Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 7085-7094 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

EVALUATION OF EFFECTIVE PROPERTIES OF HETEROGENEOUS MEDIA THROUGH A GENERAL PURPOSE GRAPHICS PROCESSING UNIT (GPGPU) BASED ALGORITHM

Bárbara de M. Quintela, Michèle C. R. Farage and Marcelo Lobosco

Mestrado em Modelagem Computacional, Universidade Federal de Juiz de Fora, 36036-330, Juiz de Fora, MG, Brasil, http://www.ufjf.br/mmc

Keywords: Homogenization, Finite Elements Method, Parallel implementation, GPGPU.

Abstract. The aim of this paper is to present the application of a parallel Finite Elements Method (FEM) code based on General Purpose Graphics Process Units (GPGPU) to the evaluation of effective mechanical properties of periodic media. The multiscale technique adopted herein is the Asymptotic Expansion Homogenization (AEH), which demands considerable computational effort even for simple geometrical and mechanical material models. In this work, the AEH technique is implemented on a GPGPU basis. The preliminary results indicate a considerable speedup in comparison with a sequencial version of the employed code.

1 INTRODUCTION

The Asymptotic Expansion Homogenization (AEH) is a multiscale technique applied to the estimation of effective properties of heterogeneous media with periodic microstructure.

During a preliminary study (Ferreira et al., 2007; Farage et al., 2008) developed at the NUMEC (Research Group on Computational Methods in Engineering, Federal University of Juiz de Fora, Brazil), the AEH was employed to evaluate the effective mechanical properties of plane periodic structures. Even though the adopted models were quite simple, the need for refined meshes resulted in considerable computational efforts, demanding rather long processing time. In order to reduce the processing time, the use of parallelization is essential.

In a previous work, it was developed a parallel version of the code using OpenMP. The parallelization provided up to 20% improvement in application performance. However, better improvement in performance was expected to be achieved: a memory contention problem contributed to limit the gains obtained by code parallelization (Quintela et al., 2010). This results encouraged the development of a new parallel version of that code using General-purpose Graphics Processing Units (GPGPUs).

GPGPUs were chosen due to their ability to process many streams simultaneously. The present work describes the new parallel implementation as well as evaluates its performance. Experimental results showed that the parallelization was effective in improving the simulator performance, yielding speedups up to 19. For comparison purposes, the OpenMP results are presented herein.

The paper is organized as follows. Section 2 presents the Asymptotic Expansion Homogenization technique, while section 3 presents its sequential implementation. In section 4 discusses the parallelization approach. Sections 5 and 6 present the experiments, and section 7 a conclusion.

2 THE ASYMPTOTIC EXPANSION HOMOGENIZATION TECHNIQUE

The *A*symptotic *E*xpansion *H*omogenization (AEH) is a multiscale technique based on the assumption that a heterogeneous medium may be represented by a homogeneous counterpart since its microstructure is periodic or repetitive. The technique is based on the uncoupling of the different scales of a material, extrapolating the results from inferior or heterogeneous scales in order to obtain global or homogenized properties (Sanchez-Palencia, 1980; Murad et al., 2001; Murad and Moyne, 2002; Romkes and Oden, 2004). As applying the AEH, a very important aspect is the definition of the geometric characteristics of the periodic cell, which is the smallest microstructural volume able to statistically represent the global constitutive behavior of the medium. By knowing the heterogeneous properties of a cell, it is assumed that those properties are periodically repeated over the structure. Basically, in periodic structures that present two scales, the AEH consists of uncoupling those scales into a *microscale* and a *macroscale*.

