Computers and Structures 88 (2010) 773–784

Contents lists available at ScienceDirect

Computers and Structures

journal homepage: www.elsevier.com/locate/compstruc

Boundary-element parallel-computing algorithm for the microstructural analysis of general composites

F.C. Araújo [a,*], E.F. d'Azevedo [b], L.J. Gray [b]

[a] Dept. Civil Eng., UFOP, 35400-000 Ouro Preto-MG, Brazil
[b] Computer Science and Math. Div., ORNL, Oak Ridge, P.O. Box 2008, USA

ARTICLE INFO

Article history:
Received 1 September 2009
Accepted 8 March 2010
Available online 26 March 2010

Keywords:
3D standard BEM
Parallel computing
CNT-based composites
Thin-walled elements
Subregion-by-subregion technique

ABSTRACT

A standard continuum-mechanics-based 3D boundary-element (BE) algorithm has been devised to the microstructural modeling of complex heterogeneous solids such as general composites. In the particular applications of this paper, the mechanical properties of carbon-nanotube–reinforced composites are estimated from three-dimensional representative volume elements (RVEs). The shell-like thin-walled carbon nanotubes (CNTs) are also simulated with 3D BE models, and a generic subregion-by-subregion (SBS) algorithm makes the microstructural description of the CNT–polymer systems possible. In fact, based on this algorithm, a general scalable BE parallel code is proposed. Square and hexagonal fiber-packing patterns are considered to simulate the 3D composite microstructures.

1. Introduction

With the increasing capability of digital computers, nowadays reaching the exascale era, more accurate descriptions of engineering systems are feasible. For example, if three decades ago a simple 3D continuum finite-element (FE) modeling was unrealistic for practical design purposes in structural engineering, today, one has even talked about atomic-level simulations of matter [1]. Particularly important in this context has been the development of the *microstructural mechanics* [2,3], which, starting from the microstructural conception of materials, allies continuum-mechanics (CM) principles and computational methods to assess their response to given input functions. In fact, an approach able to capture three-dimensional deformation mechanisms at the constituent level in a composite, generally having a complex phase morphology, is the only reliable way to measure their effective physical properties [4,5]. Homogenization strategies for estimating them or calculating their bounds are limited to composites having non-complex microstructure and not involving highly-contrasting physical properties [6].

Given a realistic microstructural representation of the material, a boundary-value problem has to be solved. In many works, the finite-element method (FEM) along with high-performance computing (HPC) techniques has been applied to deal with the usually huge resulting models [3,5]. Nevertheless, it has been found out that, in general composites, the presence of thin-walled components, such as fibers or shell-like layers, poses special difficulties to the accurate prediction of stresses and strains via FEM. In addition to usual numerical problems connected with plate/shell formulations (e.g. element-distortion sensitivity and locking phenomena), the simulations can produce incorrect description of the zigzag-like displacement function along the thickness, and the stress continuity between layers or at matrix–fiber interfaces will not be fulfilled. This leads to unacceptable stress results in the composite [7–9]. In [10], an element-layering technique based upon conventional shell finite elements is proposed to increase global and local response accuracy at laminated tile–reinforced composites. However, as 2D formulations are adopted, local details of the field variables may not be reconstituted with high-fidelity in complex cases. On the other hand, modeling thin domains by using 3D brick finite elements may require a large prohibitive number of elements to arrive at algebraic systems with acceptable condition numbers. Moreover, in the case of composites, mesh generation is also a bottleneck in FE analysis.

To avoid these difficulties, the direct application of 3D standard boundary-element formulations has been employed to model general composites and thin-domain problems [11–13]. 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 allows more flexibility for generating boundary-element models as long as the integrals involved are accurately evaluated. Indeed, this is the basis

* Corresponding author. Tel.: +55 31 3559 1468; fax: +55 31 3559 1548.
E-mail addresses: fcelio@pq.cnpq.br (F.C. Araújo), dazevedoef@ornl.gov (E.F. d'Azevedo), graylj1@ornl.gov (L.J. Gray).

0045-7949/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compstruc.2010.03.001

of discontinuous boundary elements, very useful for the BE subregion-by-subregion (BE-SBS) algorithm [12], considered in this work for the microstructural modeling of composites and development of the parallel code.

1.1. Parallel computing

Computing power has grown exponentially over recent decades [14], and computational methods and techniques have evolved accordingly to adjust to the computer architectures. Thus, it has been possible to take into account more and more analysis options (physical details) in the virtual simulations in engineering and science. Among all of the improvements brought about, the parallelization of codes has been one of the most important, speeding up and also extending memory resources for dealing with large-order problems. Vector computers were first developed, allowing the automatic processing of data sets as single units (mathematical vectors). Later, SMP (symmetric multiprocessor) and MPP (massively parallel processors) architectures have been introduced. SMP machines, usually expensive, possess a shared memory, and codes for them are developed by adopting multithreading programming techniques, e.g. based on OpenMP standard [15]. MPP architectures consist of a series of computing nodes, each one with its own main memory and one or more processors, interconnected by a network (distributed memory). To write codes for the latter architectures, MPI (message passing interface [16,17]), which allows for the interprocess communication, has been conveniently used. MPP systems present the following advantages: they are much cheaper than SMP ones, and, most importantly, they may overcome the inherent hardware limitations of SMP architectures, namely saturation of the bus when the number of processors increases [17]. However, programming techniques for distributed-memory systems are the most difficult of all parallel programming techniques, definitely depending on the program developer for efficiently circumventing load-balance and communication overhead issues.

For BEM, the first works on parallel computing date from the 80s [18,19], with codes for solving potential problems on distributed array processor (DAP) architectures developed. In 1994, Kane [20] presented a survey on strategies known at that time for developing BE parallel codes. Later, a series of papers on the parallelization of both assembly and solution of the BE algebraic systems of equations appeared, mainly focusing distributed-memory computing platforms. In [21], the 3D Laplace and Helmholtz equations are treated, and the parallel algorithm proposed basically consists of the cyclic mapping of the coefficient matrix onto the computing nodes according to a suitable block partitioning. A problem in this approach is to define the block size (dependent on the processor-network architecture) leading to an optimum parallel performance. For working with this matrix partitioning/mapping, a LU-based parallel solver is applied. In [22], the coefficient matrix is generated in row blocks and cyclically scattered among the nodes. In addition to a simple Gaussian elimination solver, the CGS method [23,24] is also applied to solve the row-wise partitioned BE system of equations arising from 3D potential problems. In the same year (1997), following a work by Cwik et al. [25], the CARLOS-3D code [26] was developed to solve large-order three-dimensional electromagnetic problems in massively parallel computer systems via BEM. In this code, the computational load imbalance is minimized by mapping the coefficient matrices onto the nodes through the torus-wrap decomposition algorithm, and a LU-based parallel solver with column pivoting was employed. Despite its complexity, the torus-wrap mapping has shown to be superior to column or row mappings in terms of communication volume during the LU decomposition [27]. This mapping has also been adopted by Ingber and Papathanasiou [28] for developing a BE code for the micromechanical analysis of short-fiber–reinforced composites.

In [29], a parallel implementation of a multiple-reciprocity-based BE formulation (MR-BEM) for 2D potential problems is presented. Again, a block-cyclic distribution among the computing nodes is effected, and the ScaLAPACK library [30], which contains highly optimized LU-based parallel factorization routines for dense matrices, is directly applied to solve the resulting system of equations. More recently, Cunha et al. [31] also directly used LAPACK and ScaLAPACK libraries for the parallel solution of BE systems in shared- and distributed-memory architectures, respectively. The algebraic systems have been parallelized by assuming that each pair of rows (for 2D elastostatics) are independent of each other and can be concurrently generated. This fact has also been used for defining the matrix blocks adopted by the ScaLAPACK library (instead of 2D block-cyclic distribution) in their code version for distributed-memory architectures.

Papers exploiting parallelism in fast BE methods, which bases upon matrix-compression techniques as the fast multipole method (FMM) or precorrected-FFT, have also been recently published [32–34]. The main idea of these methods is to replace the matrix–vector products involved in the iterative solver by accurately reasonable and computationally efficient approximations. The source-point-related near-field contributions are evaluated in conventional way while the far-field contributions are approximated (e.g. by using series expansion). This procedure significantly reduces memory and operation requirements; ans $O(n\log n)$ or even an $O(n)$ algorithm may be attained, where n is the number of degrees of freedom of the model. In [32], a multithreading technique is considered to parallelize a precorrected-FFT-based BE code for 3D electrostatics and elastostatics at shared-memory architectures, while in [34], a FMM-based BE formulation for distributed-memory architecture is developed. In the latter paper, 3D fiber–reinforced composites have been analyzed. In [33], vectorization, multithreading, and multiprocessing approaches for a FMM-accelerated BE formulation are discussed. Applications to electrostatic problems are discussed. In all fast BE implementations cited above, the GMRES [23] solver has been applied.

