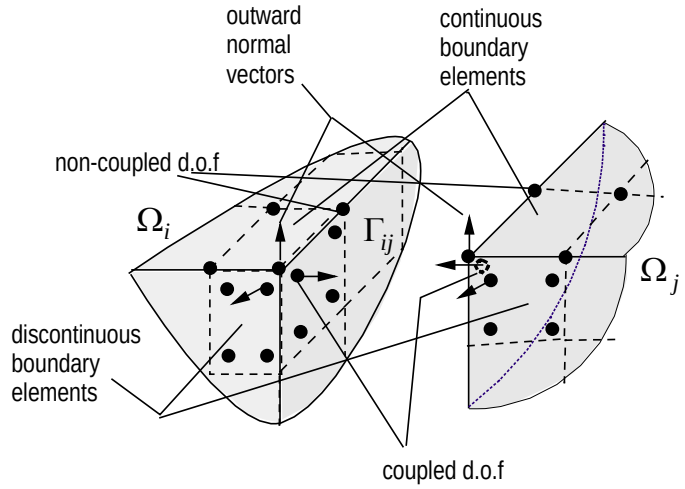


Figure 2. Discontinuous elements and coupled nodes.

Search for coupled degrees of freedom

In the coupling strategy developed here, a degree of freedom (*dof*) is coupled exclusively with another degree of freedom of another subregion (see Figure 2). Thus, for an interface element, variables $icoupn(id)$ and $icoups(id)$ are created to indicate, respectively, the degree of freedom and subregion numbering coupled with degrees of freedom pertaining to that element. In the algorithm, given the information on the element interface conditions, an automated process is then executed to identify the coupled degrees of freedom (see flowchart in Figure 3), and variables $icoupn(id)$ and $icoups(id)$ are created. In the solution phase, discussed later in this paper, these variables are used to impose the coupling conditions, which in this study are just defined as

$$\begin{cases} u_k^{(i)}(\mathbf{x}) = u_k^{(j)}(\mathbf{x}) \\ p_k^{(i)}(\mathbf{x}) = -p_k^{(j)}(\mathbf{x}) \end{cases} \text{, if } \mathbf{x} \in \Gamma_{ij} \tag{1}$$

where Γ_{ij} is the interface between subregions i and j.

Matrix data structure and solution of the coupled system

In the BE-SBS algorithm considered here [2], [4], inspired in the element-by-element (EBE) technique developed by Hughes for large-order FE models [13], the highly sparse global coupled matrix is optimally handled by merely not explicitly assembling it. By employing an

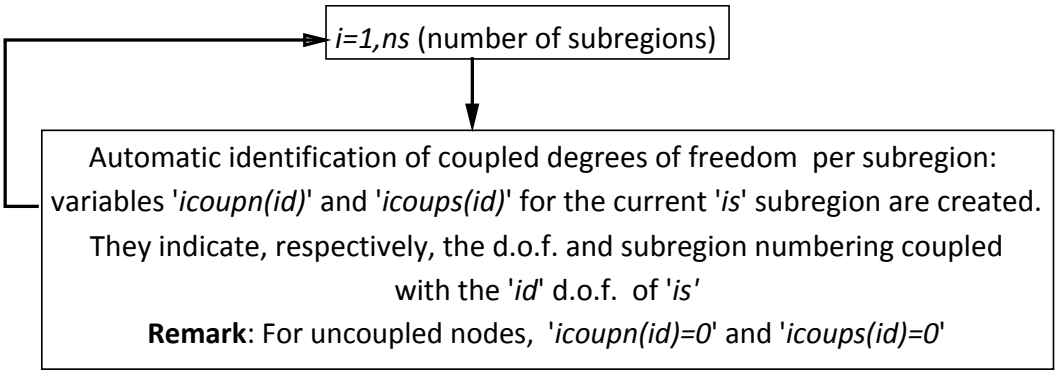


Figure 3. Search for coupled degrees of freedom.

iterative solver, a solution strategy for general coupled problems may be derived, in which the matrix-vector products (involving the coefficient matrix and possibly its transpose) are calculated from the separate contributions from from the matrices for each subregion (or substructure). In the algorithm, the interface conditions (given by relations in 1) are imposed, iteration-by-iteration, on a direct way, during the solution process. Thus, the global matrix need not be explicitly assembled. The subsystems are treated as they were physically independent from one another, so that only the exact memory space for storing all of them independently is allocated.

After introducing the boundary conditions, the BE subsystems associated with the ns BE models are given according to the relation

$$\sum_{j=1}^{i-1}(\mathbf{H}_{ij}\mathbf{u}_{ji} - \mathbf{G}_{ij}\mathbf{p}_{ij}) + \mathbf{A}_{ii}\mathbf{x}_i + \sum_{j=i+1}^{ns}(\mathbf{H}_{ij}\mathbf{u}_{ij} - \mathbf{G}_{ij}\mathbf{p}_{ji}) = \mathbf{B}_{ii}\mathbf{y}_i\ ,\ i = 1, ns, \tag{2}$$

where $\mathbf{H}_{ij}$ and $\mathbf{G}_{ij}$ denote regular BE matrices for subregion Ω_i and associated, respectively, with the interface values $\mathbf{u}_{ij}$ and $\mathbf{p}_{ij}$ at Γ_{ij}. Notice that Γ_{ii} is the outer boundary of Ω_i. In addition, $\mathbf{H}_{ij} = \mathbf{H}_{ji} = \mathbf{G}_{ij} = \mathbf{G}_{ji} = \mathbf{0}$ if the there is no coupling between i and j subdomains (fact that accounts for the sparsity of the coupled system).

