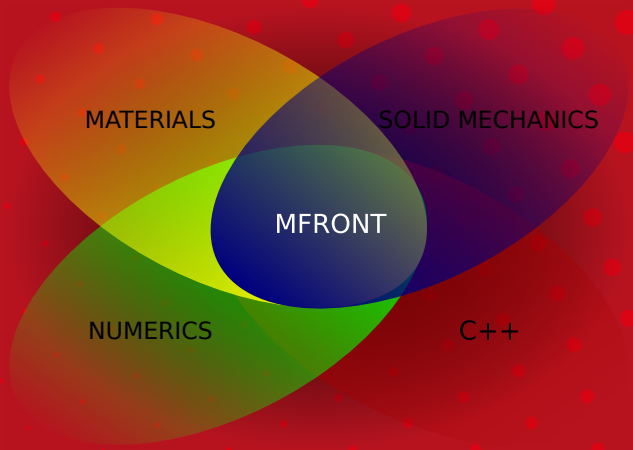
An overview of the MFront code generator



DE LA RECHERCHE À L'INDUSTRIE

OpenGeoSys User Days

12/03/2020

T. Helfer, T. Nagel, J. Bleyer and (so) many others

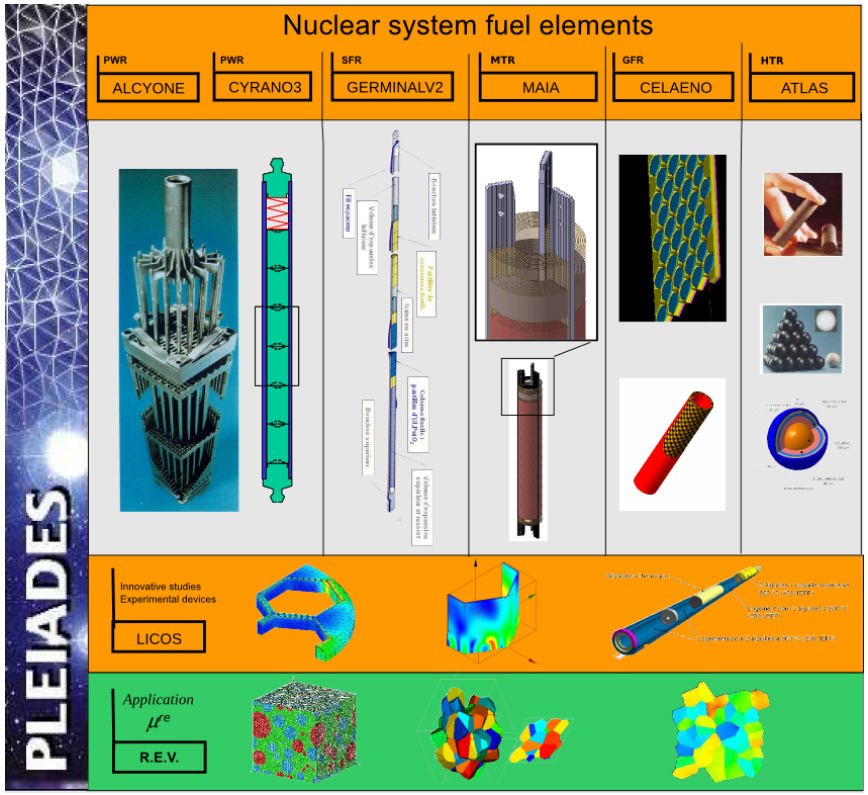
Outline

- **Overview of the `MFront` code generator**
- **Generic behaviours**
- **MFront and OpenGeoSys : Connecting two open-source initiatives**
- **`MFront` related projects of interest for the `OpenGeoSys` community**
- **Conclusions**

Material knowledge management in nuclear simulations

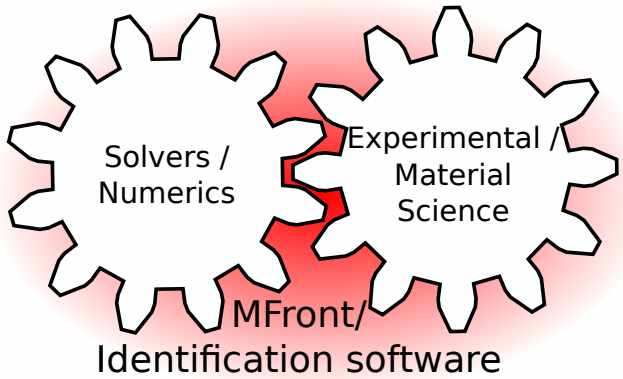
Commissariat à l'énergie atomique et aux énergies alternatives - www.cea.fr

The Pleiades platform



- A wide range of materials (ceramics, metals, composites).
- A wide range of mechanical phenomena and behaviours.
  - Creep, swelling, irradiation effects, phase transitions, etc..
- A wide range of mechanical loadings.

Material knowledge management

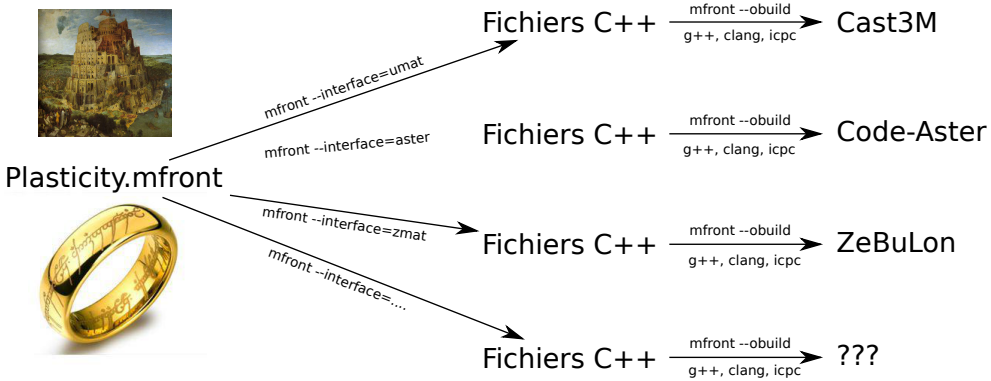


- The need to guarantee the quality of engineering studies has never been so high and is constantly growing.
- Every part of a study must be covered by strict AQ procedures :
  - The finite element solver on the one hand (see the `code_aster` documentation and unit tests).
  - The material knowledge (material properties, **mechanical behaviours**) and experimental data on the other hand.
- **One must guarantee a complete consistency from experimental data to engineering studies**
- See also the `MFrontGallery` project

Overview of the `MFront` code generator

Commissariat à l'énergie atomique et aux énergies alternatives - www.cea.fr

MFront goals



- `MFront` is a code generation tool dedicated to material knowledge (material properties, mechanical behaviours, point-wise models) :
  - Support for small and finite strain behaviours, cohesive zone models, **generalised behaviours** (non local and or multiphysics).
- Main goals :
  - Numerical efficiency (see various benchmarks on the website).
  - Portability (`Cast3M`, `Cyrano`, `code_aster`, `Europlexus`, `TMFTT`, `AMITEX_FFTP`, `Abaqus`, `CalculiX`, `MTest`).
  - **Ease of use** : *Longum iter est per praecepta, breve et efficax per exempla* (It's a long way by the rules, but short and efficient with examples).

