Efficiency upgrade of different desktop-type computers when solving numerical problems

Artem Sergeevich Khoroshev, Vladimir Sergeevich Puzin, Denis Aleksandrovich Tchoutchkin, Ekaterina Viktorovna Shevchenko, Denis Vladimirovich Batishchev, Andrey Arturovich Gummel

Platov South-Russian State Polytechnic University (Novocherkassk Polytechnic Institute), Novocherkassk, Russia

Abstract. The article deals with the problems of direct solution methods for ill-conditioned systems of higher order linear equations when simulating electromagnetic fields by finite difference method, using computer systems with limited hardware resources, such as personal computers. Author gives recommendations on selection of direct solution techniques and ordering methods, as well as the software for implementing these methods to ensure the effective use of available hardware resources and increase the speed of numerical problems solution.

[Khoroshev A.S., Puzin V.S., Tchoutchkin D.A., Shevchenko E.V., Batishchev D.V., Gummel A.A. **Efficiency upgrade of different desktop-type computers when solving numerical problems.** *Life Sci J* 2014;11(2s):141-145] (ISSN:1097-8135). http://www.lifesciencesite.com. 24

Keywords: Ill-conditioned systems, simulating electromagnetic fields, selection of direct solution techniques and ordering methods.

Introduction

Numerical simulation of electromagnetic field, using the finite element method (hereinafter FEM), is currently used very widely. After building a finite element mesh, most of the computational work is required to solve the simultaneous linear algebraic equations (SLAE), arising in the discretization of the equations, describing the simulated physical processes. Solution of the corresponding SLAE in case of small order (up to 200), usually does not cause any difficulties when using the majority of both iterative and direct methods. However, increasing the order of the matrix and/or the conditioning number may cause complexity due to insufficient convergence rate of iterative methods [1] or a sharp increase in solution time, if based on direct solution methods due to their high computational complexity.

The need to solve numerical problems using FEM requires partitioning the computational domain into a large number of finite elements and may occur in the simulation of electromagnetic and other fields in the computational domains, containing explicit geometric heterogeneity that leads to the need for local multiple condensation of a finite element mesh. The same technique may be used when approximating computational domain objects with very high accuracy. In such cases, the number of finite elements in a mesh of the computational domain may reach several tens of millions. Such values are typical for computational domains that are based on complex non-symmetric geometrical models. This requires the solution of SLAE of a specific order. In this case, simulation of electromagnetic fields may be accompanied by complications due to limitations in available computing resources. This is relevant to personal computers used by design engineers, even in the case of high-performance computing systems, so-called "workstations".

High order SLAE is usually solved by iterative methods, because direct methods exhibit a high computational complexity and require a lot of random access memory (RAM). The efficiency of iterative methods for solving SLAE has been well studied, and areas of their effective application are well identified [1, 2]. However, in case of system matrices with a large conditioning number (of 6-8th order), the iterative methods may require the fulfillment of a very large number of iterations for providing a given accuracy when solving SLAE. This will require considerable time, which is especially true for system matrices of the order of several hundreds of thousands or even millions. In such cases it may be advisable to apply direct methods for solving SLAE or a combination of the direct method, as a preconditioning technique, and iterative method for solving SLAE. Efficiency and productivity of such problems solution, when using direct solution methods, significantly depends on the choice of mathematical methods, algorithms and proper software, used in the mathematical simulation. Further we consider special aspects when choosing mathematical methods and algorithms for solving the high resolution problems (based on the finite element mesh, containing several millions finite elements) using direct solution methods of SLAE on the example illustrating selection of a general environment for the treatment of discrete problems (GetDP) [3, 4].

Main part.

Environment for solving discrete problems of GetDP, if used the PETSc toolkit, provides the possibility of using direct methods for solving SLAE: LU - decomposition, QR-decomposition, and Cholesky decomposition. Besides, there is an opportunity to select appropriate software (solver) to implement these methods. In addition, direct methods for solving SLAE can be used as SLAE preconditioning methods with subsequent solution of SLAE using Krylov subspace methods, such as GMRES, CG, BiGGStab, MINRES, and other [5]. Next, we will consider the selection features of software components that implement the mathematical operations, required for solving SLAE using MUMPS solver.

