Advances in Engineering Software 47 (2012) 35–50

Contents lists available at SciVerse ScienceDirect

Advances in Engineering Software

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

Object-oriented, parallel finite element framework with dynamic load balancing

B. Patzák *, D. Rypl

Czech Technical University, Faculty of Civil Engineering, Thákurova 7, 166 29 Prague, Czech Republic

ARTICLE INFO

Article history:
Received 15 August 2011
Received in revised form 9 November 2011
Accepted 13 December 2011
Available online 9 January 2012

Keywords:
Parallel computing
Dynamic load balancing
Adaptive analysis
Object-oriented FEM
Message passing
Domain decomposition
Nonlinear fracture analysis

ABSTRACT

The present paper deals with the design and implementation of parallel load-balancing framework in an object-oriented finite element environment. The parallelization strategy is based on domain decomposition and message passing paradigms. The algorithmic and implementation aspects are discussed in detail. Paper also describes components of a complete adaptive strategy, i.e., the error estimator/indicator, projection operator and remeshing. The capabilities and performance of the developed framework are demonstrated on advanced engineering problems, showing the scalability of the implemented algorithm and advantages of dynamic load balancing when used in dedicated and nondedicated environments.

1. Introduction

The recent developments in many scientific and engineering disciplines bring in new challenges for computational science. The engineering community is facing growing demands for state-of-the-art modeling and realistic predictions in many fields. The Finite Element Method (FEM) has become a widely used tool for solving problems described by partial differential equations. Its application often leads to nonlinear models of very complex geometries and many degrees of freedom. The quality of the obtained solution is dependent on many aspects (including the adopted spatial and time discretization, material model, equation solver, and its parameters, etc.). Thus it is very important to keep the solution error under control. This can be conveniently (and usually also most economically) accomplished by the application of the adaptive analysis.

A very natural goal of the adaptive finite element analysis is to calculate solution of the governing partial differential equation(s) with uniformly distributed error not exceeding a prescribed threshold in the most economical manner. This is achieved by improving the discretization in areas where the finite element solution is not adequate. It is therefore essential to have an assessment of the quality of the approximate solution and a capability of discretization enrichment.

For linear problems, error analysis of the finite element solution can be developed in a mathematically rigorous way [1,2]. In the nonlinear range, however, rigorous error estimates can be constructed only for a restricted class of problems. A general theory is not available since there are various sources and forms of nonlinearities. In the linear elastic case the error arises essentially from the discretization of the domain (so-called spatial error). In the nonlinear case, the error depends on the time discretization for history-dependent problems, and a part of the error is always induced by the incremental-iterative technique. The path dependency renders the problem more complex and, consequently, a reliable error estimation becomes more difficult, especially for nonconventional theories of enriched continua. Nevertheless, considerable progress has been made in recent years. For instance, Rodriguez-Ferran and Huerta [3] proposed a sophisticated error estimator for nonlocal damage models. Ladevèze and coworkers [4,5] developed a *posteriori* estimators based on the error in the onstitutive relation, and Comi and Perego [6] adapted this technique to their nonlocal damage theory [7]. However, the implementation of these complicated estimators requires a considerable effort. A simple and convenient alternative to rigorous error estimators is provided by heuristic error indicators (see, for example [8]). They are often based on physical intuition and insight into the problem at hand.

There are three main directions of the adaptive discretization enrichment. The first one, a natural way for most engineers, is the h-version [9–11], which refines the computational finite element mesh while preserving the approximation order of the elements. The p-version [12] keeps the mesh fixed but increases hierarchically the order of the approximation being used. The hp-version [10,13,14] is a proper combination of h- and p-versions and exhibits an exponential convergence rate independently of the smoothness of the solution. However, its implementation is not trivial. Similarly, the treatment of higher order elements in the p-version is rather complicated, especially when nonlinear analysis is considered.

* Corresponding author. Tel.: +420 224354375; fax: +420 224310775.
E-mail address: borek.patzak@fsv.cvut.cz (B. Patzák).

0965-9978/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2011.12.008

Complex adaptive analyses initiate demands for large-scale computing, which must be feasible from the view of both time and available resources. This naturally leads to utilization of parallel processing that allows not only to obtain results in acceptable time by significantly speeding up the analysis, but also to perform large and complex analyses, which often do not fit into a single, even well equipped, machine with one processor unit (regardless of achieved speedup). Typical parallel application decreases the demands on memory and other resources by spreading the task over several mutually interconnected computers and speeds up the response of the application by distributing the computation to individual processors.

The design of parallel algorithms requires the partitioning of the problem into a set of tasks, the number of which is greater than or equal to the number of available processors. The partitioning of the problem can be fixed at runtime (static load balancing) or can change during the solution (dynamic load balancing). The latter option is often necessary in order to achieve good load balancing of work assigned to individual processors and thus optimal scalability. The dynamic load balancing is the subject of active research in many fields. An overview of the algorithms suitable for structural mechanics was given in [15]. The distributed data structures and corresponding load balancing techniques for finite element computations have been described in [16,17].