In fact, in all of the works mentioned above, concerning either SMP or MPP architectures, the parallelism was exploited based on different ways to generate and to scatter the global BE system of equations onto the available processors. Its solution, in most cases carried out by applying available high-performance packages as LAPACK or ScaLAPACK, benefited then from the special data structures adopted in each particular implementation. Unlike these works, Kamiya et al. [35,36] used a non-overlapping Schwarz domain decomposition method (DDM), also termed substructuring method, for solving 2D potential problems. This work directly scattered the subdomain systems onto the processors and obtained the solution from the independent solution of each subdomain. Iterative schemes were used to introduce the coupling conditions. An issue in their formulation is how to specify the corresponding iteration parameters so as to assure convergence of the process. In [37], 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. Schur-like complements must be calculated, and this can be time-consuming for 3D problems with complex morphology. Moreover, the generalization of the procedure for a generic number of subregions seems to be cumbersome, requiring memory space beyond that necessary for the subregion matrices.

1.2. Present work

In this paper, the parallel version of the BE subregion-by-subregion (BE-SBS) algorithm [38,13], a generic non-overlapping domain decomposition method, is presented. Substructuring techniques

are a natural way to develop parallel codes, irrespective of the computer architecture. However, as opposed to displacement-based FE formulations, wherein traction discontinuity at the element corners or edges is not a concern, BE formulations are mixed and require dealing with traction discontinuity at the subdomain boundaries. In the case of solids with complex internal geometries, subregion modeling is therefore a "tedious" task (quoting Lou et al. [39]). This was demonstrated in Araújo et al. [40], where continuous boundary elements are used to model 3D frequency-dependent elastodynamic problems. Discontinuous boundary elements avoid this issue, but require special integration algorithms for the quasi-singular integrals [38,12,41]. This approach has been adopted herein. In this algorithm, coupled nodes across interfaces boundaries are automatically identified, and the corresponding coupling conditions directly imposed.

The other crucial 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. [35,36] proposed iterative coupling procedures that perfectly treat the matrix sparsity but are not reliable concerning convergence. In [37], 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. A likely better way to proceed is find a way to optimize the memory requirements as a function of the position of the nonzero blocks in the global matrix, as is classically done for FE models. A parallel strategy along these lines was used by Kane about 15 years ago [20]; to exploit the matrix sparsity, iterative solvers, which do not transform the coefficient matrix, were applied.

In the BE-SBS algorithm [38,13] adopted in this work, ideas originated in FE formulations are also exploited, specifically the element-by-element (EBE) technique [42]. Observing that a boundary-element subregion is comparable to a single finite element, and employing an iterative solver, a solution strategy for general coupled problems is derived. A global matrix is not explicitly assembled, only memory space for the subregion subsystems is needed, and the coupling conditions are directly (not iteratively) enforced. In this work, a simple diagonal-preconditioned Bi-CG solver is applied, although superior Krylov subspace solvers and preconditioners are known [43]. To make the BE-SBS algorithm still more efficient, structured matrix–vector product (SMVP) and matrix–copy options have been implemented. The former option reduces the solver CPU time per iteration while the latter one speeds up the matrix assembly in case of (many) identical subregions (e.g. identical fibers in a composite) [41].

Carbon-nanotube–reinforced composites (CNT composites) have been examined using the parallel BE-SBS algorithm. Considering the dimensions of the physical systems, the length scale being the nanoscale, molecular dynamics (MD) formulations should be applied. However, as MD-based analyses are limited to very small specimens, over very short analysis times, continuum-mechanics (CM) formulations are very useful. Numerical and physical experiments have shown that CM-based modeling can provide a satisfactory description of responses at nanotubes/solids [44–49]. Chen and Liu [50] have applied CM-based formulations to study CNT–reinforced composites, employing 2D and 3D FE models to extract engineering material parameters from representative volume elements (RVEs) for different fiber-packing patterns.

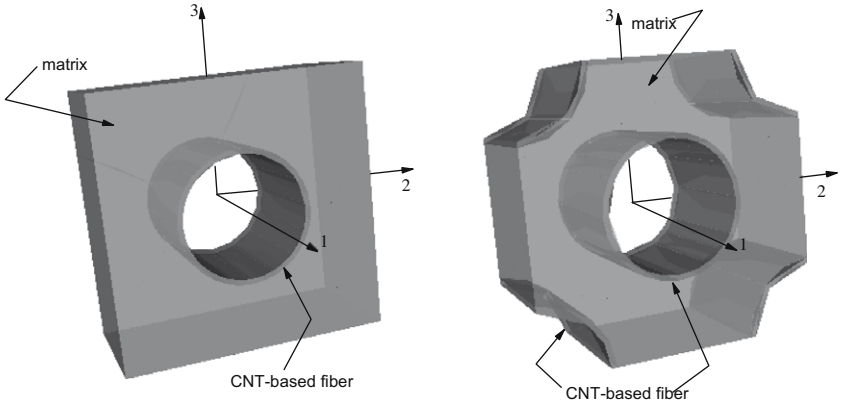
To verify the parallel performance of the BE-SBS algorithm proposed in this paper, 3D simulations of CNT composites based on square-packed and hexagonal-packed fiber arrays have been successfully carried out.

2. 3D RVEs for fiber–reinforced composites

In general, for predicting engineering properties of fiber–reinforced composite materials on the microstructural level, specified fiber-packing patterns for idealizing the fiber smearing inside the matrix material are adopted. To be more realistic, random fiber distribution should be taken into account. In the particular applications of this study, however, all RVEs base on square-packed or hexagonal-packed arrays (see Fig. 1) containing long or short fibers, a single or several cells. By evaluating displacements and tractions on the surfaces of the specimens for specific loadings, the effective engineering properties of composites can then be predicted. Below, the corresponding numerical experiments involved are described.

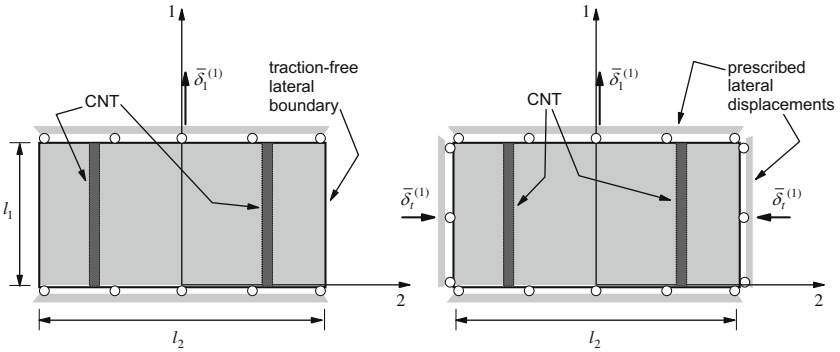
2.1. Predicting E_1,υ_{12} and υ_{13}

For predicting these properties, the specimen is stretched (or shortened) in the 1 direction (fiber direction) while, on the lateral surfaces perpendicular to the 2 and 3 directions, the surrounding medium (*in-situ* boundary conditions) is simulated by allowing them to move freely and to change length as long as they remain straight and free of any net force [51]. Here the following strategy is adopted to enforce these boundary conditions. First, the transverse displacements, $\delta_2^{(1)}$ and $\delta_3^{(1)}$, happening as a function of a prescribed axial displacement $\overline{\delta}_1^{(1)} = 1$, are calculated. Then, a second analysis for the prescribed displacements $\overline{\delta}_1^{(1)} = 1$ and



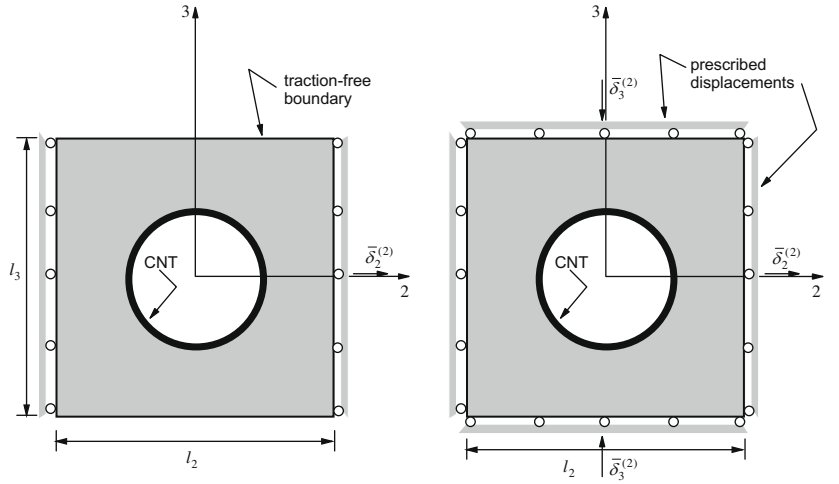
(*a*) Square-packed array (*b*) Hexagonal-packed array

Fig. 1. Single-cell RVEs with long fiber.



