Computers and Structures 88 (2010) 178–188

Contents lists available at ScienceDirect

Computers and Structures

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

Incomplete factorization preconditioners for the iterative solution of Stochastic Finite Element equations

Dimos C. Charmpis *

Department of Civil and Environmental Engineering, University of Cyprus, 75 Kallipoleos Str., P.O. Box 20537, 1678 Nicosia, Cyprus

ARTICLE INFO

Article history:
Received 8 November 2008
Accepted 29 September 2009
Available online 11 November 2009

Keywords:
Monte Carlo simulation
Stochastic Finite Element
Conjugate gradient method
Iterative solution
Preconditioner
Incomplete factorization

ABSTRACT

This work is focused on enhancing the computational efficiency in Monte Carlo simulation-based Stochastic Finite Element (SFE) analysis of large-scale structural models. Such analyses require the solution of successive systems of equations derived during simulations, which can be efficiently treated using customized versions of the iterative Preconditioned Conjugate Gradient (PCG) solution method. PCG-customization is localized at the preconditioning matrix employed to accelerate convergence. Thus, specialized preconditioners following the rationale of incomplete factorization are presented, which retain only essential numerical information during factorization. The resulting PCG-based solution schemes allow for computationally efficient SFE analyses with low storage demands in computer memory.

1. Introduction

The Stochastic Finite Element (SFE) method is a widely appreciated approach to treat structural mechanics applications involving uncertain material and geometric properties with spatial distribution. The research efforts devoted during the last decades to the field of stochastic structural mechanics have led to the development of various rather sophisticated SFE formulations, however considerably less progress has been reported on the computational efficiency and feasibility of such formulations when confronted with large-scale real-world problems. As a result, SFE applications usually involve intentionally simple and small-scale structural models and the results obtained are of limited practical importance. This unfavorable situation is expected to improve with the development, continuous upgrade and wide availability of powerful and efficient SFE software.

The computational burden associated with SFE analyses may be substantially reduced by appropriately handling the most demanding tasks in terms of processing power and storage space needs. The Monte Carlo (MC) simulation technique, which is the most effective and widely applicable method for handling large-scale SFE problems with complicated structural response, involves expensive computations due to the successive analyses required. More specifically, assuming deterministic loads and a linear static SFE problem, successive linear systems of equations with multiple left-hand sides have to be processed, since the stiffness matrix changes in every simulation. The standard direct method based on Cholesky factorization remains the most popular scheme for solving such equations, however this solution approach exhibits poor performance for large-scale problems and may lead to practically infeasible computations (in terms of required computing time) when the number of MC simulations to be performed is not small. Hence, the effective and efficient handling of SFE equations has emerged as a research topic of special interest and importance within the SFE community and a number of related publications have appeared dealing with MC simulation-based and other SFE formulations (e.g. [1–8]).

The deficiencies of the direct solution approach can be overcome with the use of a customized version of the Preconditioned Conjugate Gradient (PCG) method, which allows the adaptation of this solution scheme to the special features of nearby problems encountered in Finite Element (FE) reanalyses [2,5]. PCG can be customized to take into account the relatively small differences between stiffness matrices in successive simulations, avoiding this way the treatment of each simulation's system as a stand-alone problem. PCG-customization is localized at the preconditioning matrix employed to accelerate PCG convergence during the successive FE solutions. Hence, the reanalysis problems can be effectively solved using the PCG algorithm equipped with a preconditioner following the rationale of incomplete Cholesky preconditionings. According to this rationale, the preconditioning matrix may be taken as the complete factorized stiffness matrix of the initial simulation. With the preconditioning matrix remaining the same during the successive FE reanalyses, the repeated solutions required for the preconditioning step of the PCG algorithm can be efficiently

* Tel.: +357 22 89 2202; fax: +357 22 89 2295.
E-mail address: charmpis@ucy.ac.cy

0045-7949/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compstruc.2009.09.010

treated as problems with multiple right-hand sides. Therefore, the stiffness matrix of the initial simulation is retained in memory in a factorized form throughout all simulations.

The present work further enhances the aforementioned customized PCG solution approach by proposing alternative preconditioning schemes, which are based on incomplete factorizations of the stiffness matrix of the initial simulation. Such preconditioners aim in retaining only the essential numerical information during the factorization of the initial stiffness matrix. Thus, the matrix terms stored during factorization are selected based on their position within the initial stiffness matrix (incomplete factorization by position) or their magnitude (incomplete factorization by magnitude). As a result, preconditioner storage demands are reduced, while PCG iteration performance is not strongly affected when incomplete factorization yields a sufficiently strong preconditioner, therefore gains also in terms of overall required computing time can be achieved.