In fact, the BE matrices for all subregions, $\mathbf{H}_i$ and $\mathbf{G}_i$, $i = 1, \ldots, ns$, are ordered according to the data structure given by

$$\begin{array}{rccccc} & \overbrace{\qquad\qquad}^{\text{block 1}} & & \overbrace{\quad}^{\text{block 2}} & & \overbrace{\qquad\qquad}^{\text{block 3}} \\ \mathbf{H}_i = & \left[\mathbf{H}_{i1} \ldots \mathbf{H}_{i(i-1)}\right. & \Big| & \mathbf{A}_{ii} & \Big| & \left.\mathbf{H}_{i(i+1)} \ldots \mathbf{H}_{i(ns)}\right] \\ \mathbf{G}_i = & \left[\mathbf{G}_{i1} \ldots \mathbf{G}_{i(i-1)}\right. & \Big| & \mathbf{B}_{ii} & \Big| & \left.\mathbf{G}_{i(i+1)} \ldots \mathbf{G}_{i(ns)}\right] \end{array}\ , \tag{3}$$

which is considered to calculate the matrix-vector products (needed by the iterative solver) from the subdomains of the problem. With aid of the variables $icoupn(id)$ and $icoups(id)$, it is possible to implement these matrix-vector products without structuring the subdomain's BE matrices. However it is much less efficient than considering structured matrix-vector products (SMVP) based on the data structure shown in the relations (3) (see [3]).

THE KRYLOV SOLVERS

For a system of equations

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

Q-preconditioned Krylov solvers may be defined as iterative solvers for that the pseudo-residual at the i-th iteration, $\boldsymbol{\delta}_i$, is given by

$$\boldsymbol{\delta}_i = \psi_i(\mathbf{Q}^{-1}\mathbf{A})\boldsymbol{\delta}_0 \in K_{i+1}(\mathbf{Q}^{-1}\mathbf{A}, \boldsymbol{\delta}_0) \,,$$

where $K_{i+1}(\mathbf{Q}^{-1}\mathbf{A}), \boldsymbol{\delta}_0)$ is the $(i+1)$-dimensional Krylov subspace associated with matrix $(\mathbf{Q}^{-1}\mathbf{A})$ and $\boldsymbol{\delta}_0$, and $\mathbf{Q}$ is the preconditioning matrix [21]. Thus, the pseudo-residual reduction polynomial above, ψ_i, is of the form

$$\psi_i(\mathbf{Q}^{-1}\mathbf{A}) = \mathbf{I} + \sum_{k=1}^{i} \beta_k^{(i-1)} (\mathbf{Q}^{-1}\mathbf{A})^k \,, \tag{4}$$

where the scalars $\beta_k^{(i-1)}$ are the components of the pseudo-residual on the Krylov basis $(\mathbf{Q}^{-1}\mathbf{A})^k\boldsymbol{\delta}_0$, $k = 1, \ldots, i$.

Many Krylov subspace methods have been devised in the last 5-6 decades [16], [17], [21]. Based on the the length of their recurrence formulas, they may be subdivided into long-recurrence methods as the GMRES [15] and short recurrence methods as the BiCG (biconjugate gradient) [11], the QMR (quasi-minimal residual) [12], and the BiCGSTAB(*l*) (*l*-dimensional biconjugate gradient stabilized) [19]. In fact, long-recurrence methods are often ruled out for solving very large problems, because of their high memory requirements, whereas short-recurrence solvers, as the BiCG and the BiCGSTAB(l), are more convenient. Considering previous experiences by the authors [4], [5], the BiCG and BiCGSTAB(*l*) will be applied in this study.

The BiCG method. This method has been proposed by Fletcher [11] in 1976, takes into account the two-term iterative formulas

$$\begin{cases} \mathbf{x}_{i+1} = \mathbf{x}_i + \lambda_i \mathbf{p}_i \\ \mathbf{p}_i = \boldsymbol{\delta}_i + \alpha_i \mathbf{p}_{i-1} \end{cases},$$

where, for certain scalars $\gamma_k^{(i-1)}$ (c.f. equation 4), $\mathbf{p}_i$ (search-direction vector) is given by

$$\begin{cases} \mathbf{p}_i = \phi_i(\mathbf{Q}^{-1}\mathbf{A})\boldsymbol{\delta}_0 \in K_{i+1}(\mathbf{Q}^{-1}\mathbf{A}, \boldsymbol{\delta}_0) \,, \\ \phi_i(\mathbf{Q}^{-1}\mathbf{A})\boldsymbol{\delta}_0 = \mathbf{I} + \sum_{k=1}^{i} \gamma_k^{(i-1)} (\mathbf{Q}^{-1}\mathbf{A})^k \,, \end{cases}$$