The homogenization deals with partial differential equations related to heterogeneous materials with periodic structure considering the assumption that the amount of periodic cells tends to infinity. The scale parameter ϵ is the characteristic dimension of the period. Figure 1, adapted from reference (Cioranescu and Donato, 1999), illustrates the physical meaning of periodicity and the scales uncoupling in a two-dimensional problem. In practical purposes, the main interest is getting to know the global behavior of a composite by considering that the heterogeneities dimensions - or that the scale parameter ϵ - tends to zero (Cioranescu and Donato, 1999). The coefficients of the describing differential equations are the characteristics of the studied material, depending on ϵ. Concerning a composite with a ϵ-periodic distribution, those coefficients

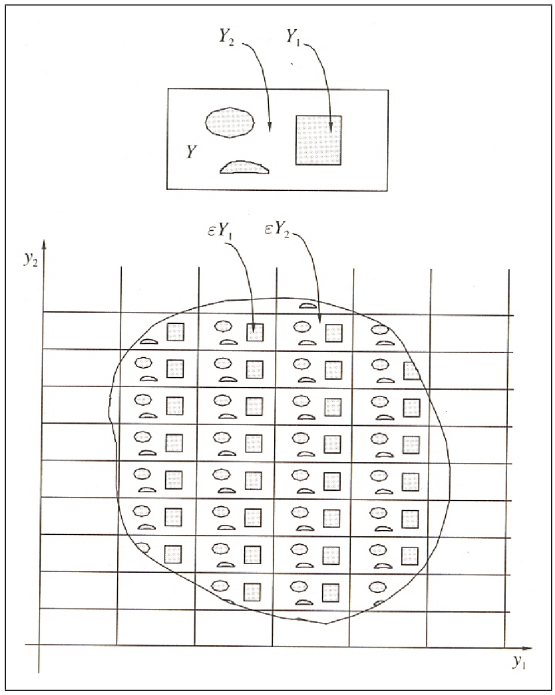

Figure 1: *Periodic structure adapted from (Cioranescu and Donato, 1999)*

are not easy to evaluate. Consideration of the limit case of $\epsilon \to 0$ leads to a *homogenized* problem, with constant coefficients that may be calculated with the help of numeric techniques, such as the Finite Element Method (Sanchez-Palencia, 1980; Cioranescu and Donato, 1999). In the present work, the AEH is employed to evaluate the homogenized elastic properties of composites. The primary variable is a smooth displacement function $u^i(x, y)$, which presents periodicity in the microscale Y, related to the macroscale by means of the equation: $y = x/\epsilon$.

2.1 AEH applied to Linear Elasticity

By considering a material whose microstructure is composed of multiple phases, periodically distributed over the body (Sanchez-Palencia, 1980; Chung et al., 2001), the periodic elastic material properties are defined by the following relation:

$$D^{\epsilon}_{ijkl} = D_{ijkl}\left(\frac{x}{\epsilon}\right) \tag{1}$$

where $(\,)^{\epsilon}$ denotes quantities related to the *actual* non-homogeneous medium; the function D^{ϵ}_{ijkl} stands for the material's properties variations in the heterogeneous microstructure Y. The linear elasticity problem is described by the equilibrium equation (Eq. 2), boundary conditions (Eq. 3) and (Eq. 4), strain-displacement relation (Eq. 5) and constitutive relation (Eq. 6):

$$\frac{\partial \sigma^{\epsilon}_{ij}}{\partial x^{\epsilon}_j} + f_i = 0 \text{ in } \Omega \tag{2}$$

$$u^{\epsilon}_i = 0 \text{ in } \partial_1\Omega \tag{3}$$

$$\sigma^{\epsilon}_i n_j = F_i \text{ in } \partial_2\Omega \tag{4}$$

$$\varepsilon_{ij}(u^{\epsilon}) = \frac{1}{2}\left(\frac{\partial u^{\epsilon}_i}{\partial x^{\epsilon}_j} + \frac{\partial u^{\epsilon}_j}{\partial x^{\epsilon}_i}\right) \tag{5}$$

$$\sigma^{\epsilon}_{ij} = D^{\epsilon}_{ijkl}\varepsilon_{kl}(u^{\epsilon}) \tag{6}$$

