23rd International Conference
ENGINEERING MECHANICS 2017
Svratka, Czech Republic, 15 – 18 May 2017

# ON PARELLELIZATION OF LINEAR SYSTEM EQUATION SOLVER IN FINITE ELEMENT SOFTWARE

**M. Bošanský[*], B. Patzák[**]**

**Abstract:** *Numerical modelling using high performance computers brings in new opportunities in finite element computations. The contribution focuses on comparing the efficiency of different existing libraries to solution of large, sparse, non-symmetric systems of linear equations, based on direct or iterative algorithms. In particular, the direct solver from SuperLU library (X.S. Li, 1999), iterative solver from Portable, Extensible Toolkit for Scientific Computation (PETSc) are compared. Additionally, the performance and scalability of parallel SuperLU solver is studied, using OpenMP (Chapman, 2008, Barney, 2014) and POSIX threads (Nicols, 1996) based on shared memory model. Also, the scalability of PETSc solver on distributed memory model using Message passing interface (MPI) (Pacheco, 1997) is evaluated. The paper shows that parallelization can efficiently exploit the power of modern available hardware and significantly reduce the computation time.*

**Keywords: Parallel computing, Memory, Direct solver, Iterative solver.**

## 1. Introduction

Recent developments in computer hardware bring in the new opportunities in computational science. Traditional codes run sequentially on a computer with a single processing unit. The future of scientific computing seems to be in parallel computing, that allows to overcome limitations of traditional sequential processing units, where permit us to run simulation codes only sequentially. Parallelization can significantly reduce computational time by more efficient use of available hardware.

Parallel computing is based on an idea of dividing the work into the smaller tasks, which are solved concurrently by the simultaneous use of multiple computing resources (Barney, 2010). Parallel programs are more difficult to design, because parallelism introduces new sources of complexity. Communication, synchronization between the tasks, and load balancing are one of the most important problems to getting scalable parallel performance. Parallel computers can be classified, for example, by type of memory architecture. The main types of memory systems are shared, distributed, and hybrid memory systems (Hughes, 2003).

The Finite Element Method (FEM) actually represents a broad spectrum of techniques that share common features. The FEM is numerical method for finding solution to boundary value problems described by ordinary differential equations. The original differential equation or system of differential equations is converted to the algebraic system of equations. For the case of linear elasticity, the resulting system has a form

$$K * r = F \tag{1}$$

where **K** and **F** are global stiffness matrix and load vector, and **r** is vector of unknown displacements. One of the key feature of the FEM is that the stiffness matrix is typically positive definite symmetric matrix with sparse structure.

A system of linear equations is called sparse if only relatively small number of stiffness matrix values are non-zero. An efficient algorithm for solving linear system must exploit this property. There are different

[*] Ing. Michal Bošanský, Department of Mechanics, Faculty of Civil Engineering, CTU in Prague, Thákurova 7/2077; 166 29, Prague; CZ, michal.bosansky@fsv.cvut.cz

[**] Prof. Bořek Patzák, Department of Mechanics, Faculty of Civil Engineering, CTU in Prague, Thákurova 7/2077; 166 29, Prague; CZ, borek.patzak@fsv.cvut.cz

schemes for efficient storage of sparse matrices and solution of related linear problem. The problem is about solve equation (1) where **r** and **F** are just vectors, **K** is a sparse matrix, and even storing a dense matrix of **K** 's size would be prohibitively expensive. There are a number of different libraries out there that solve a sparse linear system of equations. This paper compares SuperLU library as a representative of shared memory model and PETSc as a distributed memory model representative. SuperLU contains a set of sparse direct solvers for solving large sets of linear equations. The kernel algorithm in SupeLU is sparse Gaussian elimination. First step of solution algorithm is a triangular factorization. In addition to complete factorization, SuperLU library also has limited support for incomplete factorization (ILU) preconditioner which approximately solves equation (1). Permutation matrices can be set up to improve sparsity, numerical stability and parallelism. The SuperLU routines appear in three different variants: sequential, multi-threaded (shared memory systems) and parallel (distributed memory systems).

PETSc consists of a variety of libraries similar to classes in C++ (Stroustrup, 1997) with an abstract interface to provide clean and effective implementations for solving partial differential equations. Different algorithms, for example Krylov subspace methods (KSP) or truncated Newton methods are provided. After creating the matrix and vectors that define linear system (1) the user can then use KSP to solve system of linear equations. PETSc supports MPI as a distributed memory parallelism.