All above mentioned specialized versions of the PCG solution method are applicable to SFE software, for which access to modify the structural analysis kernel is provided, since the routines for solving the FE system of equations need to be re-programmed. This is possible when self-developed or open source structural analysis code is built into the global SFE software. However, a common approach followed lately is to employ a powerful and widely used deterministic (usually commercial) FE program as the structural analysis kernel of the SFE application. Such deterministic general-purpose FE packages are typically available as closed source software and, with few exceptions (e.g. MSC/NASTRAN DMAP), users cannot incorporate enhancements to the routines solving FE equations. Each of these two SFE software development approaches (with free or prohibited access to modify/replace structural analysis solution code) has its own features and offers certain advantages. Thus, it seems that a SFE software developer or user has typically to choose among the potential for efficiency of the one approach and the potential for generality of the other.

The remainder of this paper is organized as follows. Section 2 discusses the advantages and disadvantages associated with the use of closed and open source structural analysis software in SFE applications. The form of equations arising in linear static SFE analyses is overviewed in Section 3. Sections 4 and 5 present PCG-based solution methods and describe corresponding preconditioning techniques for efficiently treating SFE equations. The storage schemes used for stiffness and preconditioning matrices by the solution methods are specified in Section 6. Numerical results demonstrating the computational advantages offered by iterative solution procedures and incomplete factorization preconditioners are presented in Section 7. The paper concludes with some final remarks given in section 8.

2. Closed vs. open source structural analysis software in SFE applications

MC simulation-based SFE analysis can be viewed as a process of repeated deterministic FE analyses, since it involves structural response calculations associated with various instances generated each time for the uncertain properties considered. Consequently, the overall quality of SFE software is strongly related to the capabilities of the underlying conventional FE code. Thus, the types of elements implemented in the FE library, the material models available, the types of FE analyses supported (linear/nonlinear, static/dynamic), etc. impose or raise constraints regarding the degree of sophistication in SFE analyses offered by the software. To ensure maximum conventional FE capabilities, SFE software can be linked through appropriate interfaces to well-known and widely used FE programs (ANSYS, MSC/NASTRAN, etc.). This SFE software development approach requires explicit programming of specialized routines for the handling of stochastic information (mainly for the determination of stochastic parameters and corresponding data, the generation of discretized random field samples and the post-processing of SFE analysis results) and relies on deterministic FE codes for performing repeated FE analyses based on the generated samples. The capability to invoke powerful deterministic FE programs is regarded as a key feature of 'general-purpose' SFE software [9] and is already incorporated in a number of existing SFE software packages (e.g. [10,11]).

The coupling of SFE software with powerful deterministic (usually commercial) FE programs provides remarkable FE capabilities (associated with current FE programs versions, but also with future releases), however this implies that (with few exceptions) the access to the FE solution code is prohibited, because such FE packages are usually distributed as closed source software (access is given to compiled executables only). The aforementioned few exceptions are the commercial FE packages that already exist, which offer specialized modules/APIs (Application Programming Interfaces) to modify/replace prewritten FE code (e.g. MSC/NASTRAN DMAP). It should also be kept in mind that, in general, the interoperability of FE programs is expected to improve in the future. Nevertheless, even when capabilities to modify/replace code of the FE software kernel are offered, the development, debugging and maintenance of enhancements through a customized, non-conventional and complicated programming environment providing limited access to code is a rather difficult and non-attractive task. Moreover, enhancements programmed within such FE software can be exploited only as long as the license to use the software is renewed and need to be re-programmed, in order to be linked to another FE software. Therefore, the repeated analyses performed in the framework of SFE applications utilizing commercial FE software are typically treated as successive stand-alone FE problems, although the special features of SFE equations allow for the use of more appropriate customized solution procedures, as already stated in the introduction of the present work. The series of FE equations systems encountered in SFE applications have relatively small differences between successive stiffness matrices and form sets of so-called nearby problems. Such problems can be efficiently treated using appropriately adapted iterative solution schemes, which have however to be built into the FE analysis code. Consequently, the favorable features of nearby problems can be exploited only when the FE solution code is upgraded by implementing specialized and highly efficient SFE solvers. A suitable platform to program such an upgrade is a self-developed or open source FE code, which typically offers substantially less capabilities compared to several commercial closed source FE programs being systematically developed and improved for decades.

Following the discussion above, it appears that one has to choose among two SFE software development approaches. The first approach focuses on the generality and functionality conveyed to SFE software by the utilization of deterministic FE programs (ANSYS, MSC/NASTRAN, etc.). In that case, access to the FE solution code is typically not provided to the user (or if it is provided, it is usually difficult and non-attractive to program enhancements), therefore SFE solution efficiency is not improved by employing customized methods. The second approach essentially sacrifices generality to some extent, in order to facilitate the programming of software enhancements that yield SFE analyses results in more affordable computing times. The applicability of SFE software developed with this approach is usually restricted by the limited capabilities of the underlying FE code, however SFE solutions may be drastically accelerated by implementing efficient iterative schemes tailored to the special features of SFE equations. The most appropriate SFE software development approach to follow depends on a number of factors, like the type of application and its model-