TFEL-5.0

MFront Users Meeting

(1) CEA, DES, IRESNE, DEC, SESC, LMPC, Cadarache, France

T. Helfer(1), Maxence Wangermez(1), Antoine Martin(1) (and many others!)

Outline

1. Highlights

A departure...

MFront dev. session

CSMA 2024

- Maxence Wangermez has left CEA for new adventures at Simulia
  - A great loss for the project.. and CEA

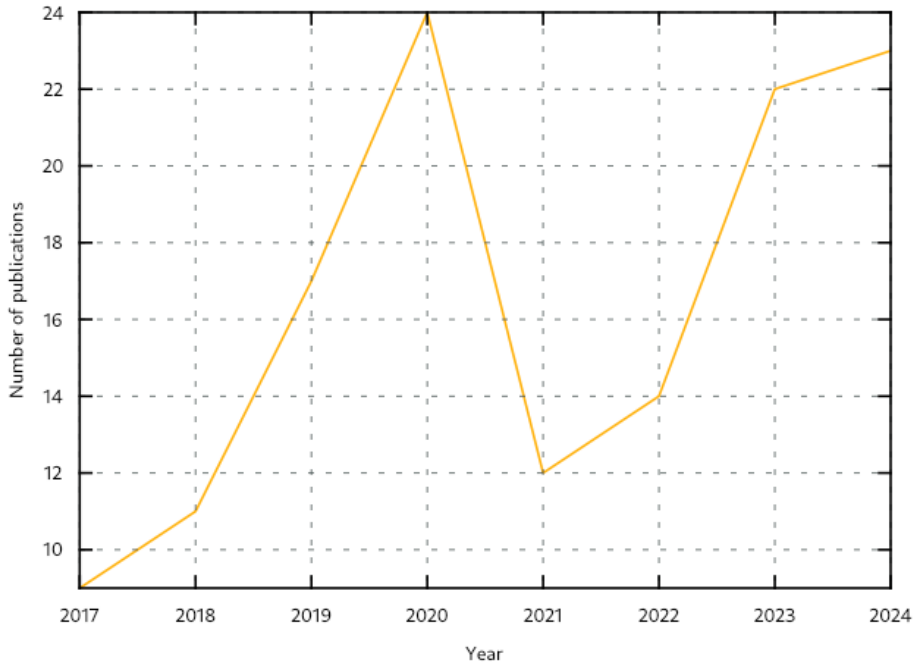
... and an arrival

- Antoine started his post-doctoral position at CEA Cadarache in October in the framework of the Advanced NOnlinear HOmogenization for structural aNAlysis (Anohona) project founded by ANR – France (French National Research Agency (project number AAPG2023).
  - The project gathers France' upmost experts in the mean-field homogenization theories (and also an international expert).
  - Antoine will develop the `TFEL/Material` library for standard tools related to homogenization
  - Antoine will develop efficient implicit algorithms for nonlinear behaviour resulting from mean-field homogenization theories.
  - Antoine will develop an `MFront`'s `brick` dedicated to mean-field homogenization theories.

Trainings

- One session at CEA Saclay in June, hosted by ISAS/DRMP:
  - Participants from CEA, EDF and Onera.
- Three sessions at CEA Cadarache (October, November and December):
  - Two days of introduction to MFront
  - Participants from CEA, IRSN, CNRS, Framatome and various european partners.
- Two trainings already planned in 2025:
  - École Centrale de Nantes.
  - EDF Energy UK, Manchester.

Publications using MFront



- Here the term publication encompasses journal papers, conference papers, PhD manuscripts, public technical reports, etc..
  - https://thelfer.github.io/tfel/web/publications.html
- Since 2021, publications are mostly journal papers, conference papers and PhD manuscripts.
- Two papers under review MFrontGallery and MFEM/MGIS.

Code reckons audit

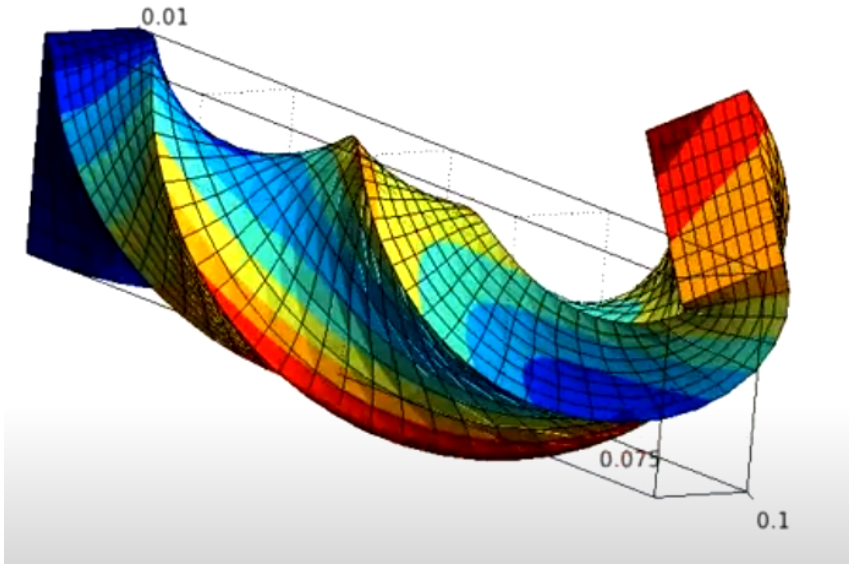
- `TFEL` is currently being audited by `Code Reckons`:
  - https://www.codereckons.com
  - Several training sessions to improve `TFEL` with the `C++` and HPC expert Joel Falcou.
    - Focus on GPU support and the `TFELMathEnzyme` library.
- A general overview led to this preliminary appreciation (https://github.com/thelfer/tfel/issues/474, translated with `Deepl`):

  The overall quality of project management is very high. The effort put into structuring the code, the willingness to follow language evolutions, the quality and quantity of documentation and tests all contribute to this.

