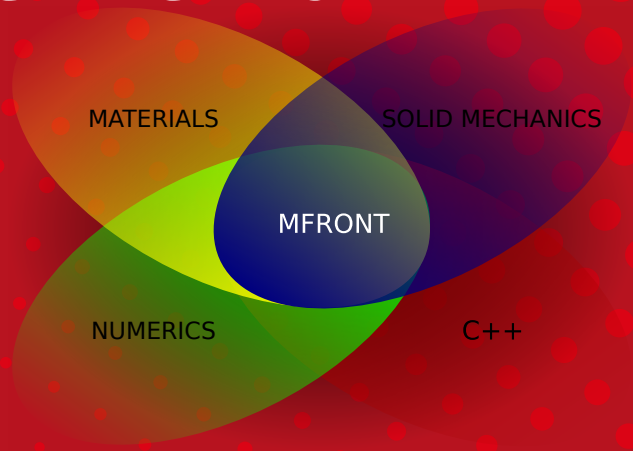
Overview of TFEL-4.0



DE LA RECHERCHE À L'INDUSTRIE

MFront User Meeting

25/11/2021

T. Helfer[(1)] and (so) many others
[(1)] CEA, DES, IRESNE, DEC, SESC, LSC, Cadarache, FranceCEA

Outline

- **A very coarse overview of `MFront` and `MGIS`**
- **Some noticeable applications of `MFront` in 2021 (not described in the other talks)**
- **New features in Versions `TFEL-3.4.1`, `TFEL-3.4.2` and `TFEL-3.4.3`**
- **New features in Version `TFEL-4.0`**
- **Conclusions**

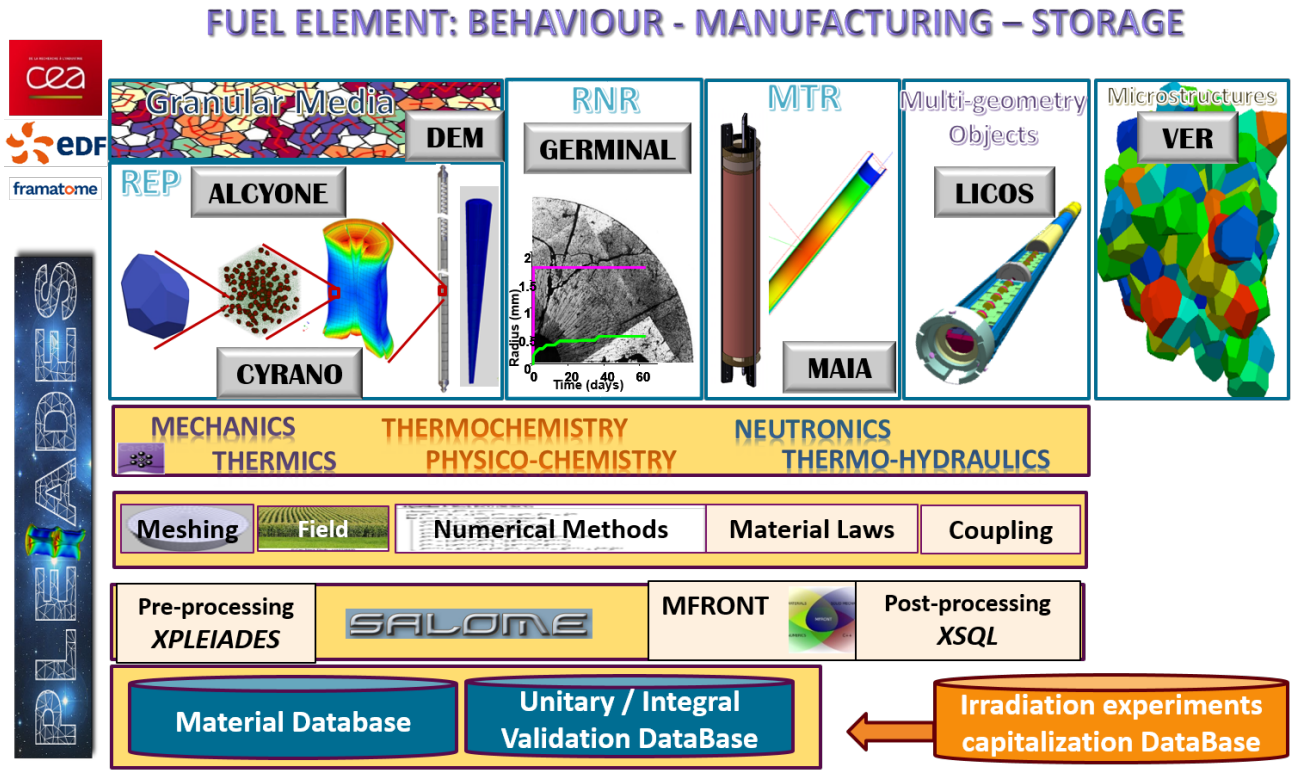
A very coarse overview of MFront and MGIS