and the parameters λ_i and α_i in the iterative formulas are determined from the orthogonality conditions

$$\begin{cases} \boldsymbol{\delta}_i^{*T}\boldsymbol{\delta}_j = 0 \\ \\ \mathbf{p}_i^{*T}(\mathbf{Q}^{-1}\mathbf{A})\mathbf{p}_j = 0 \end{cases}, \; i \neq j \; ,$$

where

$$\begin{cases} \boldsymbol{\delta}_i^* = \psi_i[(\mathbf{Q}^{-1}\mathbf{A})^T]\boldsymbol{\delta}_0^* \in K_{i+1}[\mathbf{Q}^{-1}\mathbf{A})^T, \boldsymbol{\delta}_0^*] \\ \\ \mathbf{p}_i^* = \phi_i[(\mathbf{Q}^{-1}\mathbf{A})^T]\boldsymbol{\delta}_0^* \in K_{i+1}[\mathbf{Q}^{-1}\mathbf{A})^T, \boldsymbol{\delta}_0^*] \end{cases} .$$

One obtains [17], [21]:

$$\begin{cases} \lambda_i = \frac{\boldsymbol{\delta}_i^{*T}\boldsymbol{\delta}_i}{\mathbf{p}_i^{*T}(\mathbf{Q}^{-1}\mathbf{A})\mathbf{p}_i} \\ \\ \alpha_i = \frac{\boldsymbol{\delta}_i^{*T}\boldsymbol{\delta}_i}{\boldsymbol{\delta}_{i-1}^{*T}\boldsymbol{\delta}_{i-1}} \end{cases} . \tag{5}$$

The BiCGSTAB(*l*) method. An interesting theoretical contribution concerning the formulation of iterative methods was proposed by Sonneveld [20], who devised the CGS method aiming to square the convergence rate of the BiCG method for about the same computational cost per iteration, and to avoid working with the auxiliary transpose matrix of the system. The basic idea of the CGS was to calculate the BiCG parameters by applying twice the pseudo-residual reduction polynomial $\psi_i(\cdot)$. Although the performance of this method was not good at all, it opened the theoretical possibilities for the devising of the stabilized biconjugate gradiente method (BiCGSTAB) by van der Vorst [21], and its generalization, BiCGSTAB(*l*), by Sleijpen and Fokkema [19]. The basic idea of these methods is to calculate the same BiCG parameters in relations (5) by imposing additional conditions associated with the minimization of the pseudo-residual, $\boldsymbol{\delta}_j$.

In the BiCGSTAB(*l*) method, the pseudo-residual, $\boldsymbol{\delta}'_j$, and the search direction vector, $\mathbf{p}'_i$, are given by

$$\begin{cases} \boldsymbol{\delta}'_j = \eta_j(\mathbf{Q}^{-1}\mathbf{A})\psi_j(\mathbf{Q}^{-1}\mathbf{A})\boldsymbol{\delta}_0 \; , \\ \\ \mathbf{p}'_{j-1} = \eta_j(\mathbf{Q}^{-1}\mathbf{A})\phi_{j-1}(\mathbf{Q}^{-1}\mathbf{A})\boldsymbol{\delta}_0 \; , \end{cases} \tag{6}$$

where the additional polynomial $\eta_j(\cdot)$ is of the form

$$\eta_j(\mathbf{Q}^{-1}\mathbf{A}) = \mathbf{I} - \sum_{k=1}^{l} \omega_{kj}(\mathbf{Q}^{-1}\mathbf{A})^k \; ,$$

and the parameters ω_{kj} are determined so as minimize $\boldsymbol{\delta}'_j$ among a *l*-dimensional subspace. Notice that in relations (6), the minimization polynomial, $\eta_j(\cdot)$, is applied on the BiCG pseudo-residual, $\psi_j(\mathbf{Q}^{-1}\mathbf{A})\boldsymbol{\delta}_0$, and on the BiCG search direction vector, $\phi_{j-1}(\mathbf{Q}^{-1}\mathbf{A})\boldsymbol{\delta}_0$. By applying then the BiCGSTAB(*l*) method, the BiCG parameters in equation (5) are calculated by

$$\begin{cases} \lambda_{j+k} &= \dfrac{\boldsymbol{\delta}_0^{*T}\eta_{j+k}(\mathbf{Q}^{-1}\mathbf{A})\boldsymbol{\delta}_{j+k}}{\boldsymbol{\delta}_0^{*T}(\mathbf{Q}^{-1}\mathbf{A})\eta_{j+k}(\mathbf{Q}^{-1}\mathbf{A})\mathbf{p}_{j+k}} \\ \alpha_i &= \dfrac{\boldsymbol{\delta}_0^{*T}\eta_{j+k}(\mathbf{Q}^{-1}\mathbf{A})\boldsymbol{\delta}_{j+k}}{\boldsymbol{\delta}_0^{*T}\eta_{j+k}(\mathbf{Q}^{-1}\mathbf{A})\boldsymbol{\delta}_{j+k-1}} \end{cases} . \tag{7}$$