  From my point of view as an expert, MFRONT is a software of exemplary quality, compatible with use in an industrial context with high safety requirements.

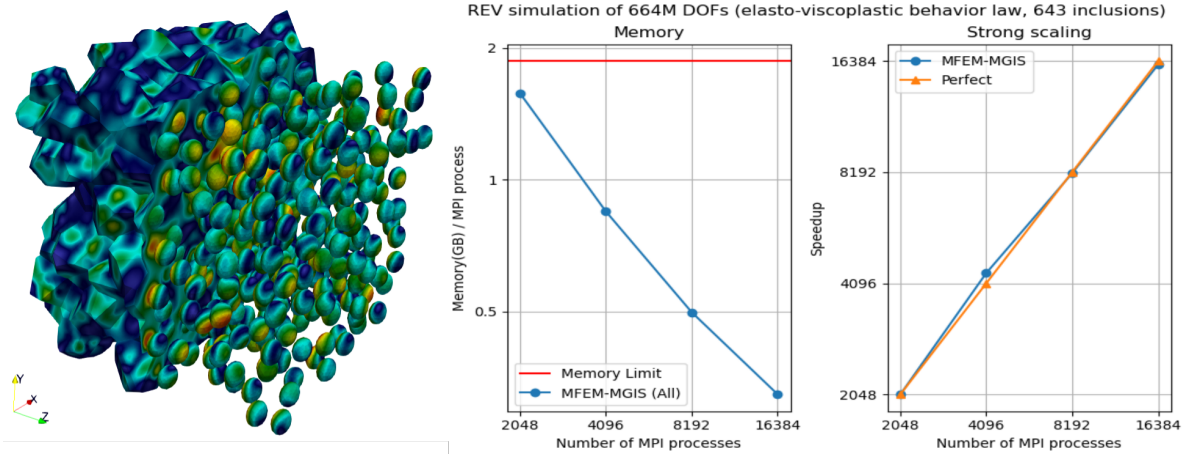
  It is important to allow this project to continue its efforts to provide quality software.

Integration of MGIS in MBDyn



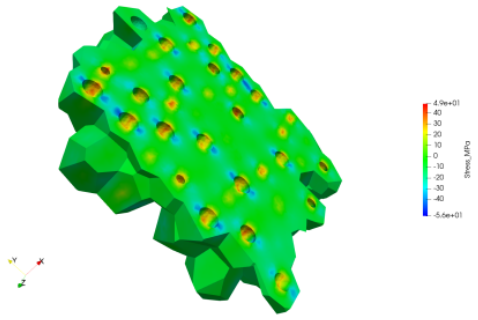
- MBDyn is a multibody analysis code developed by the Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano
  - https://public.gitlab.polimi.it/DAER/mbdyn
- https://www.youtube.com/watch?v=I8HENx5mszA

Viscoplastic simulation of REV of the MOX fuel



- This simulation leverages the `MFEM` platform:
  - https://thelfer.github.io/mfem-mgis
  - https://github.com/thelfer/mfem-mgis
- Nonlinear Representative Elementary Volume of $5.3\,10^9$ dofs have been simulated on the CEA supercomputing center with more than 16 000 computing cores.
  - Good scalability
  - The behaviour integration takes less than 1% to the total computation time.
- A paper of in the Journal Of Open Source Software has been submitted.

Analysis of the fragmentation of nuclear fuels

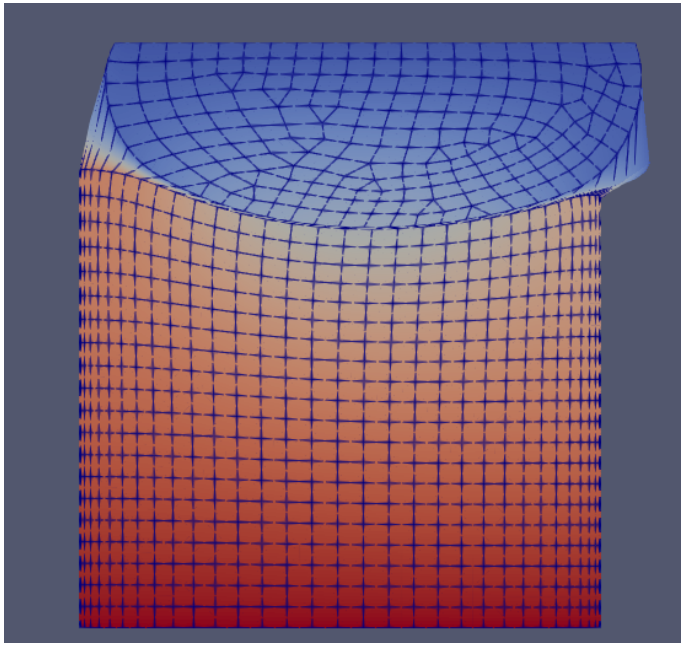


- Evaluation of the fragmentation of fuels due to pressurized bubbles in High Burnup Structure.
- Study made in the framework of the OperaHPC project using `MFEM-MGIS`:
  - Development of open source 3D HPC simulation tools with parallel computing capabilities at the microstructure and fuel element scales.
  - https://www.operahpc.eu/
- Computations made by T. Barani (CEA DEC/SESC/LEVA)
- Microstructures generated by `Mérope`:
  - https://github.com/MarcJos/Merope