There are in general two basic factors causing a load imbalance between individual sub-domains: (i) one coming from the nature of the application, such as switching from linear to nonlinear response in certain regions or local adaptive refinement, and (ii) external factors, caused by resource reallocation, typical in nondedicated cluster environments, where individual processors are shared by different applications and users, leading to a variation in the allocated processing power. The application has to continuously monitor the solution process and to detect work imbalance. When an imbalance is identified, the decision has to be made whether to recover load balance or to continue with the existing work distribution, depending on the magnitude of load imbalance and the cost of load recovery. Work transfer requires serialization of the problem data (representing parts of solution vectors, elements, nodes, etc.) into a byte stream that is sent over the network and unpacked, followed by a topology update reflecting the new partitioning.

As the spatial distribution of error is not uniform, the number of elements can vary significantly between partitions after a few remeshing steps. This is particularly true for problems with strong localization, where error is localized into narrow bands leading to enormous refinement in these regions. Thus, the load rebalancing on the fly is necessary to recover the load balance and to obtain scalable implementation. The load balance recovery is achieved by repartitioning the problem domain and transferring the work (represented typically by finite elements) from one sub-domain to another. The repartitioning is an optimization problem with multiple constraints. The optimal algorithm should balance the work while minimizing the work transfer and keeping the sub-domain interfaces as small as possible. Other constraints can reflect the differences in the processing power of individual processors or may be induced by the topology of the network. The graph partitioning problem is NP-complete. However, in recent years a lot of attention has been focused on developing suitable heuristics and many powerful mesh (re)partitioning algorithms have been developed. Algorithms based on spectral methods [18,19] have been shown to be reasonably efficient (in terms of the quality of the partitioning) for partitioning unstructured problems in many applications, but they have a relatively high computational complexity. Geometric partitioning methods [20,21] are quite fast but they typically provide worse partitioning than other (more expensive) methods. Recently, a number of researchers have investigated a class of multilevel graph partitioning algorithms that have a moderate computational complexity and provide excellent (even better than spectral) partitioning [22,23].

The software developers have to address above mentioned demands for large-scale computations. From the software development perspective, the object-oriented approach is a tool that can significantly facilitate the design and implementation of complex software systems meeting the above criteria. It is based on the uniform application of the principles for managing complexity – abstraction, inheritance, association, and communication using messages. In the past several years, a number of articles on applying an object-oriented approach to the finite element analysis have been published. In 1990, Fenves [24] described the advantages of an object-oriented approach for development of engineering software. Forde et al. [25] presented one of the first applications of object-oriented programming to finite elements. Many authors presented the complete architectures of OO finite element codes, notably, a coordinate free approach by Miller [26], a nonanticipation principle by Zimmermann et al. [27], Dubois-Pelerin et al. [28–30], and Commend and Zimmermann [31]. Recent contributions include the work of Mackie [32–34], Archer et al. [35,36], and Menetrey and Zimmermann [37]. The present contribution deals with the design of an object-oriented framework for dynamic load balancing implemented in the frame of the OOFEM [38,39] code. OOFEM is an open-source finite element solver with the object-oriented architecture, distributed under GNU public license, written in C++ programming language.

The aim of this paper is to show the implementation and integration of individual components of the parallel, h-adaptive, dynamically load balanced analysis into a single framework that can be used for efficient adaptive simulations on memory distributed computing platforms. The described framework introduces several class-hierarchies providing high-level communication services and load-balancing layer, consisting of (i) load monitoring, keeping track of the solution process and detecting load imbalance, (ii) load balancing, responsible for transparent work transfer, and (iii) interface to external domain partitioners. The application of presented methodology is illustrated on nonlinear analyses of concrete fracture. Concrete as well as other quasibrittle materials is characterized by the development of large nonlinear fracture process zones. Modeling of progressive growth of microcracks and their gradual coalescence leads to constitutive laws with softening, i.e., with a descending branch of the stress–strain diagram. Regularization technique is necessary to enforce objective, mesh-independent response (see [40–43]). In the presented work, objectivity is ensured by using regularized material models based on the concept of nonlocal averaging which is applied to isotropic and anisotropic damage formulations (see [44]).

The paper starts with the short introduction to the overall design of OOFEM environment. Next the adaptive methodology is recalled. Then, parallelization strategy based on the domain decomposition is presented. The design of communication layer, built on top of message passing library is outlined. Further, the design and implementation of load-balancing layer describing the load monitoring, load balancing, and load transfer phases is presented. Next section describes parallel remeshing strategy based on 2D and 3D subdivision algorithms. Finally, the potential of the proposed approach is illustrated on a few examples of adaptive nonlinear fracture analyses of concrete specimens.

2. Overall design of OOFEM code

The general structure of the OOFEM is shown in Fig. 1, using the Unified Modeling Language (UML) notation [45]. In short, abstract classes are represented by rectangles. Lines with a triangle mark represent the generalization/specialization relation (inheritance),