Preconditioning. Since the convergence rate of iterative solvers depends on the spectral conditioning of the system matrix, preconditioning plays a fundamental role in the performance of iterative solvers. For general non-symmetric systems of equations, including BEM systems, a series of preconditioners have been reported in the technical literature [21], [17], [9], [10]. In general, the splitting matrix of basic iterative methods as the Jacobi, block Jacobi, Gauss-Seidel or of the ILU (incomplete LU decomposition) methods can be used to construct preconditioners. Furthermore, domain decomposition methods (DDM) allied with direct methods have also been employed to construct global preconditioners [21].

In this paper, besides the Jacobi preconditioner, the SBS algorithm is employed to construct a global block-diagonal (BD) preconditioner, the SBS-BD (subregion-by-subregion block-diagonal) preconditioner. The SBS algorithm is nothing other than a DDM-based strategy to substructure a physical problem via the BEM, and as the subsystems are independently assembled, the block-diagonal matrices corresponding to each subregion can be easily decomposed in their $\mathbf{L}$ and $\mathbf{U}$ factors. Moreover, implementing this preconditioners on parallel-processing platforms is straightforward.

Organizing the boundary variables for the i-th subregion according to the boundary-value sequence $\left\{\mathbf{p}_{i1}\ \mathbf{p}_{i2}\ \ldots\ \mathbf{p}_{i(i-1)}\ \mathbf{x}_{ii}\ \mathbf{u}_{i(i+1)}\ \ldots\ \mathbf{u}_{in_s}\right\}$, it results from equation (2) that the SBS-BD preconditioner is given by

$$\mathbf{Q}_i = [-\mathbf{G}_{i1}\ \ldots\ -\mathbf{G}_{i(i-1)}\ \mathbf{A}_{ii}\ \mathbf{H}_{i(i+1)}\ \ldots\ \mathbf{H}_{in_s}]\ ,\ i = 1, ns.$$

In this study, the BE subdomain matrices are straightforwardly considered to construct the global SBS-BD preconditioner for the coupled system of equations. As the global system matrix, the global SBS-BD preconditioner is not explicitly assembled either. It is separately stored per subregion at an additional memory space of the size $(nno \times ndofn) \times (nno \times ndofn)$, where nno is the number of nodes of the model, and $ndofn$ is the number of degrees of freedom per node. In the results presented later on, the BE-SBS-based preconditioner is employed to left precondition the BiCG and BiCGSTAB(l) solvers, which means that systems like $(\mathbf{L}_i\mathbf{U}_i)\mathbf{x}_i = \mathbf{x}_i$ have to be solved, where $\mathbf{x}_i$ is the iterative solution for the i-th subregion [5]. Notice that for the BiCG solver, $(\mathbf{L}_i\mathbf{U}_i)^T\mathbf{x}_i = \mathbf{x}_i$ has aditionally to be solved, whereas the BiCGSTAB(l) iterations does not need any transpose-matrix operation. The flowchart shown in Figure 4 gives a general idea of the whole SBS algorithm.

APPLICATIONS

To show the performance of the SBS algorithm along with the implemented preconditioned solvers, the cantilever beam in Figure 5, with variable channel cross section, under torsion, is analyzed. The beam has the following material properties: $E = 2.1 \times 10^6\ kg/cm^2$, $G = 8.05 \times 10^5\ kg/cm^2$. The geometrical details of the cross section variation are furnished in Figure 5b. This problem was analyzed for a torque $M_t = 300\ kg{\times}cm$ in [18], where the values