code_aster moved to MGIS

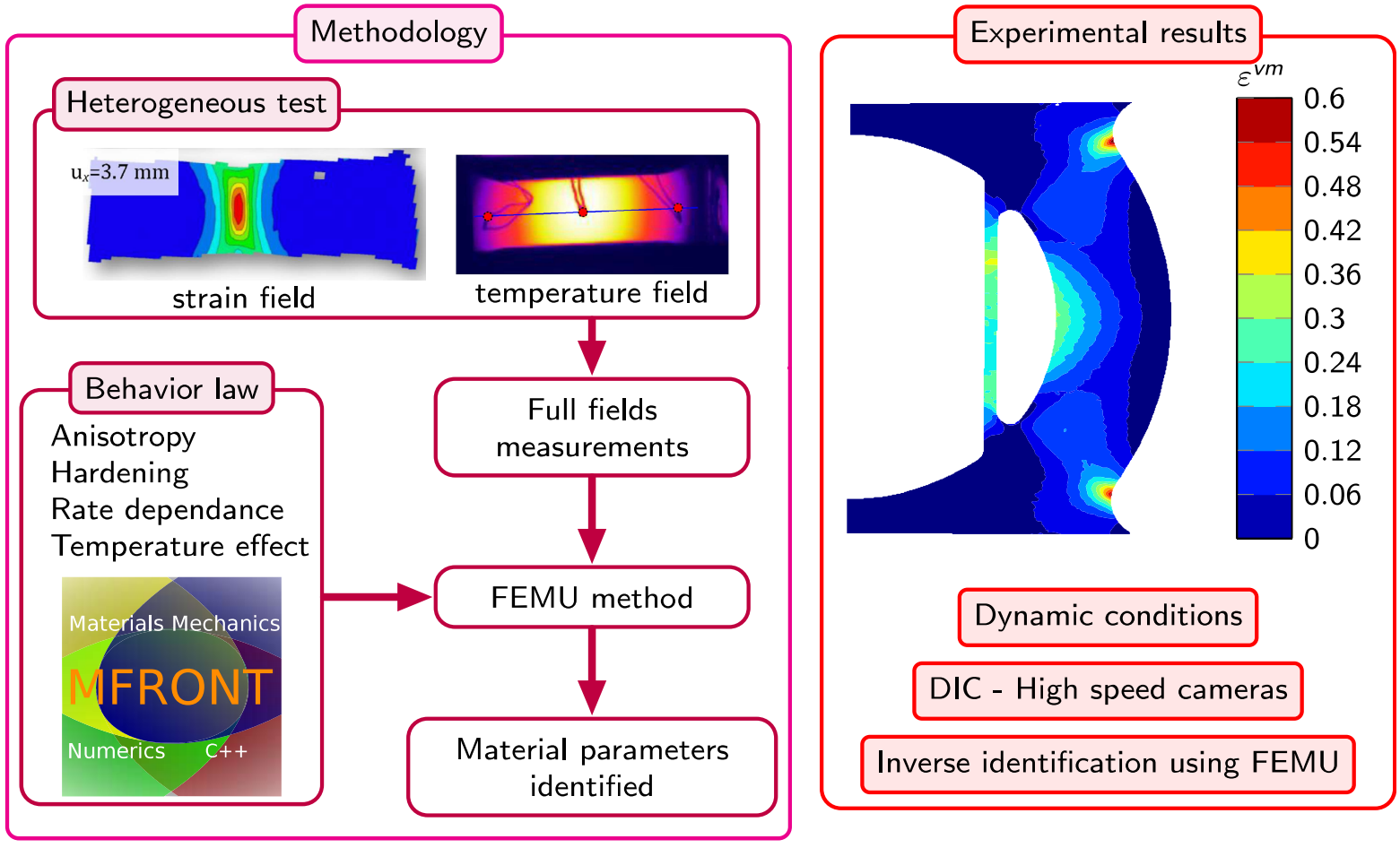
- code_aster finished to integrate MGIS and removed their historical interface.
- Thanks to all the code_aster's team for their confidence and support.

Third medium contact in Manta



- https://en.wikipedia.org/wiki/Third_medium_contact_method
- The medium contact consist in assigning a hyperelastic behaviour to the air with negligible stiffness.
  - Stabilization (HuHu-regularization) is not implemented.

