Engineering Analysis with Boundary Elements 37 (2013) 1267–1275

Contents lists available at SciVerse ScienceDirect

Engineering Analysis with Boundary Elements

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

A SBS-BD based solver for domain decomposition in BE methods ☆

F.C. de Araújo [a,*], E.F. D'Azevedo [b], L.J. Gray [b], R. Degenhardt [c]

[a] Department of Civil Engineering, UFOP, 35400-000 Ouro Preto-MG, Brazil
[b] Computer Science and Math Division, ORNL, P.O. Box 2008, Oak Ridge, USA
[c] Institute of Composite Structures and Adaptive Systems, German Aerospace Center, Lilienthalplatz 7, 38108 Braunschweig, Germany

ARTICLE INFO

Article history:
Received 9 May 2013
Accepted 18 June 2013

Keywords:
3D standard BEM
Subregioning techniques
Krylov solvers
Preconditioning
General composites

ABSTRACT

In boundary element methods (BEM), subregioning may be needed either to model complex solids (with cracks, stiffeners, layers, inclusions, etc.) or simply to decompose a problem by computational reasons (e.g. for parallelization). Since the development of the first BEM codes, many attempts have been made to efficiently devise generic boundary-element subregioning techniques. Crucial points are how to profit from the sparsity of the global matrix, and how to deal with traction discontinuities. In this work, the most fundamental steps for efficiently devising reliable and efficient subregioning algorithms are discussed. The subregion-by-subregion (SBS) algorithm and the preconditioning of the embedded Krylov solver are addressed. Besides the BiCG solver, the BiCGSTAB(l) is newly incorporated into the BE-SBS code. The 3D microstructural analysis of carbon-nanotube-reinforced composites (CNT composites) is considered to verify the performance of the algorithm. Numerical results showing the efficiency of the preconditioned solvers studied are presented.

© 2013 Elsevier Ltd. All rights reserved.

1. Introduction

Realistic engineering problems commonly concern modeling a number of coupled subsystems with different materials and special interfacial (contact) constitutive laws. Such problems typically happen in the microstructural analysis of general composites, wherein fibers or particles are smeared within a host material. Describing the macroscopic behavior of the composite from the direct modeling of its constituents (particles or fibers, and matrix material) is not a simple task, and even for small specimens may lead to huge systems of algebraic equations. By using the finite element method (FEM), domain-decomposition (substructuring) techniques as balancing domain decomposition (BDD) [1–4] and finite element tearing and interconnecting (FETI) [4,5,6,7] methods have been largely employed to deal with this class of problems.

These domain decomposition methods (DDMs) have been proposed as general parallel solvers for the FEM, as for the analysis of models with a very large number of degrees of freedom (e.g. for the analysis of composites on the microscale), regular FE algorithms are unacceptably memory/time-consuming. For boundary element methods (BEMs), besides the parallel-computing aspect, substructuring techniques are also needed for modeling material heterogeneity, as boundary-integral-based formulations assume a homogeneous region for stating the boundary-value representations of the corresponding responses. Substructuring (or subregioning) are then fundamental techniques in BEM, and have been pursued since the first implementations of BEMs [8].

Many 3D engineering applications such as in seismology, computational fluid dynamics (CFD), the micromechanical analysis of composites, etc., require a large number of variables. In these cases, iterative solvers are commonly employed since direct solvers (though robust in solving general dense systems [9,10]) generate a large amount of fill-ins and are computationally expensive. In fact, in most practical applications by using the FEM or the BEM, large sparse systems are generated, so that employing direct solvers additionally requires devising complex re-ordering (pivoting) strategies to reduce the fill-ins, and to improve the numerical stability and scalability (for parallel processing) of the algorithms [11–14]. Furthermore, high-precision direct solvers also take into account iterative refinement of the computed solution [15]. Thus, for truly large computational models, iterative solvers, though not completely predictable concerning convergence, have been, sometimes, the tool of choice for their analysis [16,17].

Main advantages of iterative solvers include the complete exclusion of fill-ins, reduced solution time (for few iterations), easy implementation along with diverse matrix storage formats (e.g. compressed sparse row/column [16], element-by-element

☆This paper is the full and peer reviewed version of an original presentation made at the Mini-Symposium 009—Advances in Boundary Element Methods and Mesh Reducing Techniques, which was part of the 10th World Congress on Computational Mechanics (WCCM 2012—www.wccm2012.com), held in São Paulo, Brazil, on July 8–13, 2012.

* Corresponding author. Tel./fax: +55 31 3559 1548.
E-mail addresses: fcelio@em.ufop.br, dearaujofc@gmail.com, fcelio@pq.cnpq.br (F.C. de Araújo), dazevedoef@ornl.gov (E.F. D'Azevedo), len@bssi-tt.com (L.J. Gray), richard.degenhardt@dlr.de (R. Degenhardt).

0955-7997/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.enganabound.2013.06.006

[18], and edge-by-edge [19] data structures), scalability, and simplicity of implementation in parallel. Moreover, iterative solvers are the only alternative for fast BEM formulations such as fast multipole methods (FMMs) [20,21] or precorrected-FFT methods [22], as the matrix coefficients associated with the far-field variables are not explicitly constructed.

Fast scalable (preconditioned) parallel Krylov solvers have been under intense research in the engineering community in the last years [3,23,24]. However, a hard and tricky way has been followed towards devising engineering-reliable iterative solvers, sometimes being attempted to get more advanced and more robust formulations as BiCGSTAB(l) [25], GPBiCG [26], sometimes focused on the construction of efficient preconditioners [27,17]. Many Krylov subspace methods have been devised in the last 5–6 decades [28,16,17]. They may be subdivided into two broad classes based on the length of their recurrence formulas: long-recurrence methods as the GMRES [29] and short recurrence methods as the BiCG (biconjugate gradient) [30], the QMR (quasi-minimal residual) [31], and the BiCGSTAB(l) (l-dimensional biconjugate gradient stabilized) [25]. 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 much more suitable.

Krylov iterative solvers enjoy many advantages but may be less robust compared to sparse direct solvers. However, convergence failure or slowness may be an indication of non-adequate models for describing the physical response at hand, and preconditioners have been successfully employed to accelerate the iterative process [27,17]. In fact, it can even turn out that employing a simpler solver as BiCG with a good preconditioner may bring about more efficiency than using a more robust solver as the pure BiCGSTAB(l) or the GPBiCG. Preconditioning is of fundamental importance since the convergence rate and performance of iterative solvers depend on the spectral conditioning of the preconditioned system matrix. For BEM systems, a series of preconditioners have been reported in the technical literature [27,32,33,28]. 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 [17].

