On calculation of effective material properties using RVE method by parallelized FE code for shell applications

Paweł Jarzębski[1], **Krzysztof Wiśniewski**[2*]
[1,2]*IFTR, Polish Academy of Sciences*
Pawińskiego 5B, 02-106 Warsaw, Poland
e-mail: pjarz@ippt.pan.pl [1], *kwisn@ippt.pan.pl* [2]

Abstract

The paper concerns parallelization of an FE code for machines with shared memory in order to speed up computations of large models. The loop was parallelized over elements in the research code FEAP using OpenMP, which required several modifications of the code and a specific method of synchronization for assembling, see [2] for details. The parallel solver was also applied.

Performance of the parallelized FEAP, designated as 'ompFEAP' is demonstrated in calculations of effective properties of materials using the RVE method. Two RVE examples are computed, for a heterogenous metal-ceramic composite and for a ceramic foam with a complicated micro-structure. We conclude that ompFEAP provides a very good speedup and efficiency causing only a small increase in memory usage.

Keywords: parallelization, OpenMP, finite element method, FEAP, RVE, shells

1. Introduction

The Representative Volume Element (RVE) method is a computational technique developed to determine average properties of composite, porous and cellular materials. In the two-scale approach, these properties can be subsequently used in computation of shells [3], and the solid-shell elements are particularly well suited for this purpose. In the RVE computations, typically, 3D finite elements are used, models have the size of millions of degrees of freedom (see [6]) and must be repeated many times while building a database of material properties.

Concurrent capabilities of multi-core processors and multi-machine clusters should be used, requiring advanced computational techniques. The minimum is to use a FE code with an available parallel solver, such as PARDISO, MUMPS, PaStiX, HSL and others, relatively easy to implement. The next step is parallelization of computations of the FE matrices and vectors, e.g. using OpenMP. In the third step, the so-called hybrid approach, with the domain decomposition is performed, e.g. by METIS, computations for sub-domains are scattered over a cluster of computers, which requires, e.g. MPI. Implementation of these techniques in a large and complicated existing serial FE code requires a significant programming effort.

In the paper a parallelization of the research code FEAP [1] is considered using the OpenMP library to make computations on a single machine (not a cluster) more effective. The parallelized code is designated as 'ompFEAP'.

Efficiency of parallelization is assessed by computing average properties of two advanced materials: (1) a metal-ceramic composite and (2) a ceramic foam. It is shown that the applied parallelization significantly reduces times of execution and causes only a small increase in memory usage.

2. Parallelization of FEAP

The FEAP [1] is a research FE code used at many universities, developed by Prof. R. L. Taylor at the UCB. A serial and a cluster (MPI) versions of FEAP exist, the version for one shared memory machine with multi-core processors does not, so the task was undertakenof developing it, see [2]. The parallelization of FEAP involves: (1) implementing a parallel loop over elements, and (2) adding a parallel solver.

2.1. *Parallelization of the loop over element*

The main difficulty in parallelization of the loop over elements is caused by an existing architecture of this code. Several modifications in FEAP were required, see [2] for details. The main features of the implementation are as follows:

1. assembling of elemental matrices is performed immediately after generating, without an additional storage,

2. only standard directives of OpenMP are used. The crucial part of the loop is the assembling of elemental matrices into a global matrix, in some papers designated as the 'reduction', as there is the possibility of the race condition. All the available directives for the mutual exclusion synchronization of OpenMP were implemented and tested; the directive ATOMIC provided the best scalability in the tests. Using this directive, many critical sections, exist being very small and their execution is supported by hardware.

2.2. *Parallel solver for the system of equations*

The parallel direct sparse solver HSL MA86, also suitable for indefinite matrices, was interfaced to ompFEAP. The source code is available for this solver, the method implemented is described in [4].

3. Numerical results

In order to assess the speedup of computations caused by the OpenMP parallelization of FEAP, calculations were performed for two RVE material models. In both models, the 3D 8-node standard (Lagrangian) finite element was used.

The first model, for a metal-ceramic composite, was obtained

* Corresponding author.

Table 1: Total time of parallel parts of ompFEAP and memory usage.

	Metal-ceramic composite		Ceramic foam	
Number of threads	Time [secs]	Memory [GB]	Time [secs]	Memory [GB]
1	325.03	5.47	453.76	15.45
12	37.46	6.22	50.89	17.69

in [6] from the micro-CT scans, transformed into an FE mesh shown in Fig. 1 using a specialized software. This model involves 0.39 million dofs.

The second model, for a ceramic foam, is shown in Fig. 2. It was developed in [5] to correspond to the micro-structure recorded on CT scans. This model involves 3.8 millions dofs.

