Advances in Engineering Software 94 (2016) 1–13

Contents lists available at ScienceDirect

Advances in Engineering Software

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

An object-oriented symbolic approach to the automated derivation of finite element contributions

R. Saad [a], D. Eyheramendy [b,*], L. Zhang [b]

[a] Université Saint-Esprit de Kaslik, Jounieh, Lebanon
[b] LMA, CNRS UPR 7051, Centrale Marseille, Aix Marseille Université, 4 impasse Nikola Tesla, CS 40006, 13453 Marseille CEDEX 13, France

ARTICLE INFO

Article history:
Received 7 July 2015
Revised 18 December 2015
Accepted 15 January 2016
Available online 9 February 2016

Keywords:
Finite elements
Object-oriented programming
Symbolic computations

ABSTRACT

In this paper, we present a symbolic approach to automating the elaboration of numerical tools for the simulation of physical processes using the finite element method. We aim at developing a generic environment to automate discretization schemes in the context of PDEs. The approach is implemented in a consistent object-oriented tool built in Java. We propose a symbolic environment to automatically build the symbolic forms of elemental contributions corresponding to a variational formulation. These contributions are automatically introduced into a classical simulation tool. The basic object-oriented framework covering the elaboration of the elemental contributions derived from the weak statement is briefly discussed. The approach can be naturally extended to any discretization model in space or time. This approach is illustrated by the example of hyperelasticity.

1. Introduction

Numerical simulation is a key tool in modern engineering today. Engineers and researchers are faced with increasingly complex problems. One aspect of this complexity is that simulation tools involving different physics must be developed. A physical problem is usually modeled by a set of equations: partial differential equations, ordinary differential equations, etc. A space and time discretization procedure is then applied to the mathematical model. Some specific algorithms are needed to solve the problem (time integration scheme, linearization procedure, etc.). Moreover, these simulation tools which are integrated into computer systems can themselves be complex (multiprocessor systems, computing grids, etc.). In this context, the development of a finite element code turns out to be a tough job as soon as complex formulations are involved.

The traditional approach consists in either integrating the discrete forms in a computer tool or developing homemade computational applications. Structured programming approaches, and in particular object-oriented ones, have helped to significantly change not only the design of mechanical simulation tools but also the overall efficiency of the developments. It is clear that most of these programming tasks usually made by hand can be automatized which frees the engineer/researcher from fastidious programming tasks, allowing him to concentrate on modeling tasks. This issue of automation has been addressed since the 70s (see e.g. [30,43] or more recently [10,27] or [29]) and even partially considered in industrial software such as COMSOL (see e.g. [70]). The limit of COMSOL is that specific numerical algorithms may be mandatory to solve a special set of partial differential equations.

The approach we propose in this paper aims at allowing the user of a finite element code to introduce new formulations in simulation software, as in [10,27] and [29]. It can be considered as a generalization of the work previously proposed in [10] and an alternative to the work proposed in [10] and in [27]. The approach, applied here to the finite element method, allows the development of finite element contributions in a symbolic form for coupled partial differential equations, and their integration into a classical finite element code. We have developed generic concepts to automate the finite element discretization scheme. The approach is based on a tensor mathematical formalism to describe the discretization of a variational formulation as in [28] or [29]. We introduce an object-oriented framework that can handle the symbolic developments of elemental contributions. The integration of the elemental contributions into a classical finite element code is automated. The code produced is compiled in a classical way. From the point of view of the end-user, the derivation of a new formulation consists of two steps. The first step is the construction of each variational term, including both the choice of the discretized terms and the generation of the symbolic forms of elemental contributions. The second step is the automatic compilation of

* Corresponding author. Tel.: +33 4 84 52 56 26.
E-mail addresses: roysaad@usek.edu.lb (R. Saad), dominique.eyheramendy@centrale-marseille.fr (D. Eyheramendy), lei.zhang@centrale-marseille.fr (L. Zhang).

http://dx.doi.org/10.1016/j.advengsoft.2016.01.010
0965-9978/

these elemental forms in order to integrate them into a conventional computer code.

In Section 2, we present a review of, first, the object-oriented paradigm applied to computational mechanics and, second, symbolic approaches in the context of computational mechanics. This cover the main tracks in the structured approaches to designing computation tools in mechanics and related topics. The section is concluded with the description the proposed approach. In Section 3, we describe the concepts of a generic approach for the derivation of finite element formulations. We present the mathematical formalism on which the whole approach is based. The trial example of linear elasticity is used to illustrate this formalism. In Section 4, we introduce the object-oriented principles of the new package. We describe the main objects and the class hierarchy needed to manage the symbolic manipulations (representation and computation of symbolic forms of the elemental contributions). A trial illustration in linear elasticity is given. In Section 5, we consider a step-by-step derivation of a formulation for large strains in a hyperelastic material and some numerical tests.