The BE subregion-by-subregion (SBS) algorithm presented in previous works [34,35] is based on iterative solvers, which allow the treatment of coupled subregions as independent domains, i.e. each subregion matrix is independently assembled and stored, and comes into the solution of the system of equations only at the matrix–vector products during the solver iterations. In this paper, the subregion matrices are employed to construct a global block-diagonal (BD) preconditioner. The SBS algorithm is nothing other than a decomposition-domain-based (DDM) 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 L and U factors. Herein it should be noticed that the construction of the SBS-BD (subregion-by-subregion block-diagonal) preconditioner is relatively cheap if few degrees of freedom, say, less than 3–4 thousands, are considered per subregion. Moreover, the price paid for constructing this preconditioning (much higher than e.g. for a plain diagonal preconditioner) is in fact insignificant if convergence reliability and convenience for developing general parallel boundary-element codes is attained.

In this study, the efficiency brought about by the SBS-BD preconditioning applied to the BicG solver is compared with that of the Jacobi-preconditioned BiCGSTAB(l) solver, newly implemented along with the SBS data structure. A standard collocation 3D BE formulation underlies the SBS technique. However, as the BE formulation itself employed to assemble the BE matrices for the many subregions is not the main goal of this study, but the efficiency of the SBS-BD preconditioning in comparison to the use of more elaborate Krylov solvers as the BiCGSTAB(l), no further comments on it will be made. It will be just assumed that the BE matrices are available for the many subdmains. Indeed, any BE formulation or iterative solver may underlie the SBS data structure. Different complex carbon-nanotube (CNT) composites (containing up to several tens of thousands of degrees of freedom) are analyzed to verify the performance of the coupling algorithm. The appropriateness of the preconditioning proposed for developing general scalable BE parallel codes is also commented.

2. The BE-SBS technique

Particularly difficult issues for developing subregioning techniques (in serial or parallel) are the unavoidable simulation of traction discontinuity around internal corners or edges of subdomains, and the devising of optimized techniques to deal with the highly sparse global matrices. Below, the generic BE subregion-

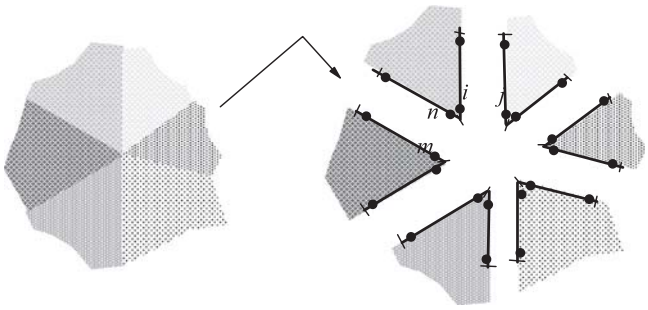

Fig. 2. Corners at subdomains of a heterogeneous material (2D); node m is coupled with node n, and node i with j.

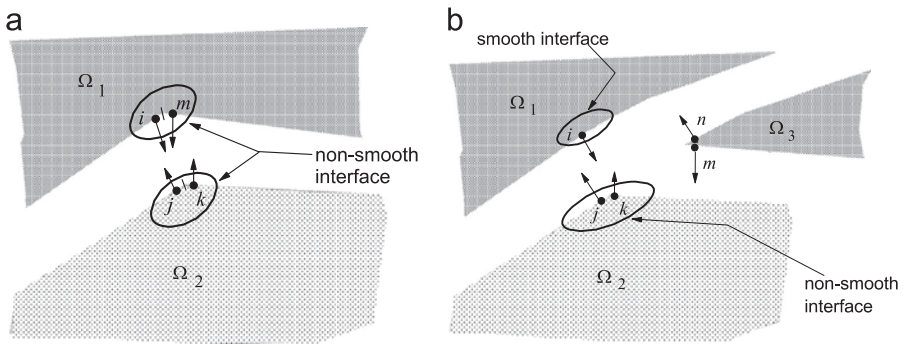

Fig. 1. Coupled 2D subdomains with non-smooth interface.

by-subregion (BE-SBS) algorithm is described, wherein both these issues are conveniently treated, respectively, by employing discontinuous elements and applying Krylov solvers to solve the global (but not explicitly assembled) system of equations.

2.1. Traction discontinuity simulation

In this paper, to simulate traction discontinuity at inner corners and edges (when the normal outward vector at interfaces changes), the concept of discontinuous elements is employed. Herein, the boundary nodes, when needed, are slightly shifted towards the element interior. This artifice conveniently applies to describe discontinuous flux/traction components, but on the other hand gives rise to quasi-singular integrals, for we will have source points very close to boundary elements, and so special integration algorithms will be required. The evolving of the numerical integration quadratures implemented in code is detailed in Refs. [34,35,36], and will be no longer addressed here.

In fact, without the use of discontinuous elements, it is simply impossible to devise a competitive BE subregioning code for general practical purposes, being a truly awkward task to model problems defined in complex coupled domains as 3D solids with inclusions of complex geometries. To explain this affirmative, we first consider a node at a non-smooth interface between two subdomains of a 2D scalar boundary-value problem (for it is simpler to illustrate the issue; see Fig. 1a). We see that by employing only continuous elements, double nodes are needed around the non-smooth interface point because of the change in the normal vector, and so, it will not be possible to solve the problem in this way as for these particular nodes only two linearly independent equations are available (one for each subdomain) to calculate three unknowns ($u_i = u_j = u_m = u_k$, $p_i = -p_j$, and $p_m = -p_k$). Thus, an alternative for solving this problem is to create an additional (physically unnecessary) subdomain so as to generate an additional equation but no additional unknown, as depicted in Fig. 1b, wherein we have now three equations (one for each subdomain) to calculate three unknowns ($u_i = u_j = u_m = u_k = u_n$, $p_i = -p_j = -p_n$, $p_m = -p_k$). However, we observe then that, besides coupled nodes (defined here as nodes with the same coordinates, belonging to different subdomains, and having opposite outward unit normal vectors), we also have flux-continuity nodes, such as nodes j and n, with the same coordinates, belonging to different subdomains, but having the same outward unit normal vector. Thus, the generation of the coupled systems becomes a bit more complicated. Furthermore, for 3D vector (elasticity) problems, it starts to be very cumbersome to generate the BE models based only on continuous elements, so as about to kill one of the main benefits of the BEM (in comparison to the FEM): its simpler meshing by describing a problem in terms only of its boundary values instead of its domain quantities. This fact was observed in [37], where continuous boundary elements were used to model 3D frequency-dependent elastodynamic problems.