An very simple example

```
@DSL Implicit;
@Behaviour Norton;
@Brick StandardElasticity;

@MaterialProperty stress E;
E.setGlossaryName("YoungModulus");
@MaterialProperty real ν, A, nn;
ν.setGlossaryName("PoissonRatio");
A.setEntryName("NortonCoefficient");
nn.setEntryName("NortonExponent");

@StateVariable real    p;
p.setGlossaryName("EquivalentViscoplasticStrain");

@Integrator{
  constexpr const auto Me = Stensor4::M();
  const auto μ = computeMu(E, ν);
  const auto σe = sigmaeq(σ);
  const auto iσe = 1 / (max(σe, real(1.e-12) · E));
  const auto vp = A · pow(σe, nn);
  const auto ∂vp/∂σe = nn · vp · iσe;
  const auto n = 3 · deviator(σ) · (iσe / 2);
  // Implicit system
  fεel += Δp · n;
  fp -= vp · Δt;
  // jacobian
  ∂fεel/∂Δεel += 2 · μ · θ · dp · iσe · (Me - (n ⊗ n));
  ∂fεel/∂Δp = n;
  ∂fp/∂Δεel = -2 · μ · θ · ∂vp/∂σe · Δt · n;
} // end of @Integrator
```

- Implicit integration.
- Implicit system :

$$\begin{cases} f_{\underline{\epsilon}^{el}} = \Delta\,\underline{\epsilon}^{el} - \Delta\,\underline{\epsilon}^{to} + \Delta\,p\,\underline{\mathbf{n}} \\ f_p = \Delta\,p - A\,\sigma_{eq}^{n} \end{cases}$$

- Jacobian :

$$\begin{cases} \dfrac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta\,\underline{\epsilon}^{el}} = \underline{\underline{\mathbf{I}}} + \dfrac{2\,\mu\,\theta\,\Delta\,p}{\sigma_{eq}}\left(\underline{\underline{\mathbf{M}}} - \underline{\mathbf{n}} \otimes \underline{\mathbf{n}}\right) \\ \dfrac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta\,p} = \underline{\mathbf{n}} \\ \dfrac{\partial f_p}{\partial \Delta\,\underline{\epsilon}^{el}} = -2\,\mu\,\theta\,A\,n\,\sigma_{eq}^{n-1}\,\Delta\,t\,\underline{\mathbf{n}} \end{cases}$$

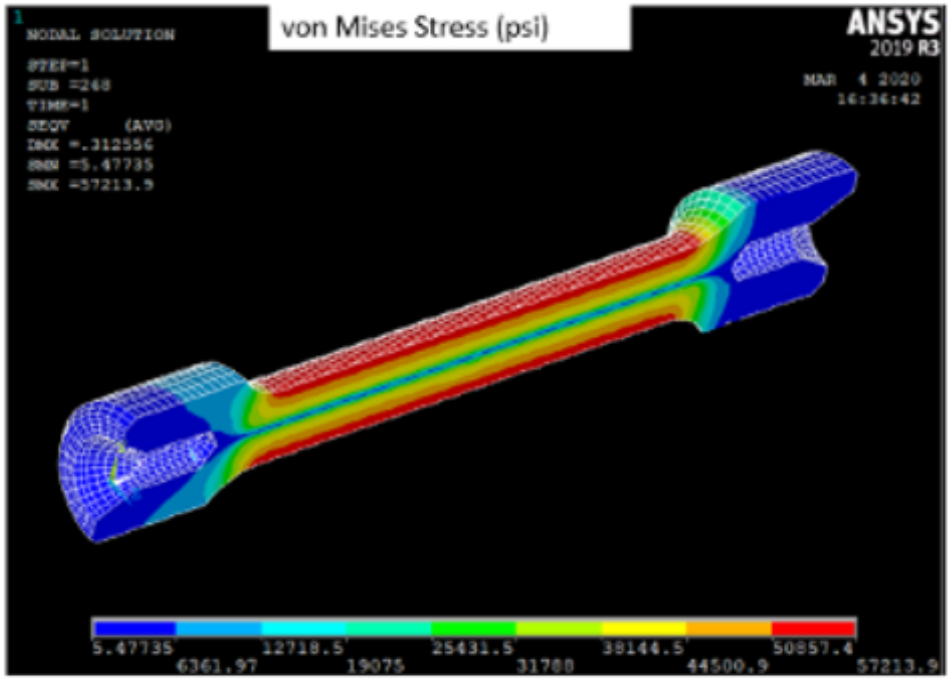
- **All programming and numerical details are hidden (by default).**

An example of the StandardElasticity-VicoPlasticity brick

```
@DSL Implicit;
@Behaviour MohrCoulomAbboSloan3;

@Epsilon 1.e-14;
@Theta 1;

@Brick StandardElastoViscoPlasticity{
  stress_potential : "Hooke" {
    young_modulus : 150.e3,
    poisson_ratio : 0.3
  },
  inelastic_flow : "Plastic" {
    criterion : "MohrCoulomb" {
      c : 3.e1,        // cohesion
      phi : 0.523598775598299,   // friction angle or dilatancy angle
      lodeT : 0.506145483078356, // transition angle as defined by Abbo and Sloan
      a : 1e1          // tension cuff-off parameter
    },
    flow_criterion : "MohrCoulomb" {
      c : 3.e1,        // cohesion
      phi : 0.174532925199433,   // friction angle or dilatancy angle
      lodeT : 0.506145483078356, // transition angle as defined by Abbo and Sloan
      a : 3e1          // tension cuff-off parameter
    },
    isotropic_hardening : "Linear" {R0 : "0"}
  }
};
```

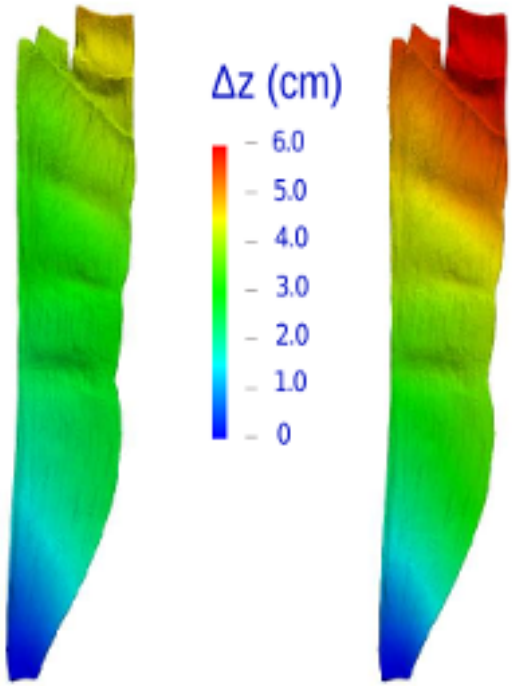
- The `StandardElasticityVicoPlasticity` brick allows a declarative syntax using predefined components.

