Advances in Engineering Software 105 (2017) 1–8

Contents lists available at ScienceDirect

Advances in Engineering Software

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

Research paper

MUESLI - a Material UnivErSal LIbrary

David Portillo[b,c], Daniel del Pozo[c], Daniel Rodríguez-Galán[c], Javier Segurado[a,c], Ignacio Romero[b,c,*]

[a] Dept. Material Science ETSI Caminos, Technical University of Madrid, Spain
[b] Dept. Mechanical Engineering ETSI Industriales, Technical University of Madrid, Spain
[c] IMDEA Materials Institute, C/Eric Kandel 2, 28906 - Getafe, Madrid, Spain

ARTICLE INFO

Article history:
Received 4 July 2016
Revised 14 October 2016
Accepted 17 January 2017
Available online 21 January 2017

Keywords:
Material modeling
Open source
Software library
Automatic testing

ABSTRACT

This article describes MUESLI, an open source library with constitutive models of continuum materials for solid, fluid, and thermal problems available at http://www.materials.imdea.org/Muesli. The library is object oriented, and includes the most commonly employed material models in Computational Mechanics. It is designed for easy extension, and includes classes for tensor manipulation and automatic testing. The library can be linked to existing codes, including commercial ones, for some of which specific interfaces are provided.

1. Introduction

Material modeling is at the core of Computational Mechanics. As a result, most research and commercial simulation codes for Mechanics have routines in their kernels that compute material response. These are responsible for calculating the response of material points when subjected to deformation, thermal loads, chemical changes, etc. Many researchers and code developers in this discipline have, at some point, implemented standard material models such as elasticity, plasticity, viscoelasticity (for small and large strain kinematics), Fourier's law, compressible and incompressible fluid response, complex fluids, coupled models, etc. Despite the apparent common interests identified above, there does not seem to be, to the authors' knowledge, a common effort to provide a clear, robust, extendable, reusable, and reliable collection of material routines that can speed up the development of new models, and simplify the exchange of existing ones.

The formulation of material models for continuum applications has reached a mature state that reflects on the publication of books and monographs that share much of their contents, in this respect. After the monumental work of Truesdell and Noll [1], much of the formalism for continuum material modeling, and even the notation to some extent, has become standard. One can find (parts of) this common body of knowledge in well-known books such as [2–7].

One of the aspects of this theory that has become unanimously accepted is the use of tensor algebra as the most convenient language to describe material behavior. Whether for working with stresses, or the material tangent elasticities, tensors provide the natural arena and mathematical formalism to manipulate these mechanical concepts. In Computational Mechanics, however, Voigt's notation has been very widely employed to represent second and fourth order tensors [8]. This notation maps second and fourth order tensors to vectors and matrices, respectively, at the expense of using cumbersome algebraic operations (cf., e.g., [9]). Although using Voigt's notation is not strictly required for many numerical methods [10], it can be argued that its widespread use follows from the choice of Fortran in the initial developments of simulation codes. These days, more sophisticated programming languages such as C++, possess operator overloading allowing the implementation of material models in a way that closely mimics the mathematical description of standard references. Either using index or compact notation, overloaded operations can be made as fast as matrix operations in Voigt's, or even faster, if using, for example, sophisticated template metaprogramming [11].

A second aspect in the use of material models that has changed over the years is related to the proliferation of commercial and research simulation codes. Indeed, as a result of the relative maturity of Computational Mechanics and the wide availability of open

* Corresponding author.
E-mail addresses: david.portillo@upm.es (D. Portillo), daniel.pozo@imdea.org (D. del Pozo), daniel.rodriguez@imdea.org (D. Rodríguez-Galán), javier.segurado@imdea.org (J. Segurado), ignacio.romero@imdea.org, ignacio.romero@upm.es (I. Romero).

http://dx.doi.org/10.1016/j.advengsoft.2017.01.007
0965-9978/

source libraries most research groups possess their own code and also access to commercial software such as Abaqus, Ansys, Fluent, Nastran, etc. Since the data structures of research and commercial codes need not be the same, and often not even the programming language, sharing the material models is not straightforward. Some may decide upfront to develop material models for commercial codes using the available *user routines*, adhering to the software's interface *and* programming language. The alternative route, developing models for one's own code might be faster and simpler but does not allow to benefit from all the tools at one's disposal when using commercial codes.

The reasons listed above indicate that it would be beneficial to the Computational Mechanics community to have access to a material library that includes common models, simplifies the development of new ones, and allows for easy sharing between commercial and research codes. Having already identified these needs, we describe in this article MUESLI, a Material UnivErSal LIbrary, designed to overcome all of them and to be accessible as open source code for all developers. One of the main initial decisions has been to develop the library in C++, an object oriented language that can take advantage of this feature to structure materials in families and subfamilies. In addition, and as mentioned above, the operator overloading capacity of this language makes the implementation as natural as possible.