As a consequence of the use of discontinuous boundary elements, the procedure for coupling the BE subdomains is exclusively based on coupled nodes (no flux/traction continuity nodes are needed), and thereby, the BEM keeps holding one of its most important characteristics: easier modeling (see Fig. 2). However, noting that the system order considerably increases with the use of discontinuous elements, discontinuous boundary elements are generated (by means of a totally automated process) just when strictly needed (depending on the boundary/interface conditions). This implementation detail makes the code more efficient.

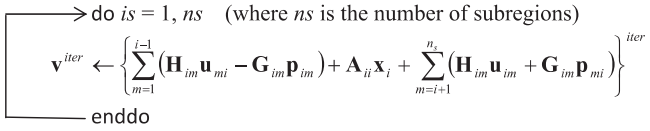

Fig. 3. Implicit matrix–vector product for the iterative solver.

2.2. The solution of the sparse system

Regarding ways to deal with the highly sparse resulting matrices, many strategies have been proposed in the literature

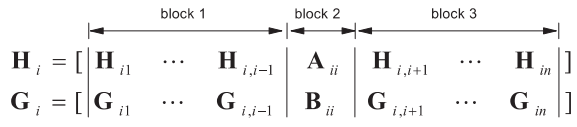

Fig. 4. Data structure for the BE matrices.

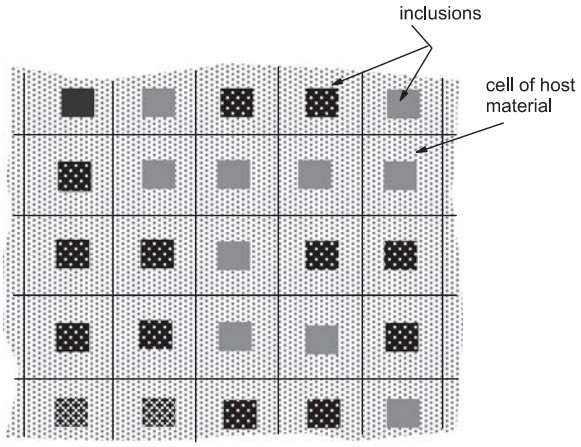

Fig. 5. Particle-reinforced composite.

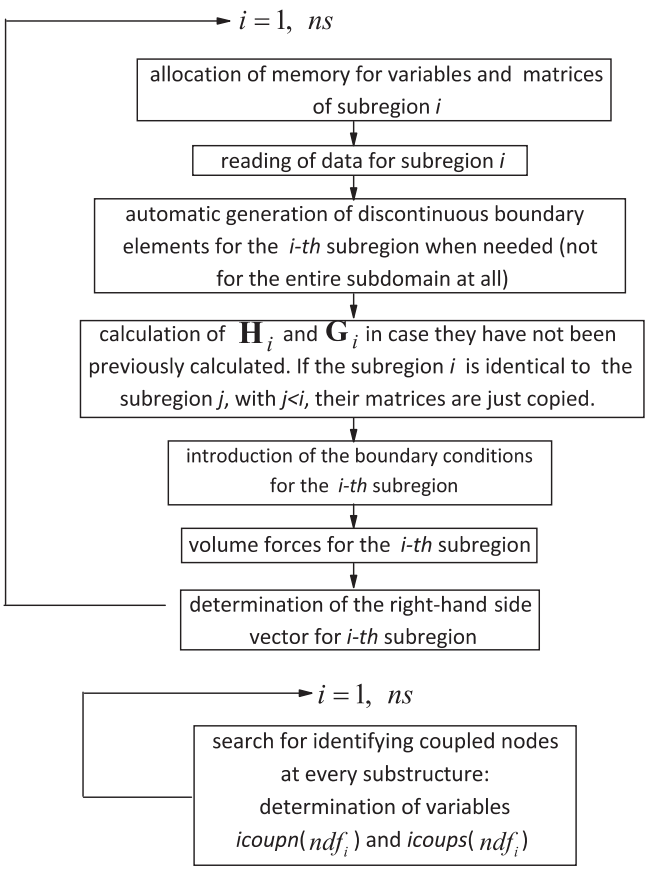

Fig. 6. Matrix-assembly of the SBS algorithm (including search for coupled nodes).

[38–41]. Kamiya et al. [39,40] proposed a substructuring technique based on the DDM to analyze coupled potential problems, wherein an iterative procedure is employed to introduce the interface conditions. This technique perfectly treats the matrix sparsity but does not state any clear way to choose the parameters needed for the iterative procedure so as to assure convergence of the coupling process. In [41], condensing the system unknowns to the interface tractions is proposed to solve 2D elasticity problems with cracks. The strategy is also based on the DDM and allows the independent assembling of the subdomain matrices. However, it involves the calculation of Schur-like complements, and this may be awkward and time-consuming for complex 3D models. Another possibility is to find a way to optimize the memory requirements as a function of the position of the non-zero blocks in the global

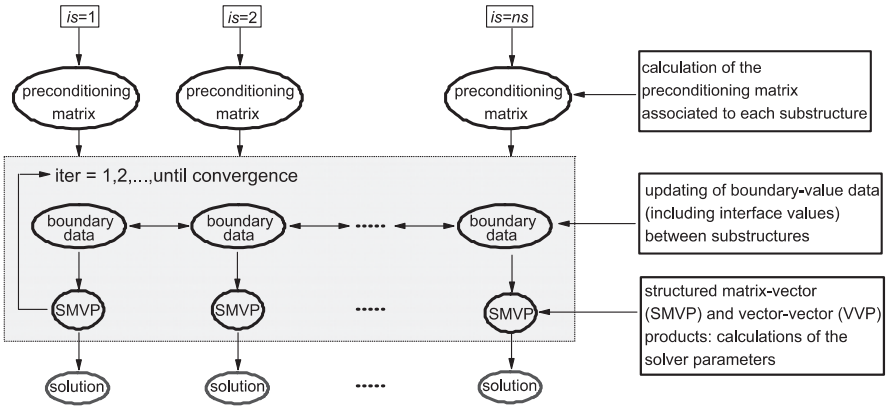

Fig. 7. Solution phase of the SBS algorithm.