2. An overview of object-oriented computational approaches and symbolic approaches

2.1. Object-oriented finite elements in mechanical engineering

Software engineering in computational mechanics has significantly evolved since the 50s following the evolution of languages. Until the 90s, the main approaches were based on procedural languages like Fortran, C or Pascal. At that time, the improvement of code maintainability and extendibility was traditionally achieved through the improvement of modularity. This could be obtained using a sequentially organized code: sequential call to functions or routines. Data structuring capabilities appeared when using languages such as C or Pascal providing sequential organization with more adequate argument management capabilities. As software complexity rapidly increased with the type of problem addressed, code maintainability and extendibility have become very difficult in this context. The application of object-oriented programming to the finite element method emerged in the late 80s. The key concept of object-oriented programming is the object. It is an entity that contains both data (attributes) and actions (methods). Objects communicate with each other by sending messages. An object is thus only an instance of class. Classes are organized in a hierarchy that allows inheritance. Thus, object-oriented programming offers a powerful alternative way of structuring codes. The pioneering work of Miller [37] and of Rehak [46] presents some basic structuring concepts of the Finite Element Method (FEM). In [37], a LISP framework for finite element computations is described. In [73], the properties of modularity and code reuse are highlighted. In [65], the efficiency in maintenance and implementation is described as a key idea of the approach. The program developed is limited to linear elasticity. Roughly speaking, the main objects proposed in these pioneering studies are related to basic structures such as nodes and elements, and to some linear algebra features. At the same time, complete approaches were developed for static and dynamic finite element analysis [3,8,49,57] and [9]. The global structuring of the FEM for linear elasticity was addressed, new features were introduced, such as the object degree of freedom in [57], and time integration algorithm objects. The nonlinear FEM was addressed later, e.g. applied to elasto-plasticity in [36]. One of the most complete approaches to nonlinear material modeling was the one proposed in [4] and [20]. Note that at the same time, some successful attempts to structure classical FE codes were made: see e.g. SIC (Interactive System Design) [22] and CAST3M [53]. In the latter, some structuring capabilities were developed based on advanced memory management systems in Fortran. However, the object-oriented design remains one of the most efficient strategies to manage complexity,

Since its origin, object-oriented programming has been widely applied in all the fields of computational mechanics and related domains (see e.g. [17,32] for an attempt at a comprehensive bibliography). We can mention a large number of papers covering a wide range of applications and computing frameworks. Among them, let us cite: numerical tools for linear algebra [56], creation of interactive codes [33], integration of artificial intelligence in finite element systems [58], fractures and damage problems [19], parallel computing in solid and fluid mechanics [1], mechanics of deformable solids in large transformations [48], multiphysics problems [7,15,51], contact problems [19] and [31]. This list, of course non-exhaustive, shows that the object-oriented paradigm is now in widespread use within the scientific computing community. This object-oriented paradigm is today a popular modeling tool to tackle the most challenging problems. Following the pioneering work, new approaches were developed in the mid 90s, bringing both additional structuring capabilities and better software integration. Among these approaches, the most popular is based on the Java language. Roughly speaking, the key features of Java are:

- It is an object-oriented programming language allowing high abstraction level data structures
- The Java virtual machine ensures a wide portability of the applications
- The Java platform provides a large number of predefined classes: I/O, object persistency, networking, multiple process management, GUIs development, security, internationalization, etc.

Java initially retained some attention for its networking capabilities and its easy Internet portability. E.g. in [44], a trial application based on a boundary element method is proposed. Similarly, a web-based application for fracture mechanics can be found in [39]. In these studies, only a few innovative structuring features are proposed. A unique way of using Java has been to use it to couple and manage traditional codes written in C/C++/Fortran. This permits the developers to use ancient codes or part of code in coupled applications, preserving the original computational efficiency. E.g. in [38], an interactive finite element application based on a coupled C++/Java is described. Comparative tests with Fortran and C are conducted on small problems using direct solvers based on tensor computations; this aims at illustrating the high efficiency computational potential of Java in the context of code coupling. In order to go further, similar conclusions are drawn in [5,13,24], where the good performances of a pure Java application are exhibited on simple matrix/vector products. In [35], the development of GUIs is put in prominent position on an unstructured mesh generator. Most computational applications, including computational mechanics applications, have been conducted within the computer science community. In [45] and [52], CartaBlanca, a Java environment for distributed computations of complex multiphase flows, is presented. This code is based on a finite volume approach, and a Newton–Krylov algorithm solution scheme is used. CartaBlanca exhibits good performances, as shown in [45]. A similar environment is developed to simulate electromagnetisms problems in [2]. Both applications show the high potential of the approach to design more complex and general computational tools in mechanics including complex parallelism paradigms. These developments exhibit the networking facilities provided by Java. The efficiency of Java has been emphasized in numerous publications focusing on various contexts of numerical analysis: direct solution of linear systems [40], FFT and iterative and direct linear systems solvers on Euler type flows [5], solution of Navier–Stokes flows [24] and [47]. More recently in [41] and [42], the description of