Computers and Structures 187 (2017) 1–23

Contents lists available at ScienceDirect

Computers and Structures

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

A computational framework for constitutive modelling

Lapo Gori *, Samuel Silva Penna, Roque Luiz da Silva Pitangueira

Department of Structural Engineering, Engineering School, Federal University of Minas Gerais (UFMG), Avenida Antônio Carlos, 6627, Pampulha, 31270-901 Belo Horizonte, MG, Brazil

A R T I C L E   I N F O

Article history:
Received 23 September 2016
Accepted 19 January 2017

Keywords:
Finite element method
Object-oriented programming
Elastic-degradation
Elasto-plasticity
Continuum damage mechanics

A B S T R A C T

The field of computational constitutive modelling for engineering applications is an active research tread in academia. New advanced models and formulations are constantly proposed. However, when dealing with implementation aspects, often the main concern is to provide a minimal environment to show a certain model and its applications, with implementations made from scratch. Though advanced, usually such implementations lack of generality and are well-suited for a certain numerical method while not compatible with other ones, making it difficult to reuse the code in other contexts. The Object-Oriented Paradigm (OOP) to programming have been widely applied in the last years for the realization of numerous academic numerical simulation softwares, due to its fundamental properties of *abstraction*, *inheritance* and *polymorphism* that allow the creation of programs well-suited for an easy collaboration between developers with expertise in different fields of engineering and mechanics. As showed in this paper, the same properties can be effectively extended also to the constitutive aspects of a model. The application of the OOP to the constitutive modelling of a wide range of materials of engineering interest is investigated, aiming to the creation of a computational framework for constitutive models that is fully independent on the other components of a code and easy to expand.

1. Introduction

The last years have seen a wide spread of academic softwares based on the Object-Oriented Paradigm (OOP) devoted to the numerical simulation of different physical problems. The well-known concepts of *abstraction*, *inheritance* and *polymorphism* of the OOP allow to achieve generality, reusability, extendibility and ease of maintenance of a code. These properties are of fundamental importance for academic softwares that, besides to be *user-friendly*, must also guarantee an easy collaboration of experts in different fields.

The advantages of the OOP in the Finite Element Method (FEM) have been pointed out in a number of works (see, e.g., the works by Mackie [1,2] or the book by Nikishkov [3]). Among the numerous academic finite element codes based on the OOP, the first that have been proposed are the softwares *SIC* [4], *CHARLY* [5], *CASTEM* [6], *OBJECT NAP* [7], while more recent applications are represented by the softwares *FEMOBJ* [8], *OOFEM* [9] and *KRATOS* [10], for example. Besides finite element applications, the OOP has been successfully used to represent also different numerical methods, such as the Boundary Element Method (BEM) [11,12], the Generalized/Extended Finite Element Method (G/XFEM) [13,14], and Mesh-Free Methods (MFM) [15,16]. A recent comprehensive OOP code is the open-source software **INSANE**[1] (INteractive Structural ANalysis Environment) [17], able to represent in a common framework different numerical methods like FEM, BEM [18], G/XFEM [19], and MFM [20], as well as different continuum descriptions, like classic and generalized ones [21,22].

A critical issue of most codes devoted to the analysis of physically non-linear problems is the effective implementation of constitutive models, as well as the treatment of non-linear systems solution. Constitutive models are usually represented inside a software in a matricial/vectorial form (the Voigt representation) using array objects, and that representation, in general, strongly depends both on the kind of analysis model of the problem (i.e., three-dimensional, plane-stress, etc.) and on the numerical method. This dependence poses a serious limit to the generality and extendibility of a code since, in this case, an operation like the introduction of a new constitutive model requires the modification of parts of the code that are not strictly related to the constitutive modelling.

A first attempt of generalization of the constitutive aspects in an object-oriented environment can be found in the work of Jeremić and Yang [23]. Taking advantage of the abstraction and inheritance mechanisms, the authors define a *template class* able to sythesize

* Corresponding author.
E-mail address: lapo@dees.ufmg.br (L. Gori).

[1] The development code is available at the Git repository URL http://git.insane.dees.ufmg.br/insane/insane.git.

http://dx.doi.org/10.1016/j.compstruc.2017.01.012
0045-7949/© 2017 Elsevier Ltd. All rights reserved.

the main aspects of elasto-plastic materials. Peculiar constitutive models are then introduced in terms of inherited classes that define specific yield criteria, flow rules and hardening-softening rules. Such generalization is made possible by the use of a peculiar C++ library that allows to handle tensorial objects in a computational environment [24,25], easing the representation of solid mechanics tensorial equations. The concept of tensors representation in an object-oriented code can be found also in the work of Weller [26]. Again, the presence of tensor objects allows to represent continuum mechanics equations in a computational environment with a syntax that is as close as possible to their mathematical representation. Within this approach, the author proposes a C++ library suitable for the solution of fluid dynamic problems with the *finite volume method*.