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

Word of caution

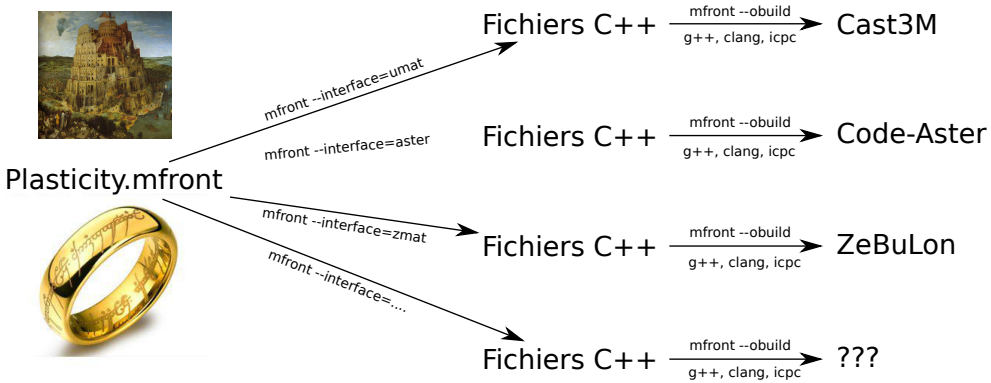
- For a more complete overview of `MFront`, see :
  - `https://thelfer.github.io/tfel/web/index.html`
  - Introducing the open-source mfront code generator : Application to mechanical behaviours and material knowledge management within the PLEIADES fuel element modelling platform. Computers & Mathematics with Applications. Available from : `http://www.sciencedirect.com/science/article/pii/S0898122115003132`
- For a more complete overview of `MGIS`, see :
  - `https://thelfer.github.io/mgis/web/index.html`
  - The MFrontGenericInterfaceSupport project. Journal of Open Source Software. `https://doi.org/10.21105/joss.02003`
- `https://www.youtube.com/watch?v=nldf7IEtnpM`

The Pleiades platform



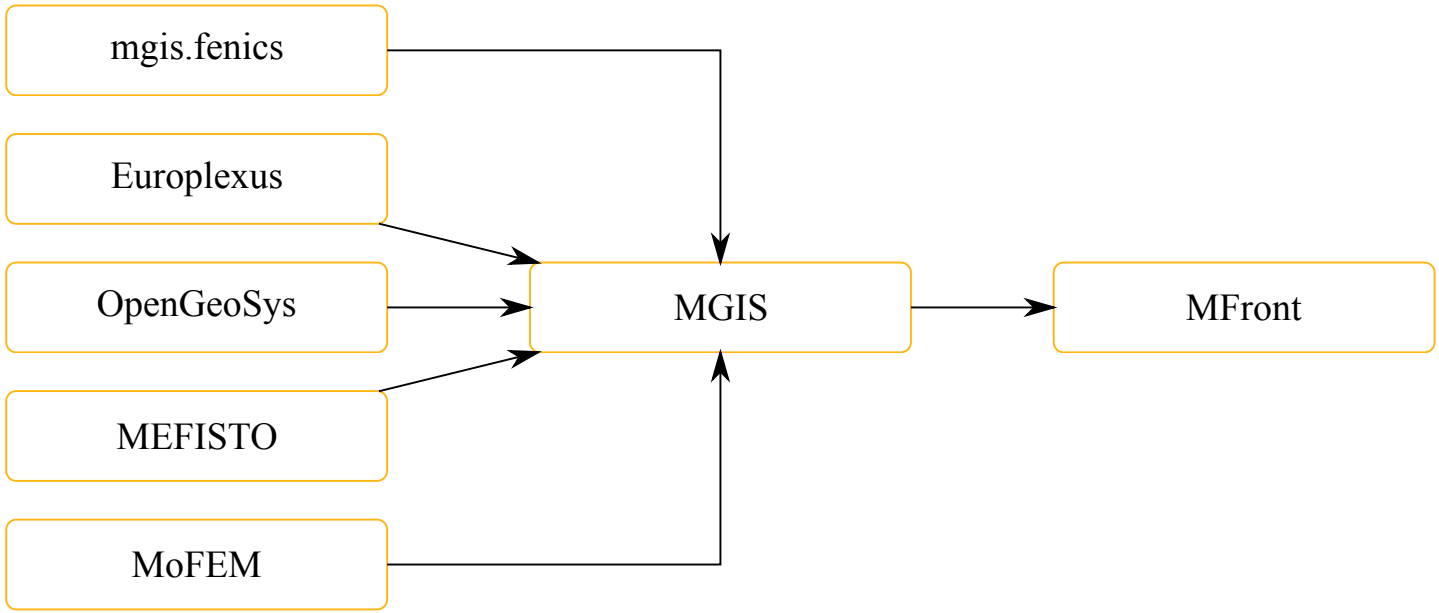
- A wide range of materials (ceramics, metals, composites).
- A wide range of mechanical phenomena and behaviours.
  - Creep, swelling, irradiation effects, phase transitions, etc..
- A wide range of mechanical loadings.