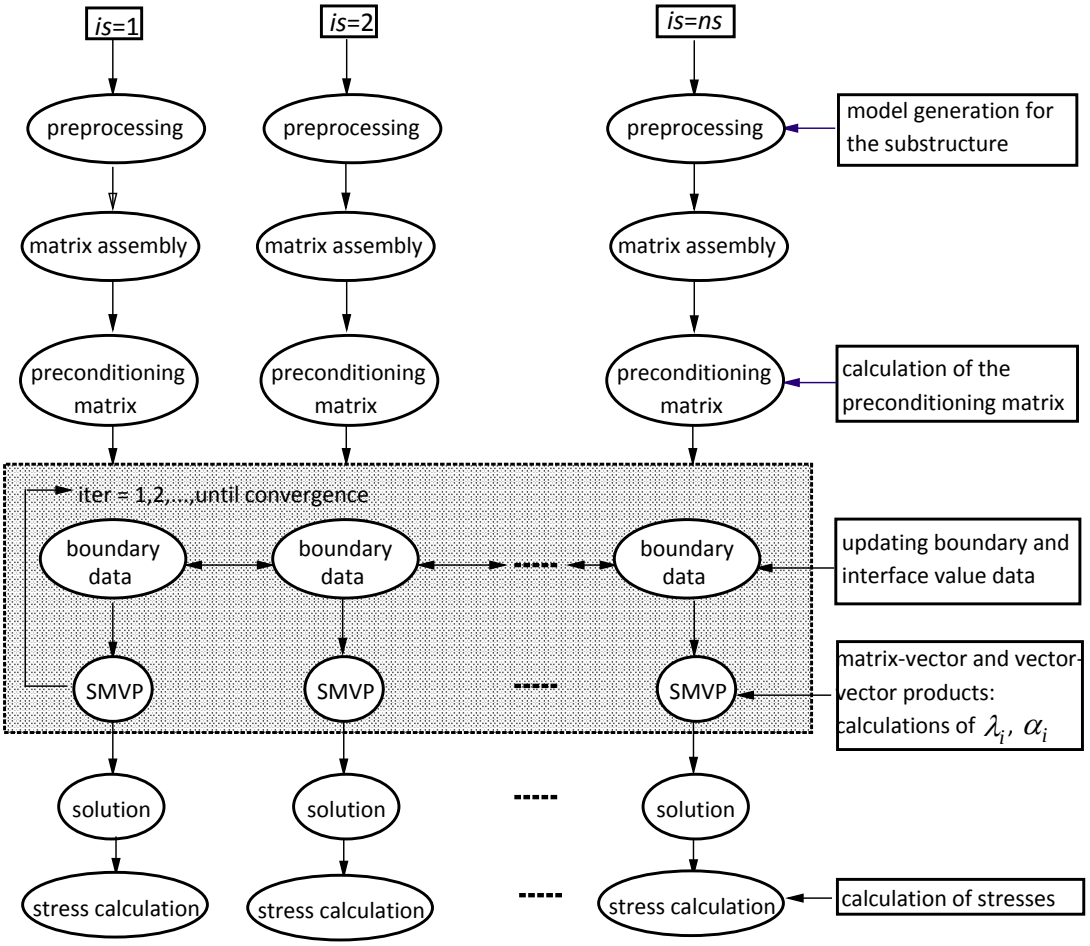
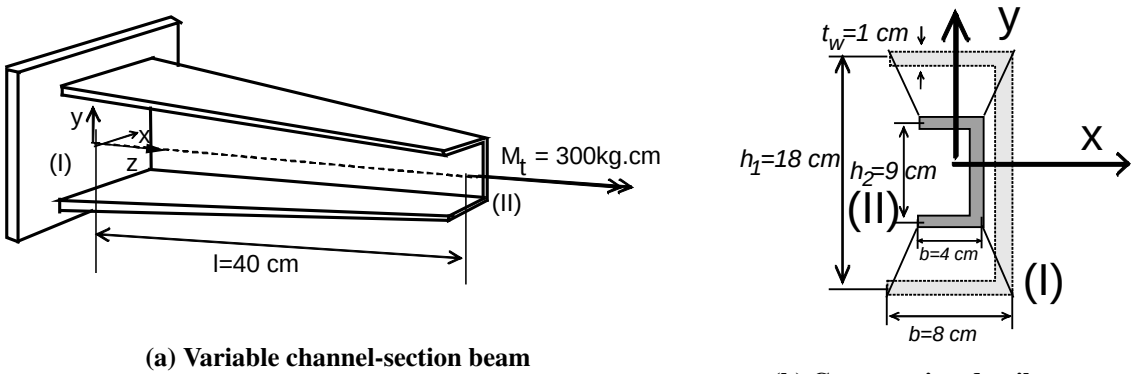


Figure 4. Flowchart of the SBS-based code.

of torsional, GI_t, and warping, EC_M, rigidities are presented. In that paper, the authors presented results obtained via a special BE formulation, and using the 4-noded quadrilateral shell and 10-noded tetrahedron solid finite elements available at the MSC/NASTRAN 4.0 program. In this study, this problem has been solved employed the 3-subdomain BE model shwon in Figure 6, which employs 88 8-noded quadrilateral boundary elements for subdomains Ω_1 and Ω_3 (flanges), and 160 boundary elements of the same type for subdomain Ω_2 (web). In Figure 6c, use of discontinuous elements is considered to simulate traction discontinuity between front and interface elements.

