THE USE OF THE SUBREGION-BY-SUBREGION ALGORITHM FOR DEVELOPING GENERAL PARALLEL-PROCESSING BOUNDARY-ELEMENT CODES

F. C. Araújo, fleio@pq.cnpq.br
Dept. Civil Eng., UFOP, 35400-000 Ouro Preto-MG, Brazil

E. F. d’Azevedo, dazevedoef@ornl.gov
L. J. Gray, graylj1@ornl.gov
Computer Science and Math. Div., ORNL, Oak Ridge, P.O. Box 2008, USA

Abstract. In this work, a generic subregion-by-subregion (SBS) algorithm is employed to develop general BE parallel codes. In this algorithm, the interactions between the subdomains are taken into account only during the solution of the system by a Krylov iterative method. Thereby, significant reduction of memory and CPU-time consumption is achieved as the global system matrix is not explicitly treated. On the other hand, special integration procedures for calculating nearly-strongly-singular integrals make the use of discontinuous boundary elements (in which quasi-singular integrals occur) possible. In fact, discontinuous elements are very useful for establishing BE models for complex heterogeneous domains. A matrix-copy option, useful for modeling systems with repeated parts, as identical fiber reinforcements, is also available. To verify the performance of the code, the 3D microstructural analysis of carbon-nanotube-reinforced composites (CNT composites) is considered. Particularly, mechanical properties of composites are measured. The representative volume elements (RVEs) adopted consist of carbon-nanotubes (shell-like elements) coupled with a polymeric material matrix.

Keywords: 3D standard BE formulation, parallel processing, CNT-based composites, thin-walled elements, subregion-by-subregion technique

1. INTRODUCTION

Although, today, the Finite-Element Method (FEM) is still the tool of choice in engineering analysis of structures and solids, its application to thin-walled solids and composites has been accompanied with a series of issues as element-distortion sensitivity, locking phenomena, and the non-fulfillment of stress continuity between layers or at matrix-fiber interfaces. Especially in the case of composites, the mesh generation itself is a bottleneck in finite-element (FE) analysis. To escape these difficulties, in recent works, the direct application of 3D standard boundary-element formulations has then been considered as an alternative to solve general composites and thin-domain problems (Chen and Liu, 2005; Araújo and Gray, 2008a; Araújo et al., 2007). Besides advantages as high accuracy, fulfillment of radiation conditions, and easier mesh generation, the Boundary Element Method (BEM) also presents the following interesting characteristic: it is derived from the exact integral representation of the problem response and does not require any interelement compatibility (in the FE sense) for assuring solution convergence. This actually allows more flexibility for generating boundary-element models as long as the integrals involved are accurately evaluated. Indeed, this is the basis of discontinuous boundary elements, very useful for the BE subregion-by-subregion (BE-SBS) algorithm (Araújo and Gray, 2008a), considered in this work for the development of the parallel code.

In many of the works concerning the development of BE parallel codes (Natarajan and Krishnaswamy, 1996; Song and Baddour, 1997; Cunha et al., 2004), either for symmetric multiprocessor (SMP) or massively parallel processor (MPP) architectures, the parallelism has been based on different ways to generate and to scatter the global boundary-element (BE) system of equations onto the available processors, its solution, in most cases, being carried out by applying available high-performance packages as LAPACK or ScalAPACK. Unlike these works, Kamiya et al. (1996) used a domain decomposition method (DDM) for solving 2D potential problems. They directly scattered the subdomain systems onto the processors and got the solution for the whole problem from the independent solution of each subdomain, wherein iterative schemes were used to introduce the coupling conditions. In Lu and Wu (2005), a DDM-based strategy is also considered to solve 2D elasticity problems with cracks. This procedure allows the independent assembling of the subdomain matrices, and is based on the condensation of the problem response to the interface tractions. In fact, this strategy may be cumbersome and time-consuming for 3D problems with complex composite morphology.

In the present paper, the parallel version of the BE subregion-by-subregion (BE-SBS) algorithm (Araújo et al., 2006; Araújo et al., 2007), a generic non-overlapping domain decomposition method, is presented. Besides being a fundamental technique in BE formulations, substructuring techniques are a spontaneous way to develop parallel codes, irrespective of the computer architecture. However, differently from displacement-based FE formulations, wherein one should never worry about traction discontinuity at the element corners or edges, BE formulations are mixed formulations and require simulating traction discontinuity at some corner/edge nodes of the subdomains. Certainly, this
is a fact that makes the modeling with subregions, in case of solids with complex internal geometries, a somehow tedious task. This was verified in Araújo et al. (2004), where continuous boundary elements are used to model 3D frequency-dependent elastodynamic problems. To get rid of this issue, discontinuous boundary elements have been adopted (Araújo et al., 2006; Araújo and Gray, 2008a, 2008b). Doing so, coupling conditions can be directly imposed, and BE-subdomain models are then a lot easier generated.