Identification of behaviours for metal forming - I

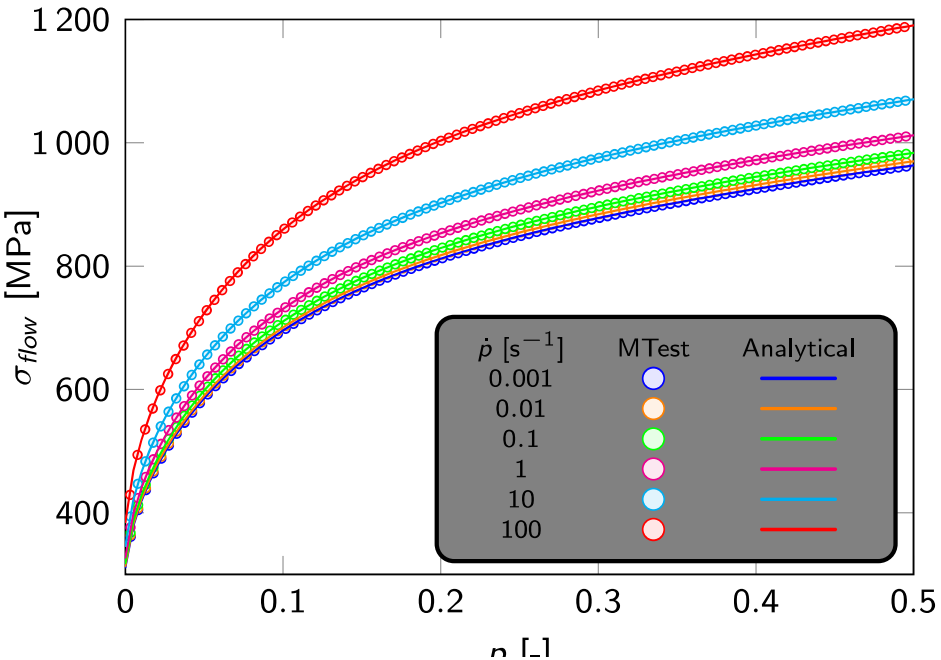


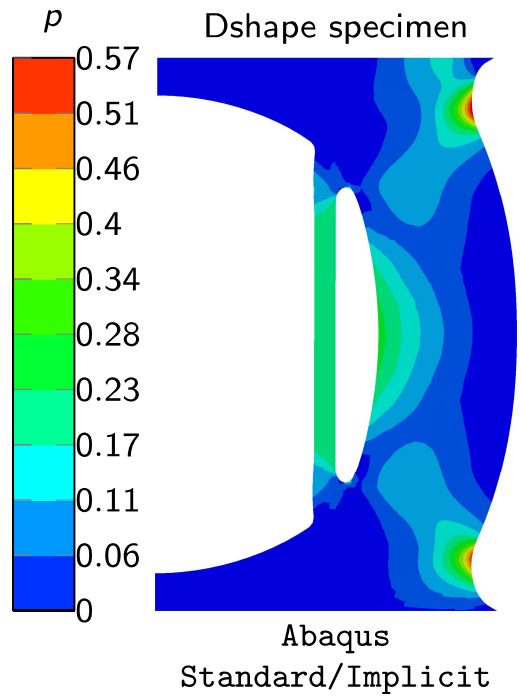
- Courtesy of T. Barret (IRDL)

Identification of behaviours for metal forming - II

$$\sigma_{flow} = \underbrace{\left[\alpha \cdot K\left(\varepsilon_0 + p\right)^n + (1-\alpha)\cdot\left(Q_1\left(1-Q_2\right)\mathrm{e}^{-Q_3 p}\right)\right]}_{\text{Swift-Voce}} \cdot \underbrace{\left[1+\left(\frac{\dot{p}}{D}\right)^{\frac{1}{e}}\right]}_{\text{Cowper-Symonds}}$$

- Implemented with the `@DSL IsotropicPlasticMisesFlow`
- Currently used for inverse identification

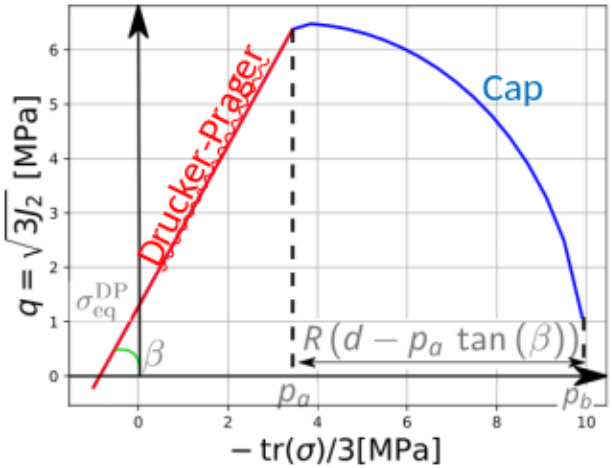




- Courtesy of T. Barret (IRDL)

Powder compaction-I: the Drucker-Prager-Cap model

- The yield surface is composed by two surfaces:
  - $F_1(p, q) = q + \tan(\beta)\, p - d$
  - $F_2(p, q) = \sqrt{(p - p_a)^2 + (R\, q)^2} - R\,(d - p_a \tan\beta)$

