CSMA 2026
17ème Colloque National en Calcul des Structures
18-22 Mai 2026, Presqu'île de Giens (Var)

New functionalities of Versions 5.0 and 5.1 of the TFEL/MFront project and Versions 3.0 and 3.1 of the MGIS project

Thomas Helfer[(1)], Antoine Martin[(2)], Jérémy Hure[(3)], Thibault Barret[(4)]

[(1)] *CEA, DES, IRESNE, DEC, SESC, Cadarache, F-13108 Saint-Paul-Lez Durance, France, thomas.helfer@cea.fr*
[(2)] *CEA, DES, IRESNE, DEC, SESC, Cadarache, F-13108 Saint-Paul-Lez Durance, France, antoine.martin@cea.fr*
[(3)] *Université Paris-Saclay, CEA, Service d'Étude des Matériaux Irradiés, 91191, Gif-sur-Yvette, France, jeremy.hure@cea.fr*
[(4)] *Université Bretagne Sud, UMR CNRS 6027, IRDL, Lorient F-56100, France thibault.barret@univ-ubs.fr*

Abstract — As part of the open-source `TFEL` project,`MFront` simplifies the implementation of complex mechanical behaviours efficiently and portably across various finite element and FFTsolvers. This paper highlights important new features introduced in `TFEL/MFront` Versions 5.0 and 5.1. It also outlines how the `MFrontGenericInterfaceSupport` project enables straightforward integration of MFront behaviours into a range of open-source or commercial solvers

Keywords: Constitutive equations, Material knowledge management, Homogenization

Introduction

The projects `TFEL/MFront` and `MGIS` have been actively developed since the talk given at the last session of this conference (1), where Version 4.2 of `TFEL/MFront` and Version 2.0 of `MGIS` were presented. Those new developments have been made with the contributions and feedback of a large community of academic and industrial users, as demonstrated by the various talks at the MFront user days[1] and the list of publications using `MFront`[2].

This paper is devoted to highlighting a selected set of features introduced in versions 5.0 and 5.1 of the `TFEL/MFront` project and versions 3.0 and 3.1 of the `MGIS` project [3].

This paper is organized as follows:

- Section 1 provides an overview of `TFEL`, `MFront`, `MTest`, and `MGIS`.
- Section 2 provides a description of the main new features in `TFEL` of Versions 5.0 and 5.1. and in `MGIS` of Versions 3.0 and 3.1.
- Section 3 provides a description of some side projects such as `MFrontGallery` and `TFELMathEnzyme`.

1 Overview of `TFEL`, `MFront`, `MTest` and `MGIS`

The `TFEL` project provides mathematical libraries that are the basis of the `MFront` code generator and the `MTest` solver (2). It is an open-source collaborative development of the French Alternative

[1] https://github.com/thelfer/tfel-doc/tree/master/MFrontUserDays
[2] https://thelfer.github.io/tfel/web/publications.html
[3] The interested reader may refer to the release notes of those versions available on the website of each project for a comprehensive and detailed description.

Energies and Atomic Energy Commission (CEA) and Électricité de France (EDF) within the framework of the PLEIADES platform (3).

TFEL/MFront is available on a large number of platforms, including LinuX, macOS, Windows, FreeBSD with various C++ compilers (gcc, clang, Microsoft's Visual Studio, NVIDIA's nvhpc, and Intel's icpx). TFEL/MFront can be installed in several popular package managers, including spack, conda, and homebrew. Moreover, it is distributed with the Cast3M and code_aster solvers.

1.1 The TFEL/Math library

The TFEL/Math library provides a linear algebra engine with mathematical objects (tensors of arbitrary orders) and operations on those objects required to express the constitutive equations in an efficient and natural manner i.e., as close as possible to the mathematical expressions common in the engineering sciences. These mathematical objects can have units, allowing the compiler to perform dimensional analysis at compile-time. A framework to build nonlinear solvers for small-sized problems. Efficient and robust implementations of several classical non linear algorithms (Newton-Raphson, Broyden, Levenberg-Marquart, etc.) are provided.