The other issue related to BE-subdomain algorithms (in parallel or serial versions) is how to optimally deal with the highly sparse resulting matrices. Kamiya et al. (1996) proposed iterative coupling procedures that perfectly treat the matrix sparsity but are not reliable concerning convergence. In Lu and Wu (2005), the coupling conditions are directly introduced, but condensing the system unknowns to the interface tractions is awkward, requires additional memory beyond that necessary for allocating the isolated subsystems, and may be time-consuming for complex models. In this work, the BE-SBS algorithm (Araújo et al., 2006; Araújo et al., 2007) is adopted. Employing some iterative solver, a solution strategy for general coupled problems is derived wherein no explicit global matrix have to be assembled, and only memory space for strictly allocating the subregion subsystems is needed. In this work, a simple diagonal-preconditioned Bi-CG solver is applied. However it is emphasized that the coupling conditions are directly enforced. Structured matrix-vector product (SMVP) and matrix-copy options are also implemented to increase the efficiency the code (Araújo and Gray, 2008b). 3D simulations of CNT-based composites are carried out to show its performance.

2. THE PARALLEL-PROCESSING ALGORITHM

The BE parallel code is based on the BE-SBS algorithm detailed in previous papers (Araújo et al. Araújo et al., 2007; Araújo et al., 2006). This algorithm considers a substructuring technique (non-overlapping domain decomposition method, DDM), and makes use of iterative solvers, similarly as done in element-by-element-based (EBE-based) finite-element formulations, to solve the global BE system of equations without explicitly assembling it. In general, after the boundary conditions have been introduced at each BE subregion separately, a set of \( n_s \) algebraic systems of equations given by

\[
\sum_{j=1}^{i-1} \left( H_{ij} u_j - G_{ij} p_j \right) + A_n x_j + \sum_{j=1}^{n_s} \left( H_{ij} u_j + G_{ij} p_j \right) = B_n y_j, \quad i = 1, n_s,
\]

where \( n_s \) is the number of subregions, has to be solved by enforcing continuity and equilibrium conditions at the interfaces:

\[
\begin{align*}
  u_j &= u_{ji} \quad \text{at } \Gamma_{ij}, \\
p_j &= -p_{ji} \quad \text{at } \Gamma_{ij},
\end{align*}
\]

In Eq. (1), \( H_{ij} \) and \( G_{ij} \) denote the usual BE matrices obtained for source points pertaining to subregion \( \Omega_i \) and associated respectively with the boundary vectors \( u_j \) and \( p_j \) at \( \Gamma_{ij} \). Note that if \( i \neq j \), \( \Gamma_{ij} \) corresponds to the interface between \( \Omega_i \) and \( \Omega_j \), which denote the \( i \)-th and \( j \)-th subregion respectively; \( \Gamma_{ij} \) is the outer boundary of \( \Omega_i \). The global system in Eq. (1) is then conveniently solved by applying an iterative solver. Here particularly, the diagonal-preconditioned biconjugate gradient (J-BiCG) solver.

As in the BE SBS algorithm there is no overlapping of coefficients belonging to edges or corners shared by different subregions, as it happens in finite-element models, the data structure in Eq. (1) does not need any further optimization. All zero blocks present in the highly-sparse global system matrix are perfectly excluded. Besides, the following techniques/strategies are especially important for increasing the efficiency of the BE-SBS-based code: discontinuous boundary elements, structured matrix-vector products (SMVP), special integration quadratures, and the matrix-copy option. In the references Araújo et al. (2006), Araújo et al (2007), Araújo and Gray (2008a), and Araújo and Gray (2008b), the BE SBS algorithm is thoroughly described.

In Fig. 1, the flowchart of the BE-SBS-based parallel code is presented, wherein it is assumed that \( k \) processes is considered. In fact, as in the BE SBS algorithm the subdomains are independently treated during the entire analysis, its implementation for running in a parallel-processing platform is immediate. Assembling the algebraic systems needs no information from other processes. Only during its solution, communication between the processes is needed for updating the boundary values in all subregions (Fig. 2).
3. APPLICATIONS

The performance of the BE-SBS-based parallel code detailed above has been observed by determining engineering constants for the CNT-based composites shown in Fig. 3. In all representative volume elements (RVEs), constructed by coupling together single unit cells with dimensions $l_1 = 10 \text{ nm}$, and $l_2 = l_3 = 20 \text{ nm}$ (see Fig. 3a), the following phase constants, adopted in Chen and Liu (2004), are considered:

**CNT:**

\[
E_{\text{CNT}} = 1,000 \frac{nN}{nm^2} \quad \text{(GPa)}; \quad \nu_{\text{CNT}} = 0.30 ,
\]

**Matrix:**

\[
E_m = 100 \frac{nN}{nm^2} \quad \text{(GPa)}; \quad \nu_{\text{CNT}} = 0.30 .
\]
The long CNT fibers are geometrically defined by cylindrical tubes having outer radius \( r_0 = 5.0 \text{ nm} \), inner radius \( r_i = 4.6 \text{ nm} \), and length \( l_f = 10 \text{ nm} \) (equal to the RVE thickness). When needed, discontinuous boundary elements are automatically generated by shifting the nodes interior to the elements a distance of \( d = 0.10 \) (measured in the natural coordinate system). The matrix-copy option is also conveniently considered to replicate physically and geometrically identical subdomains. The boundary element adopted is an 8-node quadrilateral one, and the tolerance for the iterative solver (J-BiCG) is taken as \( \zeta = 10^{-5} \). The analyses were carried out at the ORNL (Oak Ridge National Laboratory) institutional cluster (OIC), consisting of 80 usable nodes, each one having Dual Intel 3.4GHz Xeon EM64T processors, 4GB of memory, and dual Gigabit Ethernet Interconnects.

