PARALLEL MATSOL LIBRARY FOR SOLUTION PROBLEMS OF CONTACT MECHANICS

Z. Dostál[a], T. Kozubek[a], V. Vondrák[a], M. Sadowská[a],
R. Kučera[b], A. Markopoulos[c], and T. Brzobohatý[c]

[a]*Dept. of Appl. Math., VŠB–Technical University of Ostrava, Czech Republic*
zdenek.dostal@vsb.cz, tomas.kozubek@vsb.cz, vit.vondrak@vsb.cz, marie.sadowska@vsb.cz

[b]*Dept. of Mathematics, VŠB–Technical University of Ostrava, Czech Republic*
radek.kucera@vsb.cz

[c]*Dept. of Mechanics, VŠB–Technical University of Ostrava, Czech Republic*
alexandros.markopoulos@vsb.cz, tomas.brzobohaty@vsb.cz

Abstract: The goal of this contribution is to present in a sense optimal algorithms for problems of contact mechanics. These algorithms are implemented in the MatSol library and they are based on FETI domain decomposition methods which are well known by its parallel and numerical scalability. The performance is illustrated on both model and real world problems.

Keywords: *MatSol, Contact problem, Domain decomposition, Numerical scalability, Parallel scalability*

1. Introduction

During last several years, our research team has been focused on development of scalable algorithms for contact problems in mechanics. These algorithms are based on FETI domain decomposition methods which are well known by their numerical and parallel scalability, i.e., we are able to solve resulting quadratic programming problems in $O(1)$ iterations and in $O(1)$ seconds independently on the problem size only by adding directly proportional number of processors [2]. All the algorithms were implemented into a new library that is developed in the Mathworks Matlab environment which is equipped with many helpful functions for mathematics, plotting, debugging, quick testing and efficient implementation. We call this library MatSol (MATlab SOLvers) [1]. Nowadays, it is our primary testing and developing library. To parallelize the algorithms we use Matlab Distributed Computing Engine which allows to run Matlab functions also on parallel computers. Hence, the MatSol has full functionality to solve efficiently large problems of mechanics.

2. Structure of the MatSol Library

Todays structure of the MatSol library looks as follows. The solution process starts from the model which is stored in the model database. Models may be converted to the model database from standard commercial and non-commercial preprocessors like ANSA, ANSYS, COMSOL, PMD, etc. The list of preprocessing tools is not limited and any new one can be simply plugged into the MatSol library by creating a proper database convertor. Preprocessing part continues in dependence on the solved problem. User can solve deterministic or stochastic problems, static or transient analysis, optimization problems, problems in linear and non-linear elasticity and contact problems with various friction models. To discretize our problems we can choose between finite and boundary element methods. Furthermore, FETI or BETI based methods are used as the domain decomposition approaches. The solution process could be run either in sequential or parallel mode, but the algorithms are implemented in such a way that the code is the same for both modes. MatSol library includes also tools for postprocessing of the results and advanced tools for postplotting. The results of the problem may be converted through the model database to the modelling tools for further postprocessing.

The above described structure of the MatSol library allows to override standard solvers in commercial and non-commercial finite element packages and substitute them by those implemented in MatSol. This gives a very useful alternative to users of commercial packages and a great tool for algorithm developers to test the new algorithms on the realistic problems.

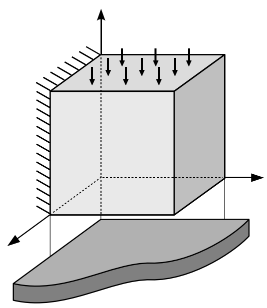

Fig. 1. Geometry and traction.

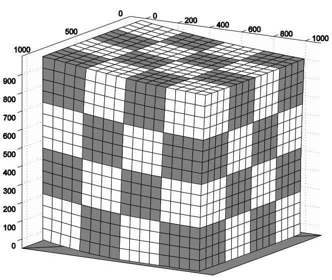

Fig. 2. Solution with traces of decomposition.

