Introduction

The ability to easily integrate user-defined constitutive equations plays a major role in the versatility of (mechanical) solvers. Constitutive equations describe how the internal state variables of a material evolve with changing external conditions or mechanical loading. Those state variables can describe many microstructural aspects of the material (grain size, dislocation density, hardening state, etc.) or be phenomenological in nature (equivalent plastic strain). The knowledge of those internal state variables allows the computation of local thermodynamic forces which affect the material equilibrium at the structural scale. Due to the large number of phenomena that can be described in this manner, such as plasticity, viscoplasticity, or damage, computational mechanics is one of the most demanding domains for advanced constitutive equations.

At each time step, the constitutive equations must be integrated to obtain the state of the material at the end of the time step. As most phenomena are nonlinear, an iterative scheme is required at the equilibrium scale to find the local loading of the material: the integration of the constitutive equations is therefore called several times with different estimates of the loading of the material. Algorithmic efficiency at the constitutive level is therefore a key aspect for the overall efficiency of a code.

The MFront open-source code generator has been designed to simplify the implementation of the integration of the constitutive equations over a time step, to minimize errors during implementation, to facilitate the portability of constitutive equations between solvers, and to help achieve a reproducible and efficient code (CEA & EDF, 2019; Helfer et al., 2015). For that purpose, MFront uses a source file with a syntax very close to a physical/engineering description of the constitutive model, and generates C++ code specific to many well-established (mostly thermo-mechanical) solvers through dedicated interfaces and compiles them into shared libraries. For example, MFront provides interfaces for Cast3M, code_aster, Europlexus, Abaqus/Standard, Abaqus/Explicit, CalculiX, etc.

To further facilitate this cross-software integration, MFront recently introduced a so-called generic interface. This paper describes the MFrontGenericInterfaceSupport project, which is denoted MGIS in the following. MGIS aims at proving tools (functions, classes, etc.)

1The term solver emphasizes that the numerical method used to discretize the equilibrium equations is not significant in the present context.
bindings to various programming languages) to handle behaviours\(^2\) generated using MFront’s generic interface (Helfer, 2019a). Those tools alleviate the work required by solver developers. Permissive licences have been chosen to allow integration in open-source and proprietary codes.

This paper is divided into three parts:

1. A brief overview of MGIS.
2. A description of the various bindings available.
3. Some examples of usage in various open-source solvers: FEniCS, OpenGeoSys and JuliaFEM.

**Overview**

The aims of the MFrontGenericInterfaceSupport project are twofold:

1. At the pre-processing state MGIS shall provide the possibility of retrieving metadata about a particular behaviour and performing proper memory allocation. At the post-processing stage, easy access to internal state variables is desired.
2. During computations, MGIS shall simplify the integration of the behaviour at integration points\(^3\) and the update of the internal state variables from one time step to the next.

**Preprocessing and post-processing stages**

When dealing with user-defined behaviours, most solvers, including Abaqus/Standard for example, delegates part of the work to the user. The user must:

1. Describe the behaviour in the input.
2. Take care of the consistency of the behaviour with the hypothesis made during the computation (e.g. a finite strain behaviour must be used in a finite strain analysis based on the appropriate deformation and stress measures as well as reference configurations).

In the authors’ experience, this is error-prone in particular for inexperienced users and may lead to spurious or even worse inexact results. For example, when material properties such as the Young Modulus or the Poisson ratio, are required by the behaviour, those are generally defined in the solver input file by an array of values and potential checks are limited to the size of the array. An user may thus invert two material properties. If those two material properties have the same order of magnitudes, computations might lead to a physically consistent result, despite this result being false.

MGIS introduces a very different approach: the user only declares the shared library, the behaviour and the modelling hypothesis (tridimensional, plane strain, etc.). With this information, the library retrieves various metadata which fully describe how to interact with the behaviour. The solver using MGIS can then check if the behaviour is consistent with the computations to be performed and checks that the data provided by the user are correct. The metadata can also be used to allocate the memory required to store the state of the material at each integration point. MGIS’ design allows the following types of storage:

\(^2\) In the following, we use the term “behaviour” to denote the result of the implementation and compilation of the constitutive equations.

\(^3\) The term “integration points” is used here as a generic placeholder. When using FFT for solving the equilibrium equations, the integration points are voxels. When using FEM, the integration points are typically the Gauss points of the elements.
An MGIS data structure per integration point. This is the most frequent choice. The memory is automatically allocated by MGIS.

An MGIS data structure that stores the states of an arbitrary number of integration points. MGIS can allocate the memory associated with the state of all specified integration points or borrow memory allocated by the solver.

For post-processing, MGIS provides a set of functions to retrieve information about the state of the material. For example, one can retrieve the value of a state variable from the described data structures.

**Computation stage**

MGIS provides a function to integrate the constitutive equations at one integration point or on a set of integration points⁴. The integration of the constitutive equations at different integration points are usually independent of each other; in other words, in most models the constitutive behaviour is local. Thus, when handling a set of integration points, MGIS can parallelize the integration calls using a granularity chosen by the solver.

**Main language and available bindings**

MGIS is written in C++11. The C++ API is described in another report, see Helfer (2019b). The following bindings are currently available:

- python.
- Julia.
- Fortran 2003.
- C.

**Examples of usage**

**FEniCS**

![Figure 1: Large strain elasto-plastic modelling of a notched bar.](image)

FEniCS is a popular open-source computing platform for solving partial differential equations (Alnæs et al., 2015; Logg, Mardal, Wells, & others, 2012).

⁴This strongly depends on the data structure chosen to store the internal state variables.
Non linear mechanics computations combining FEniCS at the equilibrium scale and MFront to describe the constitutive equations can be performed through the python bindings of MGIS as demonstrated by Bleyer et al. (see Bleyer & Helfer (2019a); Bleyer & Helfer (2019b)).