Using MUMPS involves the employment of LU-decomposition and the Cholesky decomposition as both direct solution methods and preconditioning techniques to be used with iterative methods for solving SLAE by means of PETSc tools.

Direct methods of solving SLAE are characterized by a high computational complexity. The LU-decomposition is characterized by computational complexity, which is proportional to the cube of the input data volume:

Cholesky decomposition is also characterized by the same computational complexity, proportional to the cube of the input data volume, though requires half the number of operations in comparison with LU-decomposition. Accordingly, the increase of computational complexity when increasing the scale of the numerical problem leads to a nonlinear increase in task-time. Furthermore, the use of direct solution methods for SLAE places high demands on the computer storage capacity, required to store data that is generated during the execution of the required operations. However, the solution of engineering problems often requires rapid results of concomitant numerical simulation of various processes, taking place in designed devices. This imposes a limit on the time, allowed for the numerical solution of the problems within the numerical simulation.

When solving numerical problems, including the direct methods of solving SLAE, one can shorten task-time either by using hardware with more computing power, or by selecting the software that allows the best use of processing power of available hardware.

To perform operations on matrices (for LU-decomposition and Cholesky decomposition), MUMPS uses instructions and a set of basic routines, available in the BLAS/LAPACK software (shared libraries) [6]. This allows the user to choose the alternative implementation of BLAS, which will provide the best performance on the available hardware. This remains pertinent for modern architecture of central processor units (CPU) of personal computers (PC), enabling the use of instructions that make it possible to increase significantly the processing speed of certain operations, inherent to the process of solving a numerical problem. Such procedures may include operations with vectors (set of AVX instructions) and mixed addition-multiplication of floating point numbers (set of FMA and FMAC instructions). Use of these instructions allows significant reduction in task-time when solving the numerical problems due to better use of available hardware resources.

Thus, for example, the use of alternative implementations of BLAS, such as ATLAS, GotoBLAS, MKL, and others can reduce task-time of LU-decomposition by factor of 2-10 depending on the problem's parameters, as well as related software and hardware [7]. The alternative implementation of the BLAS routines set, allowing for the application of AVX and FMA instructions, makes it possible to achieve even higher performance.

The reult shows a functional connection between the calculating speed, when solving a test problem (3 iterations) by means of GetDP, using a variety of alternative implementations of BLAS and different hardware, and a number of execution threads of BLAS routines. Thus, the use of FMA4 instructions reduced the whole task run-time by 12-15%, as compared with the employment of routines, presented in OpenBLAS, providing the same performance as MKL and ACML without additional instructions [8].

It should also be noted that to provide maximum performance one must consider the peculiarities of architecture of the processors used. For example, the architecture of Bulldozer (and Pile driver, which is its further development), embodied in the FX series of AMD processors, is characterized by modular installation of major functional block. Each module combines two blocks to operate with integers and one block to operate with floating-point numbers. Also, module includes one decoder with instructions to operate with floating point numbers. In this regard, the number of threads using FPU, which can efficiently utilize the available hardware resources, should not exceed the number of modules in the processor. Figure 2 shows the task-time of the test problem depending on the number of threads and used software components in the Pile driver-based triple core processor. It should be noted that when using OpenBLAS, a certain decrease in performance takes place with increasing a number of threads, when using more than three threads, as well as sharp, repeated slowdown, when using ACML with supporting FMA4 instructions. Such a sharp loss in efficiency is due to the insufficient number of relevant functional blocks (including instructions decoders). Classical x86-compatible architectures (each physical core contains one major functional block) are deprived of such shortcomings and allow better use of the available hardware resources.

Depending on the selected variation of BLAS implementation, user may also select a way to ensure execution of instructions in multiple threads. Such a way may involve parallel computing using MPI interface. Parallelization may be used not only for the whole process of numerical solution of the problem, for example, by means of its implementation through OpenMPI (in the case of using original BLAS), but in respect to directly the process factorization stages, as well as solution of the simultaneous equations using OpenMP. At that, the remaining steps of the solution process are performed in a single thread, which is important when using such BLAS implementations, as MKL, ACML, and OpenBLAS. One can also combine these techniques to achieve maximum performance in multiprocessor and distributed computing systems. Using OpenMPI leads to creation and transmission of redundant data between the threads. This involves using OpenMPI distributed-memory model for memory allocation between the threads. This model allows memory allocation to achieve high performance of distributed computing systems.