Some examples



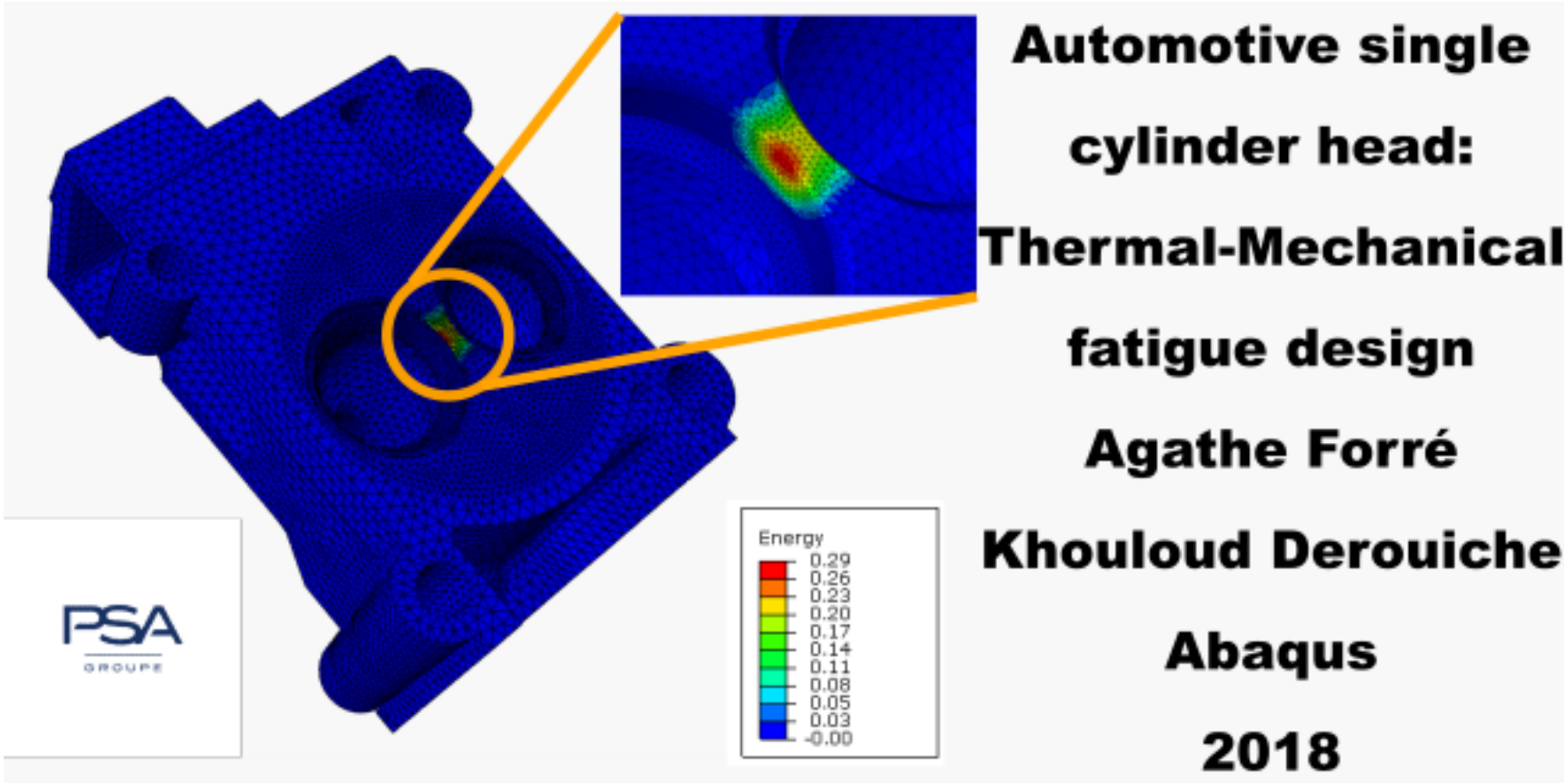
Torsional twist of a notched bar using an Hosford plastic behaviour with a bilinear hardening law
Alex Grishin
Ansys MAPDL 2020

Some examples

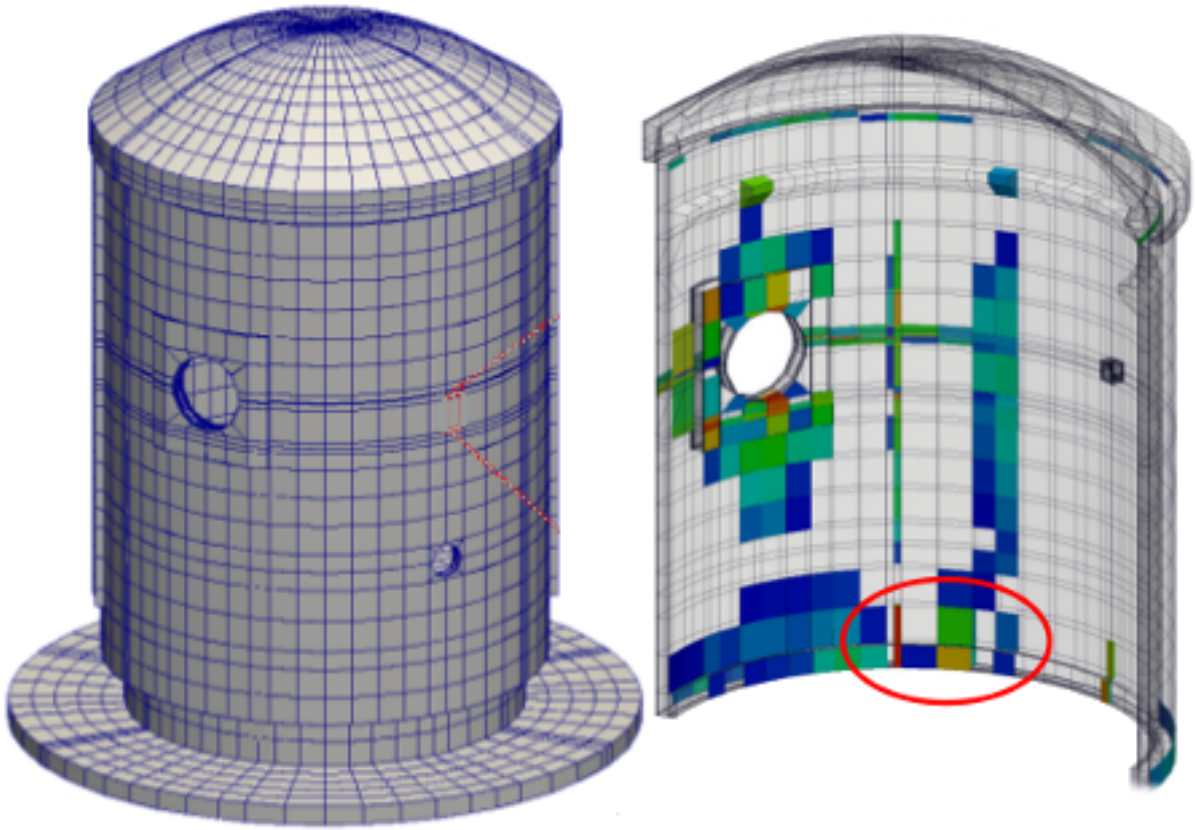


Modelling of abdominal muscles
Lluís Tuset, Gerard Fortuny, Joan Herrero, Dolors Puigjaner, Josep M. López
Code_Aster
2019