σ_{ij}^{ϵ} is the $_{ij}$ term of the internal stresses tensor and f_1 is the body force in the dominium Ω; u_i^{ϵ} is the displacement in direction i; n_j is the vector normal to the boundary $\partial\Omega$ and F_1 is the external force applied on the boundary and ε_{ij} is the $_{ij}$ term of the strains tensor. The displacements are approximated by an asymptotic series in ϵ, given by equation 7:

$$u_i^{\epsilon}(x^{\epsilon}) = u_i^{(0)}(x,y) + \epsilon u_i^{(1)}(x,y) + \epsilon^2 u_i^{(2)}(x,y) + \dots \tag{7}$$

where $u_i^{(0)}$ is the macroscopic displacement and $u_i^{(1)}, u_i^{(2)}, \dots$ stand for the periodic displacements in more refined scales. As the heterogeneous *actual* medium is represented by two coordinate systems (x and $y = x/\epsilon$), the derivatives originally in x^{ϵ} must be expanded in a chain rule given by:

$$\frac{\partial}{\partial x_i^{\epsilon}} = \frac{\partial}{\partial x_i} + \frac{1}{\epsilon}\frac{\partial}{\partial y_j} \tag{8}$$

In order to obtain the uncoupled equations that describe the microscale and the macroscale problems, the displacement u_i is replaced by equation 7 in the set of equations 2 to 6. The basis of the approximation is the assumption of $\epsilon \to 0$, indicating that the number of periodic cells tends to infinity and the actual non-homogeneous structure is then approximated by a homogeneous one. In order to validate such an approximation, the resulting coefficients of ϵ with negative exponent must be identically nulls. That leads to the conclusion that the homogenized solution $u^{(0)}$ is constant over the microscopic scale ($u^{(0)}$ does not depend on y), as indicated by the following equation:

$$\frac{\partial}{\partial y_j} D_{ijkl}\left(\frac{\partial u_k^0}{\partial y_l}\right) = 0 \tag{9}$$

The homogenized elastic properties tensor is, finally, given by expression 10:

$$D_{ijmn}^h = \frac{1}{|Y|}\int_Y D_{ijmn}\left[\delta_{im}\delta_{jn} + \frac{\partial \xi_i^{mn}}{\partial y_j}\right] dy, \tag{10}$$

The detailed description of the AEH formulation is given in reference Sanchez-Palencia (1980).

3 AEH IMPLEMENTATION

3.1 Program Description

This work derives from three previously published studies (Ferreira et al., 2007; Farage et al., 2008, 2009) in which the HEA2D program was introduced and validated for the evaluation of homogenized elastic properties of plane periodic cells. Numerical results were compared to experimental data obtained for lightweight aggregate concretes (Ke et al., 2006b,a), showing good agreement. The current versions of the program evaluate the homogenized elastic tensor of two-dimensional cells through the Finite Element Method (FEM), employing six-noded triangular elements. The HEA2D application was initially implemented in the integrated technical computing environment MATLAB©, from which some preliminary results were obtained. Those experiments indicated that complex meshes severely hurt performance, and it was decided to port the application to the C programming language. That was a first step to the generalization of the code to the solution of 3D problems, which are rather more time-consuming than the current 2D version.

The HEA2D inputs are the geometrical and mechanical characteristics of the cell and the output is the effective elastic tensor for the material. The following steps are performed to calculate the homogenized properties (Chung et al., 2001):

1. Data input: Information about the finite element mesh, boundary conditions of the periodic cell and mechanical properties of their α phasis (elastic modulus E_α and Poisson's ratio ν_α).

2. Assemblage of the stiffness matrix $[K]$ and the independent tensor $[F]$ (right-hand side of the equilibrium equation).

$$[K] = \left[\sum_{i=1}^{nelm} B^T D B J\right] \tag{11}$$

$$[F] = \left[\sum_{i=1}^{nelm} B^T D J\right] \tag{12}$$

where *nelm* is the number of elements in the mesh, ***B*** is the derivation tensor, ***D*** is the local elastic property tensor (for each α phase that composes the microstruture) and ***J*** is the Jacobian tensor, which relates the coordinate system to the parametric representation of the geometry;

3. Solution of the linear equations system. The system of equations is solved to obtain the elastic corrector tensor ***U***:

$$\left[\sum_{i=1}^{nelm} B^T D B J\right][U] = \left[\sum_{i=1}^{nelm} B^T D J\right] \tag{13}$$

4. Determination of the homogenized property tensor, by means of the averaging procedure described by Eq. (10), rewritten herein as:

$$[D_{ef}] = \frac{1}{Y}\left[\sum_{i=1}^{nelm} D(I - BU)|J|\right] \tag{14}$$

where Y is the total body volume, D is the local elastic property tensor (for each phase that composes the microstruture), I the identity matrix, B is the derivation tensor, U is the elastic corrector tensor and J is the Jacobian tensor, which relates the coordinate system to the parametric representation of the geometry.