This library has been ported to GPUs and tested in the main programming models (CUDA, HIP, SYCL, Kokkos, etc.). Section 3.3 describes how the Enzyme library can be used to add automatic differentiation to TFEL/Math.

1.2 The TFEL/Material library

The TFEL/Material library provides implementations of various utility functions frequently required in constitutive modelling (computation of the Lamé coefficients, computation of the Hill tensors). In particular, computations of many popular stress criteria and their derivatives with respect to the stress tensor (Hill 1948, Hosfort 1978, Gurson 1977, Gurson-Tvergaard-Needlman 1984, Barlat 2004, Mohr-Coulomb, etc.) have been implemented.

This library has been recently extended to provide useful functions for mean-field homogeneization, as described in Section 2.1.

1.3 The MFront code generator

MFront translates a set of closely related domain-specific languages into plain C++ on top of the TFEL libraries.

Those languages are meant to be easy to use and learn by researchers and engineers. They cover three kinds of material knowledge: material properties (Young's modulus, thermal conductivity, etc.), mechanical behaviours and simple point-wise models (such as material swelling under irradiation used in fuel performance codes). Concerning behaviours, the following kinds of behaviours are supported:

- small and finite strain mechanical behaviours.
- cohesive zone models.
- generalised behaviours, such as the one encountered in higher order theories, Cosserat media, heat transfer, strongly coupled thermomechanical analysis, damage gradient models, etc.

Authors of MFront pay particular attention to the robustness, reliability and numerical efficiency of the generated code, in particular for mechanical behaviours. Various benchmarks show that MFront implementations are competitive with native implementations available in the Cast3M, code_aster, Europlexus, Abaqus/Standard, Abaqus/Explicit (4) solvers and in the fuel performance codes Cyrano3 (5) and Galileo (6).

Portability is also a very important issue: a behaviour written in MFront shall be usable in any solver for which an interface exists. In addition to the aforementioned solvers, interfaces exist

for: Ansys, ZMat, CalculiX, DianaFEA.

1.4 The generic interface and the MFrontGenericInterfaceSupport project

To limit the number of interfaces supported by MFront, an interface called generic has been introduced, along with the MFrontGenericInterfaceSupport project (MGIS), which provides solver developers tools (functions, classes, bindings, etc...) to handle behaviours generated by this generic interface.

The MFrontGenericInterfaceSupport project has already been integrated or tested in many solvers (7), including code_aster, Manta (8), MFEM-MGIS (9), OpenGeoSys (10), dolfinx_materials, etc.

2 New features

Behind the scenes, Version 5.0 of TFEL and Version 3.0 of MGIS were rewritten to take advantage of C++-20. This section highlights some noticeable new features from the end user's point of view provided by these two projects.

2.1 The TFEL/Material/Homogenization library

Homogenization tools have been added to the TFEL library and can be used in MFront. These tools include Hill tensors and localisation tensors, classical homogenization bounds and computation of homogenized stiffness by classical schemes, for very general microstructures.

2.1.1 Eshelby tensors, Hill tensors and localisation tensors

The computation of Eshelby tensors or Hill tensors or localisation tensors is analytically possible (11) when the inclusion is ellipsoidal and embedded in an isotropic matrix.