Some examples

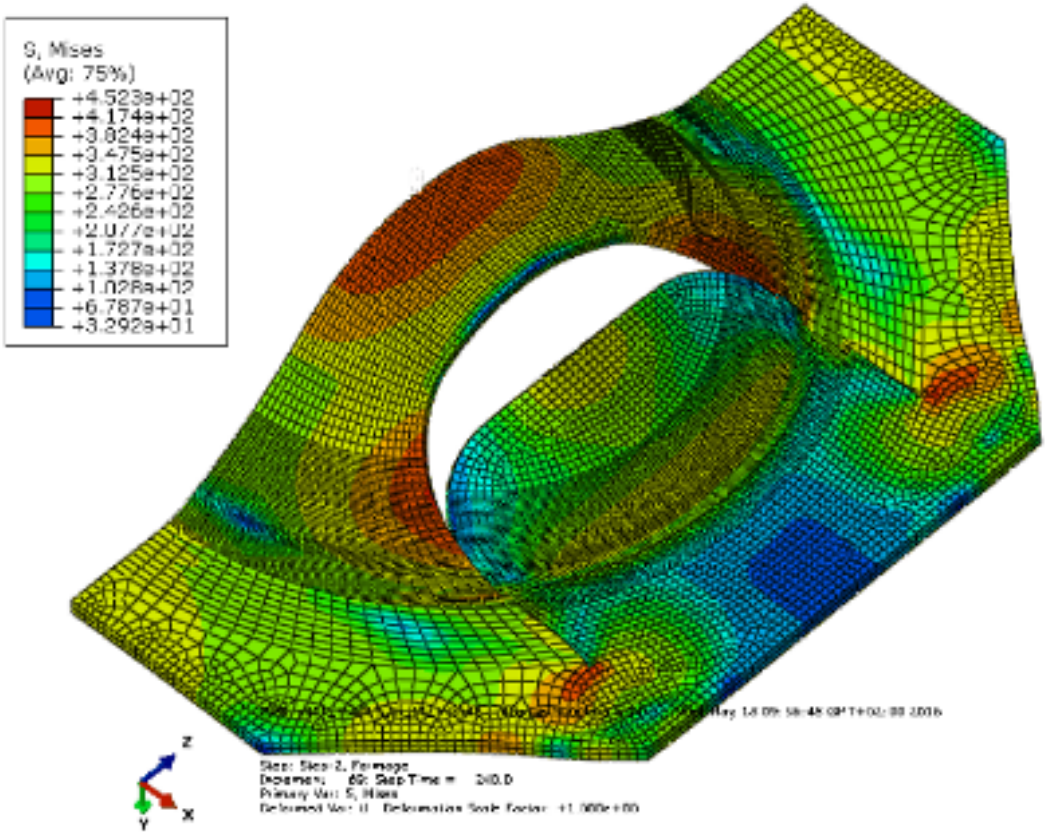


Some examples



VeRCoRs mock-up
of a nuclear reactor
building.
Code_Aster
Mehdi Asali
2016

Some examples



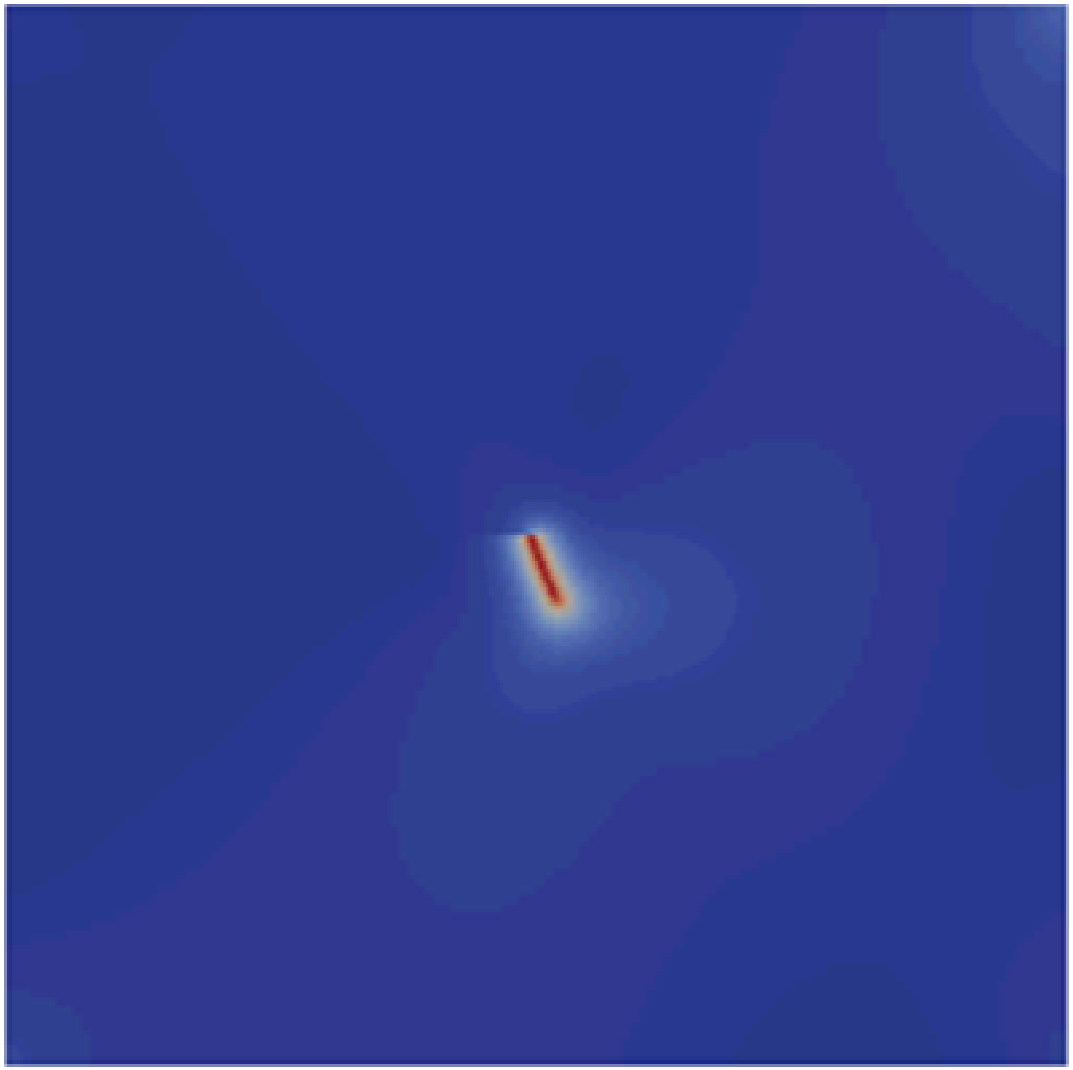
Simulation of a Punching test, used to derive a Forming Limit Diagram.