The speed-up of computation is one of the most important goals of parallel computing (Patzak, 2010). Effective parallel algorithm is considered as scalable, where with increasing number of processing threads the execution time is monotonically decreasing, ideally in a linear trend. However, when individual computing threads are mutually dependent, the ideal scalability is difficult to obtain due to the overhead of parallel algorithm (synchronization, locking) and due to the fact that some parts are essentially serial.

## 2. Implementation

Comparison of different existing libraries developed to the solution of sparse linear systems is the subject of presented paper. In this contribution, we compare efficiency of serial direct and iterative solvers on selected benchmark problems. The comparison was made using OOFEM finite element solver (Patzak, 2000). The interface to SuperLU library has been implemented, consisting of new classes to represent SuperLU solver and sparse matrix storage. The interface to PETSc already exists. SuperLU is used with expert driver based on OpenMP or POSIX threads. The driver advantageously uses symmetry of matrix in linear system. Ordering of the columns of matrix is optimized using as multiple minimum degree applied to the structure of $A^T + A$. PETSc parallel solver based on distributed memory has been configured to use conjugated gradient solver from KSP to solve linear system of equations.

## 3. Results

The individual approaches have been evaluated using a benchmark problem of a 3D finite element model of nuclear containment (Jete 250k) and a benchmark problem representing 3D finite element model of porous microstructure (Micro 250k). The nuclear containment mesh consists of 87320 nodes and 959700 tetrahedral elements with linear interpolation. The total number of equations was 260322. The model of porous microstructure mesh consist of 85184 nodes and 79507 brick elements with linear interpolation. The total number of equations was 255552. Both structures has been loaded with self-weight, resulting in nonzero contribution of every element to the external load vector. The benchmark problems are characterized by different sparsity of system matrix. The model of porous microstructure has significantly more nonzeros members than model of nuclear containment, which is really sparse problem. Number of nonzeros members in model of nuclear containment is 433555173 and to solve the problem with direct solver required 4.3 Gb of system memory. Number of nonzeros members in the model of porous microstructure is 1528773527 and to solve the problem with direct solver required 15.3 Gb of system memory. The PETSc solver needed 4673 iterations to solve problem of nuclear containment and 14687 iterations to solve porous microstructure. The relative solution relevance used was $1.e^{-6}$. The individual approaches have been tested on two Linux workstations (running Ubuntu 14.04 OS) with the two CPU Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz and 126GB RAM. The two CPU units consists of eight physical and sixteen logical cores, allowing up to thirty-two threads to run simultaneously on one workstation. All the tests fit into a system memory.