(*a*) 1st analysis (*b*) 2nd analysis

Fig. 2. Strain state 1.



(*a*) 1st analysis (*b*) 2nd analysis

Fig. 3. Strain state 2.

$\overline{\delta}_2^{(1)} = \overline{\delta}_3^{(1)} = \overline{\delta}_t^{(1)}, \overline{\delta}_t^{(1)}$ being a mean lateral displacement value (see Fig. 2), is carried out. In fact, considering $\overline{\delta}_2^{(1)} = \overline{\delta}_3^{(1)} = \overline{\delta}_t^{(1)}$ is convenient as transverse isotropy applies to the fiber-packed arrays at hand, and the minimum and maximum transverse displacements values at the lateral surfaces of the RVE are about the same. In this way, a zero-force condition is reasonably enforced. The superscripts indicate strain state 1, from which the material parameters E_1, υ_{12} and υ_{13} are determined. The following expressions, derived from basic strain–stress relationships, are employed for this purpose [41]:

$$E_1 = (\overline{\sigma}_1^{(1)} - \upsilon_{12}\overline{\sigma}_2^{(1)} - \upsilon_{13}\overline{\sigma}_3^{(1)})\left(\frac{l_1}{\overline{\delta}_1^{(1)}}\right), \tag{1}$$

$$\nu_{12} = \nu_{13} = -\left(\frac{\overline{\delta}_2^{(1)}}{l_2}\right)\left(\frac{l_1}{\overline{\delta}_1^{(1)}}\right), \tag{2}$$

where $\overline{\sigma}_i^{(1)}$ denotes the average stress over the RVE area perpendicular to the i direction, $i = 1, 2, 3$. As $\overline{\delta}_2^{(1)}$ and $\overline{\delta}_3^{(1)}$ are mean displacement values, the net forces on the lateral surfaces are not exactly zero; thus $\overline{\sigma}_2^{(1)}$ and $\overline{\sigma}_3^{(1)}$ are retained in Eq. (1).

2.2. Predicting E_2, υ_{23} and υ_{21}

The strain state 2 shown in Fig. 3, in which $\overline{\sigma}_1^{(2)} = 0$, is considered for evaluating these constants. Herein, the minimum and maximum displacement values at the 3 direction of the specimen, $\delta_3^{(2)}$, may be substantially different. Thus, for finding the $\overline{\delta}_3^{(2)}$ value corresponding to $f_3^{(2)} = 0$ (*in-situ* boundary conditions), the following strategy is adopted. First, $\overline{\delta}_2^{(2)} = 1$ is imposed, and $\delta_3^{(2)}$ is determined. If assumed that $f_3^{(2)}$ is a linear function of $\widetilde{\delta}_3^{(2)}$, prescribed displacement values at the 3 direction of the RVE, then $\overline{\delta}_3^{(2)}$ for which $f_3^{(2)} = 0$ is determined by (see Fig. 4)

$$\overline{\delta}_3^{(2)} = {}^{(b)}\widetilde{\delta}_3^{(2)} - \left(\frac{{}^{(b)}\widetilde{\delta}_3^{(2)} - {}^{(a)}\widetilde{\delta}_3^{(2)}}{{}^{(b)}f_3^{(2)} - {}^{(a)}f_3^{(2)}}\right){}^{(b)}f_3^{(2)}. \tag{3}$$

Previous numerical experiments [41] have shown that, as desired for the *in-situ* RVE boundary conditions, traction resultants $f_3^{(2)}$ approximately zero are obtained for prescribed $\overline{\delta}_3^{(2)}$ determined by relation Eq. (3). This fact hints then that an assumption of linearity between $f_3^{(2)}$ and $\widetilde{\delta}_3^{(2)}$ is appropriate.

In this work, the linear function for $f_3^{(2)}$ is constructed by taking $\widetilde{\delta}_3^{(2)} = \delta_{3,\min}^{(2)}$ and $\widetilde{\delta}_3^{(2)} = \delta_{3,\max}^{(2)}$, where $\delta_{3,\min}^{(2)}$ and $\delta_{3,\max}^{(2)}$ are the minimum and maximum displacements determined from the first analysis. With $\overline{\delta}_2^{(2)}$ and $\overline{\delta}_3^{(2)}$, an additional analysis is eventually carried out, from which E_2, υ_{23} and υ_{21} are calculated by expressions:

$$E_2 = (\overline{\sigma}_2^{(2)} - \upsilon_{23}\overline{\sigma}_3^{(2)})\left(\frac{\overline{\delta}_2^{(2)}}{l_2}\right)^{-1}, \tag{4}$$

$$\nu_{23} = -\left(\frac{\overline{\delta}_3^{(2)}}{l3}\right)\left(\frac{l_2}{\overline{\delta}_2^{(2)}}\right), \tag{5}$$

$$\nu_{21} = \nu_{12}\left(\frac{E_2}{E_1}\right). \tag{6}$$

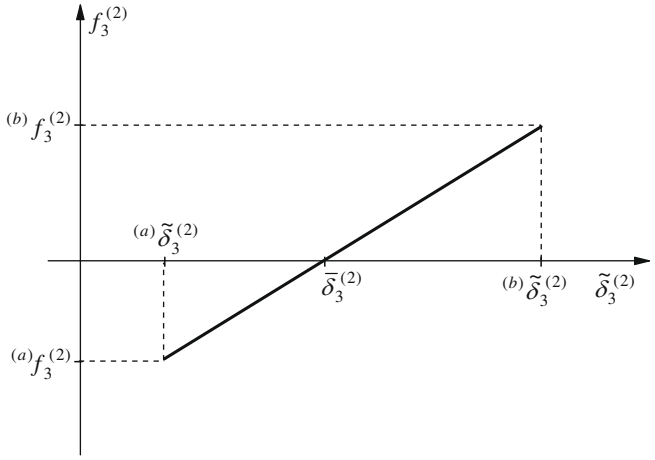


Fig. 4. Net force variation at the three direction, $f_3^{(2)}$.

Furthermore, assuming transverse isotropy in the 2–3 plane, the shear modulus G_{23} is directly calculated by

$$G_{23} = \frac{1}{2}\left(\frac{E_2}{1+\upsilon_{23}}\right). \tag{7}$$

Note that, in case of fully orthotropic materials, E_3, v_{31} and v_{32} can be evaluated from a third strain state similar to strain state 2, but with $\bar{\delta}_3^{(3)} = 1$ initially prescribed.

3. BE modeling and the parallel-processing algorithm

In this paper, a parallel version of the BE-SBS algorithm (Araújo et al. [13,38]) is implemented and applied to analyze the 3D RVEs. This algorithm is based on 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 [42], to solve the global BE system of equations without explicitly assembling it.

In general, after the boundary conditions have been introduced at each subregion separately, a set of n_s algebraic systems of equations given by

$$\sum_{j=1}^{i-1}(\mathbf{H}_{ij}\mathbf{u}_{ji} - \mathbf{G}_{ij}\mathbf{p}_{ij}) + \mathbf{A}_{ii}\mathbf{x}_i + \sum_{j=i+1}^{n_s}(\mathbf{H}_{ij}\mathbf{u}_{ij} + \mathbf{G}_{ij}\mathbf{p}_{ji}) = \mathbf{B}_{ii}\mathbf{y}_i, \quad i = 1, n_s, \tag{8}$$

where n_s is the number of subregions, has to be solved by enforcing continuity and equilibrium conditions at the interfaces. In Eq. (8),

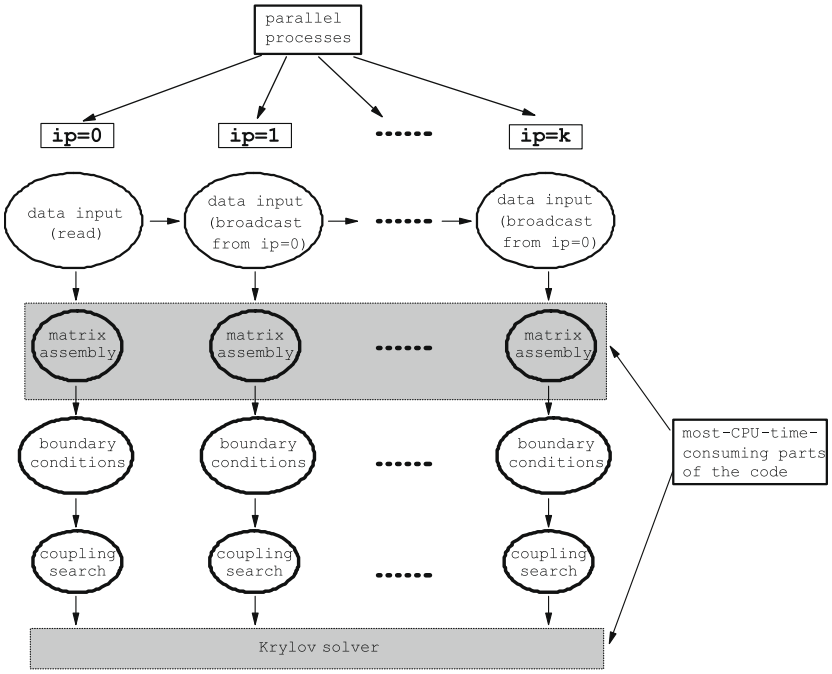


Fig. 5. Flowchart of the BE-SBS-based parallel code.