Three cases are considered in the library: spherical, ellipsoidal and spheroidal (or axisymmetrical) inclusions. Some examples are given below.

```
using namespace tfel::material::homogenization::elasticity;
const auto S0 = computeSphereEshelbyTensor<stress>(nu0);

//IM0 is a YoungNuModuli object, giving the isotropic
//moduli of the matrix phase
//n_a gives the orientation of the spheroid
const auto IM0=YoungNuModuli<stress>(E0,nu0);
const auto P0 = computeAxisymmetricalHillPolarisationTensor<stress>(IM0,n_a,e);

//here, if the axes a,b,c are all different, two vectors n_a and n_b
//are necessary to precise the orientation of the ellipsoid
const auto IMi=YoungNuModuli<stress>(Ei,nui);
const auto A = computeLocalisationTensor<stress>(IM0,IMi,n_a,a,n_b,b,c);
```

These tensors can also be computed when considering plane strain elasticity, or anisotropic elasticity of inclusions.

2.1.2 Computation in anisotropic matrix

When the inclusions are embedded in a fully anisotropic matrix, the analytical computation is no more possible and a numerical integration on a bi-dimensional domain is carried out to compute these tensors.

They can be computed as follows:

```
// With an anisotropic matrix, the elasticity C0 is a 'st2tost2' object
const auto S0 = computeAnisotropicEshelbyTensor<stress>(C0,n_a,a,n_b,b,c);
const auto P0 = computeAnisotropicHillTensor<stress>(C0,n_a,a,n_b,b,c);

//here Ci_loc is a 'st2tost2' object giving the inclusion elasticity
const auto A =
    computeAnisotropicLocalisationTensor<stress>(C0_glob,Ci_loc,n_a,a,n_b,b,c);
```

The user can also control the precision of the integration by providing some additional arguments. Moreover, the tensors can be computed when considering plane strain elasticity.

2.1.3 Homogenization bounds

Different homogenization bounds are available for an arbitrary number of phases, in dimension 2 or 3: Reuss bounds, Voigt bounds and Hashin-Shtrikman bounds (for isotropic phases).

2.1.4 Homogenization schemes

The homogenized stiffness tensors given by classical mean-field homogenization schemes are available for different types of microstructure.

A first set of functions is defined when considering simple particulate microstructures, biphasic, with isotropic phases. They do not require the instantiation of microstructure objects and are more suited to fast computation.

Different distributions of ellipsoids can be considered:

- spheres (no orientations)
- oriented ellipsoids (two vectors $\underline{n}_a$,$\underline{n}_b$ define the orientation)
- uniform isotropic distribution of orientations (the ellipsoids have no preferential orientation)
- transverse isotropic distribution of orientations (one axis $\underline{n}_a$ of the ellipsoid is fixed, the others are uniformly distributed in the transverse plane)

For example, we can do:

```
const auto IM0=YoungNuModuli<stress>(E0,nu0);
const auto IMi=YoungNuModuli<stress>(Ei,nui);
const auto KG_DS = computeSphereDiluteScheme<stress>(IM0,f,IMi);

//Here the ellipsoid is not necessarily a spheroid, and 'TransverseIsotropic'
//means that the axis relative to length a has a fixed orientation (given by
//n_a) and the two other axes rotate
const auto C_DS =
    computeTransverseIsotropicDiluteScheme<stress>(IM0,f,IMi,n_a,a,b,c);

const auto C_MT =
    computeOrientedMoriTanakaScheme<stress>(IM0,f,IMi,n_a,a,n_b,b,c);

//Here the distribution is isotropic in the sense that the ellipsoids have
//random orientations with a uniform probability
const auto KG_MT = computeIsotropicMoriTanakaScheme<stress>(IM0,f,IMi,a,b,c);
```

Ponte-Castaneda and Willis scheme (12) is also available:

```
//Here, a `Distribution` object (which defines the distribution of inclusions)
//must be created
Distribution<stress> D = {.n_a = n_a, .a = a, .n_b = n_b, .b = b, .c = c};
const auto C_PCW = computeIsotropicPCWScheme<stress>(IM0,f,IMi,a,b,c,D);
```

Some other details are provided in the documentation.