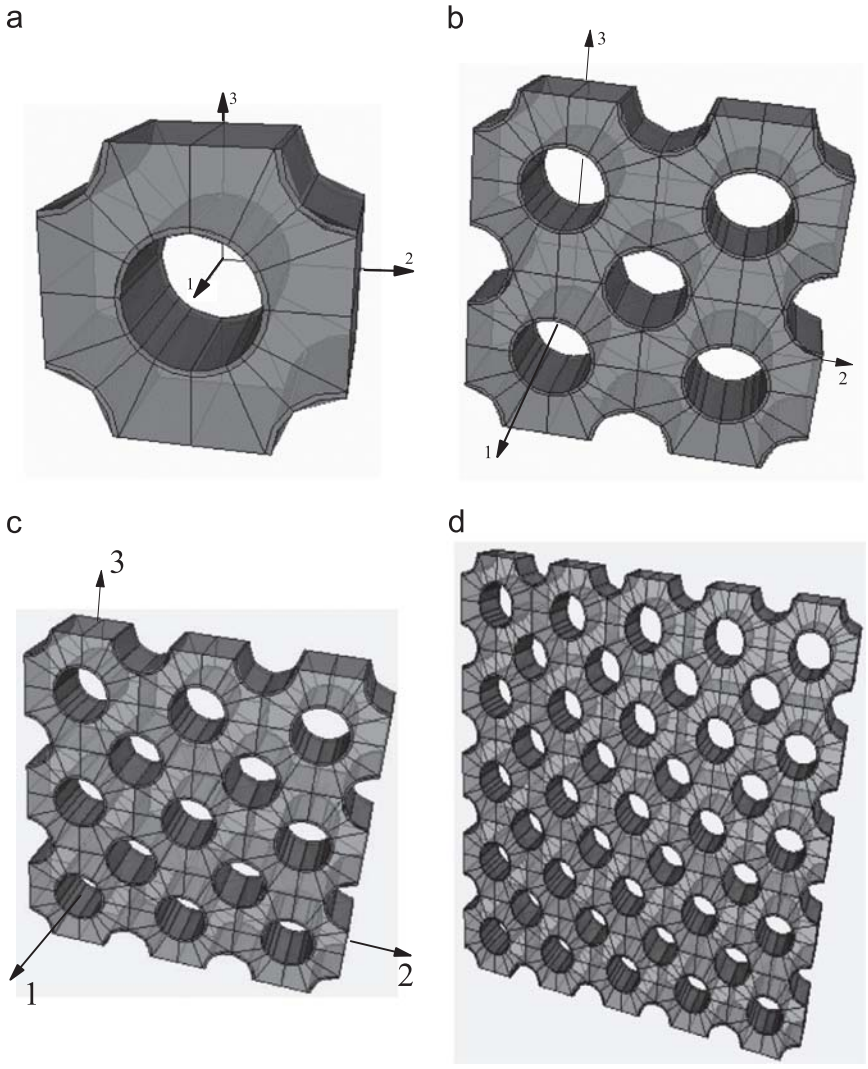

Fig. 8. Hexagonal-packed long-CNT-based RVEs.

matrix, as classically done for FE models. A parallelization strategy along these lines was proposed by Kane [38]. Herein the matrix sparsity is perfectly exploited by applying iterative solvers.

Differently from the strategies mentioned above, in the BE-SBS algorithm considered in this paper [34–36], inspired in the element-by-element (EBE) technique [18], the high sparsity of the global system matrix is optimally handled just by merely non-assembling it explicitly. In the analogy to the EBE technique, a boundary-element subregion corresponds to a single finite element. Then, by employing an iterative solver, a solution strategy for general coupled problems may be derived for which the matrix–vector products (involving the coefficient matrix and possibly its transpose) are calculated from the separate contributions from each subsystem, and the interface conditions, given by

$$\begin{cases} \mathbf{u}_{ij} = \mathbf{u}_{ji} \\ \mathbf{p}_{ij} = -\mathbf{p}_{ji} \end{cases} \text{ at } \Gamma_{ij}, \tag{1}$$

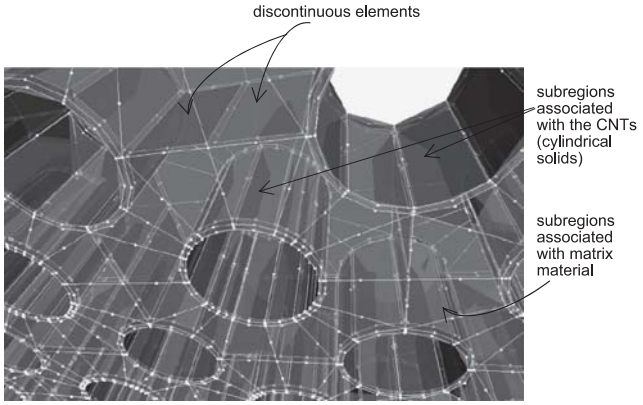

Fig. 9. Mesh details of the long-CNT-reinforced RVE.

are imposed iteration-by-iteration on a direct way. Thus, the global matrix needs not be explicitly assembled. The subsystems are then treated as physically independent from one another, so that only the exact memory space for storing all of them independently is allocated.

For n_s subregions, after introducing the boundary conditions, the BE global system of equations is then given by

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

where $\mathbf{H}_{ij}$ and $\mathbf{G}_{ij}$ denote the regular 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}. Herein Γ_{ij} denotes the interface between Ω_i and Ω_j if $i \neq j$; Γ_{ii} is the outer boundary of Ω_i. Note that $\mathbf{H}_{ij} = \mathbf{H}_{ji} = \mathbf{G}_{ij} = \mathbf{G}_{ji} = \mathbf{0}$ if the there is no coupling between i and j subdomains (which accounts for the sparsity of the coupled system). Indeed, having then been defined, when generating the mesh, which elements are boundary or interface elements, the analysis involving coupled domains, including the independent assembling of the subsystems according to Eq. (2) and the determination of the global iterative solution with no explicit assembling of the global system, flows in a totally automated process.