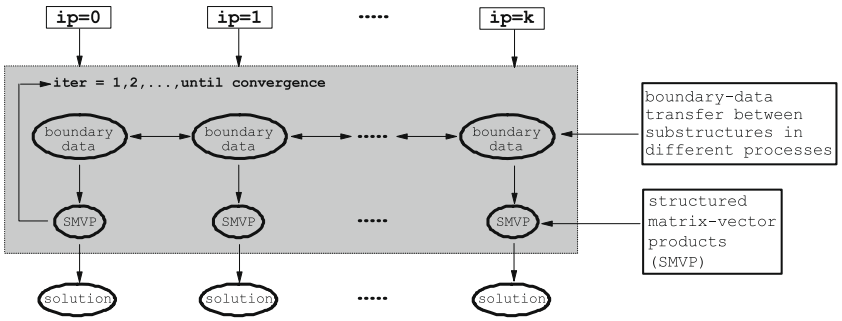


Fig. 6. Solution phase (Krylov solver).

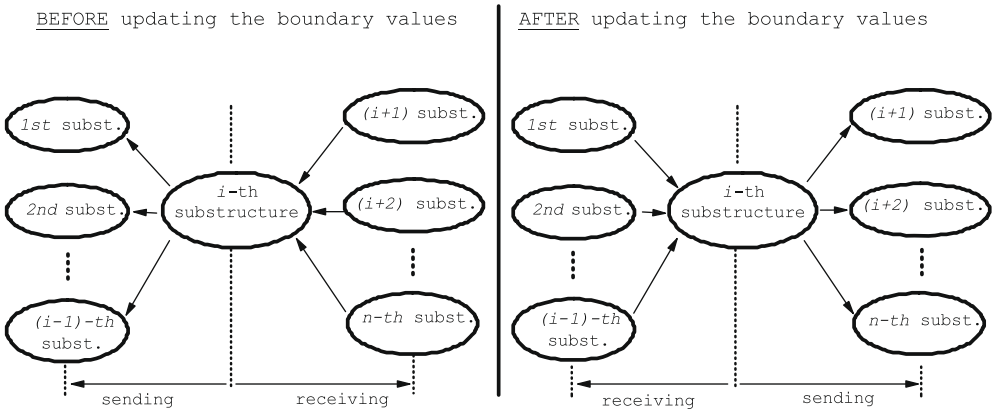


Fig. 7. Boundary-data transfer between processes.

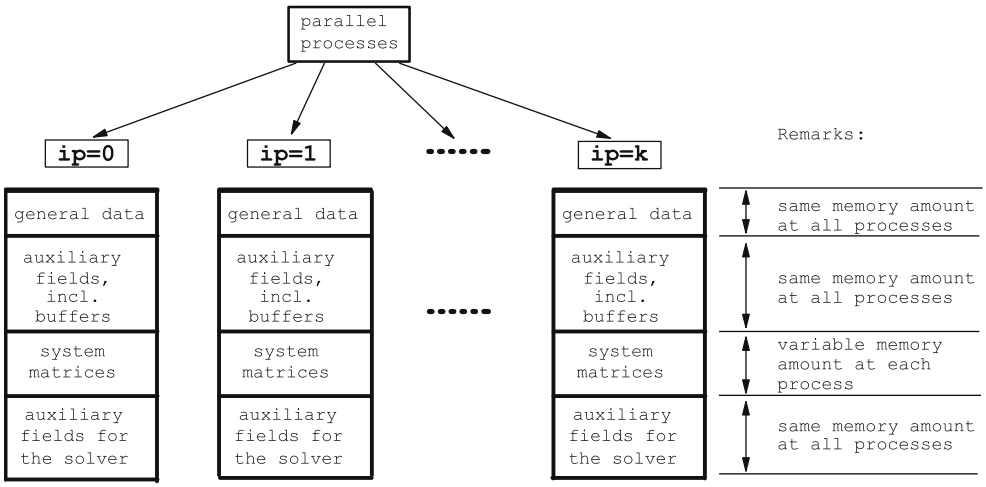


Fig. 8. Memory-allocation scheme.

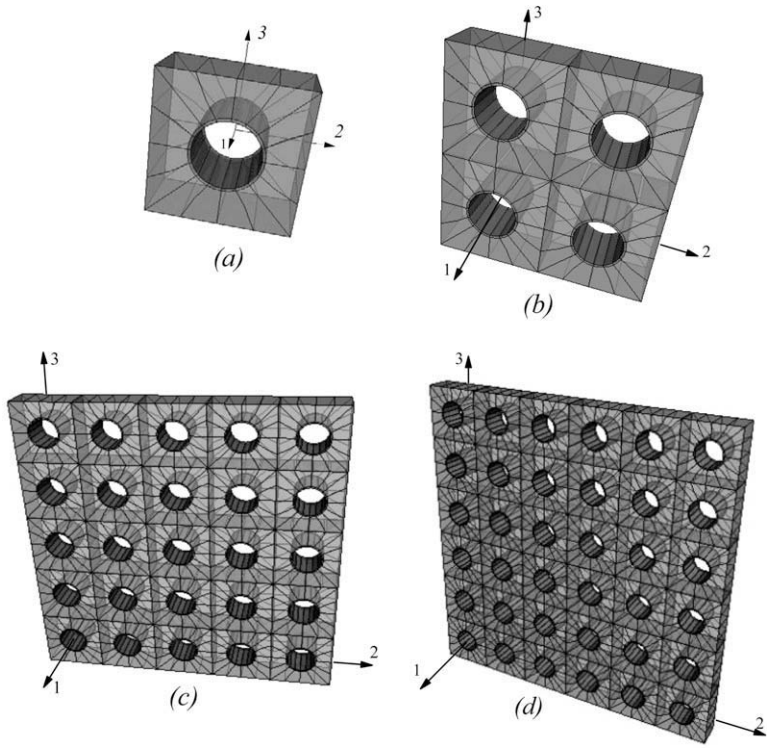


Fig. 9. Square-packed long-CNT-based RVEs.

Table 1
Model data for the square-packed long-CNT RVEs.

Model	nsub[a]	nel[b]	nnodes[c]	ndof[d]	Sparsity (%)
l × l	2	128	608	1824	29
2 × 2	8	512	2660	7980	81
5 × 5	50	1344	17,456	52,368	97
6 × 6	72	2048	25,268	75,804	98

[a] No. of subregions.
[b] No. of elements.
[c] No. of nodes.
[d] No. of degrees of freedom.

Table 2
Engineering constants for the square-packed long-CNT RVEs.

Model	E_1/E_m	E_2/E_m, E_3/E_m	ν_{12}, ν_{13}	ν_{23}
l × l	1.3227	0.8302	0.2974	0.3595
2 × 2	1.3228	0.8319	0.2973	0.3600
5 × 5	1.3228	0.8319	0.2972	0.3580
6 × 6	1.3228	0.8319	0.2972	0.3587
Chen and Liu (3D FE)	1.3255	0.8492	0.3000	0.3799
Rule of mixture[a]	1.3255	–	–	–

[a] RVE volume fraction is $V_f = 3.617\%$.

$\mathbf{H}_{ij}$ and $\mathbf{G}_{ij}$ denote the usual BE matrices obtained for source points pertaining to subregion Ω_i and associated respectively with the boundary vectors $\mathbf{u}_{ij}$ and $\mathbf{p}_{ij}$ at Γ_{ij}. Note that if $i \neq j$, Γ_{ij} corresponds to the interface between Ω_i and Ω_j; Γ_{ii} is the outer boundary of Ω_i.