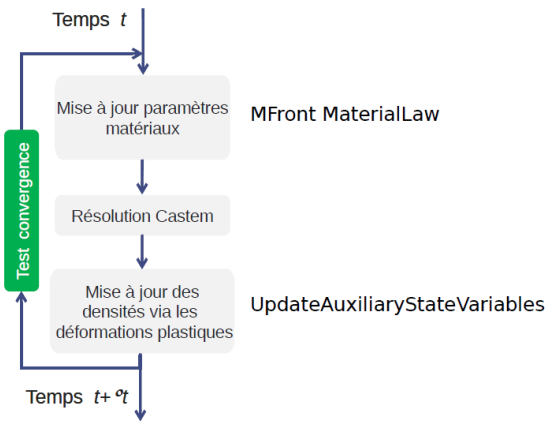


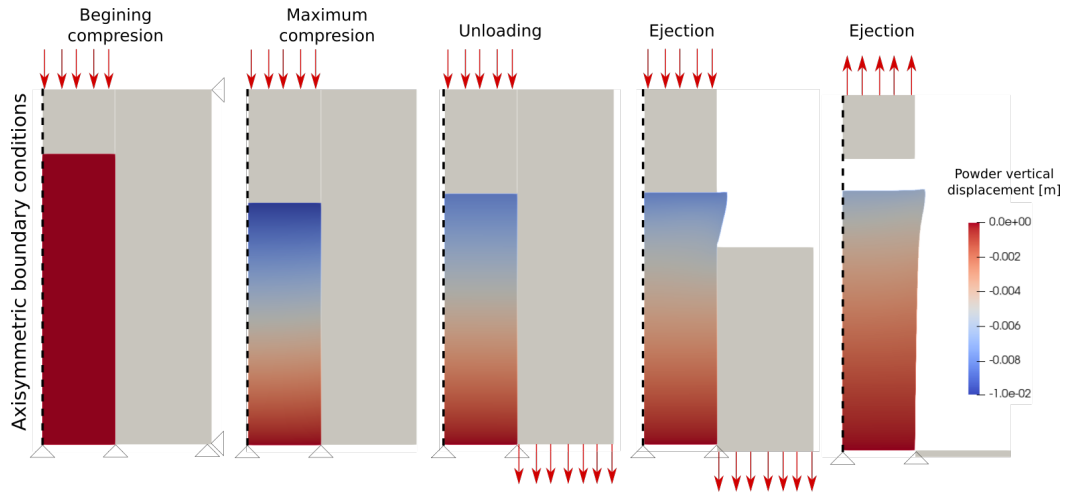
- Evolution of the mass density ρ:

$$\dot{\rho} = -\rho\ \mathrm{tr}\left(\frac{\mathrm{d}\underline{\underline{\varepsilon}}^p}{\mathrm{d}t}\right) \Rightarrow \rho = \rho_0 \exp\left(-\mathrm{tr}\ \left(\underline{\underline{\varepsilon}}^p\right)\right)$$

- The plastic parameters depends on the mass density.
- Finite strain are treated using the Hencky strain measure.
- Implementation described in the `MFront` gallery:
  - https://thelfer.github.io/tfel/web/drucker-prager-cap.html
- Courtesy of A. Socié (DEC/SESC/LDOP), V. Topin (DEC/SESC/LDOP), B. Spanu (DMRC/SPTC/LSEM), J.-. Bayle (DMRC/SPTC/LSEM)

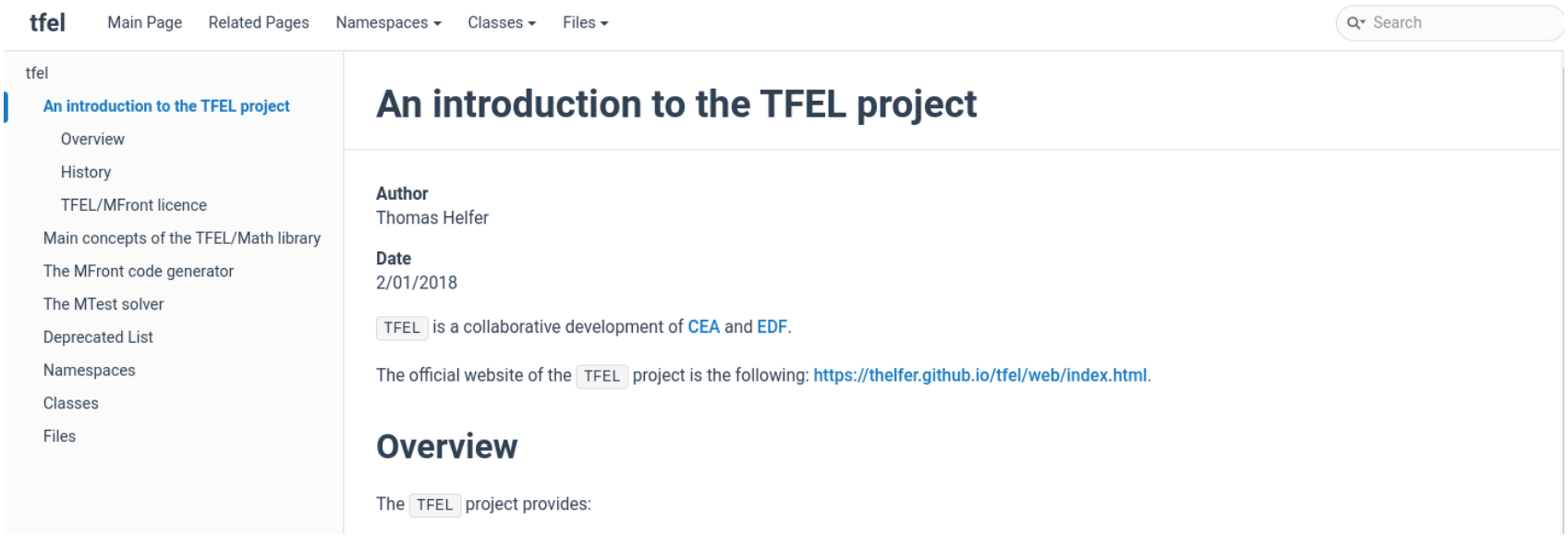
Powder compaction-II: example of simulation





- Simulation made with the `Licos` fuel performance code, based on the `Cast3M` finite element solver.
  - Case of an excessive pressure during the the ejection phase.
- Courtesy of A. Socié (DEC/SESC/LDOP), V. Topin (DEC/SESC/LDOP), B. Spanu (DMRC/SPTC/LSEM), J.-. Bayle (DMRC/SPTC/LSEM)

Update of the doxygen documentation



- https://thelfer.github.io/tfel/doxygen/index.html
- Automatically updated using Github Action.

Packages

- Up-to-date packages exists for `brew`, `spack`, `conda`, `mingw`:
  - `brew install tfel` (`Linux` and `Mac Os`)
  - `spack install tfel` (tested on `Linux`)
  - `conda install conda-forge::mfront` (`Linux` and `Windows`)
  - `pacman -S mingw-w64-x86_64-tfel` (`Windows`)
- `TFEL` ships with `Manta`, `Cast3M` and `code_aster`/`salome_meca`.