Dominique Deloison

Abaqus Standard

2016

Some examples



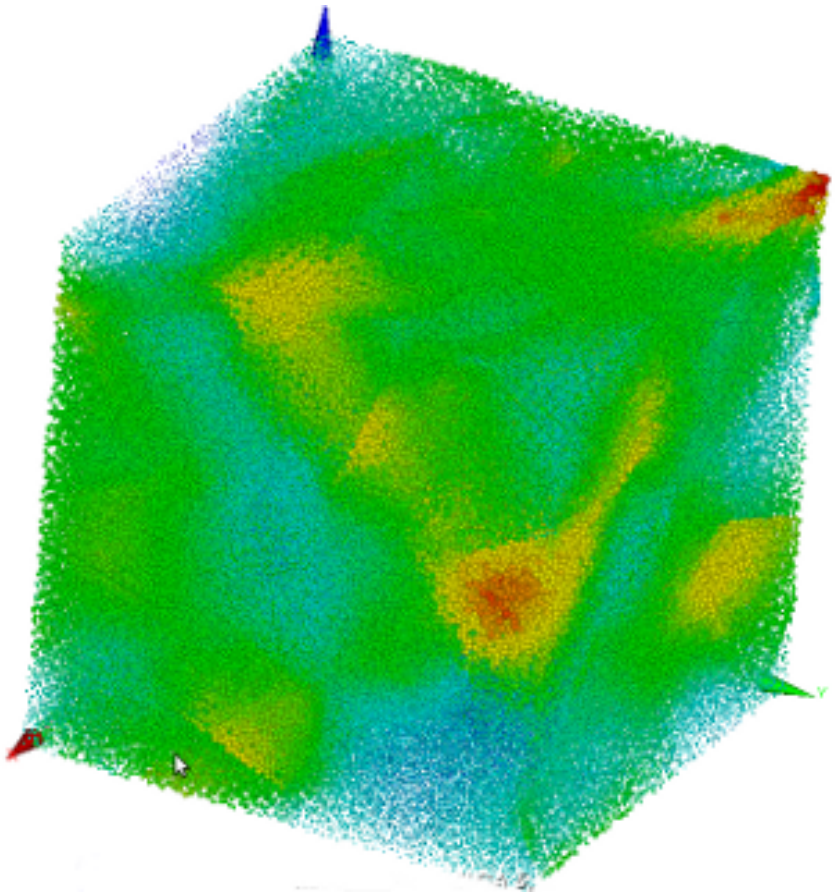
Phase Field approach to brittle fracture

Tran Thang Dang
Benoit Bary

Cast3M

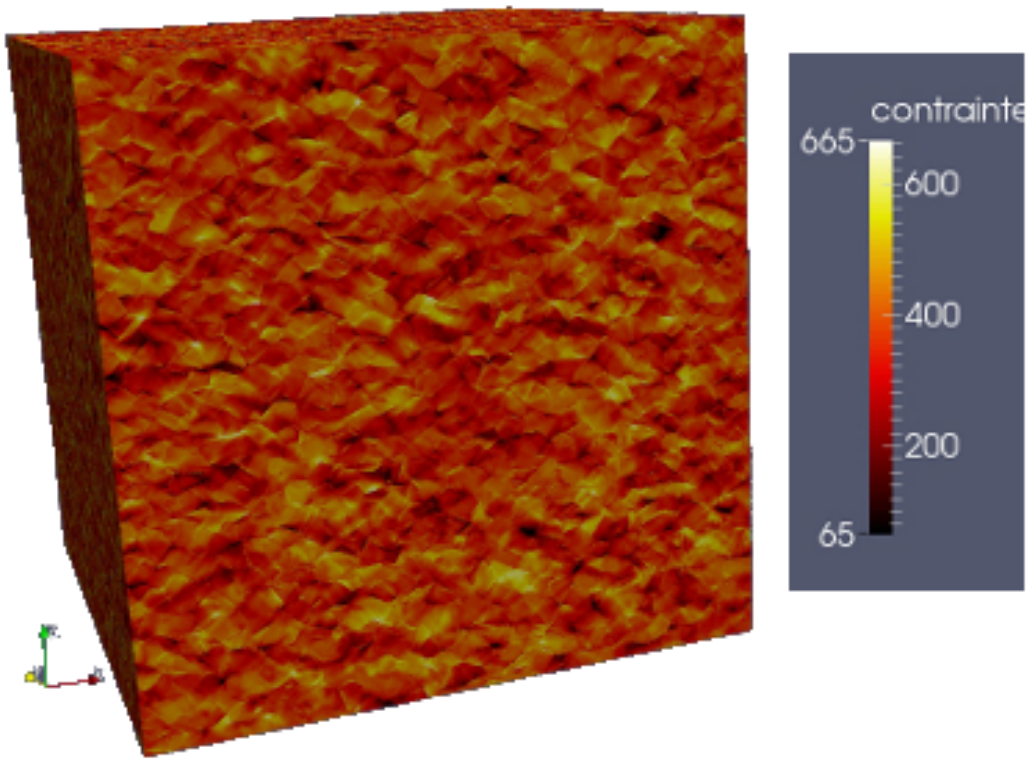
2016

Some examples



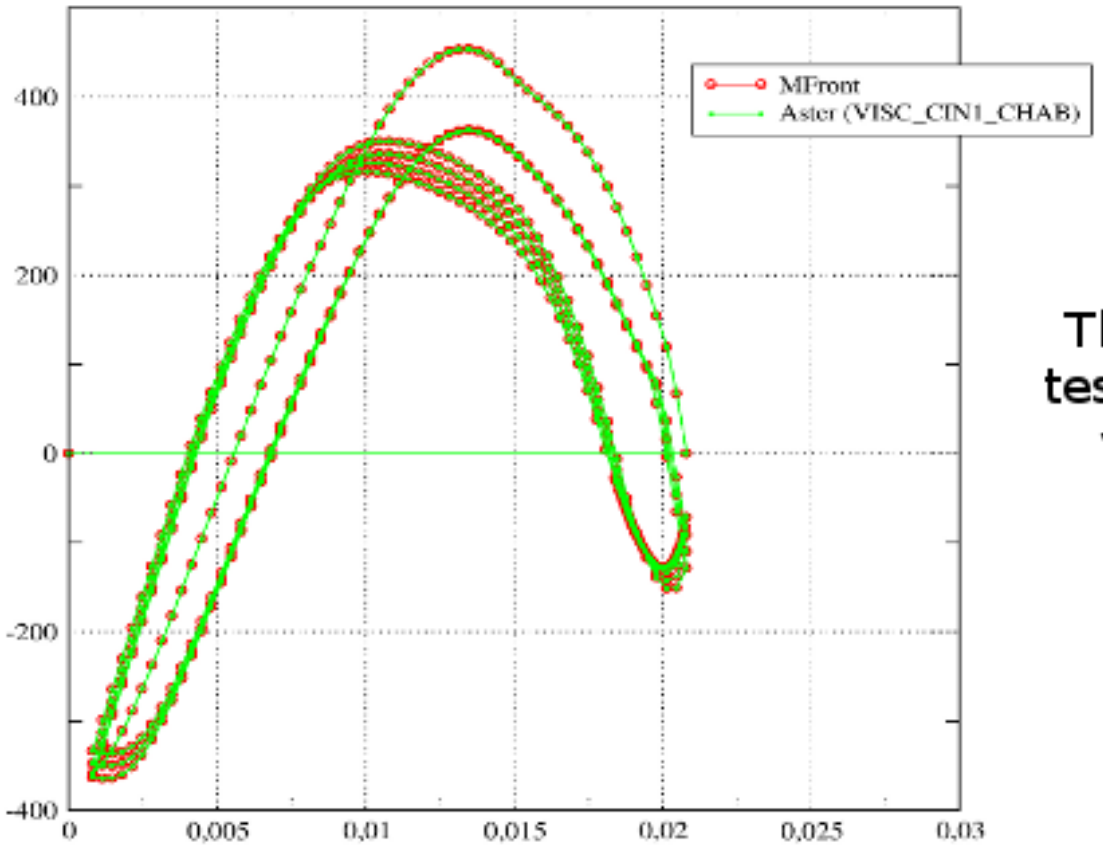
Polycrystals
computations.
Code Aster
J.-M. Proix
2014