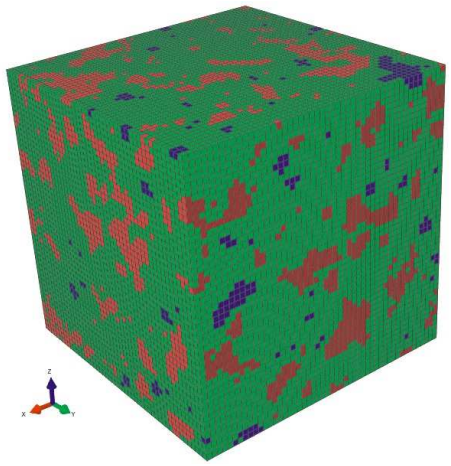

Figure 1: FE mesh for metal-ceramic composite.

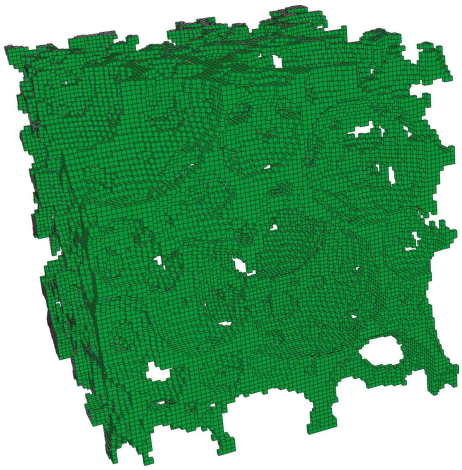

Figure 2: FE mesh for ceramic foam. (Much finer mesh than shown was used in computations.)

The computations were performed on the machine with 2 processors Xeon X5650 2.66GHz (6 cores each) and 24GB DDR3 1333MHz RAM memory. The code was compiled using Intel Compiler ver. 14.0.0, optimization flag 2.

In order to verify the correctness of parallelization, additionally to Intel Inspector 2015 tool, the computed average material parameters were compared to the reference paper results. They were identical regardless of the number of threads used.

The total time of parallel parts and the memory usage are reported in Table 1. The total time includes the time in the loop over elements (of matrix generation and assembling) and of the solution of the system of equations. Due to the metal-ceramic composite, the time is 8.7 times shorter for 12 threads (parallel execution) than for 1 thread (serial execution) and the memory usage is increased by 14%. Due to the ceramic foam, the time is 8.9 times shorter for 12 threads than for 1 thread and the memory usage increased by 14%.

Additionally, the scalability of particular parts of the parallel code is shown in Fig. 3 for the metal-ceramic composite. According to the loop over elements, which includes the matrix generation and assembling, the speedup is 10.97 while for the solver HSL MA86 the scalability is 8.64, both for 12 threads. The first speedup indicates that no more sophisticated approach to the parallelization of the loop over elements is required.

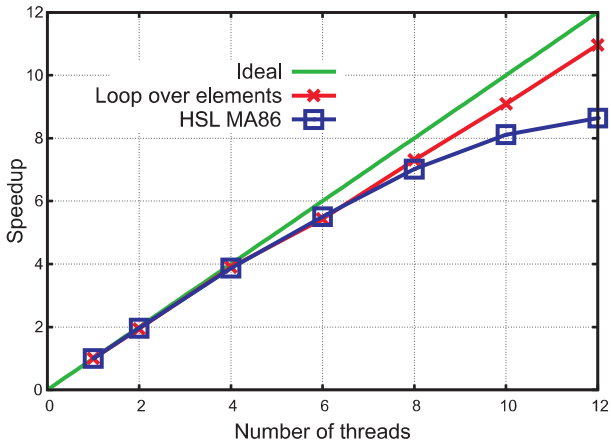

Figure 3: Scalability of ompFEAP for metal-ceramic composite.

References

[1] Taylor, R.L., *FEAP*, Ver. 8.4., 2014.

[2] Jarzebski, P., Wisniewski, K., Taylor, R.L., On paralelization of the loop over elements in FEAP, *Computational Mechanics*, 2014 (accepted).

[3] Wisniewski K., *Finite Rotation Shells*. Springer, 2010.

[4] Hogg, J.D., Scott, J.A., An indefinite sparse direct solver for multicore machines, *TR-RAL-2010-011*, 2010.

[5] Nowak, M., et al., On the reconstruction method of ceramic foam structures and the methodology of Young's modulus determination, *Archive of Metallurgy and Materials*, 58, pp. 1219–1222, 2013.

[6] Weglewski, W., et al., Comparative assessment of Young's modulus measurements of metald'ż˝ceramic composites using mechanical and non-destructive tests and micro-CT based computational modeling, *Computational Materials Science*, 77, pp. 19–30, 2013.