MFront goals



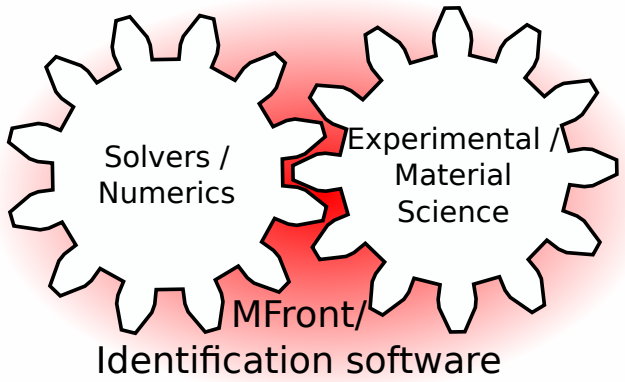
- `MFront` is a code generation tool dedicated to material knowledge (material properties, mechanical behaviours, point-wise models) :
  - Support for small and finite strain behaviours, cohesive zone models, **generalised behaviours** (non local and or multiphysics).
- Main goals :
  - Numerical efficiency (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).

The MFrontGenericInterfaceSupport project (MGIS)



- The MGIS project provides classes on the solver side to retrieve **metadata** from an MFront behaviour and call the behaviour integration over a time step.
- Written in C++. Bindings exists for C, Fortran2003, python, Julia.
- Used/tested in mgis.fenics, OpenGeoSys, Manta, XPer, MoFEM, Moose, MFEM, Disk++, Kratos Multiphysics, JuliaFEM, NairmMPM, esys.escript, DUNE, OOFEM and more...

Material knowledge management

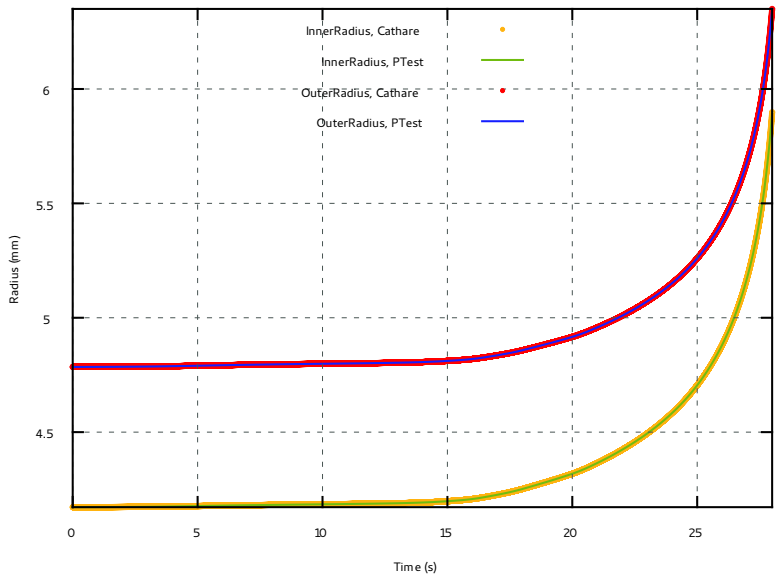


- The need to guarantee the quality of engineering studies has never been so high and is constantly growing.
- Every part of a study must be covered by strict AQ procedures :
  - The finite element solver on the one hand (see the `code_aster` documentation and unit tests).
  - The material knowledge (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

Some noticeable applications of `MFront` in 2021 (not described in the other talks)

