Available online at www.sciencedirect.com

SciVerse ScienceDirect

journal homepage: www.elsevier.com/locate/acme

Original Research Article

Iterative methods for solving large-scale problems of structural mechanics using multi-core computers

S.Yu. Fialko

Department of Physics, Mathematics and Applied Computer Science, Cracow University of Technology, Kraków 31-155, Poland

ARTICLE INFO

Article history:
Received 18 January 2013
Accepted 30 May 2013
Available online 6 June 2013

Keywords:
Finite element method
Preconditioned conjugate gradient method
Sparse matrices
Multi-core computers
Multistory buildings

ABSTRACT

The paper studies the conjugate gradient method for solving systems of linear algebraic equations with symmetric sparse matrices that arise when the finite-element method is applied to the problems of structural mechanics. The main focus is on designing effective preconditioning and parallelizing the method for multi-core desktop computers. Preconditioning is based on the incomplete Cholesky "by value" factorization method and implemented based on the technique of sparse matrices, which allows increasing convergence considerably without a significant increase of the computer's resources. Parallelization is implemented for the incomplete factorization as well as for iterative process stages. The method is integrated into the SCAD software package (http://www.scadsoft.com). The paper includes a discussion of the results of calculations done with direct and iterative methods for large-scale design models of tall buildings, originally from the SCAD Soft[1] problem collection.

1. Introduction

Intense development of multistory construction, combined with the rapid development of computation equipment used for strength calculations in small and medium-sized engineering bureaus, created a necessity to improve methods for solving linear algebraic equations (solvers) that arise when the finite element method is applied to problems of structural mechanics. Since the dimension of the design model of a contemporary multistory building is between 800,000 and 5,000,000 equations, a solver for a contemporary FEA software must be highly efficient.

Modern desktop computers are progressively more capable of solving large-scale and complicated engineering problems, eliminating the need for clusters, powerful and expensive workstations, and computer networks. However, they have a limited amount of RAM and small system bus bandwidth. This makes it necessary to develop computational methods that take into consideration the above specifics of computers with such architecture, because methods proven to be effective on distributed-memory computers (clusters or computer networks) are not always effective on desktop computers with SMP (symmetrical multiprocessing) architecture. In other words, the solver must be able to use the disc memory if the dimension of the problem exceeds the RAM capacity, demonstrate high performance while working in the RAM, and speed up reliably as the number of cores (processors) is increased.

E-mail address: sfialko@poczta.onet.pl

[1]SCAD Soft (http://www.scadsoft.com)—IT company, developer of the SCAD FEA software, one of the most popular software used in the CIS countries for structural analysis and design, certified according to the regional norms.

1644-9665/$ - see front matter © 2013 Politechnika Wrocławska. Published by Elsevier Urban & Partner Sp. z o.o. All rights reserved.
http://dx.doi.org/10.1016/j.acme.2013.05.009

Contemporary FEA software most frequently employs the sparse direct solvers for analysis of problems of structural mechanics. The decisive factor here is employing an effective ordering method that significantly decreases the number of non-zero entries during matrix factorization [4], as well as the solver's ability to use the disc memory when solving large-scale problems on computers with small amounts of RAM, and to maintain high performance and speed up when processing data blocks stored in the RAM.

In the latter case, the key issue is extracting rectangular dense submatrices from a sparse matrix and subjecting them to high-performance BLAS level 3 routines [2], particularly matrix multiplication procedures. Furthermore, with the use of direct methods, solution time does not depend on the conditioning number associated with the linear equation set and depends only slightly on the number of right hand parts, if the number of the latter is relatively small. Direct methods also allow detecting the geometric instability of the design model.

Until recently, the multifrontal solver [1,7,14], has been the most widespread direct method. It has all of the above advantages, however, it also contains an excess number of memory–memory and memory–disc–memory data transfer. On multi-core SMP computers, this shortcoming prevents achieving maximum possible performance and speeding up with the increase in the number of processors.

In the recent years, the PARDISO method [22] from the high-performance Intel Math Kernel Library (Intel MKL) [10] has become widespread, demonstrating high performance with one processor and good speed up on desktop multi-core computers. However, the OOC (out of core) mode, which utilizes disc memory, does not work for large-scale problems and demonstrates low performance with small ones [13]. Therefore, in practice, this method can only be used for problems solved in RAM.