2.1.5 Homogenization of complex microstructures

In some cases (multi-phasic microstructures) the functions described above are not sufficient to compute homogenized stiffness tensors. Hence, more complex features can be introduced by creation of `ParticulateMicrostructure` objects. It permits to consider an arbitrary number of phases, each phase being a distribution of ellipsoidal inclusions with very general properties.

For example, a construction of such a microstructure can be:

```
const auto IMO=tfel::material::KGModuli<stress>(1e7,1e7);
const auto microstructure=ParticulateMicrostructure(IMO);

Sphere<length> sphere();
Spheroid<length> spheroid(a,b);

const auto KGi=tfel::material::KGModuli<stress>(Ki,Gi);
SphereDistribution<stress> sphere_distribution(sphere,f,KGi);
IsotropicDistribution<stress> spheroid_distribution(spheroid,f,KGi);

microstructure.addInclusionPhase(sphere_distribution);
microstructure.addInclusionPhase(spheroid_distribution);
```

Three classical schemes are implemented for these microstructures, dilute, Mori-Tanaka and self-consistent.

Hence, we can do

```
auto hm_MT=computeMoriTanaka<3u,stress>(microstructure);
//Some features like 'homogenized_stiffness' or
//'mean_strain_localisation_tensors' can be obtained:
auto Chom=hm_MT.homogenized_stiffness;
auto A_i=hm_MT.mean_strain_localisation_tensors;
```

These latter attributes have been implemented in view of non-linear schemes based on linear comparison composites.

2.1.6 Python module `tfel.material.homogenization`

The new functionalities described above are also available via the `Python` package `tfel.material.homogenization`.

2.2 Main improvements to `MFront`

2.2.1 Warnings

Many warnings have been added to detect potential misuses of `MFront` or known bad practices. A typical example of this feature is the following:

```
$ mfront --interface=generic Plasticity.mfront
[warning]: using the default value for the convergence threshold.
This value is generally considered too loose.
You may want to consider a more stringent value (1e-14 is a good choice).
See the `@Epsilon` keyword for details.
```

The user has many ways of controlling how warnings are treated, or how to activate/deactivate them.

2.2.2 Strain rate sensitive isotropic hardening rule in the `StandardElastoViscoplasticity` brick

The `StrainRateSensitive` isotropic hardening rule is defined by:

$$R(p,\dot{p}) = R_0(p)\, R_{rs}(\dot{p}),$$

where:

- $R_0(p)$ is the yield radius corresponding to an infinitely slow loading. It can be built by summing any isotropic hardening rule already implemented in the `StandardElastoViscoPlasticity` brick.
- $R_{rs}(\dot{p})$ is a correction describing the strain rate sensitivity. $R_{rs}(0)$ must be equal to 1.

This isotropic hardening rule can be parametrised using the following options:

- `rate_independent_isotropic_hardening`: this option introduces a contribution to $R_0(p)$. This option can be repeated multiple times.
- `rate_sensitivity_factor`: this option introduces the rate sensivity factor $R_{rs}(\dot{p})$.

In Version `5.1`, two classical rate sensitivity factors have been implemented, namely `CowperSymonds` (13) and `JohnsonCook` (14). The following snippet illustrates this feature:

```
@Brick StandardElastoViscoPlasticity{
  stress_potential : "Hooke" {young_modulus : 210e9, poisson_ratio : 0.3},
  inelastic_flow : "Plastic" {
    criterion : "Mises",
    isotropic_hardening : "StrainRateSensitive" {
      rate_independent_isotropic_hardening : "Voce" {
        R0 : "Q1 * (1 - Q2)", Rinf : "Q1", b : "Q3"
      },
      rate_sensitivity_factor :
          "CowperSymonds" {dp0 : "dp0cs", n : "ncs", Rs_eps : 1e-8}
    }
  }
};
```

2.2.3 Behaviour variables

A behaviour variable is a practical way to call another `MFront` behaviour from a main behaviour. A behaviour variable can be declared as follows:

```
@BehaviourVariable first_phase_plastic_behaviour {
  file: "Plasticity.mfront",
  variables_suffix: "1",
  external_names_prefix: "FirstPhase",
  store_gradients: true,
  store_thermodynamic_forces: true,
  shared_material_properties: {".+"},
  shared_external_state_variables: {".+"}
};
```

The main point is to declare what variables are shared with the main behaviour and the behaviour variable. In the previous example, the material properties are all shared.