Some examples



Polycrystals
computations
10^9 voxels
AMITEX_FFTP
L Gélébart
2014

Some examples



Thermo-mechanical unit testing of a Chaboche-like viscoplastic behaviour
MTest vs Code-Aster
J.M. Proix, 2014

Timeline : more than 10 years of development

- 2006 : first line of codes :
  - Mainly focused on the `Cast3M` finite solver.
- 2009 : officially part of the `Pleiades` project :
  - Development of the `Cyrano` interface.
- 2013 : first contact with the `code_aster` team.
- 2014 : TFEL-2.0 is released as an open-source project :
  - Officially co-developped by CEA and ÉDF.
  - Implementing an interface to `ZMAT`.
- 2015 : Officially part of `code_aster/Salomé-Méca`.
- 2016 : TFEL-3.0 is released (C++ -11)
  - Support of `Europlexus`, `Abaqus/Standard`, `Abaqus/Explicit`
  - `StandardElasticity` brick
- 2017 : TFEL-3.1 is released :
  - Experimental support of `Ansys`
- October 2018 : TFEL-3.2 is released :
  - The **generic** interface. The `StandardElastoViscoplasticity` brick
- October 2019 : TFEL-3.3 is released

Generic behaviours

Generic behaviours - I

- Generic behaviours relates pairs of gradients and fluxes.

Generic behaviours - I

- Generic behaviours relates pairs of gradients and fluxes.
- The consistent tangent operator at least contains the derivatives of the flux with respect to the gradients.

Generic behaviours - I

- Generic behaviours relates pairs of gradients and fluxes.
- The consistent tangent operator at least contains the derivatives of the flux with respect to the gradients.
- However, new terms in the consistent tangent operator may be needed :
  - derivative of state variable with respect to a gradient (non linear heat transfer).
  - derivative of state variable with respect to an external state variable (transient heat transfer).

Generic behaviours - I

- Generic behaviours relates pairs of gradients and fluxes.
- The consistent tangent operator at least contains the derivatives of the flux with respect to the gradients.
- However, new terms in the consistent tangent operator may be needed :
  - derivative of state variable with respect to a gradient (non linear heat transfer).
  - derivative of state variable with respect to an external state variable (transient heat transfer).
- Currently limited by the fact that some mathematical objects are not defined yet :
  - derivative of vectors with respect to symmetric tensors and non symmetric tensors...

Generic behaviours - I

- Generic behaviours relates pairs of gradients and fluxes.
- The consistent tangent operator at least contains the derivatives of the flux with respect to the gradients.
- However, new terms in the consistent tangent operator may be needed :
  - derivative of state variable with respect to a gradient (non linear heat transfer).
  - derivative of state variable with respect to an external state variable (transient heat transfer).
- Currently limited by the fact that some mathematical objects are not defined yet :
  - derivative of vectors with respect to symmetric tensors and non symmetric tensors...
- Currently only supported by the `generic` interface.

Heat transfer examples

```
a_Newton = inner(grad(v), dot(Ct, grad(T_)))∗dxm
res = −inner(grad(T_), j)∗dxm
```

```
@DSL DefaultGenericBehaviour;
@Behaviour StationaryLinearHeatTransfer;

@Gradient TemperatureGradient gT;
gT.setGlossaryName("TemperatureGradient");

@Flux HeatFlux j;
j.setGlossaryName("HeatFlux");

@MaterialProperty thermalconductivity k;
k.setGlossaryName("ThermalConductivity");

@Integrator{
  j = −k ∗ (gT + dgT);
} // end of @Integrator

@TangentOperator {
  Dt = −k ∗ tmatrix<N, N, real>::Id();
}
```

- On the left, the variational form in `FEniCS`
- On the right, the ‘MFront‘ implementation

Heat transfer examples

```
a_Newton = (inner(grad(v), dot(as_tensor(dj_ddgT), grad(T_)))+
            inner(grad(v),  as_vector(dj_ddT))*T_)*dxm
res = -inner(grad(v), j)*dxm
```

```
@DSL DefaultGenericBehaviour;
@Behaviour StationaryNonLinearHeatTransfer;

@Gradient TemperatureGradient gT;
gT.setGlossaryName("TemperatureGradient");

@Flux HeatFlux j;
j.setGlossaryName("HeatFlux");

@AdditionalTangentOperatorBlock dj_ddT;

@LocalVariable thermalconductivity k;

@Integrator{
  constexpr const auto A = 0.0375;
  constexpr const auto B = 2.165e-4;
  const auto T_ = T + dT;
  k = 1 / (A + B * T_);
  j = -k * (gT + dgT);
} // end of @Integrator

@TangentOperator {
  dj_ddgT = -k * tmatrix<N, N, real>::Id();
  dj_ddT =  B * k * k * (gT + dgT);
} // end of @TangentOperator
```

- On the left, the variational form in `FEniCS`
- On the right, the 'MFront' implementation

Heat transfer examples

```
a_Newton = -(v*dH1_ddT*T_-
                dt*theta*(inner(grad(v), dot(as_tensor(dj1_ddgT),
          grad(T_)))+
                          inner(grad(v),  as_vector(dj1_ddT))*T_))
          *dxm
res = (v*(H1-H0)-dt*(theta*inner(grad(v), j1)+(1-theta)*
          inner(grad(v), j0)))*dxm
```

```
...
@StateVariable real H;
H.setEntryName("Enthalpy");
@AdditionalTangentOperatorBlock dH_ddT;
...

@Integrator {
  const auto T_ = T + dT;
  k = (T_<Tm) ? ks :  kl;
  j = -k*(gT+dgT);
  // enthalpy
  if (T_<Tm){
    Ce = Cs;
    H = Cs*T_;
  } else {
    Ce = Cl;
    H = Cl*(T_-Tm)+dHsl+Cs*Tm;
  }
}  // end of @Integrator

@TangentOperator {
  dj_ddgT = -k * tmatrix<N, N, real>::Id();
  dj_ddT = tvector<N, real>(real(0));
  dH_ddT = Ce;
}  // end of @TangentOperator
```

- On the left, the variational form in `FEniCS`
- On the right, the 'MFront' implementation

Heat transfer examples

```
a_Newton = -(v*dH1_ddT*T_-
                dt*theta*(inner(grad(v), dot(as_tensor(dj1_ddgT),
          grad(T_)))+
                          inner(grad(v),  as_vector(dj1_ddT))*T_))
          *dxm
res = (v*(H1-H0)-dt*(theta*inner(grad(v), j1)+(1-theta)*
          inner(grad(v), j0)))*dxm
```

```
...
@StateVariable real H;
H.setEntryName("Enthalpy");
@AdditionalTangentOperatorBlock dH_ddT;
...

@Integrator {
  const auto T_ = T + dT;
  k = (T_<Tm) ? ks :  kl;
  j = -k*(gT+dgT);
  // enthalpy
  if (T_<Tm){
    Ce = Cs;
    H = Cs*T_;
  } else {
    Ce = Cl;
    H = Cl*(T_-Tm)+dHsl+Cs*Tm;
  }
}  // end of @Integrator

@TangentOperator {
  dj_ddgT = -k * tmatrix<N, N, real>::Id();
  dj_ddT = tvector<N, real>(real(0));
  dH_ddT = Ce;
}  // end of @TangentOperator
```

- On the left, the variational form in `FEniCS`
- On the right, the 'MFront' implementation