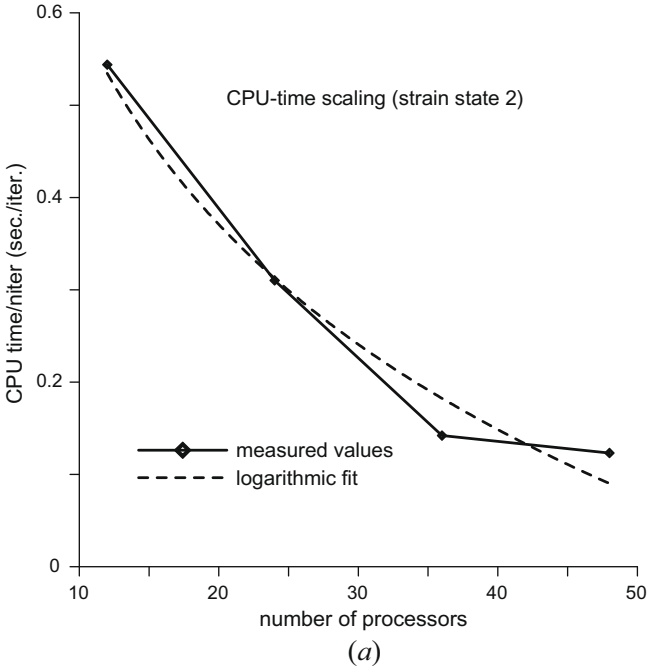
The global model is then stored and iteratively solved according to the data structure shown in Eq. (8). During the solution, wherein in this particular study the diagonal-preconditioned BiCG solver is applied, the interface conditions, given by

$$\begin{cases} \mathbf{u}_{ij} = \mathbf{u}_{ji} \\ \mathbf{p}_{ij} = -\mathbf{p}_{ji} \end{cases} \text{ at } \Gamma_{ij}, \tag{9}$$

are directly enforced, iteration by iteration. 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. (8) 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 Refs. [38,13,12,41], the evolution of the BE-SBS algorithm is documented; there a detailed description of all strategies mentioned above is found.

In the flowchart in Fig. 5, the generic structuring of the BE-SBS-based parallel code is presented. 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 (see Fig. 5). Only during its solution, communication between the processes is needed for updating the boundary values in all subregions (Figs. 6 and 7).

In Fig. 8, the generic memory-allocation scheme is sketched. The system matrices, one by one stored at the work vectors allocated at each processor, will take the largest chunk of the total memory used per processor. According to the allocation strategy, $\mathbf{A}_m$ and $\mathbf{B}_m$, with $m \leqslant k$, will be sequentially allocated in the k first processes. If $m > k$, $\mathbf{A}_m$ and $\mathbf{B}_m$ will be allocated in the process with so far least allocated memory. The total storage memory for the system matrices at each process will then depend upon the corresponding number of substructures allocated at it. An advantage of this allocation strategy is that it allows larger messages in the communication among the processes, reducing then the communication overhead. For example, sending $\mathbf{A}_m$ and $\mathbf{B}_m$ from a certain process to another one will just correspond to sending a single contiguous chunk of the work vector. Herein, notice that communication efficiency increases with message size.



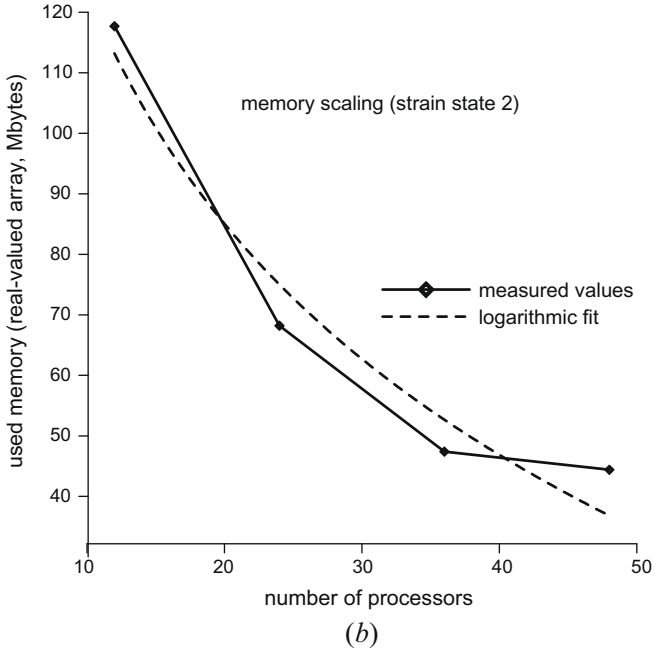


Fig. 10. Scalability curves: 6 × 6 square-packed long-CNT model (strain state 2).

4. Applications

The performance of the BE-SBS-based parallel code detailed above has been observed by determining engineering constants for complex CNT-based composites. To construct the analysis models (RVEs), long and short CNT fibers arranged according to square

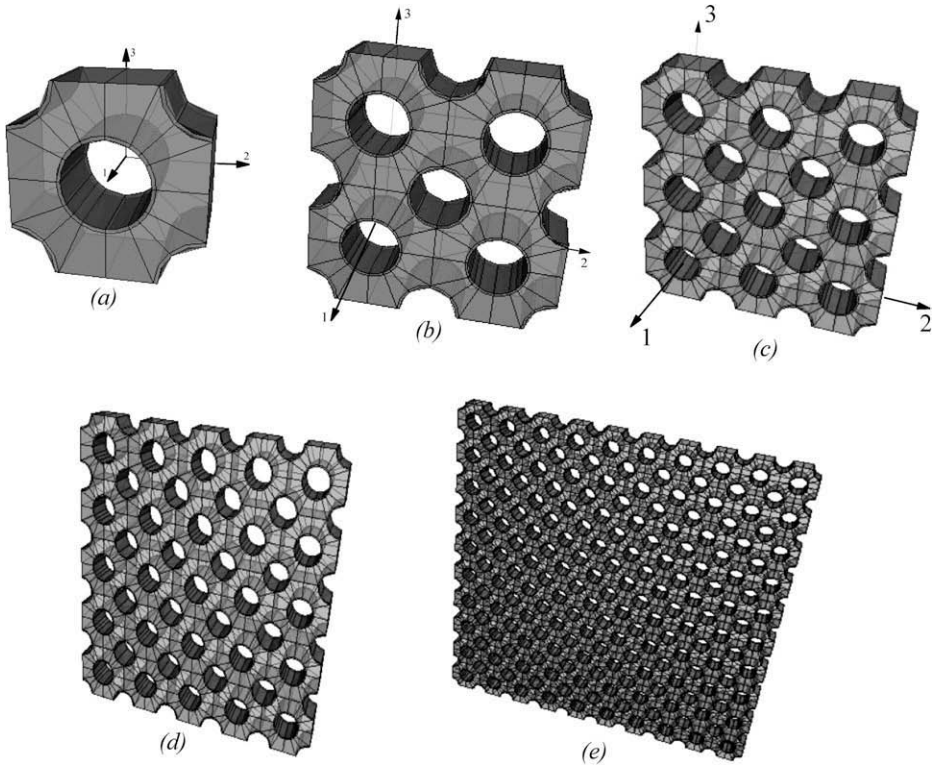


Fig. 11. Hexagonal-packed long-CNT-based RVEs.

Table 3
Model data for the hexagonal-packed long-CNT RVEs.

Model	nsub[a]	nel[b]	nnodes[c]	ndof[d]	Sparsity (%)
1×1	6	138	856	2568	72
2×2	17	656	3456	10,368	86
3×3	34	1464	7800	23,400	93
5×5	86	4040	21,720	65,160	97
10×10	321	16,080	87,040	261,120	99

[a] No. of subregions.
[b] No. of elements.
[c] No. of functional nodes.
[d] No. of degrees of freedom.

Table 4
Engineering constants for the hexagonal-packed long-CNT RVEs.

Model	E_1/E_m	E_2/E_m, E_3/E_m	υ_{12}, υ_{13}	υ_{23}
1×1	1.8081	1.0889	0.2943	0.5107
2×2	1.8074	1.0839	0.2936	0.5107
3×3	1.8074	1.0916	0.2931	0.5185
5×5	1.8126	1.0813	0.2927	0.4997
10×10	1.8014	1.0805	0.2926	0.5103
Rule of mixture[a]	1.8131	–	–	–

[a] RVE volume fraction is $V_f = 9.035\%$.

and hexagonal packing patterns inside the polymer matrix have been adopted. Herein, a single or several coupled composite unit cells have been used.

Concerning the integration quadratures applied, the combined coordinate-transformation-based procedure, defined in [13] and [38], and the line-integral procedure, as described in [41], are employed to evaluate, respectively, the weakly/nearly-weakly-singular, and the strongly-nearly-singular integrals. In all analyses, 8×8 and six integration points are used for all surface and line quadratures involved, respectively. In all RVEs, the following pure phase constants, adopted in [50], are considered:

$$\text{CNT}: \; E_{\text{CNT}} = 1000\ \text{nN/nm}^2\text{(GPa)}; \; \upsilon_{\text{CNT}} = 0.30,$$
$$\text{Matrix}: \; E_m = 100\ \text{nN/nm}^2\text{(GPa)}; \; \upsilon_{\text{CNT}} = 0.30.$$

The long CNT fibers are geometrically defined by cylindrical tubes having outer radius $r_0 = 5.0$ nm and inner radius $r_i = 4.6$ nm, and length $l_f = 10$ nm. The short CNTs have cross section and hemispherical caps with same previous outer and inner radius; its length (including both caps) is $l_f = 50$ nm. In this respect, noticing that RVEs representing long-fiber composites have their fibers all the way through their length, then solving long-CNT composites can be satisfactorily accomplished based on 2D models. Thus, the fiber lengths in 3D models must not necessarily be long at all. That is why in the 3D models in this paper (considered to verify the performance of the general 3D BE-SBS-based parallel code), the long CNTs ($l_f = 10$ nm) are shorter than the short CNTs ($l_f = 50$ nm).