The above became a stimulus for developing the PARFES (parallel finite element solver) method, intended for solving finite-element problems with multi-core desktop computers [13]. PARFES demonstrates the performance and speed up that approximate those of PARDISO, but, unlike the latter, includes two disc usage modes, OOC and OOC1. In the OOC mode, the number of I/O operations is minimal, resulting in only a small decrease in performance and speed up compared to CM (core mode—utilizing RAM only). However, if the requirements to RAM are still exceeded in the OOC mode, PARFES switches to the OOC1 mode. In this mode, the number of I/O operations is increased considerably, and the decrease in performance and speed up is also considerable—however, this mode allows solving large-scale problems on computers with small amounts of RAM [17]. The mode is selected automatically.

The common disadvantage of direct methods includes the quadratic dependence of the number of operations on the dimension of the problems. Plus, for problems with the dimension exceeding the RAM capacity, the matrix factorization time and forward-back substitution time are considerably increased.

Iterative methods do not guarantee detecting the geometric instability of the design model; the iterative process is started anew for each right hand side; and when solving poorly conditioned problems, iterative methods result in slower or no convergence.

Design models of tall multistory buildings contain different types of finite elements (FE)—thin-walled plate and shell FE, FE of spatial frame, volumetric FE, specialized FE (elastic supports, rigid links that carry penalty parameters in SCAD, compatible nodes, etc.). This also results in a wide dispersion of stiffness values. In realistic problems, due to complex geometry, mesh generators cannot always provide for the optimal ratios between FE sides and angles for shells and plates. Due to this, design models for multistory buildings usually result in poorly conditioned stiffness matrices and therefore require iterative methods that are stable against ill conditioning [21]. The use of preconditioning is an effective way for overcoming of poor conditioning [6,8].

The main operations of iterative methods are matrix–vector multiplication and solving the system of linear algebraic equations regarding preconditioning. These operations are classified as BLAS level 2 routines. Unlike matrix multiplication used in direct methods, which employs cache reuse (data, once placed in the cache, is read into the processor registers multiple times, from the fast cache rather than the slow RAM), register blocking, vectorization of computation with the use of XMM and YMM registers, and other high-performance techniques, BLAS level 2 routines are carried out at the speed of the slow memory system rather than the fast processor [8]. The use of multithreading in the SMP architectures allows for good speed up of the matrix multiplication routine, since the memory system is not overloaded thanks to the cache reuse. In BLAS level 2 routines, on the other hand, the number of memory–cache–memory transfers is of the same order as the number of arithmetic operations. The memory system is incapable of efficiently serving several processors, and each of them is forced to perform many idle cycles while waiting for the required data to be loaded into the cache from the RAM. Due to this, the performance and speed up of matrix–vector multiplication and solving of linear equation sets regarding preconditioning is far less compared to matrix multiplication (see Fig. 1).

Here, $S_p = T_1/T_p$ is speed up, T_1 is solution time with one processor, T_p is solution time with p processors. The results were obtained on a computer with Intel® Core™2 Quad CPU Q6600 @2.40 GHz processor, RAM DDR2 333.7 MHz, 8 GB,

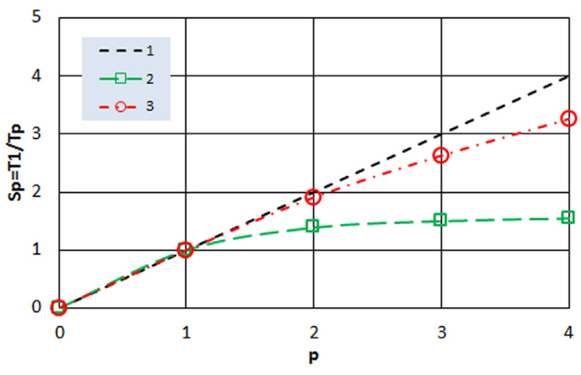

Fig. 1 – Speed up (S_p) with the increase in the number of processors (p). 1—ideal, 2—for dense matrix–vector multiplication algorithm, 3—for dense matrix multiplication algorithm.