Behaviour variables are useful in many contexts:

- It allows switching from a numerical implementation to another, for instance from a fast implementation to a robust one if the fast one diverges.

- It allows easy implementation of various non-linear homogenization schemes. Tutorials are available on the TFEL website for:
  - Implementing the Sachs/Reuss homogenization scheme.
  - Implementing the Taylor/Voigt homogenization scheme.
  - Implementing the β-rule homogenization scheme.

2.2.4 Extension of the `@Model` keyword

In previous versions, the `@Model` keyword allowed calling from a behaviour point-wise models using the historical `Model` DSL.

The `@Model` keyword now allows using point-wise models implemented using the following DSLs: `DefaultModel`, `RungeKuttaModel` and `ImplicitModel`. In this case, a behaviour variable is automatically associated with every point-wise model. This behaviour variable shares all its material properties and external state variables with the calling behaviour. Every persistent variable of the point-wise model is declared as an auxiliary state variable in the calling behaviour, and a local variable is declared, which contains the increment of this variable over the time step.

This feature allows easy coupling of behaviours with phase transition, swelling under irradidation, etc.

2.3 The `MGIS/Function` library

The main new feature of `MGIS` is the `MGIS/Function`[^functions] librar,y which has been introduced to provide standard mechanical post-processings based on the `TFEL/Math` and `TFEL/Material` libraries. `MGIS/Function` allows reusing the large amount of documented and verified functionalities provided by those libraries at the structural scale, including: rotation from/to material frame, conversion between stress measures, Hencky and Green-Lagrange strains, Tresca, von Mises, Hosford, Hill, Barlat equivalent stresses, principal values, Lode's angle, etc.

[^functions] A function denote a set of values associated with the elements (nodes, quadrature points, cells) of a discretized space. Functions are often called `fields` in many solvers.

The following snippet computes the Cauchy stress in the global frame from the first Piola-Kirchhoff stress known in the material frame.

```
const auto evaluator = pk1_function | as_tensor<3>                    |
                       from_pk1_to_cauchy (F_function | as_tensor<3>) |
                       rotate_backwards (R_function | as_matrix<3,3>);
const auto ok = assign(ctx, sig | as_stensor<3>, evaluator);
```

The library is developed in `C++-20` with high-quality requirements and strives to be `constexpr` friendly, memory-safe, thread-safe, exception-safe, and `GPU-friendly`.

The data structures of the functions depend on the targeted solver, as well as the implementation of the `assign`, which depends on the programming model chosen by the targeted solver (`OpenMP`, `CUDA`, `SYCL`, `Kokkos`, etc.).

For standard data structures, such as the one used by `Numpy`'s array, so-called `views` are provided allowing direct usage of the library. In this case, an implementation of the `assign` algorithm is provided by the parallel version of the Standard library. Depending on the compiler and implementation of the standard library, various parallel programming models are used (multithreading and GPU-offloading).

3 A description of some side projects of `TFEL/MFront`

Aside from `MGIS`, several side projects built on top of `TFEL/MFront` have been actively developed.

3.1 MFrontGallery

The MFrontGallery project addresses the management of MFront implementations, including their compilation, unit testing, and deployment (15).

The project has two main, almost orthogonal, objectives:

1. Show how solver developers may provide their users a set of ready-to-use (mechanical) behaviours that can be parametrized to meet specific needs.
2. Describe how to set up a high-quality material knowledge management project based on MFront, capable of meeting the requirements of safety-critical studies.