In general, the scheme shown in Fig. 3 is considered to calculate the matrix–vector products from the decomposed domain. 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 by re-ordering the matrix columns of the ith subregion in three separate blocks: one associated with interfaces Γ_{ij} for which $i > j$, a second one associated with the outer boundary Γ_{ii}, at which boundary values are prescribed, and a third one associated with interfaces Γ_{ij} for $i < j$ (see Fig. 4). Thus, compared to unstructured matrix–vector product (UNSMVP), many conditional tests, per every single degree of freedom to identify its type of boundary condition (if boundary or interface value), are avoided during the calculation of matrix–vector products along the solver iterations (see [34,36]). For re-ordering the system

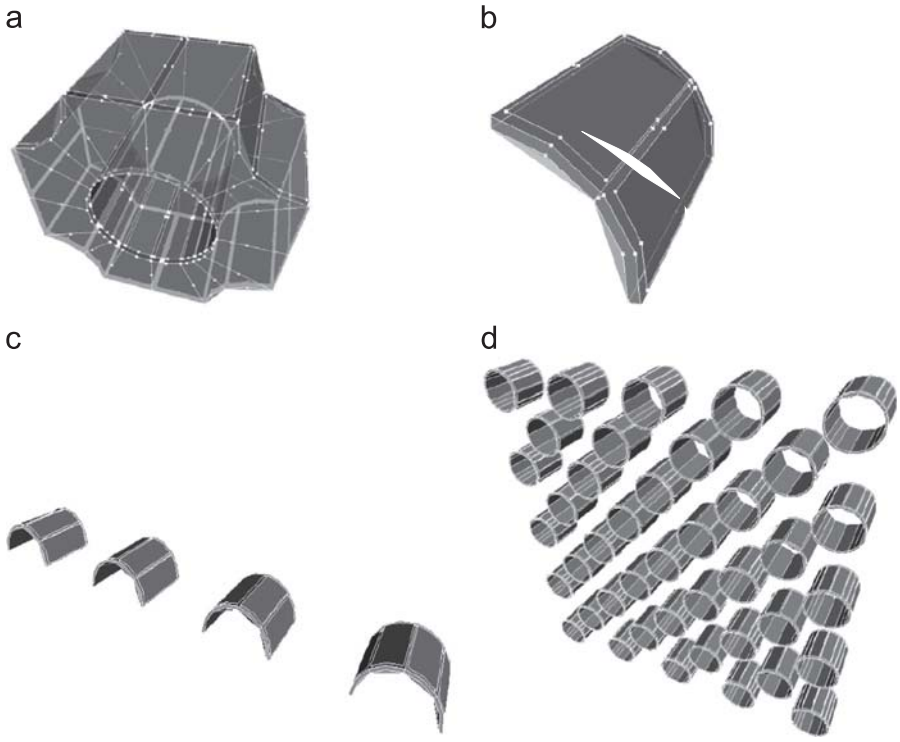

Fig. 10. Independent BE subregions composing the 3D RVE. (a) Matrix material. (b) Quarter-cylinder CNTs. (c) Half-cylinder CNTs. (d) Complete CNTs.

matrices, the variables '*icoupn*(.)' and '*icoups*(.)', automatically generated for every subregion during the search for coupled nodes, are used. These variables, say for the *i*th subregion, are of the form $icoupn(m)=j$ and $icoups(m)=k$, where $m=1,\ ndf_i$, ndf_i is the number of degrees of freedom of the subregion '*i*', '*j*' is the numbering of the degree of freedom coupled with the *m*th degree of freedom of subregion '*i*', and '*k*' is the numbering of the subregion of '*j*'.

The matrix-copy option mentioned above speeds up the matrix assembly in case of (many) physically and geometrically identical subregions (e.g. identical fibers in a composite) under the same boundary/interface conditions (see Fig. 5) [35].

As iterative solvers, the BiCG and the BiCGSTAB(*l*) methods are employed along with the BE-SBS code. The BiCG solver follows the classical implementation proposed by Fletcher in 1974 (see [30,16]) while the more complex BiCGSTAB(*l*) routine [25] was directly downloaded from Sleijpen's web site (⟨http://www.math.uu.nl/~sleij101/⟩) and included into the BE-SBS code. Notice that, as long as the SMVP routines for the BE-SBS algorithm have been implemented, the inclusion of any other iterative solver is straightforward for the matrix–vector products are the basic operations needed for iterative solvers.

Table 1
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

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

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

Model	E_1/E_m	$E_2/E_m, E_3/E_m$	ν_{12}, ν_{13}	ν_{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
Rule of mixture[a]	1.8131	–	–	–

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

3. The SBS-BD-based preconditioner

Organizing the boundary variables for the *i*th subregion according to the sequence $\{\mathbf{p}_{i1}, \mathbf{p}_{i2}, \ldots, \mathbf{p}_{i,i-1}, \mathbf{x}_{ii}, \mathbf{u}_{i,i+1}, \ldots, \mathbf{u}_{i,n_s}\}$, one sees from Eq. (2) that the block-diagonal matrices of the coupled system are given by

$$\mathbf{Q}_i = \begin{bmatrix} -\mathbf{G}_{i1} & \cdots & -\mathbf{G}_{i,i-1} & \mathbf{A}_{ii} & \mathbf{H}_{i,i+1} & \cdots & \mathbf{H}_{in} \end{bmatrix}, \quad i=1,\ n_s. \tag{3}$$

In this study, the $\mathbf{Q}_i$ matrices are considered to straightforwardly construct the global SBS-based block-diagonal (SBS-BD) preconditioner for the coupled system of equations. As the global system matrix, the global 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 only the BicG solver, which means that systems like $(\mathbf{L}_i\mathbf{U}_i)\mathbf{x}_i = \mathbf{x}_i$ and $(\mathbf{L}_i\mathbf{U}_i)^T\mathbf{x}_i = \mathbf{x}_i$ have to be solved, where $\mathbf{x}_i$ is the iterative solution for the *i*th subregion (for more details see Araújo, d'Azevedo, and Gray [42]). For the BiCGSTAB(*l*) iterations, only the plain Jacobi preconditioning is left applied. Notice that the latter solver does not need any transpose–matrix–vector product.