The use of such a memory allocation model in the local computer systems will lead to a significant increase in the consumption of RAM and some loss of efficiency in comparison with the shared-memory model of memory allocation, used in OpenMP. Figure 3 shows dependence of the solution time of test problem on the number of threads when using OpenMPI and OpenMP. Solution of the test problem is performed using In-Core algorithm when employing the central processing unit of INTEL Core i7-870.

The process of solving numerical problem in GetDP can be divided into several stages according to the use of hardware. The first stage consists in reading of preliminary preprocessed problem topology file, and loading geometry of the computational domain in the form of a finite element mesh from the corresponding file. It takes negligible time, depending almost entirely just on the read rate from the disk subsystem. The second stage consists in generation of matrices according to the problem topology and the rules of its allocation in RAM. Execution speed of this stage depends on the CPU performance and can be carried out just in one thread. The time required to complete this stage is also not great. After that the system matrix and its subsequent ordering are analyzed. Time to complete this stage largely depends on the selected software, algorithms, and ordering parameters of matrix and its inherent characteristics. Characteristics of the matrix itself largely determine the choice of ordering algorithm that is especially important when using the Out-Of-Core algorithm. The algorithm chosen (as well as the software that implements this algorithm) affects directly on run-time of collating stage and factorization stage performances, as well as the subsequent solution of the matrix. Use of MUMPS as an SLAE solver allows one to select high performance hybrid sets of routines METIS and PORD for matrix ordering. Using these sets of routines in most cases allows one to reduce the time required for the matrix factorization by factor of 2 or 3, as compared with the use of other sets of routines, such as AMD, QAMD [9] or AMF. Though for certain matrices, time required for ordering stage, may be magnified [10].

The next stage includes matrix factorization (decomposition). At this stage, either solution of the matrix is carried out by direct method, or matrix preconditioning, in the case of using direct methods as preconditioning techniques. Factorization stage, when using LU-decomposition and Cholesky decomposition, takes quite a long time, up to 70-80% of the iteration time, as it requires execution of a large number of operations with floating-point numbers. Required number of operations can be estimated by the expression (1) or by diagnostic messages of MUMPS upon completion of SLAE analysis [11]. The processing speed of current stage is largely dependent on the performance of the CPU. Run-time of factorization stage depends to some extent on the RAM bandwidth, and when using Out-Of-Core algorithm, it slightly depends on recording performance of the disk subsystem. The next stage after factorization is a stage of solution of the resulting simultaneous equations (in form of a matrix). At this stage, in the case of using Out-Of-Core algorithm for solving numerical problem, run-time largely depends on the performance of the disk subsystem of a personal computer, rather than on the processor, though to a certain limit (usually up to the point where subsystem performance disc becomes commensurable with the RAM). After that the results of solving the equations system are stored in the memory. This stage is performed in a negligible time.

During the FEM-based simulation of electromagnetic fields, when describing electromagnetic processes, one obtains usually square symmetric positive-definite SLAE. This allows use of both Cholesky decomposition, and LU-decomposition. Such SLAE properties allow successful application of any of the available ordering algorithms. For symmetric matrices, we can employ hybrid methods of ordering; for example PORD shows a very high efficiency. Figure 4 presents the run-time of solution stages and the total run-time required for solving SLAE when performing ordering by means of PORD, AMF, and subsequent LU-decomposition.

It is worthwhile to note a threefold increase in execution time of ordering procedure, when using the hybrid PORD method instead of AMF, which, however, leads to a reduction in LU-decomposition run-time by 150% that accounts for about 90% of the total computational effort at solving SLAE by direct method. This ultimately reduces the SLAE solution time by 140%.