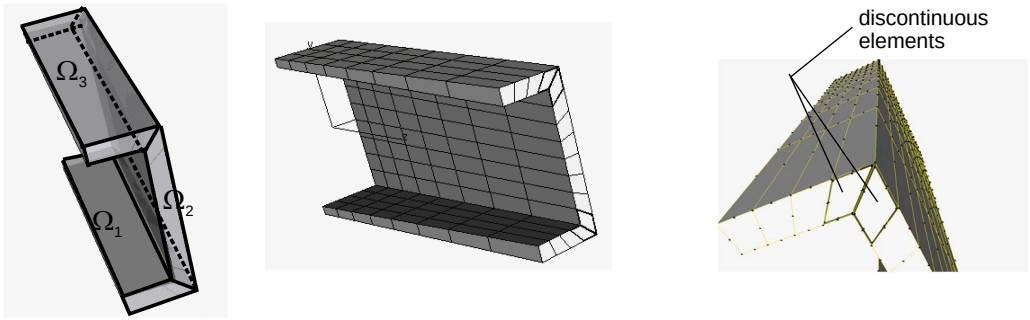
In the graphs in Figure 7, the results obtained with the proposed 3D BE-SBS method are confronted with those published in [18] (using 4-noded shell and 10-noded solid finite elements of the MSC/NASTRAN 4.0), and with the results obtained in this study using 4-noded shell finite elements of the SAP2000. It is observed that the MSC/NASTRAN maximal translation values are limited to $0.29986\,cm$ (see [18]), while the 3D BE-SBS and SAP2000 (employing shell finite elements) ones are limited to $0.40\,cm$. Thus, for the displacement amplitudes, an error of about 25% is verified. In addition, the performance of the solver is shown in the graph in Figure 8. As seen, the SBS preconditioning proposed played a relevant role in the convergence of the iterative processes. In fact, as consequence of the SBS preconditioning, the performance of the BiCG solver considerably increased, what is inferred from the strong residual decay as a function of the number of matrix-vector multiplications, nmv. On the other hand, without



(a) Variable channel-section beam

(b) Cross section details

Figure 5. Cantilever beam under torsion



(a) BE subdomains

(b) BE meshes

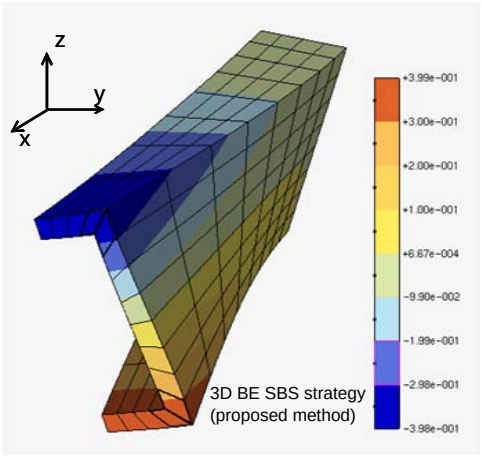
(c) Discontinuous elements

Figure 6. 3D Boundary-element model

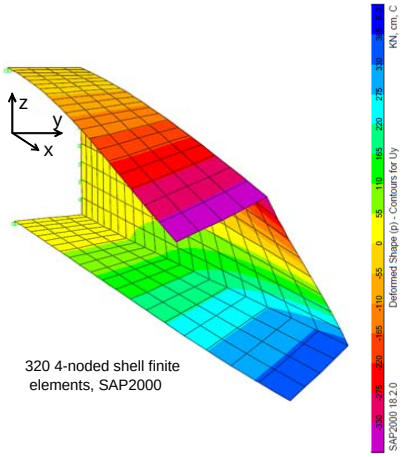
applying the SBS prenconditioner, the BiCGSTAB($l = 8$) solver had poor convergence behavior and did not overcome the BiCG one.

CONCLUSION

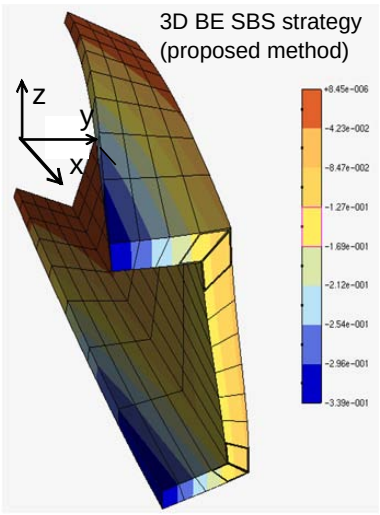
In this paper, ways to construct robust BE codes are proposed. In fact, robustness is verified by observing the solution performance of the BE-SBS algorithm by solving a beam with complex geometry description under torsion, which has also been analyzed by other authors and methods [18], including shell and solid finite elements available in the MSC/NASTRAN 4.0 and SAP2000. As seen from Figures 7 and 8, the BE-SBS method provides accurate results, and the performance of the embedded Krylov solvers, when employing the SBS-BD preconditioning, is excellent. As a conclusion, a good preconditioner may be even more relevant in the solver performance than a very complex solver as e.g the BiCGSTAB(l). In fact, in the realm of the finite arithmetic on the computer, rounding errors invalidate the theoretical convergence properties of Krylov solvers, so that developing reliable iterative solvers has been always an open question. The SBS-BD preconditioning, in the light of the present results and previous ones [5], has been a promising preconditioning option. Notice that using the BiCGSTAB(l) solver also requires defining the dimension l of the optimization subspace (usuallly $l \leq 8$), i.e. one still has an additional parameter to feed into the iterative process, depending on which convergence may not be attained. Indeed, no criterion for an optimal choice of l is available in the literature yet.