To give a general idea of the whole SBS algorithm, the flowcharts in Figs. 6 and 7 are presented. In Fig. 6, the assembling of the system of equations along with the automatic search for coupled nodes is illustrated, and that in Fig. 7, the solution phase is shown.

4. Applications

To measure the performance of the solvers and preconditioners, the hexagonal-packed long CNT-based composites shown in Fig. 8 are analyzed. Herein, 3D representative volume elements (RVEs) based on 1 × 1, 2 × 2, 3 × 3, and 5 × 5 unit cells are employed. The long CNT fibers are geometrically defined by thin cylindrical tubes having outer radius $r_0 = 5.0$ nm, inner radius $r_i = 4.6$ nm, and length $l_f = 10$ nm. Besides the practical relevance of this application, which concerns a technique for modeling general CNT (and other) composites, the problem considered also

Table 3
Performance data for the hexagonal-packed long-CNT RVEs; tol = 1.0×10^{-6}.

Model	System order	nmvp (SBS-BD BICG)	nmvp (Jacobi BICG)	nmvp (Jacobi BICGstab *l*=8)	CPU time (s) (SBS-BD BICG)	CPU time (s) (Jacobi BICG)	CPU time (s) (Jacobi BICGstab *l*=8)
1 × 1longCNT hex, load1	2568	112	532	338	1.44	2.09	1.44
1 × 1longCNT hex, load2	2568	128	722	386	1.56	2.84	1.62
2 × 2longCNT hex, load1	10,368	388	1506	Failed to converge	15.46	25.88	—
2 × 2longCNT hex, load2	10,368	452	1670	980	17.52	28.64	18.82
3 × 3longCNT hex, load1	23,400	520	2168	4183	46.82	88.73	179.71
3 × 3longCNT hex, load2	23,400	690	3034	1940	60.06	124.27	82.97
5 × 5longCNT hex, load1	65,160	944	6262	7276	230.27	740.07	906.42
5 × 5longCNT hex, load2	65,160	1314	6048	7225	313.86	716.60	894.01

n is the system order; *nmvp* is the number of matrix–vector products.

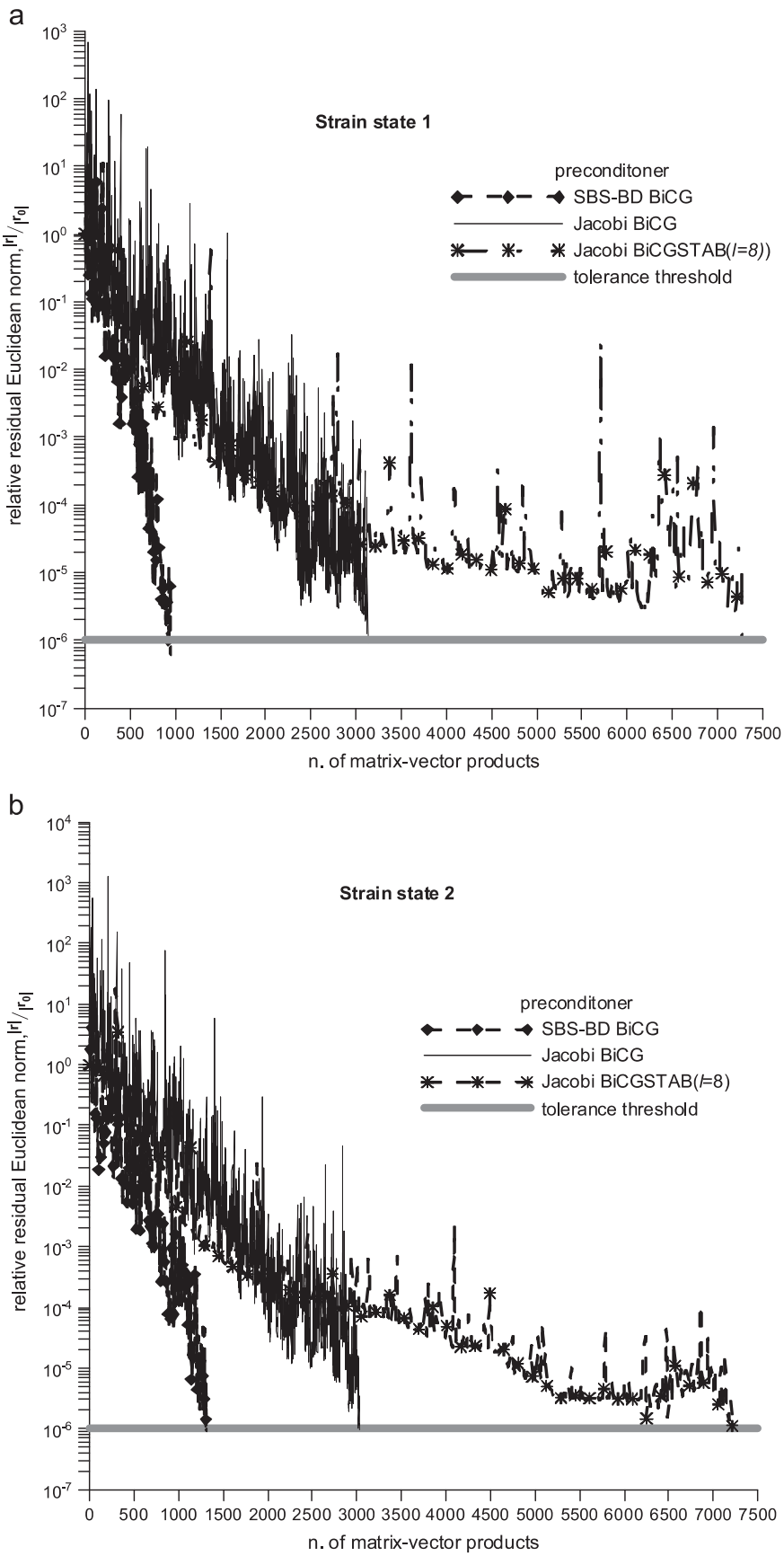

Fig. 11. Residual norm decaying vs. *nmvp*: 5×5 -unit-cell, hexagonal-packed long CNT.

offers a series of intricate issues to be dealt within BE formulations as subregioning, discontinuous elements, thin domains, nearly-singular integrations, Krylov solvers, preconditioning, etc. Thus, this application is very important to verify the overall performance of the generic SBS technique. So as to describe it more clearly, details of the BE mesh adopted are furnished in Fig. 9. In this figure, part of the coupled model for the RVE is zoomed in so as to show details of the discontinuous elements and how the many BE subdomains are coupled to each other. Additionally, in Fig. 10, some independent subregions employed to decompose the RVE are shown separately. In fact, as mentioned above, the subregions (as seen in Fig. 10) are generated, and allocated, one by one, in any order, based only on local numbering for elements and nodes. The coupling between them is considered only during the iterative solution of the system.

