Advances in Engineering Software 72 (2014) 8–17

Contents lists available at SciVerse ScienceDirect

Advances in Engineering Software

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

Code-coupling strategy for efficient development of computer software in multiscale and multiphysics nonlinear evolution problems in computational mechanics

Adnan Ibrahimbegovic [a,*], Rainer Niekamp [b], Christophe Kassiotis [c], Damijan Markovic [d], Hermann G. Matthies [b]

[a] Ecole Normale Supérieure, Laboratory of Mechanics and Technology, Cachan, France
[b] TU-Braunschweig, Institute for Scientific Computing, Braunschweig, Germany
[c] Ecole des Ponts, Laboratory St. Venant, Paris, France
[d] EDF, SEPTEN, Lyon, France

ARTICLE INFO

Article history:
Available online 18 July 2013

Keywords:
Software development
Code coupling
Multiscale
Multiphysics
Nonlinear evolution problem
Monolithic code at linkage time

ABSTRACT

In this work we seek to provide an efficient approach to development of software computational platform for the currently very active research domain of multiphysics and multiscale analysis in fully nonlinear setting. The typical problem to be solved is nonlinear evolution problem, with different scales in space and time. We show here that a successful solution to such a problem requires gathering the sound theoretical formulation, the most appropriate discrete approximation and the efficient numerical implementation. We show in particular that the most efficient numerical implementation is obtained by reusing the existing codes, in order to accelerate the code development and validation. The key element that makes such an approach possible is the Component Template Library (CTL), presented in this work. We show that the CTL allows to seamlessly merge the existing software products into a single code at compilation time, regardless of their 'heterogeneities' in terms of programming language or redundancy in use of local variables. A couple of illustrative problems of fluid–structure interaction and multiscale nonlinear analysis are presented in order to confirm the advantage of the proposed approach.

1. Introduction

From its early days and focus upon the finite element method development (e.g. [49,7]), the computational mechanics has developed into a very broad area, much driven by various industrial applications (e.g. [43]). Very diverse problems are presently of interest for research and expertise of computational mechanics community, such as biomechanics, nanotechnology, and predictive home security. Typical formulations of interest are very much interdisciplinary, and ought to be formulated in terms of multiphysics problems: such as thermomechanical coupling, fluid–structure interaction, coupling of probability and mechanics. Moreover, in mathematical interpretation, the vast majority of these problems are highly nonlinear evolution problems. Thus, in trying to provide the most efficient solution procedure for any such problem, one is often prompted to exploit multiscale modeling and computations, which consider the most appropriate spatial and temporal evolution for each particular field. Providing the best theoretical formulation for any such problem is one of the currently most important challenges that has been addressed in a number of recent works on multiscale modeling and computations (e.g. [18,26,22,24,34,44,20], among others).

However, the theoretical formulation of multiphysics and multiscale problems is not the only challenge. We argue here that providing the corresponding computational tools, or computer software, presents even a bigger challenge, with much more benefits for bringing this technology to bear upon the current engineering practice. Namely, as the problem complexity grows, many existing codes are found insufficient to meet the new requirement or even completely obsolete. Thus, a number of new developments have sprung in starting the development of new codes for multiphysics and multiscale problems [35], where higher level programming languages or existing libraries are used to accelerate the software development; case in point are: code in Java (e.g. [15]), Smalltalk (e.g. [13,14]), or Comsol (e.g. see [8]). However, the software products of this kind are not necessarily the most efficient. In fact, although some tentative to increase the code efficiency are made, many such codes are mostly used in academic research environment, very far from the industry applications.

* Corresponding author.
E-mail address: ai@lmt.ens-cachan.fr (A. Ibrahimbegovic).

0965-9978/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.advengsoft.2013.06.014

Thus we propose here very different software development strategy, where the final software product is capable to fully integrate existing computer codes. The proposed strategy not only significantly accelerates the final code development, but even more importantly allows to directly include the existing codes that have been extensively tested previously. Last but not least, the proposed strategy can directly provide a very appealing interface to user who are used to a particular software product making them accept the new code release and other novelties much faster.

We note in passing that the same strategies of coupling of existing codes are currently used by interpreters, such as Python (e.g. see [33,19]) or Matlab (e.g. see [3]). However, contrary to such an approach which sacrifices the efficiency in favor of flexibility while coupling typically the executable versions of existing codes, we target here the code coupling that is done at the compilation and linking time, and thus provide truly the single code as the final software product. Such a code, perhaps needless to say, will be both more robust and more efficient than any other code-coupling alternative.

Last but not least, in seeking the optimal results we show in this paper the need to focus upon the complete treatment of the present problem, and not only the code-coupling. More precisely, the proposed strategy ought to reconsider the theoretical formulation of multiphysics and multiscale problems, the optimal discrete approximations both in space and in time, and finally the most efficient software development based upon the code-coupling strategy. This is illustrated upon two model problems. The first one concerns the multiscale analysis of inelastic behavior of heterogeneous materials (e.g. see [26], or [37]), where the same code FEAP (e.g. [50]) is used at both scales. The second pertains to fluid–structure interaction, where FEAP is coupled with Open-FOAM (e.g. see [45]) in order to produce the final software product. Both chosen model problems allow us to illustrate the important concept of interface and the use of localized Lagrange multipliers (e.g. [46,38]). In the multiscale problem the interface between the microscale and macroscale is defined in terms of macroelement boundaries. In the multiphysics problem the interface is the wetted structure boundary. The key role in defining the interface very clearly is played in choosing the original interpretation of theoretical formulation for multiscale strategy and fluid–structure interaction. The second important point of our strategy of using the localized Lagrange multiplier pertains to the optimal choice of discrete approximation for each sub-problem. This is illustrated in the case of multiphysics problem of fluid–structure interaction, where the finite element method is used for structure and finite volume method is used for fluids in order to construct optimal discrete approximations. The seamless connection of these two approximations is again achieved by the localized Lagrange multipliers and their corresponding interpolations in terms of radial approximations.