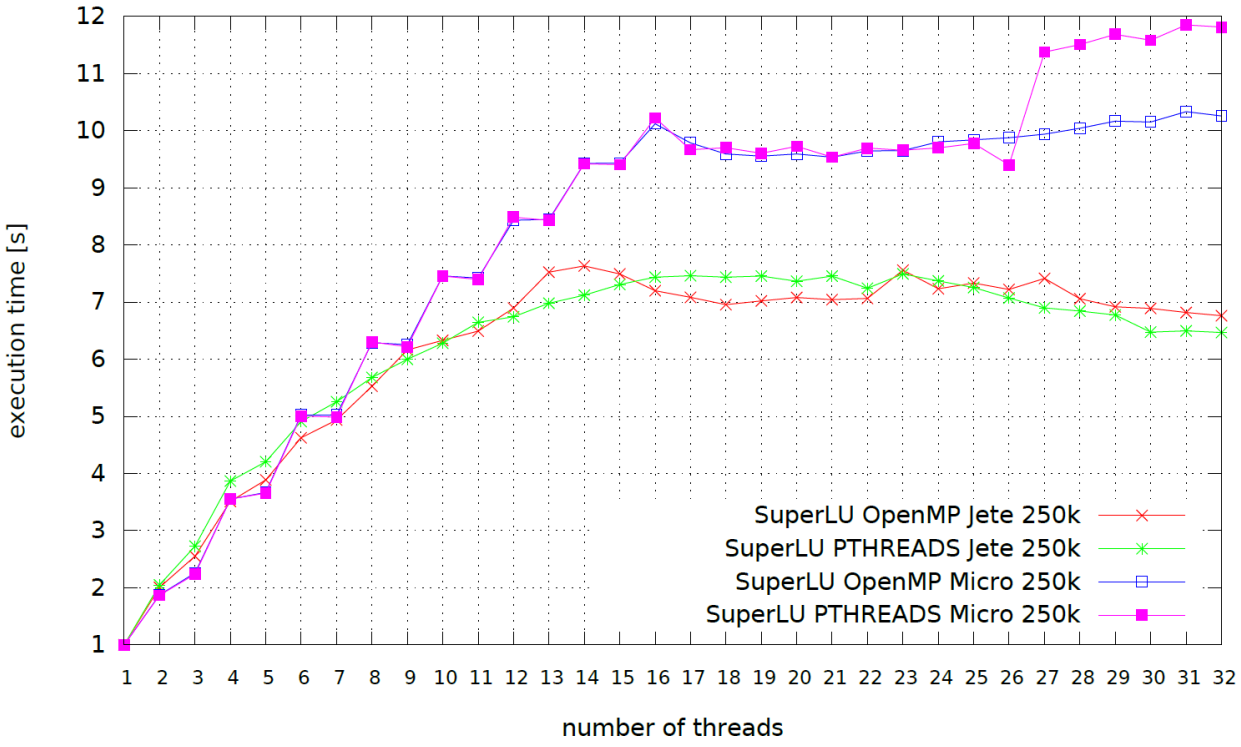


*Fig. 1: Speed-up of linear system solution of direct SuperLU solver based on shared memory.*

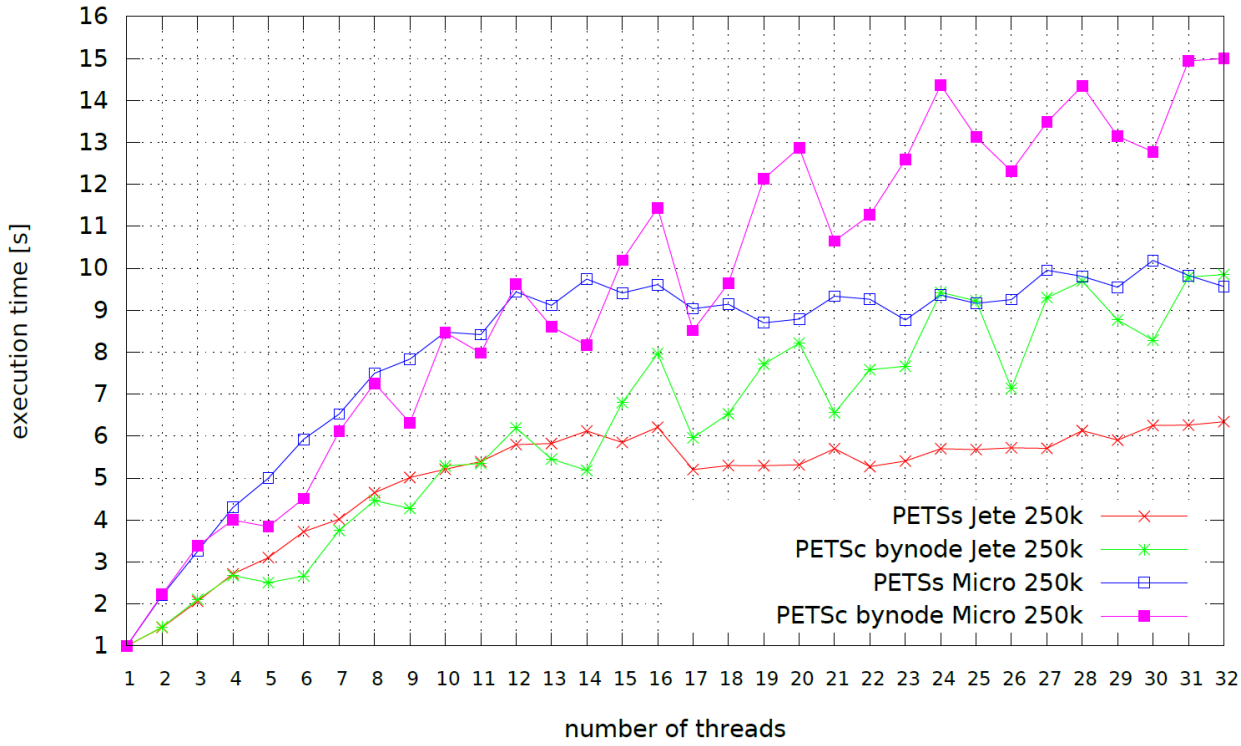


*Fig. 2: Speed-up of linear system solution of iterative PETSc solver based on distributed memory.*

Speed-up to solve linear system of equations using direct solver based on shared memory model (SuperLU with using OpenMP and POSIX threads) and iterative solver based on distributed memory (PETSc with using MPI) are presented in Fig. 1 and Fig. 2. The reported executions times have been obtained as an average of five consecutive runs. It can be clearly observed, that time needed for the execution is reduced with increasing number of threads. This can be caused by reading the local memory bus limits on single workstation.