Using Cholesky decomposition will also reduce the time for solving SLAE due to a significant reduction in the computational work on the factorization stage. Figure 5 shows the execution time per iteration in the iterative process, when solving a nonlinear problem simulating the electromagnetic field, depending on the order of the system matrix of solved SLAE. Solution of SLAE was performed employing LU-decomposition and Cholesky decomposition.

Conclusions.

When using direct methods of solving SLAE, required RAM is directly proportional to the amount of computational effort spent at the factorization stage. Therefore, when solving SLAE by direct method, one has to use hybrid methods of ordering that enable to reduce the amount of required RAM. This is especially important for personal computers, having a very limited amount of RAM, or low performance disc subsystem, which does not allow effective use of Out-Of-Core algorithm due to a significant increase in SLAE solution time because of the low read rate of matrix decomposition results from the disk.

Despite the high computational complexity of direct methods for solving SLAE, when combined with modern hybrid methods of ordering, they are capable of providing a sufficiently high speed at solving SLAE, which are formed when dealing with numerical problems, arising in the simulation of electromagnetic fields by the finite element method.

Using high-performance software along with CPUs that support modern operating instructions for computations with the floating-point numbers enables one to solve SLAE or carrying out their preconditioning by direct methods in a reasonable time. This is especially important in cases where the iterative methods for solving SLAE provide an extremely low rate of convergence.

Acknowledgments.

This article was prepared based on the results obtained in the framework of the project #SP-201.2012.2 on "Development of scientific and technical solutions to create a system for monitoring the status of steel ropes of polar cranes at nuclear power stations", supported by the scholarship of the President of the Russian Federation for young scientists and graduate students, engaged in advanced research and design in priority areas of the Russian economy, as well as the project #7.1604.2011 "Theory of creation and research of resource and energy saving electromechanical devices, systems, and complexes", carried out within the framework of the state assignment for 2013 and the planning period of 2014 and 2015.

Corresponding Author:
Dr. Khoroshev,
Platov South-Russian State Polytechnic University (Novocherkassk Polytechnic Institute),
Russia, 346428, Rostovskaya oblast, Novocherkassk, Prosvescheniya street, 132.

References

1. Marchevskiy I. K. and V.V. Puzikova, 2013. Analysis of the effectiveness of iterative methods for solving systems of linear algebraic equations, implemented in the package OpenFOAM. Proceedings of the Institute for System Programming RAS. 24. pp: 71-85.
2. Saad, Y., 1996. Iterative Methods for Sparse Linear Systems. Pws Pub Co.
3. GetDP: a General Environment for the Treatment of Discrete Problems. Date Views: 21.12.20013. www.geuz.org/getdp/
4. Horoshev A.S., Pavlenko A. V., Batishev D.V., Puzin V.S., Shevchenko E.V. and I.A. Bolshenko, 2013. Verification of complex programs GMSH + GETDP for the finite element modeling of electromagnetic fields. News of higher educational institutions. North Caucasus region.Technicheskie nauki. 6. pp:74 – 78
5. PETSc: Summary of Sparse Linear Solvers Available from PETSc. Date Views: 21.12.20013. www.mcs.anl.gov/petsc/documentation/linearsolvertable.html.
6. Lawson C. L., R. J. Hanson, D. Kincaid, and F. T. Krogh, 1979. Basic Linear Algebra Subprograms for FORTRAN usage. ACM Trans. Math. Soft., 5: 308—323.
7. Eddelbuettel, D., 2010. Benchmarking single-

and multi-core BLAS implementations and GPUs for use with R. Mathematica.
8. Faq. xianyi/OpenBLAS Wiki GitHub github.com. Date Views: 21.12.20013. www.github.com/xianyi/OpenBLAS/wiki/faq# wiki-sandybridge_perf.
9. Amestoy P., T. A. Davis and I. S. Duff, 2004. Algorithm 837: AMD, An approximate minimum degree ordering algorithm. ACM Transactions on Mathematical Software, 30 (3), pp: 381-388.
10. Gregoire, R., 2002. Coupling MUMPS and ordering software., CERFACS.
11. MUltifrontal Massively Parallel Solver Users' guide. 2011, 10.

2/9/2014