The outline of the paper is as follows. In the next section, we present the theoretical formulations of the multiscale problem for inelastic behavior of heterogeneous materials, along with the multiphysics problem of fluid–structure interaction. The discrete approximation of the latter problem is also presented in detail in Section 2, along with the corresponding proofs of convergence of sequential solution procedure and the illustrative examples showing what kind of results can readily be obtained in this manner. Section 3 provides a number of pertinent remarks on numerical implementation and the essence of proposed code-coupling procedure based upon the CTL – Component Template Library tool (e.g. see [41,42]). Concluding remarks are given in Section 4.

2. Theoretical formulation and discrete approximation of multiscale and multiphysics nonlinear evolution problems with interface

2.1. Localized Lagrange multipliers

In this section we first introduce the notion of interface and corresponding localized Lagrange multipliers, which allows reduction of the problem complexity. This is done in the simplest possible setting of domain decomposition where two domains are connected at the single point (see Fig. 1).

The classical formulation of partitioned problem introduces the Lagrange multiplier imposing the equalities of field values in points 1 and 2, which are now separated by the partition. This leads to a particular constraint:

$$c_{12} := \mathbf{u}_\Gamma^{(1)} - \mathbf{u}_\Gamma^{(2)} = 0 \tag{1}$$

In the solution method, this kind of constraint is handled by the Lagrange multiplier $\boldsymbol{\lambda}^{(12)}$, and the resulting set of equations can be written as:

$$\begin{aligned}\delta\Pi_{\text{total}} &:= \delta\Pi^{(1)} + \delta\Pi^{(2)} + \delta\pi_{\text{classical}} \\ &= \delta\mathbf{u}^{(1)T}\left[\mathbf{A}\left(\mathbf{u}^{(1)}\right) - \mathbf{f}^{(1)}\right] + \delta\mathbf{u}^{(1)T}\left[\mathbf{A}\left(\mathbf{u}^{(2)}\right) - \mathbf{f}^{(2)}\right] \\ &\quad + \delta\left[\boldsymbol{\lambda}^{(12)T}\left(\mathbf{u}_\Gamma^{(1)} - \mathbf{u}_\Gamma^{(2)}\right)\right]\end{aligned} \tag{2}$$

where $\delta\pi_{\text{classical}}$ is the classical form of interface constraint, the operator $\mathbf{A}(\mathbf{u}^{(\alpha)})$ represents the response of the system corresponding to system excitation $\mathbf{f}^{(\alpha)}$. We can see from (2) above that the two systems are 'visible' to each other through the global Lagrange multiplier $\boldsymbol{\lambda}^{(12)}$, physically representing the conjugate action to difference in field values in two subsystems. The main disadvantage of such a formulation is in keeping the partitioned systems tightly coupled, which thus leads to questionable gains of partitioned strategy.

However, an alternative to such a formulation is the use of localized Lagrange multipliers (e.g. [46,38]), which can introduce the notion of interface to separate the system into subsystems that will no longer be visible to each other. Namely, for the case illustrated in Fig. 1 where the system partitioning concerns only one point, we will introduce an interface point that belongs to neither system. If the value of the field at that point is denoted as $\mathbf{u}_f$, we can then rewrite the constraint in (1) as follows:

$$c_1 := \mathbf{u}_\Gamma^{(1)} - \mathbf{u}_f = 0; \quad c_2 := \mathbf{u}_\Gamma^{(2)} - \mathbf{u}_f = 0 \tag{3}$$

The system in (2) can also be rewritten by replacing the last term with a modified constraint equation:

$$\delta\pi_{\text{localized}} = \boldsymbol{\lambda}^{(1)T}\left(\mathbf{u}_\Gamma^{(1)} - \mathbf{u}_f\right) + \boldsymbol{\lambda}^{(2)T}\left(\mathbf{u}_\Gamma^{(2)} - \mathbf{u}_f\right) \tag{4}$$

We note that the interface degrees of freedom in this case will separate two subsystems, and would not allow either to see the other one. The advantage of the localized Lagrange multipliers, here $\boldsymbol{\lambda}^{(1)}$ and $\boldsymbol{\lambda}^{(2)}$ is that they pertain only to a single sub-system. Thus, they can be handled in purely local computations, introducing the most appropriate scaling to improve the system conditioning, etc.

2.2. Multiscale evolution problem

We could extrapolate these concepts directly towards the multiscale analysis of nonlinear evolution problem. In order to illustrate these ideas we refer to Fig. 2, showing the macro and micro scale representation of 3-point bending test (see [26]). The macro scale represents the structural level, with the discrete model constructed with what looks like the standard finite element mesh ('black nodes', with displacement denoted as $\mathbf{d}^M$).