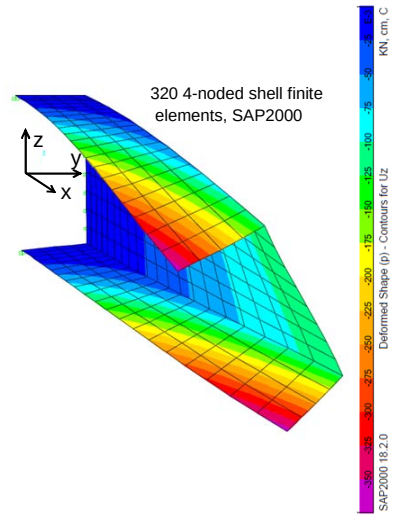
(a) Horizontal displacement (u_y)



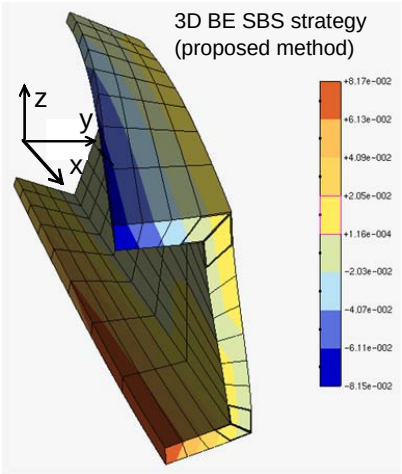
(b) Horizontal displacement (u_y)



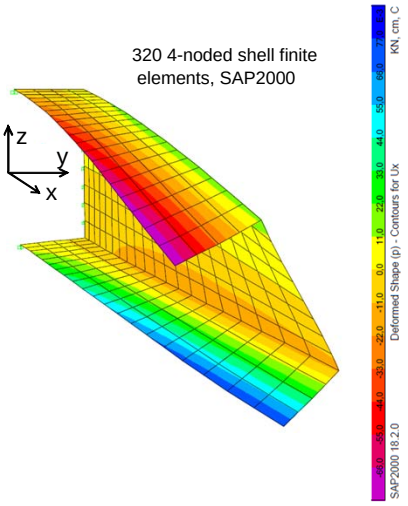
(c) Vertical displacement (u_z)



(d) Vertical displacement (u_z)



(e) Warping displacement (u_x)



(f) Warping displacement (u_x)

Figure 7. Displacement fields for the beam (3D SBS method $\times$ SAP2000)

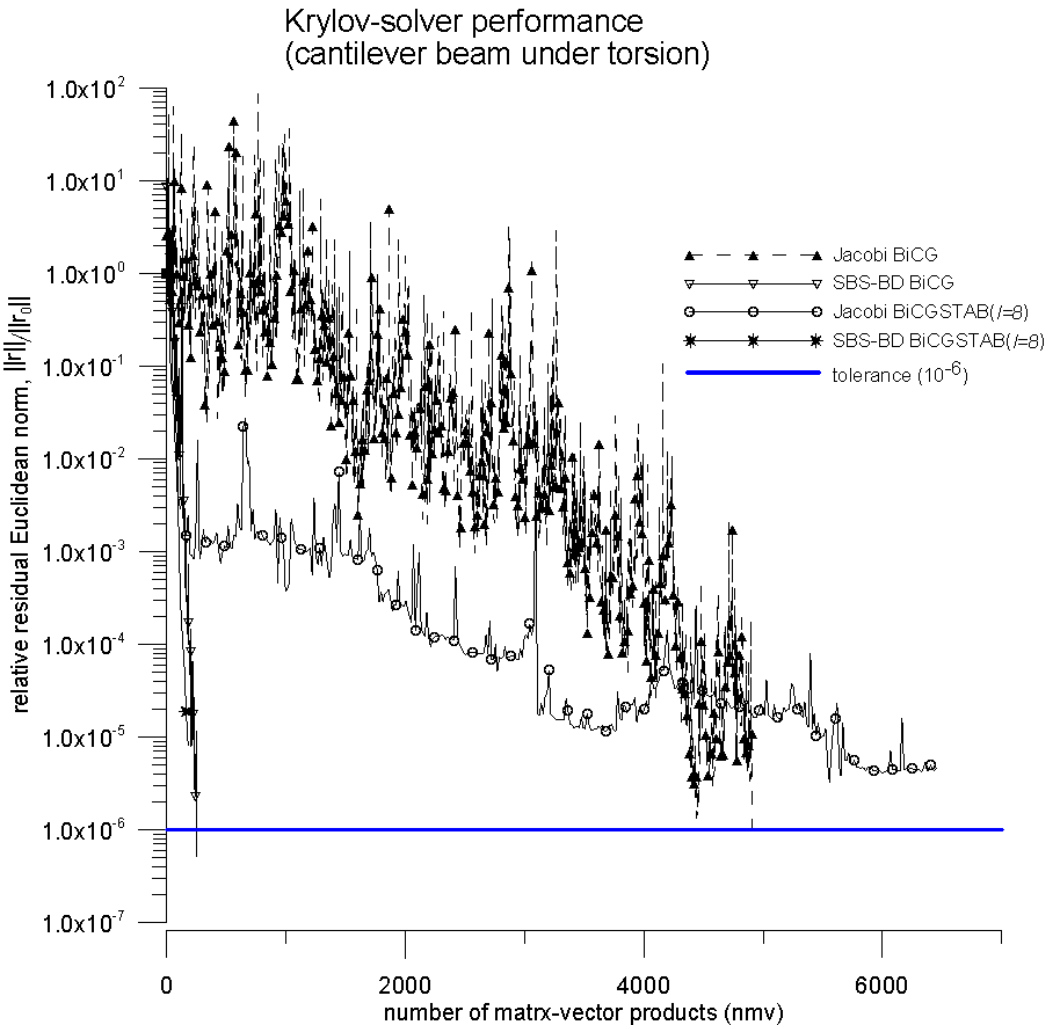


Figure 8. Jacobi $\times$ SBS BiCG and BiCGSTAB(*l*).