The C version of HEA2D (Quintela et al., 2009) was the basis to the parallel version, as described in the following sections.

4 PARALLEL VERSION

As expected, during the performance evaluation of the sequential code, the resolution of the system of algebraic equations from the finite element aproximation was identified as one hot spot. The system is represented by a matrix with dimensions gll x gll and a matrix of independent terms with dimensions gll x 3, where the unknown is a matrix with gll x 3. The dimension gll means the total number of degrees of freedom on the global structure after applying periodic boundary conditions. The function that solves the system of equations is called *solve* and it was the part of the code which was parallelized. On the sequential execution to solve the unknown matrix, the *solve* function calls three times the resolution of the chosen direct method (QR, LU or Cholesky), each one for each column of the unknown matrix to be solved.

4.1 OpenMP

The first parallel version of HEA2D (Quintela et al., 2010) was implemented using OpenMP (*Open Specifications for Multi Processing*) (Chapman et al., 2007). OpenMP offers a programming interface for shared memory parallel machines. The programmer uses compilation directives to identify the portions of the source code that should be executed in parallel. The programmer can also specify how the code should be executed.

In the OpenMP version of the code (Quintela et al., 2010), methods for solving the three systems of equations were called concurrently using the *sections* directive. The parallelization provided up to 20% improvement in application performance. A detailed analysis has shown that a memory contention problem was negatively impacting performance. Due to this fact, a new parallel version of the code was developed using General-purpose Graphics Processing Units (GPGPUs).

4.2 General-Purpose Computing on Graphics Processing Units - GPGPU

NVIDIA's CUDA (Compute Unified Device Architecture)(NVIDIA, 2006) is a massively parallel high-performance computing platform on GPGPUs. CUDA includes C software development tools and libraries to hide the GPGPU hardware from programmers.

In GPGPU, a parallel function is called kernel. A kernel is a function callable from the CPU and executed on the GPU simultaneously by many threads. Each thread is run by a *stream processor*. They are grouped into blocks of threads or just blocks. A set of blocks of threads form a grid. When the CPU calls the kernel, it must specify how many threads will be created at runtime. The syntax that specifies the number of threads that will be created to execute a kernel is formally known as the execution configuration, and is flexible to support CUDA's hierarchy of threads, blocks of threads, and grids of blocks.

Since all threads in a grid execute the same code, a unique set of identification numbers is used to distinguish threads and to define the appropriate portion of the data they must process. These threads are organized into a two-level hierarchy composed by blocks and grids and two unique coordinates, called *blockId* and *threadId*, are assigned to them by the CUDA runtime system. These two build-in variables can be accessed within the kernel functions and they return the appropriate values that identify a thread.

Some steps must be followed to use the GPU: first, the device must be initialized. Then, memory must be allocated in the GPU and data transferred to it. The kernel is then called. After the kernel have finished, results are transferred back to the CPU.

4.3 Concurrent Number Cruncher

The Concurrent Number Cruncher (CNC) library was chosen to parallelize the HEA2D code using CUDA. The CNC library is a high-performance preconditioned conjugate gradient solver on the GPGPU. The CNC is based on a general optimized implementation of sparse matrices using Block Compressed Row Storage (BCRS) blocking strategies for various block sizes, and optimized BLAS operations through massive parallelization, vectorization of the processing and register blocking strategies (Buatois et al., 2008).

The Jacobi preconditioned conjugate gradient algorithm requires a sparse matrix-vector product, a vector inner-product and vectors operations (Buatois et al., 2007; Shewchuk, 1994). These operations are performed in the GPGPU.