3. Solved Problems

Now we shall illustrate the performance of MatSol on both model and real world problems. All problems were solved on the computational cluster

HP BLc7000 with 9 nodes. Each node is equipped with 2 dual core AMD Opteron processors and 8GB RAM and interconnected by infiniband network. On this cluster we have installed 24 licences of Matlab distributed computing engine.

3.1. Cantilever cube over the obstacle

The first problem is a 3D coercive contact problem of the Signorini type. The elastic body is represented by the steel cube (see Fig. 1). The body is fixed in all directions along the left face and loaded by traction along the top face. The bottom face of the body may touch the rigid plane obstacle.

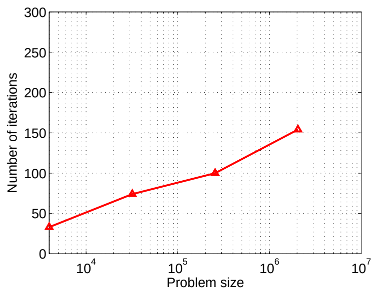

Fig. 3. Numerical scalability.

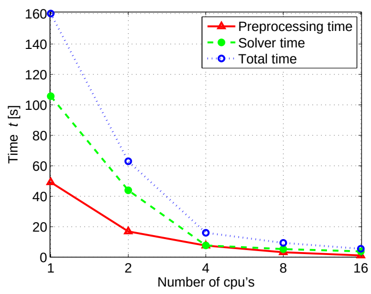

Fig. 4. Parallel scalability.

In Figure 2, we depict the deformed body together with the traces of decomposition. The numerical scalability of our algorithm is illustrated in Figure 3. We can observe that the number of iterations with increasing problem size increases only moderately in agreement with the theory. Finally, the parallel scalability is depicted in Figure 4, where we fix the number of primal variables and increase the number of partitions into subdomains accordingly to the number of used CPUs. The behaviour agrees with the theoretical results.

3.2. Real word problem: ball bearing

We have also tested our algorithms on real world problems. We considered the analysis of the stress in the ball bearing, see Fig. 5. The problem is difficult since the traction acting on the lower part of the inner ring is distributed throughout the nonlinear interface of the cage and balls to the fixed outer ring. The solution (Von Mises stress distribution in MPa) of the problem discretized by 1,688,190 unknowns (displacements) and decomposed into 700 subdomains using METIS (see Fig. 6) is depicted in Figures 7 and 8. It required 2,364 iterations and took 5,383 seconds to identify 20,843 active contact nodes. Though this number is not small, we were not able to resolve

Fig. 5. Ball bearing.

Fig. 6. Decomposed domain.

the problem by a commercial software, including ANSYS, without artificial combine elements which regularize the problem. Still we hope to improve our results, e.g., by enhancing standard FETI preconditioners.

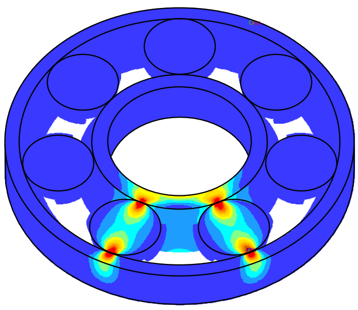

Fig. 7. Von Mises stress in the midplane.

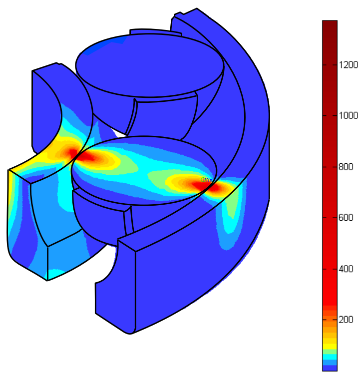

Fig. 8. Von Mises stress (zoom).

Acknowledgements

This work has been supported by the grant GAČR 101/08/0574 and by project of Ministry of Education of the Czech Republic #MSM6198910027.

References

[1] MatSol library. *http://www.am.vsb.cz/matsol.*

[2] Z. Dostál, *Optimal Quadratic Programming Algorithms, with Applications to Variational Inequalities*, 1st edition, Springer US, NY 2009, SOIA 23.