In all (RVEs), the following pure phase constants are adopted [43]:

$$\text{CNT}: \quad E_{CNT} = 1000\ \text{nN/nm}^2\ \ (\text{GPa}); \quad \nu_{CNT} = 0.30,$$

$$\text{Matrix}: \quad E_m = 100\ \text{nN/nm}^2\ \ (\text{GPa}); \quad \nu_{CNT} = 0.30.$$

Notice that discontinuous boundary elements, employed to simulate traction discontinuity, are, when needed, automatically generated by shifting the nodes interior to the elements a distance of $d = 0.10$ (measured in the natural coordinate system), and, in general, the BE models contain both continuous and discontinuous elements (see Figs. 9 and 10a and b). Furthermore, physically and geometrically identical subdomains are conveniently replicated by the matrix-copy option, avoiding then repeated assembling of their corresponding matrices. Thus, for example, for the set of CNTs shown in Fig. 10d, the coefficient matrix is assembled for one of them and copied for the other identical parts. The 8-node quadrilateral boundary element is employed to construct the models (see Figs. 9 and 10a and b), and in all analyses 8×8 and 6 integration points are used for evaluating respectively all surface and line integrals involved in the special integration quadratures embedded in the code [35]. In Table 1, important model data are provided.

Although there have been other interesting issues addressed while developing the SBS code, the main purpose of this paper is to highlight the performance of the Krylov solver (embedded in the SBS technique) preconditioned by the BE subregion matrices themselves. To do this, the SBS-BD-based and plain-diagonal (Jacobi) preconditioner are applied to accelerate the BiCG solver while the BiCGSTAB(l) solver is accelerated exclusively by the plain-diagonal preconditioner so as to highlight the efficiency brought about by the SBS-BD-based preconditioner. The iterative process is stopped when $||\mathbf{r}_m||/||\mathbf{r}_0|| < \zeta$, where $\mathbf{r}_m$ is the residual at the mth iteration, and ζ is the tolerance number, in the analysis here taken as $\zeta = 10^{-6}$. A notebook with i3 core intel 2.13 GHz processor, and 4 GB of random access memory, was used to run the models.

In Table 2, the effective engineering parameters extracted from the analysis of all the RVEs shown in Fig. 8 are confronted with results calculated by the rules of mixture [43]. Very good agreement between the results is observed. Furthermore, increasing the number of unit cells per RVE does not cause any significant change in the constant values. These values of material parameters have in fact already been shown in previous papers by the authors [42]. In this paper, mainly the contrasting of the performance between the BE-SBS-BD-preconditioned BiCG and the Jacobi-preconditioned BiCGSTAB(l), with BiCGSTAB dimension $l=8$, is focused. In Table 3, the performance data of the solvers are shown, and in the graphs in Fig. 11, the decaying of the residual Euclidean norm for the largest model, with 65,160 degrees of freedom is presented. As seen in this application, in general, the performance of the SBS-BD-preconditioned BiCG is superior to the Jacobi-preconditioned BiCGSTAB($l=8$), which even fails to converge for model with 2×2 unit cells (model '2×2longCNThex,load1').

5. Conclusions

The BE-SBS algorithm proposed in previous papers ([34], [35]) is employed in this work to straightforwardly construct block-diagonal preconditioners, the BE-SBS-BD ones, for accelerating Krylov solvers. The performance of this preconditioning strategy was verified by analyzing complex composite RVEs.

Observing Table 3 and graphs in Fig. 11, we see that the BE-SBS-BD preconditioning, compared to the Jacobi (plain diagonal) one, is considerably more efficient. In fact, the SBS-BD preconditioning states a transition between direct and iterative solvers, in the sense that the less the number of interfaces in the model, the closer to the global system matrix the preconditioning matrix, $\mathbf{Q}$, is. Furthermore, observing that the high sparse global coupled matrix has just a relatively few number of non-zero coefficients off the block diagonal, we see that the SBS-BD preconditioner is actually a good approximation of the system matrix, which is one of the requirements that good preconditioners should satisfy.

As the results show (see graphs in Fig. 11), applying the SBS-BD preconditioner to the BICG solver brought about more efficiency than just replacing the BiCG solver by the theoretically more robust BiCGSTAB(l) one. In fact, for the models analyzed, even the Jacobi BiCG solver is sometimes more efficient than the Jacobi BiCGSTAB($l=8$), which fails to converge for one of the models. Varying the BiCGSTAB dimension, l, may also lead to different solver performances but actually no criterion to choose optimal l values has been stated in the literature yet; there are only suggested values ($l = 2, 4, 8$). Particularly for the models analyzed, other l values (not reported here) led to non-convergence in many cases.

The results in this paper not only emphasize that preconditioning is a fundamental technique for iterative solvers but also encourage looking for more efficient preconditioners instead of just trying to develop more robust and complex Krylov methods. Indeed, the results hint that a combination of good solvers with good preconditioners may be the right way to generate reliable and fast solvers.

In future works, the BE-SBS-BD BiCGSTAB(l) and its parallel implementation should also be implemented. Of course, being the BE-SBS-BD preconditioner based on the BE-SBS algorithm, its parallelization is immediate. In general, we might see that solver-convergence reliability and parallel-processing suitability may be attained with the ideas discussed in this paper.

Acknowledgments

This research was sponsored by the Brazilian Research Council (CNPq), and the Research Foundation for the State of Minas Gerais (FAPEMIG), Brazil.

References

[1] Mandel J. Balancing domain decomposition. Commun Appl Numer Methods Eng 1993;9:233–41.
[2] Tallec PL. Domain-decomposition methods in computational mechanics. Comput Mech Adv 1994;1(2):121–220.
[3] Sistek J, Sousedik B, Burda P, Mandel J, Novotny J. Application of the parallel BDDC preconditioner to the Stokes flow. Comput Fluids 2011;46:429–35.
[4] Parret-Fréaud A, Rey C, Gosselet P, Feyel F. Fast estimation of discretization error for FE problems solved by domain decomposition. Comput Methods Appl Mech Eng 2010;199:3315–23.