In this paper, used memory and CPU time/niter bounds are employed to measure the parallel-processing scalability. The used memory bound is the total memory allocated for the real-valued array at the processor with largest amount of allocated memory, and the CPU time/niter bound is the solver CPU time per iteration at the slowest processor. In fact, also because of the communication between processes needed during the solution phase, the assembly CPU time for large-order problems is much less significant than the solution CPU time. Thus, only the latter one is considered in the parallel-performance analysis. As a sample, the curves showing the parallel-processing performance for one of the analyses carried out are presented in Fig. 4, actually only for the largest model (with \( 6 \times 6 \) unit cells) under strain state 2 (in Araújo and Gray, 2008b the germane strain states are defined).

In Table 1, important model data are given. In Table 2, the engineering parameters obtained employing the present code are confronted with results given by Liu and Chen (2004), and estimated by the rule of mixture (see Araújo and Gray, 2008b; Chen and Liu, 2004; and Hyer, 1998). As seen from Table 2, values estimated by the rule of mixture and by refined 3D FE models (Liu and Chen, 2004) are in very good agreement with the ones calculated with the present method. No significant change in the values is also observed as a function of the number of unit cells per RVE.
As one sees, the scalability of both CPU-time (see Fig. 4a) and memory-use (see Fig. 4b) is very good. In fact, more interprocessor communication and less processor load is expected when the number of processors increases. This should then explain the weakening of the processing speed-up after 36 processors.

### 4. CONCLUSIONS

In this paper, a general 3D BE parallel code has been derived from a robust BE-SBS technique. Particular applications of the code concern the evaluation of effective engineering constants for 3D CNT-reinforced composites. First, it is observed that as a consequence of the special quadratures available in the code, discontinuous and disproportionate boundary elements can be employed. In this way, the modeling of complex coupled solids, as composites, is greatly simplified. In addition, the matrix-copy option, which avoids the repeated mesh generation and calculation of coefficient matrices for identical substructures, considerably facilitates the modeling of very complex periodic composites. In the particular case of the applications shown above, no efficiency gain has been actually observed during the assembly phase as the corresponding CPU-time measurements were insignificant compared to the solver CPU time, which is dominant. However, for large identical subregions, this option will increase the computational efficiency. Moreover, for complex composites, boundary-integral-based models are simpler to generate than volume-based ones. Thus, the strategy proposed is believed to be very convenient for analyzing general composites.
A contribution of this study is the proposal of a general strategy for developing parallel-processing BE codes, readily applicable to any BIE-based methods. We notice that the algorithm proposed presents the following interesting general characteristics: (1) the BE models are independently generated, stored, and manipulated (no explicit global matrix assembly takes place), (2) no variable condensing is carried out, avoiding then the calculation of Schur complements, (3) the interface conditions are directly imposed, avoiding then the use of some iterative strategy, (4) discontinuous boundary elements are used to make the generation of coupled models easier, (5) an iterative (Krylov) solver is employed, (6) the high sparsity of the system is perfectly exploited. Obviously, as the models are independently stored, the memory-use scalability of the code is excellent, as we see from the results in the previous section. On the other hand, if the number of processors is incremented, the interprocessor communication will be more intense, decreasing then the processing speedup after a certain critical number of processors.

In this work, indeed focused on the BE-SBS-based parallel code, no special attention has been properly paid to the Krylov solver itself. As noted, just a plain diagonal-preconditioned BiCG solver has been employed. In fact, it is known that this particular solver presents irregular convergence behavior, sometimes even not converging, depending on the system-matrix spectrum. Anyway, considering all the development brought about on iterative solvers and preconditioning techniques in the last two decades (Barett et al., 1994), we do believe that the BE-SBS algorithm is the optimal way to solve complex coupled BE models, and a promising alternative to develop general BE parallel codes, accounting for scalability of memory requirements and processing time. In future steps of this research, other Krylov solvers, e.g. BiCG-stab(m), and advanced preconditioners should be incorporated into the parallel code.

5. REFERENCES


6. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.

ACKNOWLEDGEMENTS

This research was sponsored by the Office of Advanced Scientific Computing Research, U.S. Department of Energy under contract DE-AC05-00OR22725 with UT-Battelle, LLC, the Brazilian Research Council (CNPq), and by the Research Foundation for the State of Minas Gerais (FAPEMIG), Brazil.