While the first objective is common to all (mechanical) solvers, the originality of the MFrontGallery project is to address the second goal. In summary, the project provides:

- A CMake infrastructure that can be replicated in (academic or industrial) derived projects, which allows:
  - compiling MFront sources using all supported interfaces.
  - executing unit tests based on MTest which generate XML result files conforming to the JUnit standard, compatible with continuous integration platforms such as jenkins.
  - generating documentation associated with the stored implementations.
- A documentation of best practices for handling material knowledge implemented with MFront, such as use of consistent unit systems, bound-aware physical quantities, consistent tangent operators, and others.
- A set of high-quality MFront implementations.

3.2 Damage indicators for uncoupled analysis of brittle and ductile failures

The FailureCriteria project provides MFront's implementations of various criteria to predict brittle and ductile failure. These implementations can be used in uncoupled analyses as a post-processing, or in coupled analyses where the criteria are, for example, used along with an element-deletion strategy. The following criteria are available:

- for brittle analyses: Beremin et al. (16), Ruggieri and Dodds (17), and Forget, Marini and Vincent (18).
- for ductile failure: Freudenthal (19), Cockcroft and Latham (20), Rice and Tracey (21), Hancock and McKenzie (22), Johnson and Cook (23), Huang (24), and Bai and Wierzbicki (25).

This library can be used directly in solvers interfaced with MGIS. Using MGIS in Python allows using the library when simulations' data are available as NumPy's arrays. As an example, the documentation of the library includes a tutorial showing how to post-process simulation results exported from Cast3M as VTK files.

3.3 The TFELMathEnzyme library

The TFELMathEnzyme library leverages the Enzyme automatic differentiation tool (26, 27) to extend the TFEL/Math library [4].

To illustrate the use of this library, let us consider the Hosford equivalent stress defined as:

$$\sqrt[a]{(\sigma_I-\sigma_{II})^a+(\sigma_I-\sigma_{III})^a+(\sigma_{II}-\sigma_{III})^a}$$

where σ_I, σ_{II} and σ_{III} are the eigenvalues of the stress tensor. Implementing a plastic behaviour using this equivalent stress requires computing its first and second derivatives, the

[4] Enzyme is a plugin of the LLVM infrastructure on top of which is built the clang compiler. Enzyme generates derivatives of functions by automatic differentiation after the optimisations performed by the LLVM infrastructure. This solution is thus specific to this compiler.

implementations of which are complex. Although those computations are already implemented in `TFEL/Material`, it is interesting to see how those computations can be delegated to `Enzyme` through `TFELMathEnzyme`:

```
auto estress = [seps](const StressStensor &s) {
  return computeHosfordStress(s, 100, seps);
};
auto dseq = getDerivativeFunction<Mode::REVERSE, 0>(estress);
auto d2seq = getDerivativeFunction<Mode::REVERSE, 0>(dseq);
```

While early results are encouraging, the `TFELMathEnzyme` library is still very experimental.

Conclusions and future works

Future work aims at enlarging the set of physical features provided by the `TFEL/Math` and `TFEL/Material` libraries and integrating them in `MFront`, in particular through so-called bricks. In particular, a brick dedicated to nonlinear mean-field homogenization is planned.

In parallel, developments regarding GPU's support and automatic differentiation, which are currently experimental in `MFront`, are expected to mature.

Acknowledgements The authors are grateful to the many contributors to the `TFEL/MFront` project. This research was conducted in the framework of the PLEIADES project, which was supported financially by the CEA (Commissariat à l'Énergie Atomique et aux Énergies Alternatives), EDF (Électricité de France) and Framatome. Homogenization developments were conducted within the framework of the AnoHonA ANR project (n° AAPG2023). Work on `MGIS/Function` was performed as part of the EURATOM OperaHPC Project co-funded by the European Union.