The HEA2D code was modified to use the CNC library. The CNC library was also adapted to solve each column of the system of equations at a time. It is shown bellow an example of

CNC solver method call from HEA2D where A is the stiffness matrix, b is one right hand side column and x one solution column.

```
ok=CNCGPUSolver::solve_cg(A,b,x,max_iter,epsilon,block_size);
```

The internal CNC method include CUBLAS (Compute Unified Basic Linear Algebra Subprograms) calls which are BLAS implementation on top of the NVIDIA CUDA driver. (NVIDIA, 2008) The Jacobi-preconditioned conjugate gradient iterations are performed on CPU and each matrix-vector product, vector inner-product and vector operations are accomplished parallel on GPU.

5 NUMERICAL EXPERIMENTS

This section presents the results for numerical experiments. The experiments were performed on a 1.2 GHz Intel Core i7-860 processor, with 8 GB RAM, 8 MB L2 cache and a GTX 285 card. The system runs Linux kernel 2.6.31. The *gcc* version 4.4.2 was used to compile all versions of the program.

5.1 Experimental Methodology

Aiming to compare the performance of the sequential and parallel versions of the program, three distinct types of periodic cells were considered to represent composite materials.

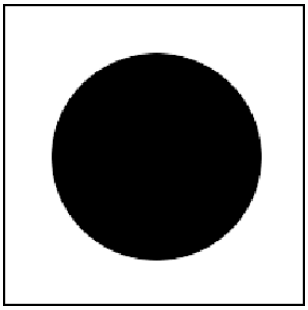

Figure 2: *circular*

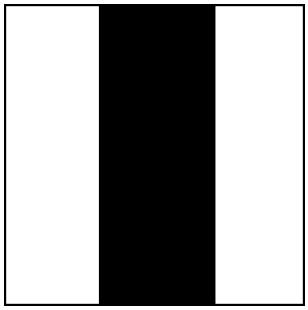

Figure 3: *laminated*

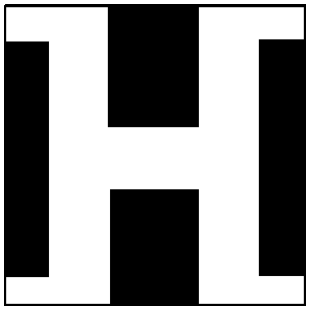

Figure 4: *short fiber*

Fig. 2 represents a square cell with a circular inclusion that may represent the cross section of a composite reinforced with long aligned fibers. This composite material exhibits isotropic behavior in the represented plane.

The cell shown in Fig. 3 is the elementary volume of a laminated composite. This structural conformation is largely employed to the construction of structures from prescribed properties (Torquato, 2001). Fig. 4 represents another important class of composite materials, standing for materials reinforced with short fibers. The two last examples, taken from reference Ghosh et al. (1995), present anisotropic behavior and the effective properties are not exactly evaluated - the quality of the approximated results depend strongly on the refinement of the finite elements mesh.

These three different types of periodic cells were used as benchmarks submitted 10 times to all versions of code, and the average execution times were used to calculate the speedups. All the standard deviation are less than 10%.

6 RESULTS

In all the numerical tests, the sequential and parallel versions of the program generated the same homogenized effective tensor so, both CPU and GPU numerical results were the same. This was expected because both codes employ single-precision arithmetic.

The circular inclusion mesh (Fig. 2) was analyzed with a $3,097$ nodes mesh. The mechanical properties of the components were adopted, for validation purposes, as: matrix elastic modulus $E_m = 1$, inclusion elastic modulus $E_i = 10$, matrix Poisson's ratio $\nu_m = 0.3$ and inclusion Poisson's ratio $\nu_i = 0.03$. The obtained effective isotropic tensor was $D_{11} = D_{22} = 1.205$, $D_{12} = D_{21} = 0.0713$ and $D_{33} = 0.5582$:

The laminated cell (Fig. 3) and the short fibers cell (Fig. 4) properties were taken from reference Ghosh et al. (1995), where the same multiscale technique was employed to analyze a composite with the following properties: $E_m = 72.5GPa$, $E_i = 400GPa$, $\nu_m = 0.33$ and $\nu_i = 0.2$.