In general, the following three main advantages of the BE-SBS algorithm should be highlighted: (1) huge memory saving as the high sparsity of the coupled system matrix is perfectly treated, (2) fast convergence during the iterative solution process (brought about by applying the SBS-BD preconditioning), and (3) easy and straightforward implementation on parallel processing platforms, fundamental for solving real-life large-order problems.

ACKNOWLEDGEMENTS

This research has been sponsored by the Brazilian Research Council (CNPq), and the Coordination for Improvement of Higher-Education Personnel (CAPES).

References

[1] Araujo, F. C., Silva, K. I., Telles, J. C. F. 2006. "Generic domain decomposition and iterative solvers for 3D BEM problems." *Int. J. Numer. Methods Engrg.*, 68, 448-472.

[2] Araujo, F. C., and Gray, L J. 2008. "Evaluation of effective material parameters of CNT-reinforced composites via 3D BEM." *Computer Modeling in Engineering and Sciences,* 24, 103-121.

[3] Araujo, F.C., and Gray, L.J. 2008 "Analysis of thin-walled structural elements via 3D standard BEM with generic substructuring", *Comp. Mech.*, 41, 633-645.

[4] Araujo, F.C., d'Azevedo, E.F. , and Gray, L.J. 2010. " Boundary-element parallel-computing algorithm for the microstructural analysis of general composites." *Computers & Structures*, 88, 773-784.

[5] Araujo, F.C., d'Azevedo, E.F., and Gray, L.J. 2011. " Constructing efficient substructure-based preconditioners for BEM systems of equations." *Eng. Anal. Boundary Elements*, 35, 517-526.

[6] Benedetti, I., and Aliabadi, M.H. 2013. " Three-dimensional grain boundary formulation for microstructural modelling of polycrystalline materials." *Computational Materials Science*, 67, 249-260.

[7] Brebbia, C. A., Telles, J. C. F. , and Wrobel, L.C. 1984. *Boundary element techniques: theory and applications in engineering.* Springer-Verlag, Berlin.

[8] Chen, X.L., Liu, Y.J. 2004. "Square representative volume elements for evaluating the effective material properties of carbon nanotube-based composites." *Comput. Mat. Sci.*,29, 1-11.

[9] Chen, K. 1998. "On a class of preconditioning methods for dense linear systems from boundary elements", *SIAM J. Sci. Comput.* 20, 684-698.

[10] Chen, K. 2005. *Matrix Preconditioning Techniques and Applications*, Cambridge, UK, Cambridge University Press.

[11] Fletcher, R. 1976. "Conjugate gradient methods for indefinite systems", *Lecture Notes in Mathematics* 506, Spriger-Verlag, Berlin, pp. 73-89.

[12] Freund, R.W., and Nachtigal, N.M. 1991. "QMR: a quasi-minimal residual method for non-Hermitian linear systems", *Numer. Math.*, 60, 315-339.

[13] Hughes, T.J.R., Levit, I. , and Winget, L. 1983. "An element-by-element solution algorithm for problems of structural and solid mechanics", *Comput Methods Appl. Mech. Engrg.*, 36 (2), 241-254.

[14] Liu, Y. 1998. "Analysis of shell-like structures by the boundary element method based on 3-D elasticity: formulation and verification." *Int. J. Numer. Methods Engrg.*, 41, 541-558.

[15] Saad, Y., and Schultz, M.H. 1986. "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems", *SIAM J. Sci. Star. Comput.*, 7, 856-869.

[16] Saad, Y., van der Vorst, H.A. 2000. "Iterative solution of linear systems in the 20th century", *Journal of Computational and Applied (Mathematics*, 123, 1-33.

[17] Saad, Y. 2003. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia.

[18] Sapountzakis, E., and Mokos, V. 2004."Nonuniform torsion of bars of variable cross section", *Computers and Structures*, 82, 703-715.

[19] Sleijpen, G.L.G., and Fokkema, D.R. 1993. "BICGSTAB(L) for linear equations involving unsymmetric matrices with complex spectrum", *Electron. Trans. Numer. Anal.*, 1, 11-32.

[20] Sonneveld, P. 1989. "CGS: a fast Lanczos-type solver for nonsymmetric linear systems", *SIAM J. Sci. Statist. Comput.* 10, 36-52.

[21] van der Vorst, H.A. 2003. *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press.