39th Solid Mechanics Conference
Zakopane, Poland, September 1–5, 2014

P153

ON PARALLELIZATION OF THE LOOP OVER ELEMENTS FOR COMPOSITE SHELL COMPUTATIONS

P. Jarzębski, K. Wiśniewski

Institute of Fundamental Technological Research, Warsaw, Poland

1. Introduction

The multi-scale models used in computation of composite shells require a significant computational power and, therefore, a finite-element code should take advantage of such techniques as: 1) parallel solvers, e.g. PARDISO, MUMPS, PaStiX, 2) parallelization of the loop over elements using e.g. OpenMP, and 3) domain decomposition and spreading tasks over a cluster of computers, using e.g. MPI. A significant programming effort is needed to convert a large and complicated existing FE code into a parallel one, especially when implementing the most effective hybrid approach.

In this paper, we focus on parallelization of the loop over elements using OpenMP standard [2], which we apply to the finite element program FEAP [1]. We shall discuss basic features of our implementation as well as demonstrate correctness and speed up ratio of the code. Several numerical examples of shell benchmarks will be presented.

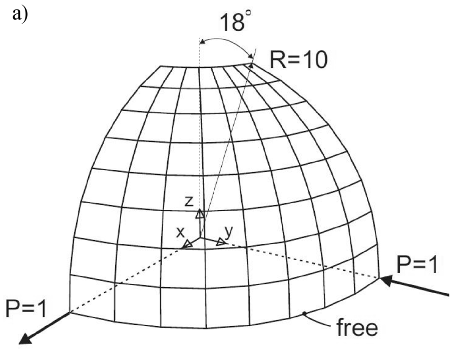
2. Parallelization of the loop over elements

Personal computers have processors with several (2–32) cores, which makes them shared memory architectures, for which communication is implicit. An appropriate parallelization technique for such architectures is a threading parallelism, which may be based on the OpenMP standard [2], specifying parallelization in terms of compiler directives, library routines and environment variables. OpenMP defines the ’fork-join’ parallelism, because it launches multiple parallel threads (fork) in parallel regions of the code and joins them into a single thread (the master one) for serial processing in non-parallel regions [3].

To parallelize FEAP, we apply the “*parallel do*” directive to the loop over elements, for which OpenMP creates a set of threads and distributes the iterations of the loop across them for parallel execution. The pivotal question is the choice as to whether a variable is shared or private. If too many variables are made private then OpenMP has to make additional and unnecessary work of initialization and copying extra variables for each thread. On the other hand, for shared variables more critical sections are needed to synchronize data. The additional difficulty is that FEAP uses many common blocks, and passes varying data through them. Finally, we have to properly treat critical sections, as they have a tremendous impact on the speed up ratio. These issues will be discussed in more detail during the presentation.

3. Pinched hemisphere shell example

Our OMP parallelization of FEAP was tested on the pinched hemispherical shell with 18° hole loaded by two pairs of equal but opposite external forces, applied in the plane $z = 0$, along the X and Y axes, so the shell undergoes strong bending, see [4] p. 445. The geometry of the shell and the load are shown in Figure 1a. The shell was computed for two thicknesses, $h = 0.4$ and $h = 0.04$ using three different elements. The numerical

a)


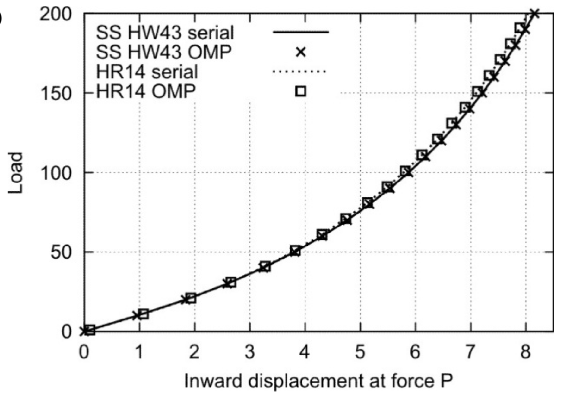
b)


Fig. 1. Pinched hemispherical shell. $E = 6.825 \times 10^7$, $v = 0.3$: a) initial geometry and load, b) displacement ($-u_y \times 100$) for serial and parallel version for thin shell.

tests were performed on 1 node of the cluster GRAFEN [6]. One node has two 6-core processors Xeon X5650 2.66 GHz and 24 GB RAM.

The thick shell ($h = 0.4$) was computed using our 3D displacement-type 8-node solid element using 10 elements through the shell thickness. The mesh consisted of $316 \times 316 \times 10$ elements (about 3.3 millions unknowns). The results for computation of a tangent matrix and a residual vector for a linear problem are given in Table 1, and we see a good speed up ratio with the number of threads.

Table 1. Time of computation and speed up ratio for thick shell.

Version	Serial	OMP, number of threads						
		1	2	4	6	8	10	12
Time [secs]	52.48	53.61	27.08	13.63	9.45	7.11	5.68	4.79
Speed up ratio	0.98	1	1.98	3.93	5.67	7.54	9.44	11.19

The thin shell ($h = 0.04$) was computed using two non-linear elements: our solid shell element HW43 as well as the shell element with 2 rotational dofs of FEAP [5] (which we denote HR14), to demonstrate that the user as well as the FEAP elements are operational in the parallel version. The Newton method was used The results of nonlinear analyses are given in Fig. 1b, and we see that both versions, the serial and the parallel one, give exactly the same results. This indicates that our implementation of the OpenMP standard in FEAP is correct indeed.

References

1. Taylor R.L. (2013). FEAP 8.3, http://www.ce.berkeley.edu/projects/feap.
2. OpenMP Architecture Review Board, OpenMP Application Program Interface (2013). http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf.
3. Pantale O. (2005), Parallelization of an object-oriented FEM dynamics code: influence of the strategies on the Speedup, *Advances in Engineering Software*, 361–373.
4. Wisniewski K. (2010). *Finite Rotation Shells*, Springer.
5. Simo J.C., Tarnow N. (1992). On a stress resultant geometrically exact shell model. Part VI 5/6 dof treatments, *Int. J. Numerical Methods in Engineering*, **34**, 117–164.
6. GRAFEN, http://info.grafen.ippt.pan.pl/