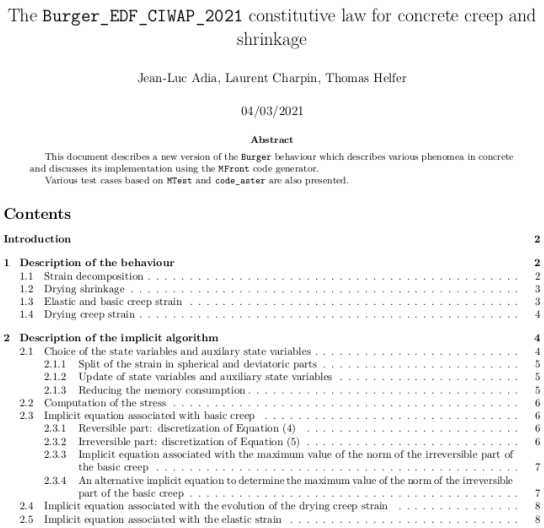
Commissariat à l'énergie atomique et aux énergies alternatives - www.cea.fr

Coupling Cathare and MTest through the ICOCO interface



- Functional coupling between Cathare (French reference thermal-hydraulic system code used for nuclear safety analysis) and MTest through the ICOCO interface :
  - Evolution of the cladding results from a local finite strain finite element resolution performed by MTest.
  - Computational times are mostly unaffected.
- Courtesy of G. Marois (CEA DES/DEC) and G. Fauchet (CEA DES/DM2S).

The Burger_EDF_CIWAP_2021 constitutive law for concrete creep and shrinkage



The `Burger_EDF_CIWAP_2021` constitutive law for concrete creep and shrinkage

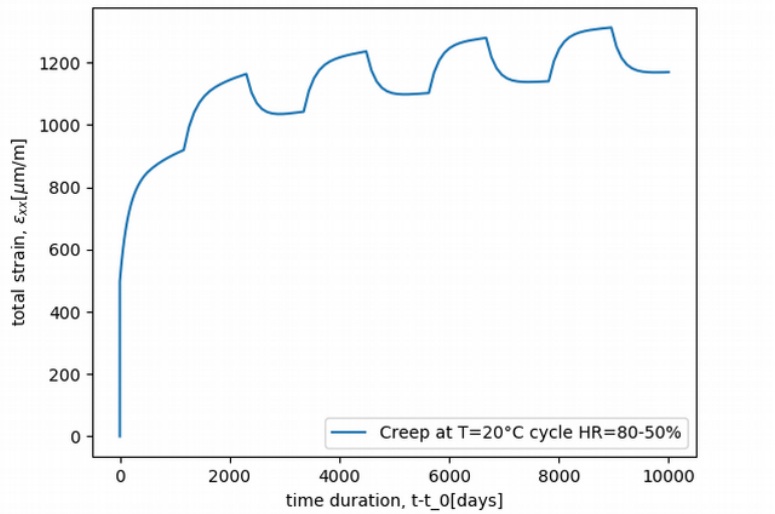
Jean-Luc Adia, Laurent Charpin, Thomas Helfer

04/03/2021

Abstract

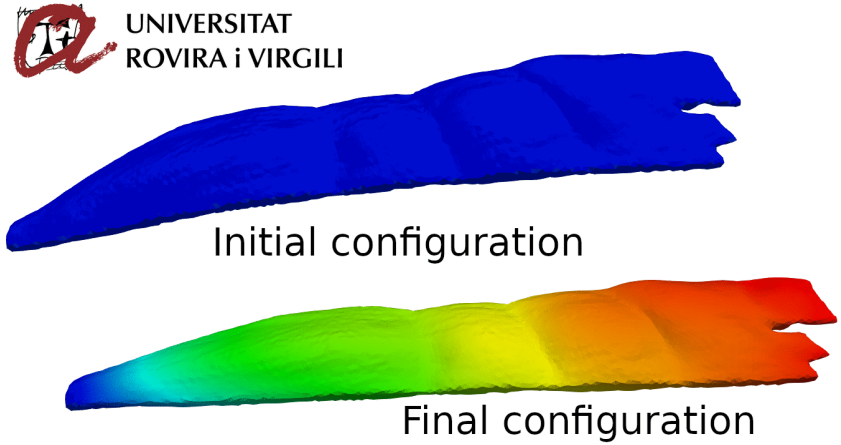
This document describes a new version of the `Burger` behaviour which describes various phenomena in concrete and discusses its implementation using the `MFront` code generator.
Various test cases based on `MTest` and `code_aster` are also presented.