References

1. Helfer, Thomas, Wangermez, Maxence, Khellal, Salem, Wang, Yushan, Prat, Raphaël, Lionel, Gélébart and Latu, Guilaume. New functionalities of Versions 4.1 and 4.2 and 4.0 of the TFEL/MFront project and Version 2.0 and 2.2 of the MGIS project. In : *Actes du 16e Colloque National en Calcul des Structures*. Giens, France, 2024.

2. Helfer, Thomas, Michel, Bruno, Proix, Jean-Michel, Salvo, Maxime, Sercombe, Jérôme and Casella, Michel. Introducing the open-source mfront code generator: Application to mechanical behaviours and material knowledge management within the PLEIADES fuel element modelling platform. *Computers & Mathematics with Applications*. September 2015. Vol. 70, no. 5, p. 994–1023.

3. Bernaud, Stéphane, Ramière, Isabelle, Latu, Guillaume and Michel, Bruno. PLEIADES: A numerical framework dedicated to the multiphysics and multiscale nuclear fuel behavior simulation. *Annals of Nuclear Energy*. 2024. Vol. 205, p. 110577.

4. Deloison, Dominique and Congourdeau, Fabrice. Testing and validation of the MFRONT interface for ABAQUS. EDF Lab Sacaly, 2016.

5. Petry, Charles and Helfer, Thomas. Advanced mechanical resolution in CYRANO3 fuel performance code using MFront generation tool. In : *LWR fuel performance meeting/TopFuel/WRFPM*. Zurich, Switzerland, July 2015.

6. Vioujard, N., Bessiron, V., Garnier, Christophe, Georget, V., Mailhé, P., Barbier, B., Deuble, D., Landskron, H., Bellanger, P. and Arimescu, V. I. GALILEO, AREVA's advanced fuel rod performance code and associated realistic methodology. In : *Proceedings of the TOPFUEL 2012 conference*. Manchester UK, 2010.

7. Helfer, Thomas, Bleyer, Jeremy, Frondelius, Tero, Yashchuk, Ivan, Nagel, Thomas and Naumov, Dmitri. The 'MFrontGenericInterfaceSupport' project. *Journal of Open Source Software*. 2020. Vol. 5, no. 48, p. 2003.

8. Jamond, Olivier, Lelong, Nicolas, Brooking, Guillaume, Helfer, Thomas, Prabel, Benoit, Prat, Raphael and Jaccon, Adrien. MANTA: An industrial-strength open-source high performance explicit and implicit multi-physics solver. In : *16ème colloque national en calcul de structures*. Giens, France : CNRS, CSMA, ENS Paris-Saclay, CentraleSupélec, May 2024.

9. Helfer, Thomas, Latu, Guillaume, Prat, Raphaël, Wangermez, Maxence and Cuteri, Francesca. MFEM/MGIS, a HPC mini-application targeting nonlinear thermo-mechanical simulations of nuclear fuels at mesoscale. *Journal of Open Source Software*. 10 April 2025. Vol. 10, no. 108, p. 7719.

10. Bilke, Lars, Flemisch, Bernd, Kalbacher, Thomas, Kolditz, Olaf, Helmig, Rainer and Nagel, Thomas. Development of open-source porous media simulators: Principles and experiences. *Transport in Porous Media*. 1 October 2019. Vol. 130, no. 1, p. 337–361.

11. Parnell, William J. The eshelby, hill, moment and concentration tensors for ellipsoidal inhomogeneities in the newtonian potential problem and linear elastostatics. *Journal of Elasticity*. 8 March 2016. Vol. 125, no. 2, p. 231–294.

12. Castañeda, Pedro P. and Willis, John R. The effect of spatial distribution on the effective behavior of composite materials and cracked media. *Journal of the Mechanics and Physics of Solids*. 29 June 1995. Vol. 43, no. 12, p. 1919–1951.

13. Cowper, G. R. and Symonds, P. S. Strain-hardening and strain-rate effects in the impact loading of cantilever beams. Fort Belvoir, VA : Defense Technical Information Center, 1957.