Four different meshes were adopted for laminated and six for the short fiber cell with continuous refinement to approximate the results obtained by Ghosh et al. (1995) and it was observed (Quintela et al., 2010) that the results tend to the reference values. However, this additional refinement directly impacts performance as the size of the problem is increased.

Tables 1 and 2 present the speedup obtained for each of the analyzed cases. The speedup figures were obtained by dividing the sequential execution time of HEA2D by its parallel version: the OpenMP version running on CPU and the CUDA version running on GPGPU. Although the CPU and GPU architectures differs, the OpenMP speedups were presented for comparison purposes. The number of threads used for parallel regions in the OpenMP version is equal to three, so the maximum speedup this version of the code could achieve is 3.

	3,097 (circular)	2,775	9,171	10,877	15,445
$CUDA$	1.31	2.50	6.02	6.60	6.76
$OpenMP$	1.90	1.94	2.33	2.37	2.37

Table 1: Speedups for the circular and laminated meshes with different refinements. The circular values are presented in the first column. (number of nodes)

	5,175	7,735	15,125	21,903	33,485	45,353
$CUDA$	6.33	8.50	9.63	17.02	12.44	19.02
$OpenMP$	2.22	2.35	2.37	2.43	2.47	2.54

Table 2: Speedups for the short fiber meshes with different refinements. (number of nodes)

As indicated in Tables 1 and 2, the OpenMP version of the code achieved better speedup than the CUDA one only for the circular mesh. For the laminated mesh with 2,775 nodes, OpenMP also achieved a speedup similar to CUDA. Such result is due to the small number of nodes used in both meshes: working on the GPU in practices becomes inefficient when only small portions of data are transferred to the device and kernels are launched frequently, so that the initialization and data transfers to and from the GPU takes the biggest portion of the compute time. For all other configurations, the CUDA outperformed OpenMP by at least a factor of 2.5 times.

As one can observe, the more refined the mesh the better is the CUDA speedup. The best result a speedup of 19 was obtained when using a mesh comprising of 45,353 nodes. This fact indicates that better speedups may be achieved when larger system configurations are employed.

A key observation is that the new experiments presented in this work reinforce the initial suspicion that the poor OpenMP performance presented in a previous work (Quintela et al., 2010) was caused by memory contention. In fact, the best OpenMP speedup obtained previously was 1.2 but, it was obtained in a different hardware architecture: a 2 GHz Intel Core 2 Quad processor (Q8200), with 4 GB RAM and 2 MB L2 cache running Linux kernel 2.6.18. It is important to notice that the same version of the OpenMP code was executed in both machines. Also, another work (Xavier et al., 2010) have also identified the same issue and presented additional experiments that reinforces the suspect that a memory contention hazard affects the Intel Core 2 Quad architecture.

7 CONCLUSIONS

This work, presents a parallel implementation of the asymptotic expansion homogenization (AEH) technique using GPGPUs. The experiments indicate that the parallelization was effective in improving the performance, providing gains up to 19 times.

The new results presented in this paper reinforce the idea that the poor OpenMP performance obtained in previous work (1.2) was due to a memory contention problem related to the internal Intel Core 2 Quad processor architecture.

We intend to continue this work by developing a 3D version of the asymptotic expansion homogenization method. We hope that, by using this new parallel 3D version, the code will be able to analyze more complex and realistic models of composite materials.

ACKNOWLEDGMENT

The authors would like to thank FAPEMIG (APQ-00259-08), CNPq (479458/2008-1), CAPES (for the M.Sc. scholarship to the first author) and UFJF for supporting this study.

REFERENCES

Buatois L., Caumon G., and Lévy B. Concurrent number cruncher an efficient sparse linear solver on the gpu. In *High Performance Computation Conference 2007*. Lecture Notes in Computer Science (LNCS), 2007.