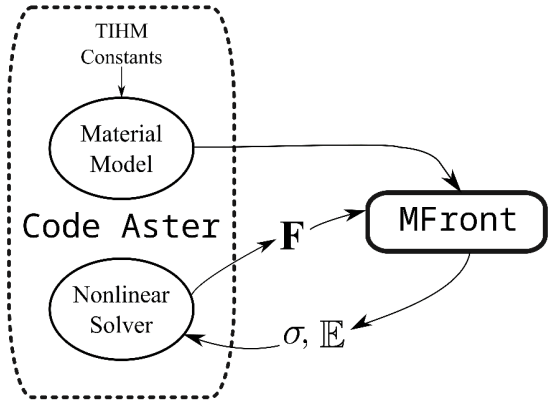
Contents



- The `Burger_EDF_CIWAP_2021` behaviour is shared on the MFront gallery in the framework of the European project ACES, in order to give acces to the VERCORS benchmark 3rd participants
- This behaviour describes the elasticity, drying shrinkage, basic creep and drying creep of concrete.
- `https://thelfer.github.io/tfel/web/Burger_EDF_CIWAP_2021.html`
- Courtesy of J.-L. Adia (EDF R&D) and L. Charpin (EDF R&D).

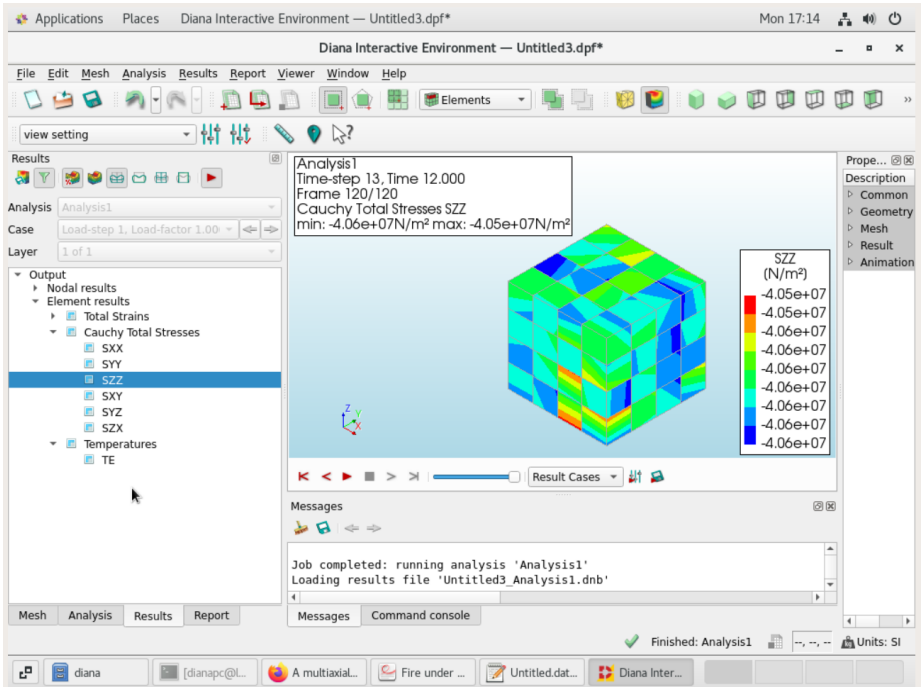
Modelling of abdominal muscles

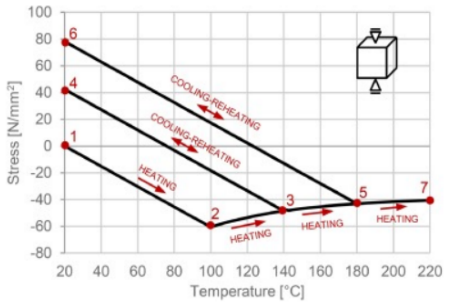




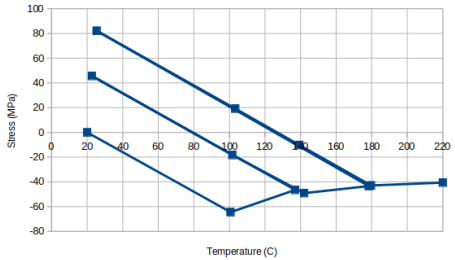
- A slightly compressible hyperelastic material formed by a ground isotropic material and one family of fibers to model muscle tissue in `code_aster`.
  - *Implementation of a new constitutive model for abdominal muscles*. Ll. Tuset, G. Fortuny, J. Herrero, D. Puigjaner, J. M. López. Computer Methods and Programs in Biomedicine. 179, October 2019
- In progress extension to viscohyperelasticity based on :
  - *Anisotropic finite strain viscoelasticity : Constitutive modelling and finie element implementation*. H. Liu, G. A. Holzapfel, B. H. Skallerud, V. Prot. Journal of the Mechanics and Physics of Solids. 124. March 2019