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. 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 3.4 GHz Xeon EM64T processors, 4 GB of memory, and dual Gigabit Ethernet Interconnects.

In this paper, the memory-use and CPU-time scalability are measured, respectively, by the following variables: *used memory* and *CPU time*/*niter bounds*. 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. Thus, the *used memory* and *CPU time*/*niter bounds* take into account the worst situations. Particularly concerning the CPU time measurements, notice that dividing it by the total number of iterations

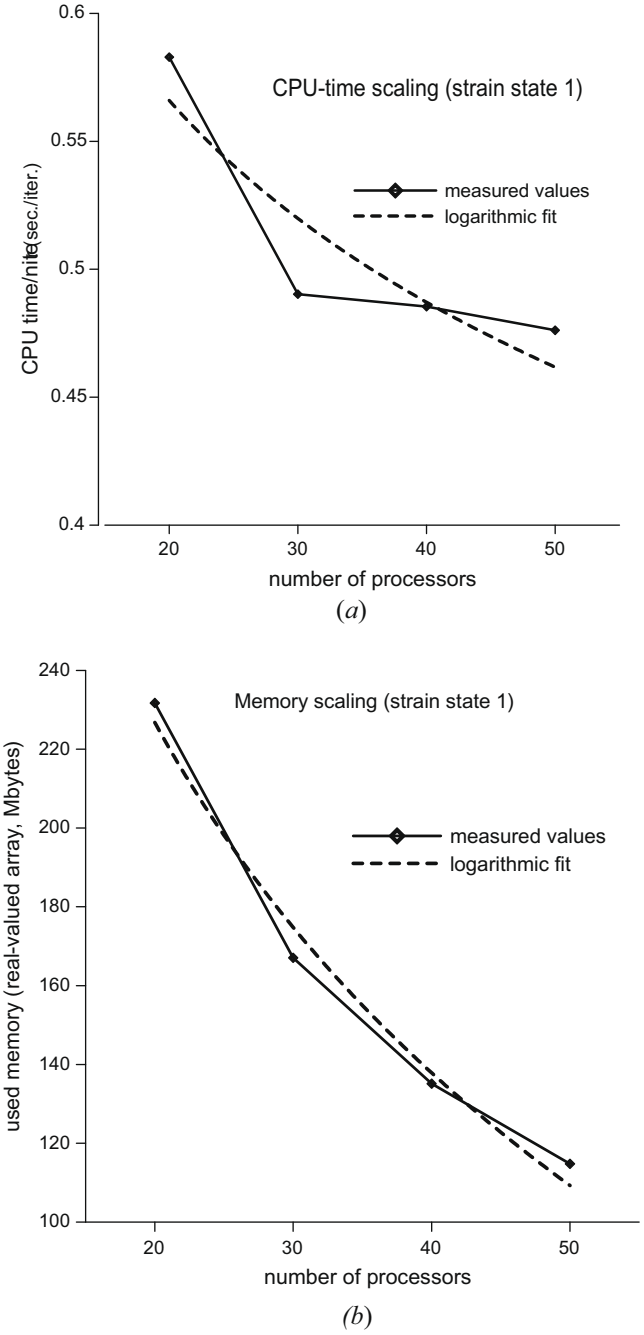


Fig. 12. Scalability curves: 10 × 10 hexagonal-packed long-CNT model (strain state 1).

for convergence makes it independent of the number of iterations; it is then convenient. Moreover, as the assembly CPU time is insignificant compared to the solution CPU time, only the latter one is considered in the parallel-performance analysis. Because of the limited space of the paper, only a few curves showing samples of the parallel-processing performance are presented. Actually, only the scalability curves for the largest model of each application under either strain state 1 or 2 is shown.

4.1. RVEs with square-packed long CNT fibers

In this application, RVEs based on 1 × 1, 2 × 2, 5 × 5, and 6 × 6 unit cells are employed for modeling long-CNT-based composites (see Fig. 9). Noting that for long-fiber–reinforced composites, the response does not vary along the fiber direction (1 direction) of the specimen, any convenient length l_1 (see Figs. 1a and 2a) can be taken; here, $l_1 = 10$ nm, which allows a single layer of boundary elements across the RVEs (see Fig. 9), is considered. The other dimensions of each unit cell (along the 2 and 3 axes; see Fig. 1) are taken as $l_2 = l_3 = 20$ nm. In Table 1, important model data are given. In Table 2, the engineering parameters obtained from the analysis, via the present code, of all the RVEs in Fig. 9 are confronted with results calculated by Liu and Chen [50] via finite-element analysis, and estimated by the rule of mixture (see Refs. [41,50,51]).

As seen from Table 2, values estimated by the rule of mixture and by refined 3D FE models [50] are in very good agreement with the material parameters 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. In Table 1, data showing the sparsity of the corresponding systems are furnished. They indicate that the systems become highly sparse when the number of degrees of freedom (ndof) increases.

In Fig. 10, results for the 6 × 6-unit-cell RVE under strain state 2, showing how the parallel processing scales, are presented. As one sees, the scalability of both CPU-time (see Fig. 10a) and memory-use (see Fig. 10b) 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.

4.2. RVEs with hexagonal-packed long CNT fibers

Here, 1 × 1, 2 × 2, 3 × 3, 5 × 5, and 10 × 10 RVEs are analyzed (see Fig. 11), each one built with unit cells having dimensions $l_1 = 10$ nm and $l_2 = l_3 = 20$ nm. Model data are given in Table 3, and the estimated material parameters, in Table 4. As reference value for comparison purposes, only E_1, estimated by the rule of mixture, is considered (see Refs. [41], [51]), and we verify that E_1 values estimated by the rule of mixture and calculated with the present method are about the same magnitude. It is also observed that increasing the number of unit cells per RVE does not significantly change the estimation of the material constants.

In Fig. 12, the parallel-processing performance is shown for the 10 × 10-unit-cell RVE under strain state 1. Again, the scalability of the memory use is very good, practically following the logarithmic fit (see Fig. 12b). Concerning the CPU-time scaling, it is observed the speedup tends to decrease when the number of processors is incremented (e.g. from 30 to 50 processors; see Fig. 12a), the explanation for that being a relative increase on the interprocessor communication compared to the load per processor.

4.3. RVEs with squared-packed short CNT fibers

In this application, the RVEs are constructed smearing 1 × 1, 2 × 2, and 5 × 5 short capsule-like CNTs according to square-packing patterns (see Fig. 13). A single-cell RVE has outer dimensions $l_1 = 100$ nm , and $l_2 = l_3 = 20$ nm. The geometrical details of the CNT were furnished above, and important model data are given in Table 5. Contrasting the results with the present method with those obtained by Liu and Chen [50], and by the extended rule of mixture [41], good agreement is found (see Table 6). Again, one sees that the material constant values do not considerably modify increasing the number of cells in the RVE.

The parallel-processing performance is shown for the 5 × 5-unit-cell RVE under strain state 1 (see Fig. 14). Comparing the graphs for the CPU-time scaling obtained here with those for the previous models (see Figs. 10a and 12a), a somehow better performance is observed. It happens because for this model, with more degrees of freedom per subregion, the relationship between

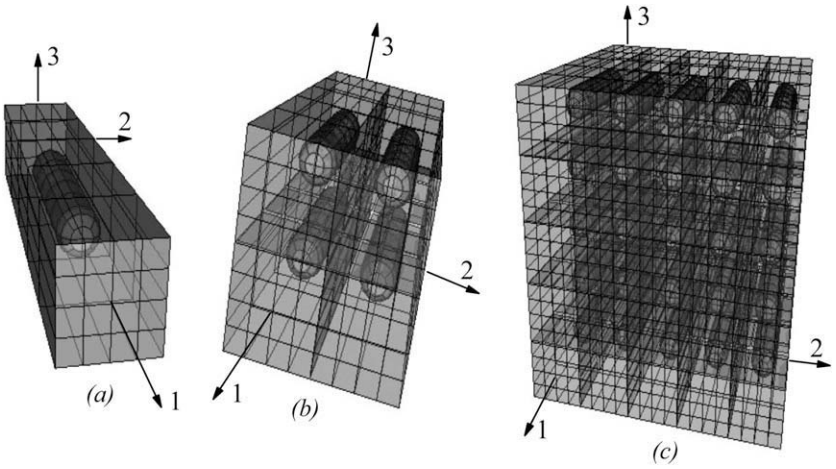


Fig. 13. Square-packed short-CNT-based RVEs.