Regarding the solution of non-linear systems, Menétrey [27] showed that also the non-linear analysis procedures can be efficiently integrated into an object-oriented approach. These aspects have been further investigated by Dubois-Pèlerin [28] and Lages [29]. In the cited works the authors show that the modularization of linear and non-linear equations solving offered by the OOP allows an easy implementation of different equation solvers and solution control modes, and facilitates their combination in numerical applications.

In this paper, a novel approach to computational constitutive modelling based on the OOP, that makes use of concepts previously discussed by Penna [30], is investigated. The initial approach proposed by Jeremić and Yang [23] for elasto-plastic materials is generalized in order to accomodate also elastic-degrading media, based on both single and multiple loading functions. The theoretical basis is represented by a unified formulation for constitutive models [31–37,30], i.e., a theoretical resource able to represent a large number of elasto-plastic and elastic-degrading models in a common framework, in terms of constitutive operators, loading functions and degradation/flow rules. From a theoretical point of view, the generality of the representation is guaranteed by the tensorial formalism of the involved constitutive equations, as illustrated by Rizzi and Carol [38]. Following concepts analogous to the ones discussed by Jeremić and Sture [24] and by Weller [26] for the handling of tensor objects in a computational environment, the constitutive models based on the unified formulation are introduced in the code directly with their tensor-based format rather than with the matricial/vectorial expressions of the Voigt representation, i.e., with a syntax that is as close as possible to their original mathematical representation. In this way the aforementioned unified formulation can be reproduced in the code taking full advantage of the object-oriented paradigm; the abstract general equations of the formulation are introduced in their tensorial format, and the peculiar constitutive models are derived within this scheme using the concept of inheritance. The proposed tensor-based approach to constitutive modelling results in a general framework for elasto-plastic and elastic-degrading media that, with respect to the aforementioned papers and to existent object-oriented codes, is able to guarantee a complete independence between the constitutive aspects and the other parts of the code, with specific emphasis on analysis models and numerical methods. In the first part of the paper, the basic aspects of the unified formulation for constitutive models are briefly recalled, and the derivation of known constitutive models within this formulation is showed. The second part focuses on the computational aspects of the proposed framework and on its implementation in the **INSANE**. The main aspects of the software are briefly presented, while the organization of the constitutive models framework, together with the relations that exist between the framework itself and the other parts of the code, is presented in details. Also the handling of tensor objects and the approach for non-linear system solving are discussed. Finally, numerical simulations are performed in order to emphasize the advantages of the proposed approach.