[5] Farhat C, Roux FX. An unconventional domain decomposition method for an efficient parallel solution of large-scale finite element systems. SIAM J Sci Stat Comput 1992;13:379–96.
[6] Dostal Z, Horak D, Vlach O. FETI-based algorithms for modelling of fibrous composite materials with debonding. Math Comput Simul 2007;76:57–64.
[7] Farhat C, Mandel J, Roux FX. Optimal convergence properties of the FETI domain decomposition method. Comput Methods Appl Mech Eng 1994;115:365–85.
[8] Kane JH. Boundary element analysis in engineering continuum mechanics. Englewood Cliffs, NJ: Prentice-Hall; 1994.
[9] Anderson E, Bai Z, Bishof C, Blackford LS, Demmel J, Dongarra J, et al. LAPACK users guide. 3rd ed. Philadelphia: SIAM; 1999.
[10] Blackford LS, Choi J, Cleary A, d'Azevedo E, Demmel J, Dhillon I, et al. ScaLAPACK users guide. Philadelphia: SIAM; 1997.
[11] Duff IS, Koster J. The design and use of algorithms for permuting large entries to the diagonal of sparse matrices. SIAM J Matrix Anal Appl 1999;20 (4):889–901.
[12] Schenk O, Gärtner K. Solving unsymmetric sparse systems of linear equations with PARDISO. Future Gener Comput Syst 2004;20:475–87.
[13] Maurer D, Wieners C. A parallel block LU decomposition method for distributed finite element matrices. Parallel Comput 2011;37:742–58.
[14] Gupta A. A shared- and distributed-memory parallel generalsparse direct solver. AAECC 2007;18:263–77, http://dx.doi.org/10.1007/s00200-007-0037-x.
[15] Wilkinson JH. Rounding errors in algebraic processes. Prentice-Hall; 1963.
[16] Saad Y. Iterative methods for sparse linear systems. Philadelphia: Society for Industrial and Applied Mathematics (SIAM); 2003.
[17] van der Vorst HA. Iterative Krylov methods for large linear systems. Cambridge University Press; 2003.
[18] 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.
[19] Elias RN, Martins MAD, Coutinho ALGA. Parallel edge-based solution of viscoplastic flows with the SUPG/PSPG formulation. Comput Mech 2006;38:365–81, http://dx.doi.org/10.1007/s00466-005-0012-y.
[20] Liu Y, Nishimura N, Otani Y. Large-scale modeling of carbon-nanotube composites by a fast multipole boundary element method. Comput Mater Sci 2005;34:173–87.
[21] Popov V, Power H, Walker SP. Numerical comparison between two possible multipole alternatives for the BEM solution of 3D elasticity problems based upon Taylor series expansions. Eng Anal Bound Elem 2003;27:521–31.
[22] Masters N, Ye W. Fast BEM solution for coupled 3D electrostatic and linear elastic problems. Eng Anal Boundary Elem 2004;28:1175–86.
[23] Lin PT, Shadid JN. Towards large-scale multi-socket, multicore parallel simulations: performance of an MPI-only semiconductor device simulator. J Comput Phys 2010;229:6804–18.
[24] Giraud L, Haidar A, Pralet S. Using multiple levels of parallelism to enhance the performance of domain decomposition solvers. Parallel Comput 2010;36:285–96.
[25] Sleijpen GLG, Fokkema DR. BICGSTAB(L) for linear equations involving unsymmetric matrices with complex spectrum. Electron Trans Numer Anal 1993;1:11–32.
[26] Zhang S-L. GPBi-CG: generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems. SIAM J Sci Comput 1997;18:537–51.
[27] Chen K. Matrix preconditioning techniques and applications. Cambridge, UK: Cambridge University Press; 2005.
[28] Saad Y, van der Vorst HA. Iterative solution of linear systems in the 20th century. J Comput Appl Math 2000;123:1–33.
[29] Saad Y, Schultz MH. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Star Comput 1986;7:856–69.
[30] Fletcher R. Conjugate gradient methods for indefinite systems. Lecture notes in mathematics 506. Berlin: Spriger-Verlag; 73–89.
[31] Freund RW, Nachtigal NM. QMR: a quasi-minimal residual method for non-Hermitian linear systems. Numer Math 1991;60:315–39.
[32] Chen K. On a class of preconditioning methods for dense linear systems from boundary elements. SIAM J Sci Comput 1998;20:684–98.
[33] Mittal RC, Al-Kurdi AH. An efficient method for constructing an ILU preconditioner for solving large sparse nonsymmetric linear systems by the GMRES method. Comput Math Appl 2003;45:1757–72.
[34] Araujo FC, Silva KI, Telles JCF. Generic domain decomposition and iterative solvers for 3D BEM problems. Int J Num Methods Eng 2006;68:448–72.
[35] Araujo FC, Gray LJ. Evaluation of effective material parameters of CNT-reinforced composites via 3D BEM. Comp Mod Eng Sci 2008;24(2):103–21.
[36] Araújo FC, Gray LJ. Analysis of thin-walled structural elements via 3D standard BEM with generic substructuring. Comp Mech 2008;41:633–45.
[37] 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;26:231–48.
[38] Kane JH. Boundary element analysis on vector and parallel computers. Comput Syst Eng 1994;5:239–52.
[39] Kamiya N, Iwase H, Kita E. Parallel implementation of boundary element method with domain decomposition. Eng Anal Bound Elem 1996;18:209–16.
[40] Kamiya N, Iwase H, Kita E. Performance evaluation of parallel boundary element analysis by domain decomposition method. Eng Anal Bound Elem 1996;18:217–22.
[41] Lu X, Wu W-L. A new subregion boundary element technique based on the domain decomposition method. Eng Anal Bound Elem 2005;29:944–52.
[42] Araujo FC, d'Azevedo EF, Gray LJ. Constructing efficient substructure-based preconditioners for BEM systems of equations. Eng Anal Bound Elem 2011;35:517–26.
[43] Chen XL, Liu YJ. Square representative volume elements for evaluating the effective material properties of carbon nanotube-based composites. Comput Mat Sci 2004;29:1–11.