For the direct solver SuperLU needed considerably more time to solve problem than iterative solver PETSc on both benchmark problems. The computational times for 17 threads and more are similar and no significant speed-up progress is achieved.

The results of PETSc solver executed on one workstation (32 threads with hyper-threading) and on two workstations (16 threads on workstation 1 and 16 threads on workstation 2 marked bynode keyword in Fig. 2) show that there are slight differences between using the solver on one or two computers caused by an additional overhead due to communication over network. PETSc solver using communication over the network on two workstations (by node) show better speed-up proportionally for higher number of threads (17 - 32 threads) than solving the same problem only on one computer. However, using both parallel solvers one can achieve better performance compared to serial code.

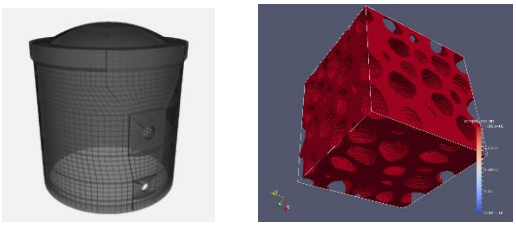

*Fig. 3: Benchmarks problems: on the left 3D finite element model of nuclear containment (Jete 250k) and on the right of 3D finite element model of porous microstructure cube (Micro 250k).*

## 4. Conclusions

Different approaches to solve system of linear equations with direct and iterative solvers have been implemented and their efficiency compared on tests examples. Results show that there are significant differences between performance of direct and iterative solvers for considered benchmarks. The PETSc iterative solver based on distributed memory has the better performance than SuperLU direct solver based on shared memory. However, the presented conclusions are related to selection of benchmark problems. In general, the possibility to have both iterative and direct parallel solvers can be an advantage. It is therefore important to have different possibilities available in general purpose finite element code.


## Acknowledgement

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS15/038/OHK1/1T/11 “Advanced algorithms for numerical modelling in mechanics of structures and materials”.


## References


Li, X.S., Demmel, J.W., Gilbert, J.R., Grigori, L., Shao, M. and Yamazaki, I. (1999) Lawrence Livermore National Laboratory, SuperLU Users’ Guide Program Interface, September 1999, LBNL-44289, http://crd.lbl.gov/xiaoye/SuperLU/.

Balay, S., Abhyankar, S., Adams, M.~F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp W~D., Kaushik, D., Knepley, M.~G., Curfman McInnes, L., Rupp, K., Smith, B.~F., Zampini, S. and Zhang, H. (2016) Argonne National Laboratory, ANL-95/11 - Revision 3.7, http://www.mcs.anl.gov/petsc.

Chapman, B., Jost, G., Pas, R.V.D. (2008) Using OpenMP - portable shared memory parallel programming. Morgan Kaufmann Publishers, Inc.

Barney, B., Lawrence Livermore National Laboratory (2014) OpenMP - An Application Program Interface. https://computing.llnl.gov/tutorials/openMP/.

Nicols, B., Buttlar, C. and Farrell, J.P. (1996) Pthreads Programming, ISBN 1-56592-115-1, O‘Reilly and Associates.

Pacheco, P.P. Univ. of San Francisco, Univ. of San Francisco, A User's Guide to MPI, 1997

Hughes, C., Hughes T. (2003) Parallel and Distributed Programming Using C++. ISBN 0-13-101376-9, Pearson Education.

Barney, B. (2010) Task scheduling for parallel systems. https://computing.llnl.gov/tutorials/parallel comp/.

Stroustrup, B. (1997) The C++ programming language. Murray Hill, AT Labs Murray Hill, New Jersey.

Patzak, B. (2010) Parallel computations in structural mechanics. ISBN 978-80-01-04565-7, Czech Technical University in Prague.

Patzak, B. OOFEM, 2000, http://www.oofem.org.