Buatois L., Caumon G., and Lévy B. Concurrent number cruncher a gpu implementation of a general sparse linear solver. *The International Journal of Parallel, Emergent and Distributed Systems*, 00(00), 2008.

Chapman B., Jost G., and van der Pas R. *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*. The MIT Press, 2007. ISBN 0262533022.

Chung P.W., Tamma K.K., and Namburu R.R. Asymptotic expansion homogenization for heterogeneous media: computational issues and applications. *Composites: Part A*, 32:1291–1301, 2001.

Cioranescu D. and Donato P. *An Introduction to Homogenization*. Oxford University Press, 1st edition, 1999.

Farage M.C.R., Beaucour A.L., Barra L.P.S., Ke Y., Sanabio D.F.S., and Ferreira A.P.G. Multiscale modeling of the elastic moduli of lightweight aggregate concretes: numerical estimation and experimental validation. *Revista da Escola de Minas*, 2009. Accepted for publication.

Farage M.C.R., Ferreira A.P.G., Barra L.P.S., and Beaucour A.L. Modelagem multiescala de concretos feitos com agregados leves. In *VIII Simpósio de Mecânica Computacional*. 2008.

Ferreira A.P.G., Farage M.C.R., and da Silva Barra L.P. Aplicação da homogeneização por

expansão assintótica à modelagem mecânica de corpos heterogêneos bidimensionais. In *X Encontro de Modelagem Computacional*. 2007.

Ghosh S., Lee K., and Moorthy S. Multiple scale analysis of heterogeneous elastic structures using homogenization theory and voronoi cell finite element method. *International Journal of Solids and Structures*, 32(1), 1995.

Ke Y., Beaucour A.L., Ortola S., Dumontet H., and Cabrillac R. Comportement mécanique des bétons de granulats légers ; étude expérimentale et modélisation. In *XXIVéme Rencontres du Génie Civil et Urbain, Construire : les nouveaux défis*. 2006a.

Ke Y., Ortola S., Beaucour A.L., Cabrillac R., and Dumontet H. Influence of aggregates on mechanical behavior of lightweight aggregate concrete: experimental characterization and modeling. In *First Euro Mediterranean in Advances on Geomaterials and Structures*. 2006b.

Murad M.A., Guerreiro J.N., and Loula A.F.D. Micromechanical computational modeling of secondary consolidation and hereditary creep in soils. *Comput. Methods Appl. Mech. Engrg.*, 190:1985–2016, 2001.

Murad M.A. and Moyne C. Micromechanical computational modeling of expansive porous media. *C. R. Mechanique*, 330:865–870, 2002.

NVIDIA. Cuda (compute unified device architecture). 2006.

NVIDIA. *CUDA - CUBLAS Library*, 2008.

Quintela B.M., Ferreira A.P.G., Farage M.C.R., and Lobosco M. Implementation of the asymptotic homogenization technique for the solution of plane multiphasic problems. In *XXX CILAMCE - Congresso Ibero-Latino-Americano de Métodos Computacionais em Engenharia*. 2009. Accepted for publication.

Quintela B.M., Ferreira A.P.G., Farage M.C.R., and Lobosco M. Parallel implementation of the aeh technique for the solution of plane multiphasic problems. In *Vetor (FURG)*. 2010. Accepted for publication.

Romkes A. and Oden J.T. Adaptive modeling of wave propagation in heterogeneous elastic solid. *Comput. Methods Appl. Mech. Engrg.*, 193:539–559, 2004.

Sanchez-Palencia E. *Non-homogeneous media and vibration theory*. J. Ehlers Lecture Notes in Physics, Springer, 1980.

Shewchuk J. An introduction to the conjugate gradient method without the agonizing pain. 1994.

Torquato S. *Random heterogeneous materials: microstructure and macroscopic properties*. Springer, 2001.

Xavier C.R., dos S. Amorim E.P., Amorim R.M., Lobosco M., Goldfeld P., Dickstein F., and dos S R.W. Performance evaluation of a reservoir simulator on a multi-core cluster. In *ICCSA proceedings*, pages 395–408. Lecture Notes in Computer Science (LNCS), 2010.