Outline

2. Evolutions and of new features in TFEL libraries

Port to C++-20

- Each major version of C++-20 allows a partial rewrite of the TFEL libraries which aims at:
  - avoiding dead code.
  - simplifying and easing maintenance the implementation by relying on standard features.
  - extending support for new compilers and devices (GPUs).
    - Visual Studio is supported again!
- The main feature of C++-20 is the introduction of concepts.
  - Concepts were introduced in TFEL in early versions using various implementation tricks (CRTP, SFINAE, requires clauses in TFEL-4.x)

Evolution of the implementation of `trace`

- `rliv-2.0` (C++-98, unmaintained):

```
template <class T>
typename tfel::meta::EnableIf<tfel::meta::Implements<T, StensorConcept>::cond,
                              typename StensorTraits<T>::NumType>::type
trace(const T &s) {
  return s(0) + s(1) + s(2);
}
```

- `rliv-3.4` (C++-11) :

```
template <class T>
typename std::enable_if<tfel::meta::Implements<T, StensorConcept>::cond,
                        StensorNumType<T>>::type
trace(const T &s) {
  return s(0) + s(1) + s(2);
}
```

- `rliv-4.2` (C++-17):

```
template <typename StensorType>
constexpr std::enable_if_t<implementsStensorConcept<StensorType>(),
                           numeric_type<StensorType>>
trace(const StensorType &s) {
  return s(0) + s(1) + s(2);
}
```

- `TFEL-5.0` (C++-20):

```
TFEL_HOST_DEVICE constexpr auto trace(const StensorConcept auto &s) noexcept {
  return s(0) + s(1) + s(2);
}
```

A new eigensolver

Algorithm	Failure ratio	Δ_∞	Times (ns)	Time ratio
TFELEIGENSOLVER	0.00058	6.94e-14	137752068	1
GTESYMMETRICQREIGENSOLVER	1e-06	2.30e-15	315593552	2.29
FSESJACOBIEIGENSOLVER	0	9.08e-16	256285090	1.86
FSESQLEIGENSOLVER	0.000202	3.04e-15	214537012	1.56
FSESCUPPENEIGENSOLVER	0.019251	5.58e-15	219113965	1.59
FSESHYBRIDEIGENSOLVER	0.081586	1.29e-10	81861668	0.59
FSESANALYTICALEIGENSOLVER	0.103935	4.11e-10	79701256	0.58
HARARIEIGENSOLVER	0.000037	2.27e-14	116977683	0.85

- The computation of eigen values is done with Harari's algorithm and the computation of eigen vectors is done with the default eigen solver for symmetric tensors of TFEL
  - This algorithm is more efficient and more accurate than the default TFEL algorithm

```
const auto [vp, m] = s.computeEigenVectors<stensor::HARARIEIGENSOLVER>();
```

Computation of eigenvalues of a real, symmetric 3 x 3 matrix with particular reference to the pernicious case of two nearly equal eigenvalues. Harari, Isaac and Albocher, Uri. International Journal for Numerical Methods in Engineering

Tools for mean-field homogeneization theories

- Functions related to Eshelby tensor and localisation tensors:
  - `computeEshelbyTensor`, `computeAnisotropicEshelbyTensor`
  - `computeEllipsoidStrainLocalisationTensor`, `computeEllipsoidStressConcentrationTensor`
- Linear Homogeneization schemes:
  - Voigt/Reuss and Hashin/Shtrikman bounds
  - Dilute scheme, Mori-Tanaka scheme
  - `computeDiluteScheme`, `computeMoriTanakaScheme`, `computeSelfConsistentScheme`, `computeIsotropicDiluteScheme`, `computeTransverseIsotropicDiluteScheme`, `computeOrientedDiluteScheme`, et

Outline

3. New MFront's features

Material properties based on data

```
@DSL MaterialProperty;
@Law LinearDataInterpolation;

@UseQt true;
@UnitSystem SI;

@Output stress E;
E.setGlossaryName("YoungModulus");

@StateVariable temperature T;
T.setGlossaryName("Temperature");

@Data {
  values: { 293.15 : 240e9, 693.15 : 180e9, 893.15 : 170e9 },
   interpolation  :  " linear "
};
```

- The data can be interpolated using a piece-wise linear regression or using cubic splines.
- The data can be extrapolated using a linear extrapolation or using the closest value.

Rate sensitive isotropic hardening rules

```
@Brick StandardElastoViscoPlasticity{
  stress_potential  : "Hooke" {young_modulus : 210e9, poisson_ratio : 0.3},
  inelastic_flow  : "Plastic" {
    criterion  : "Mises",
    isotropic_hardening : "StrainRateSensitive" {
      rate_independent_isotropic_hardening : "Linear" {R0 : 150e6, H : 2e9},
      rate_sensitivity_factor  : "CowperSymonds" {D : 9013, E : 0.319}
    }
  }
};
```

- The `StrainRateSensitive` isotropic hardening rule describes the following yield radius:

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

  $R_0\left(p\right)$ is the yield radius corresponding to an infinitly slow loading. It can be built by summing any isotropic hardening rule already implemented in the `StandardElastoViscoPlasticity` brick. $R_{rs}\left(\dot{p}\right)$ is a correction describing the rate sensitiviy.

- The Cowper-Symonds and the Johnson-Cook corrections are available.
- Thanks to T. Barret (IRDL) for his valuable contribution.