The library currently provides support for three-dimensional small strain and finite strain solid materials, fluid materials, thermal materials, and coupled models. Overall there are almost twenty models in version 1.0 of the library, covering the basic, most commonly employed materials. All these material models are developed according to the library's design guidelines, but interfaces are provided to Abaqus and LS-Dyna. Users of the library can extend it by adding their own models to the existing families. In the future, we plan to grow the library by adding more materials and interfaces.

A final feature that is often required from scientific computing codes is the ability to provide some kind of checks regarding the correctness of the implementation. For material models, it is often the case that there is an energy (elastic, thermal, etc.) from which stresses or other gradients are derived; in addition, the linearization of the latter is required for the iterative solution of nonlinear problems. With this in mind, MUESLI provides a limited suit of checks for each material family, checks that can be used to verify the correctness of the implementation, up to a certain extent. For complex models, these checks can be of great aid in identifying bugs and other sources of error.

To better understand the context in which MUESLI has appeared, we mention similar efforts developed in the Computational Mechanics and Material community. Regarding commercial software, the library Digimat [12] provides a large database of models for composite materials, including multiscale analysis; Z-mat [13] includes mechanical and thermal constitutive laws for many elastic and inelastic materials, and provides an interface with ABAQUS. Neither of these libraries can be extended by the user, and their code is closed. PolyUMod [14], focused on constitutive models for polymers, has a free version and a commercial one, but are both closed as well. The open-source MFront library [15] shares with MUESLI the goal to provide general constitutive material models for several simulation codes. The approach of the two projects, however, is different: MFront defines a *meta-language* for the high-level description of material behavior which is then converted to C++ code, in the form that is suitable for certain finite element codes. MFront is much larger than MUESLI, and as a result, more complex.

The remainder of the article is organized in the following way: Section 2 describes the main concepts and philosophy behind MUESLI. Section 3 explains how the library can be linked with existing codes and Section 4 outlines its automatic checking capabilities. Section 6 summarizes the main aspects of the library and provides details to obtain it.

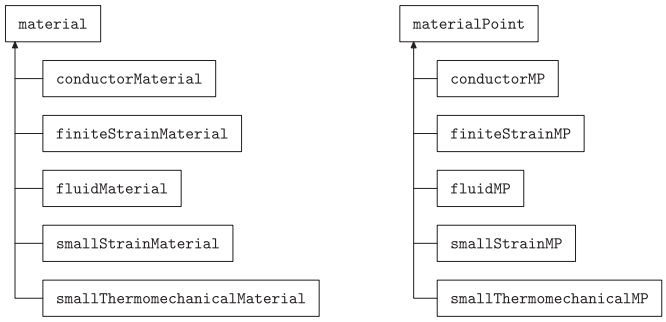

Fig. 1. Material families in MUESLI, as defined by the `material` classes and their corresponding `materialPoint` counterparts.

2. Structure and software design

MUESLI is an object oriented library designed to implement the constitutive behavior of materials at the continuum level. While the analytical definition of the material responses are fairly standard, at least for many models, the organization and structure of the library is completely original. It has been designed to offer developers the maximum simplicity, without sacrificing the capacity to implement the most involved material models currently used in Computational Mechanics.

2.1. Material families

Materials are used, in a broad sense, in every device, part, or structure. When using simulation tools to analyze these elements, material models are employed in very different ways. The design of a material library needs to encompass many of these needs if it is to be used by diverse types of users. MUESLI has been developed with this generality in mind, and classes are structured into *material families*. Each of these refers to (one or) materials with the same interface and needs. For example, the `smallStrainMaterial` family defines the interface and minimal requirements of materials employed in mechanical simulations with small strain kinematics. In some cases a material family might delegate part of its implementation to other, previously defined, family. For instance, the `smallThermomechanicalMaterial` defines the interface for coupled thermo-mechanical response. The purely mechanical response in these models is entrusted to the `smallStrainMaterial` class, while the coupled material takes care of the coupling and the thermal part. See Fig. 1 for an illustration of the currently implemented families in MUESLI.

2.2. Materials and material points

The fundamental concepts in MUESLI are those of *material* and *material point* and are implemented in the classes `material` and `materialPoint`, respectively. The first one embodies the idea of *abstract material*, an entity which can spawn `materialPoints` and hold common data for all the children. Using the terminology of software development, each specific `material` includes a *factory method*. The second one is the one that represents individual points in real continuum bodies: each of them having a specific material constitutive behavior and aware of its past history.

Fig. 2 illustrates the concepts of `material` and `materialPoint`. When creating a computational model of a continuum, the analyst must decide which materials it is going