Advanced concrete models in DIANA FEA





Published results

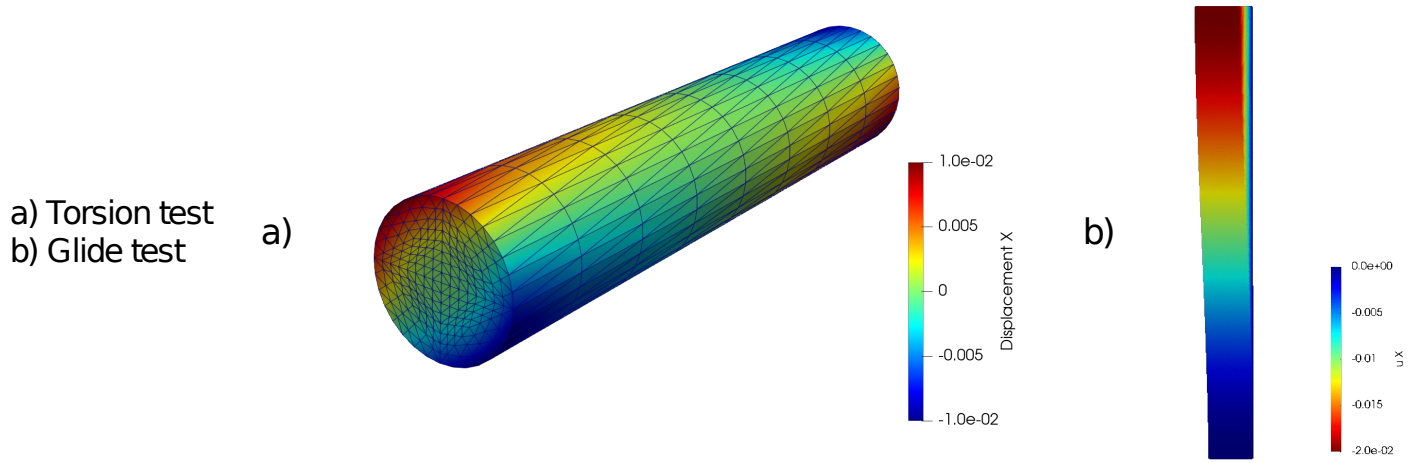


DIANA results

- Functional interface for DIANA to demonstrate the safety of AGR plants (in-operation or decommissioned) in case of LOss of Coolant Accidents.
- Coupling with `MFront` to introduce advanded concrete behaviours :
  - Load Induced Thermal Strain : `https://www.sciencedirect.com/science/article/pii/S0020768316303456`
- Courtesy of J. Draup (EDF R&D UK).

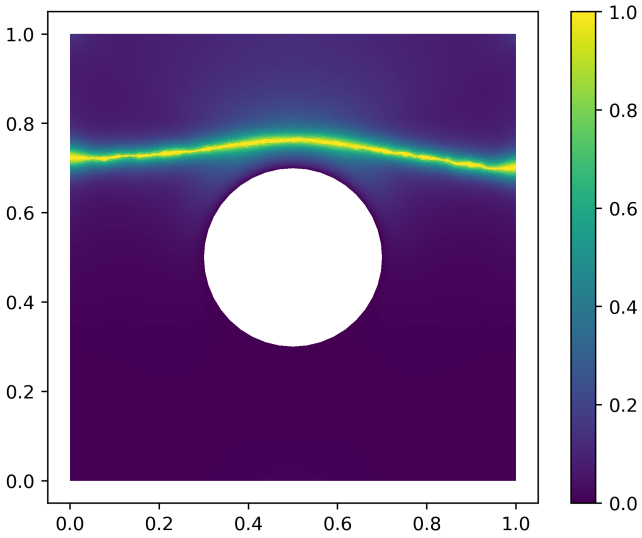
Modelling of adiabatic shear bands using a plasic Cosserat medium