New interfaces under development

- `TrioCFD` :

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`
  - See V. Faucher's talk

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`
  - See V. Faucher's talk
- `OpenFOAM` :

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`
  - See V. Faucher's talk
- `OpenFOAM` :
  - Based on `MGIS`

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`
  - See V. Faucher's talk
- `OpenFOAM` :
  - Based on `MGIS`
  - The main target is the `OFFBEAT` fuel performance code

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`
  - See V. Faucher's talk
- `OpenFOAM` :
  - Based on `MGIS`
  - The main target is the `OFFBEAT` fuel performance code
- `Plaxis`

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`
  - See V. Faucher's talk
- `OpenFOAM` :
  - Based on `MGIS`
  - The main target is the `OFFBEAT` fuel performance code
- `Plaxis`
  - https://github.com/thelfer/plaxis-udm

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`
  - See V. Faucher's talk
- `OpenFOAM` :
  - Based on `MGIS`
  - The main target is the `OFFBEAT` fuel performance code
- `Plaxis`
  - https://github.com/thelfer/plaxis-udm
  - Developed in collaboration with Miguel Angel Manica Malcom

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`
  - See V. Faucher's talk
- `OpenFOAM` :
  - Based on `MGIS`
  - The main target is the `OFFBEAT` fuel performance code
- `Plaxis`
  - https://github.com/thelfer/plaxis-udm
  - Developed in collaboration with Miguel Angel Manica Malcom
  - Based on `MGIS`

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`
  - See V. Faucher's talk
- `OpenFOAM` :
  - Based on `MGIS`
  - The main target is the `OFFBEAT` fuel performance code
- `Plaxis`
  - https://github.com/thelfer/plaxis-udm
  - Developed in collaboration with Miguel Angel Manica Malcom
  - Based on `MGIS`
- `SYSTUS`

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`
  - See V. Faucher's talk
- `OpenFOAM` :
  - Based on `MGIS`
  - The main target is the `OFFBEAT` fuel performance code
- `Plaxis`
  - https://github.com/thelfer/plaxis-udm
  - Developed in collaboration with Miguel Angel Manica Malcom
  - Based on `MGIS`
- `SYSTUS`
  - Developped with `Framatome`'s team

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`
  - See V. Faucher's talk
- `OpenFOAM` :
  - Based on `MGIS`
  - The main target is the `OFFBEAT` fuel performance code
- `Plaxis`
  - https://github.com/thelfer/plaxis-udm
  - Developed in collaboration with Miguel Angel Manica Malcom
  - Based on `MGIS`
- `SYSTUS`
  - Developped with `Framatome`'s team
  - Based on `MGIS`

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`
  - See V. Faucher's talk
- `OpenFOAM` :
  - Based on `MGIS`
  - The main target is the `OFFBEAT` fuel performance code
- `Plaxis`
  - https://github.com/thelfer/plaxis-udm
  - Developed in collaboration with Miguel Angel Manica Malcom
  - Based on `MGIS`
- `SYSTUS`
  - Developped with `Framatome`'s team
  - Based on `MGIS`
  - Hopefully compatible with `SYSWELD`

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`
  - See V. Faucher's talk
- `OpenFOAM` :
  - Based on `MGIS`
  - The main target is the `OFFBEAT` fuel performance code
- `Plaxis`
  - https://github.com/thelfer/plaxis-udm
  - Developed in collaboration with Miguel Angel Manica Malcom
  - Based on `MGIS`
- `SYSTUS`
  - Developped with `Framatome`'s team
  - Based on `MGIS`
  - Hopefully compatible with `SYSWELD`
- `Open-Radioss` and `Radioss`

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`
  - See V. Faucher's talk
- `OpenFOAM` :
  - Based on `MGIS`
  - The main target is the `OFFBEAT` fuel performance code
- `Plaxis`
  - https://github.com/thelfer/plaxis-udm
  - Developed in collaboration with Miguel Angel Manica Malcom
  - Based on `MGIS`
- `SYSTUS`
  - Developped with `Framatome`'s team
  - Based on `MGIS`
  - Hopefully compatible with `SYSWELD`
- `Open-Radioss` and `Radioss`
  - A dedicated `MFront` interface will be developed.

New interfaces under development

- `TrioCFD` :
  - Based on `MGIS`
  - See V. Faucher's talk
- `OpenFOAM` :
  - Based on `MGIS`
  - The main target is the `OFFBEAT` fuel performance code
- `Plaxis`
  - https://github.com/thelfer/plaxis-udm
  - Developed in collaboration with Miguel Angel Manica Malcom
  - Based on `MGIS`
- `SYSTUS`
  - Developped with `Framatome`'s team
  - Based on `MGIS`
  - Hopefully compatible with `SYSWELD`
- `Open-Radioss` and `Radioss`
  - A dedicated `MFront` interface will be developed.
  - Planned for early 2025

Outline

4. The MFrontJIT project

Overview

```
mfront:: jit ::compile(
      " Plasticity .mfront",
      {.dsl_options = {{"parameters_as_static_variables", true},
                       {"disable_runtime_checks", true},
                       {"out_of_bounds_policy_runtime_modification", false},
                       {"overriding_parameters",
                        tfel :: utilities ::DataMap{{"Temperature", 293.15}}}},
       .ecmds = {"@SelectedModellingHypothesis PlaneStrain"}});
```

- The `MFrontJIT` project provides a way to compile `MFront` files with specific options and knowledge of the case to be treated.

Overview

```
mfront:: jit ::compile(
    " Plasticity .mfront",
    {.dsl_options = {{"parameters_as_static_variables", true},
                     {"disable_runtime_checks", true},
                     {"out_of_bounds_policy_runtime_modification", false},
                     {"overriding_parameters",
                      tfel :: utilities ::DataMap{{"Temperature", 293.15}}}},
     .ecmds = {"@SelectedModellingHypothesis PlaneStrain"}});
```

- The `MFrontJIT` project provides a way to compile `MFront` files with specific options and knowledge of the case to be treated.
  - Constant and uniform material properties and external state variables can be treated as static variables, considerably simplying the memory access pattern and reducing data transfer.