Extensions to finite strain elastoplasticity have been recently added as shown in Figure 1 which models a tensile test on a notched bar5.

**OpenGeoSys**

OpenGeoSys (OGS) is a scientific open-source initiative for the numerical simulation of thermo-hydro-mechanical/chemical (THMC) processes in porous and fractured media, inspired by FEFLOW and ROCKFLOW concepts and continuously developed since the mid-eighties (see Kolditz (1990); Wollrath (1990); Kroehn (1991); Helmig (1993); Kolditz et al. (2012); Bilke et al. (2019)).

The OGS framework is targeting applications in the environmental geosciences, e.g., in the fields of contaminant hydrology, water resources and waste management, geotechnical applications, geothermal energy systems and energy storage.

The most recent version, OpenGeoSys–6 (OGS–6) (Naumov et al. (2018); Bilke et al. (2019)), is a fundamental re-implementation of the multi-physics code OpenGeoSys–4/5 (Kolditz & Bauer (2004); Wang & Kolditz (2006)) using advanced methods in software engineering and architecture with a focus on code quality, modularity, performance and comprehensive documentation.

Among its recent extensions are the implementation of numerical methods for the propagation of discontinuities, such as enriched finite element function spaces, non-local formulations and phase-field models for fracture (Watanabe, Wang, Taron, Görke, & Kolditz (2012); Parisio et al. (2019); Yoshioka et al. (2019)).

To simplify the implementation of new constitutive models for solids developed with MFront, OGS–6 relies on the C bindings of MGIS.

**Figure 2**: Slope stability analysis with strength reduction performed in OpenGeoSys. The image on the left shows the norm of the displacement vector for a low top load. The image on the right shows the equivalent plastic strain for a setting with an increased top load.

Figure 2 shows the results of a $\phi - c$ reduction approach to slope stability analysis. The soil is modelled by a non-associated plastic behaviour based on the Mohr-Coulomb yield criterion (Abbo & Sloan, 1995; Zienkiewicz & Pande, 1977). The implementation presented in Nagel et al. (2017) was extended by a tension cut-off. The image on the left shows the norm of the displacement vector for a low top load. The failure kinematics are clearly visible. The image on the right shows the equivalent plastic strain for a setting with an increased top load. It can be seen that the failure mechanism becomes more complex with an additional slip surface forming beneath the load.

5This case is adapted from a non-regression test of the Code_Aster finite element solver, see EDF (2011) for details
JuliaFEM

JuliaFEM (Aho et al., 2019a, 2019b; Frondelius & Aho, 2017; Rapo, Aho, & Frondelius, 2017; Rapo, Aho, Koivurova, & Frondelius, 2018; Rapo et al., 2019) is an open-source finite element solver written in the Julia programming language (Bezanson, Edelman, Karpinski, & Shah, 2017). JuliaFEM enables flexible simulation models, takes advantage of the scripting language interface, which is easy to learn and embrace. Besides, it is a real programming environment where other analyses and workflows combine with simulation.

![Block diagram showing the software layers involved in using MFront behaviours in JuliaFEM.](image)

**Figure 3:** Block diagram showing the software layers involved in using MFront behaviours in JuliaFEM.

The MFrontInterface.jl (Frondelius, Helfer, Yashchuk, Vaara, & Laukkanen, 2019) is a Julia package where MFront material models are brought to Julia via wrapping MGIS, see Figure 3. Installation is, as easy as any julia packages, i.e., `pkg> add MFrontInterface`. For example TFEL and MGIS cross-compiled binary dependencies are automatically downloaded and extracted. Lastly, Figure 4. shows a simple 3D geometry example using JuliaFEM and MFrontInterface together.

![Simple isotropic plasticity modelling of a 3D beam in JuliaFEM with MFrontInterface.](image)

**Figure 4:** Simple isotropic plasticity modelling of a 3D beam in JuliaFEM with MFrontInterface.

Conclusions

This paper introduces the MFrontGenericInterfaceSupport library which considerably eases the integration of MFront-generated behaviours in any solver. In particular, the library

provides a way of retrieving the metadata associated with a behaviour, data structures to store the physical information and functions to perform the behaviour integration over a time step. Examples of usage in various open-source solvers (FEniCS, OpenGeoSys, JuliaFEM) have been provided.

The models implemented for one code can easily be used in another without the need for re-implementation. This offers great benefits for code quality assurance. Since the constitutive integration is handled by MFront, this step of the computation is equally efficient across the different solver platforms.

Acknowledgements

This research was conducted in the framework of the PLEIADES project, which is supported financially by the CEA (Commissariat à l’Energie Atomique et aux Energies Alternatives), EDF (Electricité de France) and Framatome.

We would like to express our thanks to Christoph Lehmann, Francesco Parisio, Olaf Kolditz and the entire community of developers and users of OpenGeoSys(OGS). We thank the Helmholtz Centre for Environmental Research – UFZ for long-term funding and continuous support of the OpenGeoSys initiative. OGS has been supported by various projects funded by Federal Ministries (BMBF, BMWi) as well as the German Research Foundation (DFG). We further thank the Federal Institute for Geosciences and Natural Resources (BGR) for funding.

Also, we would like to acknowledge the financial support of Business Finland for both ISA Wärtsilä Dnro 7734/31/2018, and ISA VTT Dnro 7980/31/2018 projects.

This project uses code extracted from the following projects:

- https://github.com/bitwizeshift/string_view-standalone by Matthew Rodusek
- https://github.com/mpark/variant by Michael Park
- https://github.com/progschj/ThreadPool by Jakob Progsch and Václav Zeman
- https://github.com/martinmoene/span-lite by Martin Moene

References