Other examples, thanks to J. Bleyer

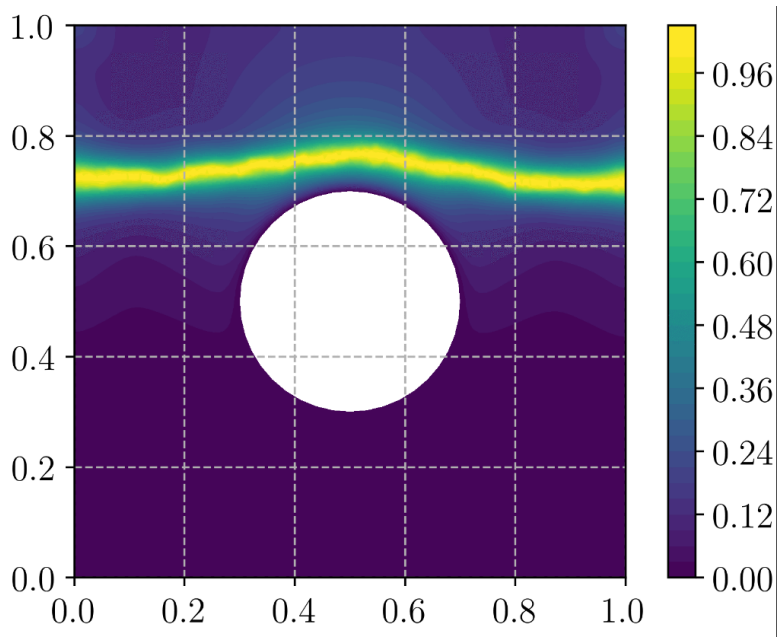
tfel-editor

File Edit Buffers Editor MFront Options Help

```
@DSL DefaultGenericBehaviour;
@Behaviour TwoPhasesGeneralizedElasticity;

@Gradient StrainStensor e1;
e1.setEntryName("MatrixStrain");
@Flux StressStensor σ1;
σ1.setEntryName("MatrixStress");

@Gradient StrainStensor e2;
e2.setEntryName("InclusionStrain");
@Flux StressStensor σ2;
σ2.setEntryName("InclusionStress");

@ProvidesTangentOperator;
@Integrator {
  // remove useless warnings, as we always compute the tangent operator
  static_cast<void>(computeTangentOperator_);

  constexpr const auto λ1 = 150e9;
  constexpr const auto μ1 = 200e9;
  constexpr const auto λ2 = 100e9;
  constexpr const auto μ2 = 300e9;
  constexpr const auto λ12 = 30e9;
  constexpr const auto μ12 = 50e9;

  ∂σ1/∂Δe1 = λ1 · (I2 ⊗ I2) + 2 · μ1 · I4;
  ∂σ1/∂Δe2 = λ12 · (I2 ⊗ I2) + 2 · μ12 · I4;
  ∂σ2/∂Δe1 = ∂σ1/∂Δe2;
  ∂σ2/∂Δe2 = λ2 · (I2⊗I2) + 2 · μ2 · I4;
  σ1 = ∂σ1/∂Δe1 · e1 + ∂σ1/∂Δe2 · e2;
  σ2 = ∂σ2/∂Δe1 · e1 + ∂σ2/∂Δe2 · e2;
}
```

test.mfront 63% (21,34) MFront 18:29

Homogeneisation



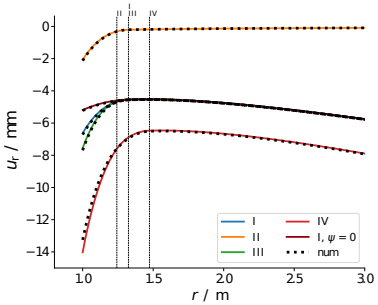
Phase-field approach of fracture

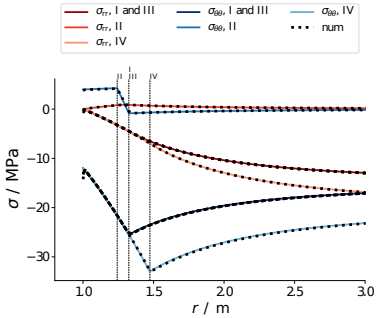
- J. Bleyer currently develops a `python` module around `MGIS` to seamlessly connect `MFront`' `generic` behaviours in `FEniCS`.
  - Very early results, but progresses (and promises) are outstanding.

MFront and OpenGeoSys : Connecting two open-source initiatives

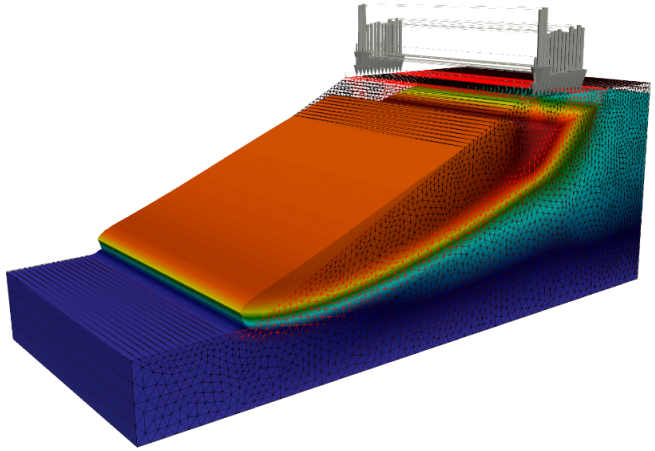
Commissariat à l'énergie atomique et aux énergies alternatives - www.cea.fr

An example based on the Mohr-Coulomb criteria





Slope failure analysis with phi-c-reduction in OpenGeoSys

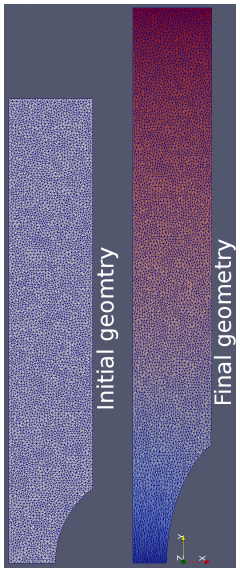


- Analytical verification (left)
- Slope failure analysis with strength reduction in `OpenGeoSys` by T. Deng and T. Nagel (right)
- For details, see `https://opengeosys.org/docs/benchmarks/small-deformations/slope_stability.pdf`.
- The implementation of the behaviour is described here : `http://tfel.sourceforge.net/MohrCoulomb.html`

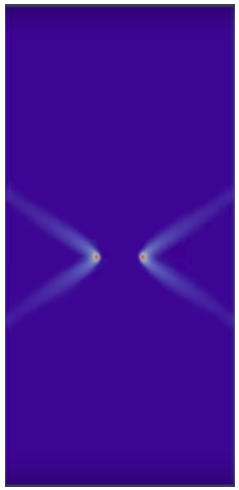
MFront related projects of interest for the OpenGeoSys community

Commissariat à l'énergie atomique et aux énergies alternatives - www.cea.fr

The MGIS project



FEniCS



Kratos

The MFrontGenericInterfaceSupport project

Thomas Helfer[1], **Jeremy Bleyer**[2], **Tero Frondelius**[3, 4], **Ivan Yashchuk**[5, 6], **Thomas Nagel**[7, 8], **and Dmitri Naumov**[7, 8]