Overview

```
mfront:: jit ::compile(
    " Plasticity .mfront",
    {.dsl_options = {{"parameters_as_static_variables", true},
                     {"disable_runtime_checks", true},
                     {"out_of_bounds_policy_runtime_modification", false},
                     {"overriding_parameters",
                      tfel :: utilities ::DataMap{{"Temperature", 293.15}}}},
     .ecmds = {"@SelectedModellingHypothesis PlaneStrain"}});
```

- The `MFrontJIT` project provides a way to compile `MFront` files with specific options and knowledge of the case to be treated.
  - Constant and uniform material properties and external state variables can be treated as static variables, considerably simplying the memory access pattern and reducing data transfer.
    - Required for GPU programming

Overview

```
mfront:: jit ::compile(
      " Plasticity .mfront",
      {.dsl_options = {{"parameters_as_static_variables", true},
                       {"disable_runtime_checks", true},
                       {"out_of_bounds_policy_runtime_modification", false},
                       {"overriding_parameters",
                        tfel :: utilities ::DataMap{{"Temperature", 293.15}}}},
       .ecmds = {"@SelectedModellingHypothesis PlaneStrain"}});
```

- The `MFrontJIT` project provides a way to compile `MFront` files with specific options and knowledge of the case to be treated.
  - Constant and uniform material properties and external state variables can be treated as static variables, considerably simplying the memory access pattern and reducing data transfer.
    - Required for GPU programming
  - This paves the way to further improvements and optimisations.

Overview

```
mfront:: jit ::compile(
    " Plasticity .mfront",
    {.dsl_options = {{"parameters_as_static_variables", true},
                     {"disable_runtime_checks", true},
                     {"out_of_bounds_policy_runtime_modification", false},
                     {"overriding_parameters",
                      tfel :: utilities ::DataMap{{"Temperature", 293.15}}}},
     .ecmds = {"@SelectedModellingHypothesis PlaneStrain"}});
```

- The `MFrontJIT` project provides a way to compile `MFront` files with specific options and knowledge of the case to be treated.
  - Constant and uniform material properties and external state variables can be treated as static variables, considerably simplying the memory access pattern and reducing data transfer.
    - Required for GPU programming
  - This paves the way to further improvements and optimisations.
    - If the elastic material properties are known, the elastic strain can be removed from the state variables of isotropic DSLs.

Overview

```
mfront:: jit ::compile(
      " Plasticity .mfront",
      {.dsl_options = {{"parameters_as_static_variables", true},
                       {"disable_runtime_checks", true},
                       {"out_of_bounds_policy_runtime_modification", false},
                       {"overriding_parameters",
                        tfel :: utilities ::DataMap{{"Temperature", 293.15}}}},
       .ecmds = {"@SelectedModellingHypothesis PlaneStrain"}});
```

- The `MFrontJIT` project provides a way to compile `MFront` files with specific options and knowledge of the case to be treated.
  - Constant and uniform material properties and external state variables can be treated as static variables, considerably simplying the memory access pattern and reducing data transfer.
    - Required for GPU programming
  - This paves the way to further improvements and optimisations.
    - If the elastic material properties are known, the elastic strain can be removed from the state variables of isotropic DSLs.
    - The number of state variables would decrease from 7 to 1 in $3D$. The memory transfer associated with state variables would be divided by 7.

Overview

```
mfront:: jit ::compile(
    " Plasticity .mfront",
    {.dsl_options = {{"parameters_as_static_variables", true},
                     {"disable_runtime_checks", true},
                     {"out_of_bounds_policy_runtime_modification", false},
                     {"overriding_parameters",
                      tfel :: utilities ::DataMap{{"Temperature", 293.15}}}},
     .ecmds = {"@SelectedModellingHypothesis PlaneStrain"}});
```

- The `MFrontJIT` project provides a way to compile `MFront` files with specific options and knowledge of the case to be treated.
  - Constant and uniform material properties and external state variables can be treated as static variables, considerably simplying the memory access pattern and reducing data transfer.
    - Required for GPU programming
  - This paves the way to further improvements and optimisations.
    - If the elastic material properties are known, the elastic strain can be removed from the state variables of isotropic DSLs.
    - The number of state variables would decrease from 7 to 1 in $3D$. The memory transfer associated with state variables would be divided by 7.
- https://github.com/thelfer/mfront-jit

Outline

5. Conclusions and perspectives

Conclusions

- The development of Version 5.0 was driven by internal evolutions:
  - Port to C++-20.
  - Internal toward GPU support.
  - Improvements and bug fixes (24 issues fixed).
- New and experimental developments are done in external libraries:
  - `TFELMathEnzyme`
  - `MFrontJIT`
- The release notes are available:
  - https://thelfer.github.io/tfel/web/release-notes-5.0.html

Perspectives

- Thanks the ANOHONA projects, support for mean field homogeneizations will considerably increase.
- Further work on GPUs will be tacked by R. Prat and F. Cuteri (DEC/SESC/LDOP):
  - Internship: Accélération de l'intégration de lois de comportement matériau sur des supercalculateurs GPU exaflopiques.
  - Phd: Implémentation d'algorithmes parallèles sur GPU pour les simulations du combustible nucléaire sur supercalculateurs exaflopiques
- A Phd is proposed to build UFL-like features for `Manta` (CEA) and `A-Set` (Mines de Paris, Onera, Safran-TECH):
  - Tightly integrated with `MFront`.
  - Focused on nonlinear resolutions.