14. Johnson, Gordon R and Cook, William H. A constitutive model and data for metals subjected to large strains, high strain rates and high temperatures. In : *Proceedings of the 7th international symposium on ballistics*. 1983. p. 541–547.

15. Helfer, Thomas, Wangermez, Maxence, Simo, Eric, Nagel, Thomas, Silbermann, Christian B. and Riparbelli, Lorenzo. The MFrontGallery project. *Journal of Open Source Software*. 23 May 2025. Vol. 10, no. 109, p. 7742.

16. Beremin, F. M., Pineau, Andre, Mudry, Francois, Devaux, Jean-Claude, D'Escatha, Yannick and Ledermann, Patrick. A local criterion for cleavage fracture of a nuclear pressure vessel steel. *Metallurgical Transactions A*. 1 November 1983. Vol. 14, no. 11, p. 2277–2287.

17. Ruggieri, Claudio and Dodds, Robert H. A local approach to cleavage fracture modeling: An overview of progress and challenges for engineering applications. *Engineering Fracture Mechanics*. 1 January 2018. Vol. 187, p. 381–403.

18. Forget, Pierre, Marini, Bernard and Vincent, Ludovic. Application of local approach to fracture of an RPV steel: Effect of the crystal plasticity on the critical carbide size. *Procedia Structural Integrity*. 1 January 2016. Vol. 2, p. 1660–1667.

19. Freudenthal Alfred.m. The inelastic behavior of engineering materials and structures. John Willey; Sons, London, 1950.

20. Cockcroft, M. G. and Latham, D. J. Ductility and the workability of metals. *Journal Institute of Metals*. 1968. Vol. 96, p. 33–39.

21. Rice, J. R. and Tracey, D. M. On the ductile enlargement of voids in triaxial stress fields*. *Journal of the Mechanics and Physics of Solids*. June 1969. Vol. 17, no. 3, p. 201–217.

22. Hancock, J. W. and Mackenzie, A. C. On the mechanisms of ductile failure in high-strength steels subjected to multi-axial stress-states. *Journal of the Mechanics and Physics of Solids*. 1 June 1976. Vol. 24, no. 2, p. 147–160.

23. Johnson, Gordon R. and Cook, William H. Fracture characteristics of three metals subjected to various strains, strain rates, temperatures and pressures. *Engineering Fracture Mechanics*. 1 January 1985. Vol. 21, no. 1, p. 31–48.

24. Huang, Y. Accurate dilatation rates for spherical voids in triaxial stress fields. *Journal of Applied Mechanics*. December 1991. Vol. 58, no. 4, p. 1084–1086.

25. Bai, Yuanli and Wierzbicki, Tomasz. Application of extended mohr–coulomb criterion to ductile fracture. *International Journal of Fracture*. 1 January 2010. Vol. 161, no. 1, p. 1–20.

26. Moses, William S., Churavy, Valentin, Paehler, Ludger, Hückelheim, Jan, Narayanan, Sri Hari Krishna, Schanen, Michel and Doerfert, Johannes. Reverse-mode automatic differentiation and optimization of GPU kernels via enzyme. In : *Proceedings of the international conference for high performance computing, networking, storage and analysis*. New York, NY, USA : Association for Computing Machinery, 13 November 2021. p. 1–16. SC '21. ISBN 978-1-4503-8442-1.

27. Moses, William S., Narayanan, Sri Hari Krishna, Paehler, Ludger, Churavy, Valentin, Schanen, Michel, Hückelheim, Jan, Doerfert, Johannes and Hovland, Paul. Scalable automatic differentiation of multiple parallel paradigms through compiler augmentation. In : *Proceedings of the international conference on high performance computing, networking, storage and analysis*. Dallas, Texas : IEEE Press, 18 November 2022. p. 1–18. SC '22.