1 French Alternative Energies and Atomic Energy Commission, DEN/DEC/SESC, Département d'Études des Combustibles, Centre de Cadarache, Saint Paul Lez Durance 13 108 CEDEX, France **2** Laboratoire Navier UMR 8205 (École des Ponts ParisTech-IFSTTAR-CNRS) **3** R&D and Engineering, Wärtsilä, P.O. Box 244, 65101 Vaasa, Finland **4** University of Oulu, Erkki Koiso-Kanttilan katu 1, 90014 Oulu, Finland **5** VTT Technical Research Centre of Finland, Kivimiehentie 3, 02150 Espoo, Finland **6** Department of Computer Science, Aalto University, Konemiehentie 2, 02150 Espoo, Finland **7** Geotechnical Institute, Technische Universität Bergakademie Freiberg, Gustav-Zeuner-Str. 1, 09599 Freiberg, Germany **8** Department of Environmental Informatics, Helmholtz Centre for Environmental Research – UFZ, Permoserstr. 15, 04318 Leipzig, Germany

DOI: 10.21105/joss.02003

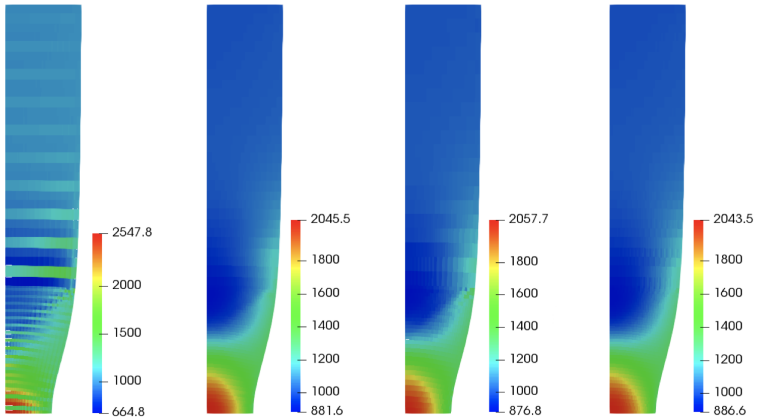
- Developped in late 2018 to allow developpers to easily and efficiently integrate MFront behaviours in their solvers.
  - Bindings for C, Fortran03, python, Julia.
  - Already usable in many solvers : OpenGeoSys, Kratos Multiphysics, FEniCS, JuliaFEM, XPer, Europlexus, etc..
  - And the first non CEA solvers was ... OpenGeoSys !
    - Thanks to C. Lehmann' work.
- https://github.com/thelfer/MFrontGenericInterfaceSupport

The MFrontGallery project

```
mfront_behaviours_library(UO2
  UO2LPCCCreep
  UO2DDIF2
  UO2L3F
  UO2CompressibleViscoplasticBehaviour_GattMonerie_ALCYONE_1_4
   ...
  UO2DDIF2IncompressibleViscoplasticBehaviour_GattMonerie_SESC_2017
  UO2DDIF2ViscoplasticDishFillingBehaviour_Julien_SESC2015
  UO2DDIF2ViscoplasticDishFillingBehaviour_Julien_SESC2017
  DEPENDENCIES
  UO2_YoungModulus
  UO2_PoissonRatio
  UO2_ShearModulus
  UO2_ThermalExpansion)
```

- The MFrontGallery project provides a cmake framework that :
  - Handles the **compilation** of material libraries and the executions of **unit-tests** (non regression tests and integration tests) for all interfaces supported by MFront (Cast3M, Cyrano, Code_Aster, etc.)
  - It allows a complete dissociation of the solver from the material knowledge while defining a strict AQ procedure.
  - https://github.com/thelfer/MFrontGallery

A portable implementation of the Hybrid High Order method



(e) Q2 (f) UPG (g) HHO(1) (h) HHO(2)

Solver

HybridHighOrder library

MGIS

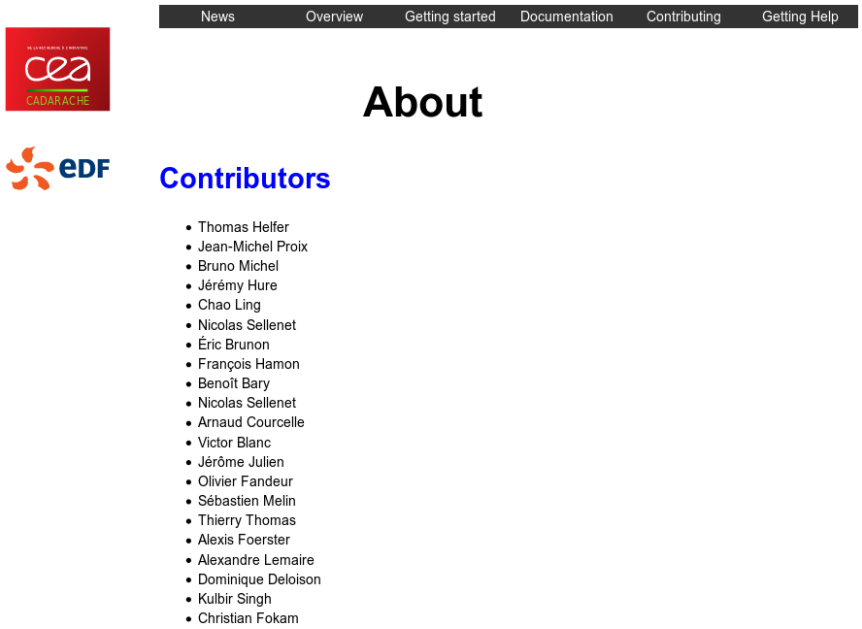
- Hybrid High Order methods are a family of non conforming approximation methods with is insensible to volumetric locking.
  - Used for incompressible hyperelastic behaviours and finite strain plasticity in the PhD thesis of N. Pignet (EDF Lab) and implemented in `code_aster` and `DisK++`
  - Can be related to Hybrid Discontinous Galerkin (HdG) Method, non conforming Virtual Element Method, etc..
- Currently working on an open-source portable implementation for phase field modelling of fracture.
  - Shall be released this year.

Conclusions

Some perspectives

- Continue working on generalised behaviours :
  - Implementation of monolithic strongly coupled resolution schemes.
  - A interesting (long-term) perspective : integrate automatic differentiation tools to define **only** the free energy.
- Extend the `StandardElastoViscoPlastic` brick to porous plasticity (e.g. GTN-like behaviours)
- Finalize support of unit checks at compile time.
- Port to C++ -17 (TFEL-4.0)

How to contribute



- Citations and illustrations
- Feed-backs, feed-backs, and feed-backs !
  - Please use the forum.
  - Enhancement suggestions (code, documentation, algorithm, etc...)
- Submit new behaviours implementation and tests.
- Submit pages to the gallery.
- Code (for the braves)

Thank you for your attention. Time for discussion!

https://tfel.sourceforge.net

https://twitter.com/TFEL_MFront

https://github.com/thelfer/

tfel-contact@cea.fr

Appendix

Licences : Open-source (again)!

- To meet CEA and EDF needs, `TFEL` 2.0 is released under a multi-licensing scheme :
  - Open-source licences :
    - GNU Public License : This licence is used by the `code_aster` finite element solver.
    - CECILL-A : License developped by CEA, EDF and INRIA, compatible with the GNU Public License and designed for conformity with the French law.
  - CEA and EDF are free to distribute `TFEL` under custom licences : Mandatory for the `PLEIADES` plateform.