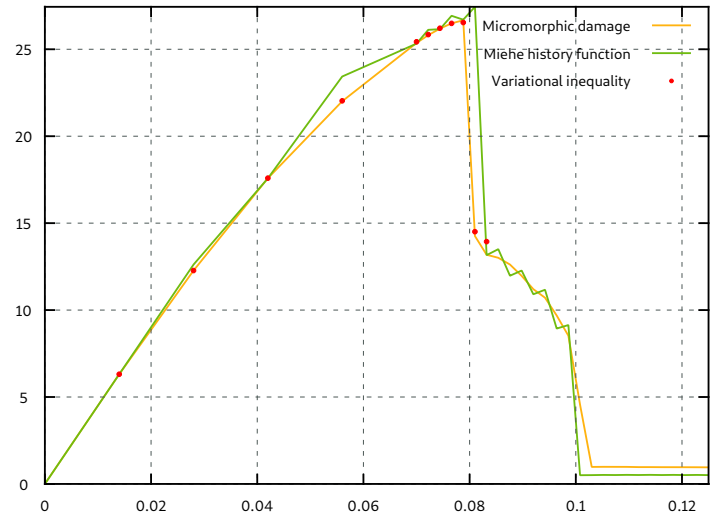
a) Torsion test
b) Glide test



- Cosserat Media in small deformation framework based on the work of Russo et al. implemented in the open-source FEM platform `FEniCS` and `TFEL/MFront`.
  - *Thermomechanics of Cosserat medium : modeling adiabatic shear bands in metals.* Russo, Raffaele, Samuel Forest, and Franck Andrés Girot Mata. Continuum Mechanics and Thermodynamics (2020) : 1-20.
- Validation and performance of the model for the glide test :
  - *mgis. fenics Part II : Cosserat media in small deformation with* `mgis.fenics`. Dancheva, Tamara, et al. FEniCS 2021. 2021
- Courtesy of , T. Dancheva (Basque Center for Applied Mathematics).

Micromorphic damage of brittle material

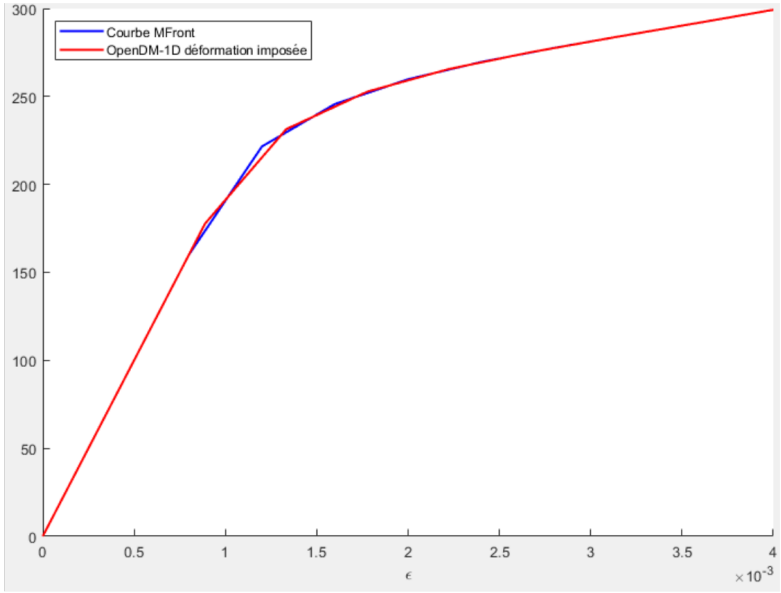




- Simulation of the fiber reinforced matrix in tension with a micromorhic damage model using `mgis.fenics`.
  - Implementation of the monolithic scheme and variours staggered schemes.
- PhD of D. Siedel (CEA, Mines ParisTech, PSL University, Centre des matériaux, CNRS UMR 7633)

Ceramic composite damage model in 1D

```
@DSL DefaultGenericBehaviourDSL;
@Behaviour OneraDamageModel1D;

@Gradient strain e;
e.setEntryName("ScalarStrain");
@ThermodynamicForce stress s;
s.setEntryName("ScalarStress");

@MaterialProperty stress E0;
E0.setGlossaryName("YoungModulus");
@MaterialProperty stress y0_s;
@MaterialProperty stress y0_c;
@MaterialProperty real d_p;
@MaterialProperty real d_c;
@MaterialProperty real chi;

@StateVariable stress ymax;
ymax.setEntryName("ElasticEnergyReleaseRate");
@StateVariable real d;
d.setGlossaryName("Damage");

@Integrator {
  // strain at the end of the time step
  const auto e_ets = e + de;
  const auto y = (e_ets >= 0) ? E0 * power<2>(e_ets) / 2 : 0;
  ymax = max(ymax, y);
  const auto g =
      (ymax > y0_s) ? (sqrt(ymax) - sqrt(y0_s)) / sqrt(y0_c) : real(0);
  const auto d_s = d_c * (1 - exp(-pow(g, d_p)));
  //
  const auto eta = (e_ets >= 0) ? 1 : 0;
  // damage update
  d = d_s;
  // stress update
  const auto Ee = E0 / (1 + eta * d);
  s = Ee * e_ets - E0 * er;
}
```