Table 5
Model data for the square-packed short-CNT RVEs.

Model	nsub[a]	nel[b]	nnodes[c]	ndof[d]	Sparsity (%)
1×1	2	352	1064	3192	27
2×2	8	1408	5156	15,468	78
5×5	50	8800	35,384	106,152	99

[a] No. of subregions.
[b] No. of elements.
[c] No. of functional nodes.
[d] No. of degrees of freedom.

Table 6
Engineering constants for the square-packed short-CNT RVEs.

Model	E_1/E_m	E_2/E_m, E_3/E_m	v_{12}, v_{13}	v_{23}
1×1	1.0378	0.9366	0.2963	0.3207
2×2	1.0379	0.9379	0.2976	0.3217
5×5	1.0379	0.9389	0.3000	0.3223
Chen and Liu (3D FE)	1.0391	0.9342	0.3009	0.3217
Rule of mixture[a]	1.0396	–	–	–

[a] The extended rule of mixture is considered.

communication overhead and processor load is small enough to compensate the increase in the number of processors (see Fig. 14a). The scalability of the memory use is again very good (see Fig. 14b). In this application, the logarithmic fit is closely followed.

5. Conclusions and prospects

A parallel-processing algorithm based upon a robust BE-SBS technique has been developed and particularly applied to estimate effective engineering constants for 3D CNT–reinforced composites. We first observe that as a consequence of the special quadratures available in the code, the reliable use of discontinuous boundary elements is possible, and so traction discontinuity at inner edges and corners of interfaces are easily simulated. These quadratures also allow employing disproportionate boundary elements, and so are very helpful for modeling thin-walled solids, such as CNTs, with relatively coarse meshes, without sacrificing accuracy (see RVE models Figs. 9, 11 and 13). 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, increases computational efficiency by reducing the total matrix-assembly CPU time and makes the modeling of very complex composites (with e.g. hundreds or thousands of fibers smeared in the matrix material) possible. 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 (dominant). However, for large identical subregions, this option will certainly increase the computational efficiency. The strategy proposed is believed to be very convenient for analyzing general composites. Notice as well that the BE-SBS-based algorithm directly furnishes the solution in terms of surface stresses, which the effective material constants depend upon, so that these constants can be straightforwardly estimated from the solution outputted. Moreover, for complex composites, boundary-integral-based models are simpler to generate than volume-based ones.

Certainly, a contribution of this study is the proposal of a general strategy for developing parallel-processing BE codes, noway restricted to the particular class of elasticity problems treated here, but readily applicable to any BIE-based methods. It should be particularly observed that the algorithm proposed, which bases on a non-overlapping DDM, presents the following interesting general characteristics: (1) the BE models are independently generated, stored, and manipulated during the solution of the problem (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 for solving the global coupled system, (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 seen 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.

Of course, an important pillar of the whole parallel code is the Krylov solver, which, based on the current advances achieved in this area, mainly after the 90s, should actually account for considerable CPU-time saving during the solution phase. Herein, relevant characteristics of efficient Krylov solvers are: (1) low number of iterations required for convergence, (2) perfect exploitation of sparsity, and (3) the suitableness for developing scalable parallel code. Particularly in this work, indeed focused on the development of the first parallel version of the BE-SBS algorithm, no special attention has been properly paid to the solver itself. As noted, just

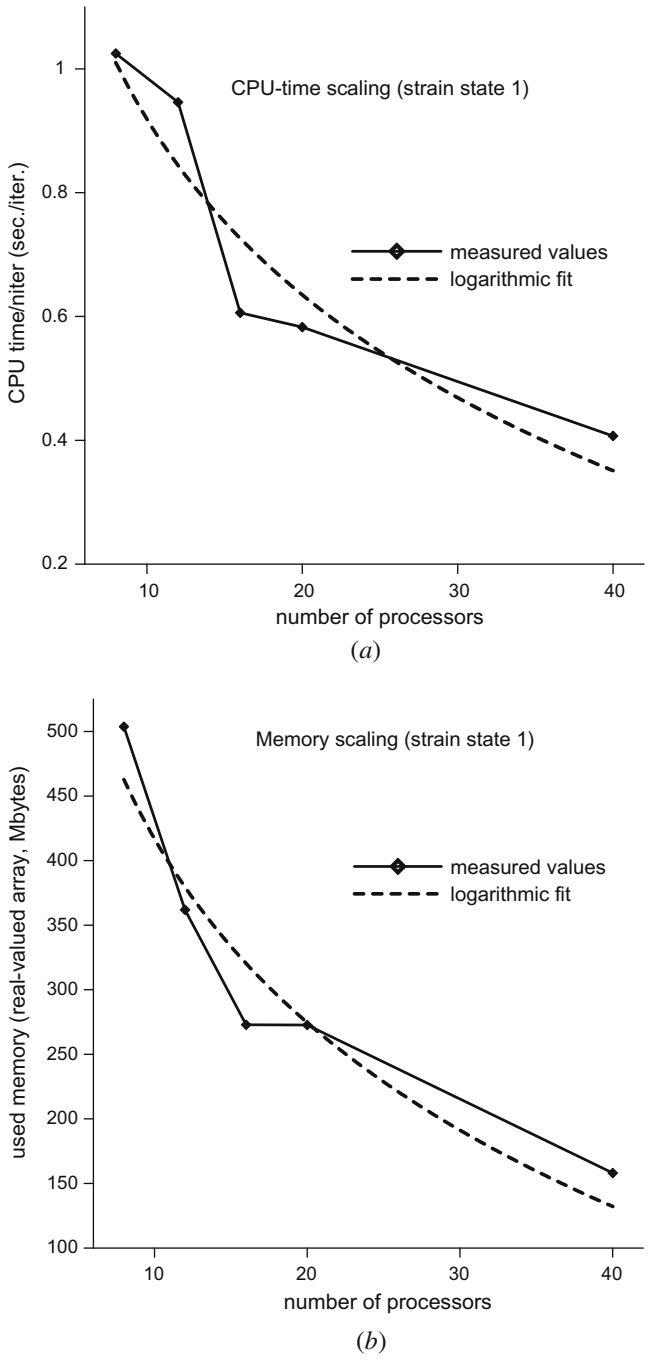


Fig. 14. Scalability curves: 5 × 5 square-packed short-CNT model (strain state 1).

a plain diagonal-preconditioned BiCG solver has been employed. In fact, although for the tolerance considered ($\zeta = 10^{-5}$), the number of iterations was less than 10% of the system order (ndof) in all problems analyzed, it is known that this particular solver presents irregular convergence behavior, sometimes even not converging, depending on the conditioning of the system. Anyway, considering all the development brought about on iterative solvers and preconditioning techniques in the last two decades, 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 this respect, relevant contributions e.g. by Sleijpen and Fokkema [52], and by Zhang [53] should be considered in the next versions of the parallel code. The preconditioning also plays a fundamental role in the efficiency of iterative solvers and will be certainly included in the next development steps of the code. Herein, it is worthy noting that the way in that the SBS data structure has been constructed allows easy implementation of ILU preconditioning, e.g. conveniently defined by the inverses of the coefficient matrices corresponding to the independent subregions. An interesting overview on iterative solvers and preconditioning techniques is given by Barett et al. [54].

On the effective material constants calculated by means of the present BE-SBS algorithm, we see they are in very good agreement with results from both FE calculations and estimated by the rules of mixture. Besides, the strategy adopted for determining the displacement boundary conditions for strain state 2 showed to be appropriate. The corresponding traction resultant in the 3 direction is less than 0.001% of that in the 2 direction for all cases analyzed, i.e. its relative value is approximately zero as it should be [51].

In immediate future steps of this strategy, which may be very useful for the microstructural analysis of general composites, besides the improvements of the Krylov solver, commented above, an analysis option to allow nonlinear matrix–fiber contact (delamination) should be considered.

Acknowledgements

This research was sponsored by the Office of Advanced Scientific Computing Research, US 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.