1.1. Notations

Some standard notations used in the body of the paper are summarized here. With the symbols $\mathbf{E}, \bar{\mathbf{E}}$ and $\mathbb{R}^N$, the *environment space* (a three-dimensional Euclidean space), its associated *vector space* and the *N-dimensional Euclidean space* are indicated. The symbols $(\bar{e}_i)$ and $(\bar{r}_\alpha)$ indicate, respectively, a basis of $\mathbf{E}$ and a basis of $\mathbb{R}^N$. Latin indexes are assumed to run from 1 to 3, while greek indexes assume values from 1 to N. *Vectors* are indicated as $\bar{x} \in \bar{\mathbf{E}}$, with $\bar{x} = x_i\ \bar{e}_i$, while *second-order* and *fourth-order* tensors respectively by $\underline{x} \in \bar{\mathbf{E}} \otimes \bar{\mathbf{E}}$, with $\underline{x} = x_{ij}\ \bar{e}_i \otimes \bar{e}_j$, and by $\hat{\mathbf{x}} \in \bar{\mathbf{E}} \otimes \bar{\mathbf{E}} \otimes \bar{\mathbf{E}} \otimes \bar{\mathbf{E}}$, with $\hat{\mathbf{x}} = x_{ijk\ell}\ \bar{e}_i \otimes \bar{e}_j \otimes \bar{e}_k \otimes \bar{e}_\ell$. The tensors of *third-*, *fifth-* and *sixth-order* used in this paper are of mixed type, and are represented respectively by $\tilde{\mathbf{x}} \in \mathbb{R}^N \otimes \bar{\mathbf{E}} \otimes \bar{\mathbf{E}}$, with $\tilde{\mathbf{x}} = x_{\alpha ij}\ \bar{r}_\alpha \otimes \bar{e}_i \otimes \bar{e}_j$, by $\check{\mathbf{x}} \in \mathbb{R}^N \otimes \bar{\mathbf{E}} \otimes \bar{\mathbf{E}} \otimes \bar{\mathbf{E}} \otimes \bar{\mathbf{E}}$, with $\check{\mathbf{x}} = x_{\alpha ijk\ell}\ \bar{r}_\alpha \otimes \bar{e}_i \otimes \bar{e}_j \otimes \bar{e}_k \otimes \bar{e}_\ell$, and by $\breve{\mathbf{x}} = \mathbb{R}^N \otimes \bar{\mathbf{E}} \otimes \bar{\mathbf{E}} \otimes \mathbb{R}^N \otimes \bar{\mathbf{E}} \otimes \bar{\mathbf{E}}$, with $\breve{\mathbf{x}} = x_{\alpha ij\beta k\ell}\ \bar{r}_\alpha \otimes \bar{e}_i \otimes \bar{e}_j \otimes \bar{r}_\beta \otimes \bar{e}_k \otimes \bar{e}_\ell$ The symbol $\cdot$ denotes both the standard dot product between vectors and the total contraction between tensors like, for example, $\bar{x} \cdot \bar{y} = x_i\ y_i, \underline{x} \cdot \bar{y} = x_{ij}\ y_j\ \bar{e}_i$, $\hat{\mathbf{x}} \cdot \underline{y} = x_{ijk\ell}\ y_{k\ell}\ \bar{e}_i \otimes \bar{e}_j$ and the other possible combinations. The same symbol is used, with a slight abuse of notation, also for contractions with mixed order tensor, like $\underline{x} \cdot \tilde{\mathbf{y}} = x_{ij}\ y_{\alpha ij}\ \bar{r}_\alpha$, since there is no risk of confusion between the different indexes. With the symbol $\otimes$, the standard tensorial product, as $\bar{x} \otimes \bar{y} = x_i\ y_j\ \bar{e}_i \otimes \bar{e}_j$ or $\underline{x} \otimes \underline{y} = x_{ij}\ y_{k\ell}\ \bar{e}_i \otimes \bar{e}_j \otimes \bar{e}_k \otimes \bar{e}_\ell$, is indicated. In case of mixed tensors, combinations are given, for example, by $\tilde{\mathbf{x}} \otimes \tilde{\mathbf{y}} = x_{\alpha ij}\ y_{\beta k\ell}\ \bar{r}_\alpha \otimes \bar{e}_i \otimes \bar{e}_j \otimes \bar{r}_\beta \otimes \bar{e}_k \otimes \bar{e}_\ell$. In some applications the Voigt notation will be used to represent second-order and fourth-order tensors; once a certain coordinates system has been fixed, a generic second-order tensor with dimension three $\underline{x}$ can be represented by means of an *array* with nine components, indicated with the symbol $\{\underline{x}\}$. In an analogus way, a fourth-order tensor with dimension three $\hat{\mathbf{x}}$ can be represented by means of a 9×9 matrix, indicated as $[\hat{\mathbf{x}}]$. It should be noted that the provided dimensions refer to a general three-dimensional case; in different situations (e.g., plane-strain or plane-stress states), the size of arrays and matrices in Voigt representation is minor. The same symbols $\{\cdot\}$ and $[\cdot]$ are also used in Section 3 to indicate, respectively, arrays and matrices in numerical equations.

2. Unified formulation for constitutive models

As discussed in the introduction, the theoretical basis for the proposed computational framework is represented by a unified formulation for constitutive models that has been developed in the last years by a number of authors (see, e.g., [31–37,30]). The core of the formulation is constituted by a set of general equations that provide a common representation for different material behaviours; from such a general representation, specific constitutive models are defined in terms of peculiar constitutive operators, loading functions, and degradation/flow rules. In the following, a brief resume of the formulation presented in the cited papers is provided, with specific emphasis on its tensorial expression; the derivation within this formulation of some known constitutive models is also discussed.

In a geometrically linear context, an elastic-degrading medium is characterized by the following *total* stress-strain relations

$$\underline{\sigma} = \hat{\mathbf{E}}^S \cdot \underline{\varepsilon}, \qquad \underline{\varepsilon} = \left(\hat{\mathbf{E}}^S\right)^{-1} \cdot \underline{\sigma} \tag{1}$$

where $\hat{\mathbf{E}}^S$ is the *secant* constitutive operator. These equations correspond to the assumption of an *unloading-reloading* process where the stiffness remains equal to the current secant one; in this case, a full unload leads to zero permanent strains. The time derivatives of the expressions in Fig. 1 result in the equations