References

[1] Ghoniem NM, Cho K. The emerging role of multiscale modeling in nano- and micro-mechanics of materials. CMES: Comput Mod Eng Sci 2002;3:147–74.
[2] Suquet P. Continuum micromechanics, CISM courses and lectures No. 377. Berlin: Springer; 1997.
[3] Cailletaud G, Forest S, Joplin D, Feyel F, Galliet I, Mounoury V, Quilici S. Some elements of microstructural mechanics. Comput Mater Sci 2003;27:351–74.
[4] Jeulin D, Ostoja-Starzewski M. Mechanics of random and multiscale microstructures. Springer; 2001.
[5] Kanit T, N'Guyen F, Forest S, Jeulin D, Reed M, Singleton S. Apparent and effective physical properties of heterogeneous materials: representativity of samples of two materials from food industry. Comput. Methods Appl Mech. Eng. 2006;195(33–36):3960–82.
[6] Torquato S. Random heterogeneous media: microstructure and improved bounds on effective properties. Appl Mech Rev 1991;44:37–76.
[7] Carrera E, Demasi L. Classical and advanced multilayered plate elements based upon PDV and RMVT. Part 1: derivation of finite-element matrices. Int J Numer Methods Eng 2002;55:191–231.
[8] D'Ottavio M, Ballhause D, Wallmersperger T, Kröplin B. Considerations on high-order finite elements for multilayered plates based on a unified formulation. Comput Struct 2006;84:1222–35.
[9] Kulilov GM, Plotnikova SV. Geometrically exact assumed stress-strain multilayered solid-shell elements based on the 3D analytical integration. Comput Struct 2006;84:1275–87.
[10] Dávila CG, Chen T-K. Advanced modeling strategies for the analysis of tile–reinforced composite armor. Appl Compos Mater 2000;7:51–68.
[11] Chen XL, Liu YJ. An advanced 3D boundary-element method for characterization of composite materials. Eng Anal Boundary Elem 2005;29:513–23.
[12] Araújo FC, Gray LJ. Analysis of thin-walled structural elements via 3D standard BEM with generic substructuring. Comput Mech 2008;41:633–45.
[13] Araújo FC, Silva KI, Telles JCF. Application of a generic domain-decomposition strategy to solve shell-like problems through 3D BE models. Commun Numer Methods Eng 2007;23:771–85.
[14] Shonkwiler RW, Lefton L. An introduction to parallel and vector scientific computing. 1st ed. Cambridge University Press; 2006.
[15] Chandra R, Dagum L, Kohr D, Maydan D, McDonald J, Menon R. Parallel programming in OpenMP. London: Academic Press; 2001.
[16] Aoyama Y, Nakano J. RS/6000 SP: practical MPI programming. IBM corporation; 1999. <http://www.redbooks.ibm.com>.
[17] Pacheco PS. Parallel programming with MPI. San Francisco: Morgan Kaufman; 1997.
[18] Symm GT. Boundary elements on a distributed array processor. Eng Anal Boundary Elem 1984;1(3):162–5.

[19] Davies AJ. The boundary-element method on the ICL DAP. Parallel Comput 1988;8:335–43.
[20] Kane JH. Boundary-element analysis on vector and parallel computers. Comput Syst Eng 1994;5:239–52.
[21] Natarajan R, Krishnaswamy D. A case study in parallel scientific computing: the boundary-element method on a distributed-memory multicomputer. Eng Anal Boundary Elem 1996;18:183–93.
[22] Song SW, Baddour RE. Parallel processing for boundary-element computations on distributed systems. Eng Anal Boundary Elem 1997;19:73–84.
[23] van der Vorst HA. Iterative Krylov methods for large linear systems. Cambridge University Press; 2003.
[24] Saad Y. Iterative methods for sparse linear systems. Philadelphia: Society for Industrial and Applied Mathematics (SIAM); 2003.
[25] Cwik T, van de Geijn R, Patterson J. Application of massively parallel computation to integral equation models of electromagnetic scattering. J Opt Soc Am 1994;11(4):1538–45.
[26] Putnam JM, Car DD, Kotulski JD. Parallel CARLOS-3D an electromagnetic boundary integral method for parallel platforms. Eng Anal Boundary Elem 1997;19:49–55.
[27] Hendrickson B, Womble D. The torus-wrap mapping for dense matrix calculations on massively parallel computers. SIAM J Sci Comput 1994;15(5):1201–26.
[28] Ingber MS, Papathanasiou TD. A parallel-supercomputing investigation of the stiffness of aligned, short-fiber reinforced composites using the boundary-element method. Int J Numer Methods Eng 1997;40:3477–91.
[29] Lobry J, Manneback P. Parallel MR-BEM using ScaLAPACK. Eng Anal Boundary Elem 1997;19:41–8.
[30] Blackford LS, Choi J, Cleary A, D'Azevedo E, Demmel J, Dhillon I, et al. ScaLAPACK users guide. Philadelphia: SIAM; 1987.
[31] Cunha MTF, Telles JCF, Coutinho ALGA. A portable parallel implementation of a boundary-element elastostatic code for shared and distributed memory systems. Adv Eng Software 2004;35:453–60.
[32] Masters N, Ye W. Fast BEM solution for coupled 3D electrostatic and linear elastic problems. Eng Anal Boundary Elem 2004;28:1175–86.
[33] Buchau A, Hafla W, Groh F, Rucker WM. Parallelized computation of compressed BEM matrices on multiprocessor computer clusters. COMPEL: Int J Comput Math Electr Electron Eng 2005;24(2):468–79.
[34] Lei T, Yao Z, Wang H, Wang P. A parallel fast multipole BEM and its applications to large-scale analysis of 3D fiber–reinforced composites. Acta Mech Sin 2006;22:225–32.
[35] Kamiya N, Lwase H, Kita E. Parallel implementation of boundary element method with domain decomposition. Eng Anal Boundary Elem 1996;18:209–16.
[36] Kamiya N, Lwase H, Kita E. Performance evaluation of parallel boundary-element analysis by domain decomposition method. Eng Anal Boundary Elem 1996;18:217–22.
[37] Lu X, Wu W-L. A new subregion boundary-element technique based on the domain decomposition method. Eng Anal Boundary Elem 2005;29:944–52.
[38] Araújo FC, Silva KI, Telles JCF. Generic domain decomposition and iterative solvers for 3D BEM problems. Int J Numer Methods Eng 2006;68:448–72.
[39] Lou G, Wu TW, Zhang P, Cheng CYR. Vector and multithread computation of silencer performance prediction on a dual-processor PC workstation. Eng Anal Boundary Elem 2002;26:61–70.
[40] Araújo FC, Dors C, Martins CJ, Mansur WJ. New developments on BE/BE multi-zone algorithms based on Krylov solvers – applications to 3D frequency-dependent problems. J Braz Soc Mech Sci Eng 2004;26(2):231–48.
[41] Araújo FC, Gray LJ. Evaluation of effective material parameters of CNT–reinforced composites via 3D BEM. Comp Mod Eng Sci 2008;24(2):103–21.
[42] Hughes TJR, Levit I, Winget L. An element-by-element solution algorithm for problems of structural and solid mechanics. Comput Methods Appl Mech Eng 1983;36(2):241–54.
[43] Balay S, Buschelman K, Eijkhout V, Gropp W, Kaushik D, Knepley M, et al. PETSc users manual. Math Comp Sci Div, Argonne National Laboratory; 2007. <http://www.mcs.anl.gov/petsc>.
[44] Yakobson BI, Brabec CJ, Bernholc J. Nanomechanics of carbon tubes: instabilities beyond linear response. Phys Rev Lett 1996;76:2511–4.
[45] He XQ, Kitipornchai S, Liew KM. Buckling analysis of multi-walled carbon nanotubes: a continuum model accounting for van der Waals interaction. J Mech Phys Solids 2005;53:303–26.
[46] Pantano A, Parks DM, Boyce MC. Mechanics of deformation of single and multiwall carbon nanotubes. J Mech Phys Solids 2004;52:789–821.
[47] Wang CM, Ma YQ, Zhang YY, Ang KK. Buckling of double-walled carbon nanotubes modeled by solid shell elements. J Appl Phys 2006;99:114317.
[48] Chen Y, Dorgan Jr BL, McIlroy DN, Aston DE. On the importance of boundary conditions on nanomechanical bending behavior and elastic modulus determination of silver nanowires. J Appl Phys 2006;100:104301.
[49] Lau K-T, Chiparab M, Linga H-Y, Hui D. On the effective elastic moduli of carbon nanotubes for nanocomposite structures. Compos: Part B 2004;35:95–101.
[50] Chen XL, Liu YJ. Square representative volume elements for evaluating the effective material properties of carbon nanotube-based composites. Comput Mater Sci 2004;29:1–11.
[51] Hyer MW. Stress analysis of fiber–reinforced composite materials. 1st ed. Boston: McGraw-Hill; 1998.
[52] Sleijpen GLG, Fokkema DR. BICGSTAB(L) for linear equations involving unsymmetric matrices with complex spectrum. Electron Trans Numer Methods Anal 1993;1:11–32.
[53] Zhang S-L. A class of product-type Krylov-subspace methods for solving nonsymmetric linear systems. J Comput Appl Math 2002;149:297–305.
[54] Barett R, Berry M, Chan TF, Demmel J, Donato J, Dongarra J, et al. Templates for the solution of linear systems: building blocks for iterative methods. 2nd ed. Philadelphia: SIAM; 1994.