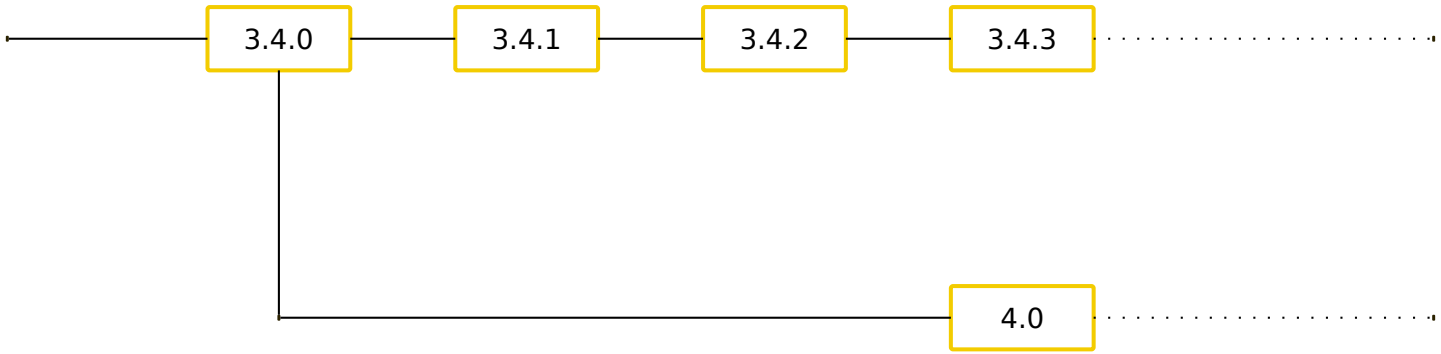


- Advanced description of ceramic composites.
- Implementation based on generic behaviours (scalar relationships).
  - In progress implementation for multi-dimensional analyses in `Abaqus/Standard`.
- Courtesy of F. Guillet (CEA-DAM, DMAT)

New features in Versions TFEL-3.4.1, TFEL-3.4.2 and TFEL-3.4.3

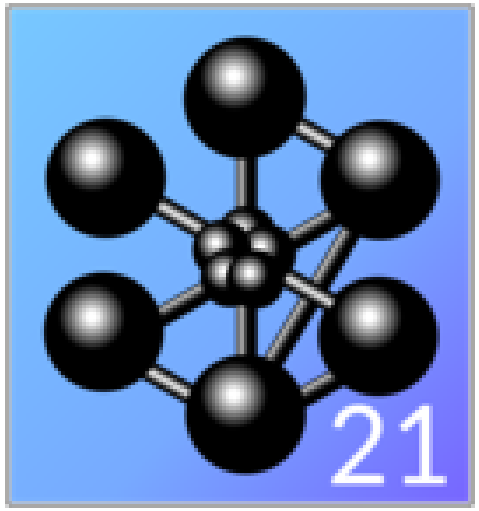
Commissariat à l'énergie atomique et aux énergies alternatives - www.cea.fr

About the TFEL 3.4.x series



- `TFEL-3.4.x` contains bug fixes and backports of new features when possible.
  - This is due to the fact that `TFEL-4.0.x` requires fairly recent compilers with decent support of C++ -17 which may not be accessible to some platforms (`code_aster`, `MAP`, etc.), so transition may take more time than usual.

TFEL 3.4.1




```
<!--
    Simple template for use in AMITEX_FFTP.

    You may copy-paste those lines in your material.xml file
    and adjust:

- the material number l,
- the path to the library (currently UmatBehaviour),
- the coefficients and initial  internal  variables values
which are currently replaced by three dots.
-->

<Material numM="l" Lib="UmatBehaviour" Law="umatplasticity">
  <!-- material property YoungModulus -->
  <Coeff Index = "1" Type = "Constant" Value = "..."  />
  <!-- material property PoissonRatio -->
  <Coeff Index = "2" Type = "Constant" Value = "..."  />
  <!-- internal state variable ElasticStrain -->
  <IntVar Index = "1"  Type = "Constant" Value = "..."  />
   ...
  <!-- internal state variable EquivalentPlasticStrain -->
  <IntVar Index = "7"  Type = "Constant" Value = "..."  />
</Material>
```

- Release notes :
  - `http://tfel.sourceforge.net/release-notes-3.4.1.html`.
  - Shipped with `Cast3M 2021`.
- Main new features :
  - Support for `Cast3M 21`.
  - The `amitex` interface with automatic input file generation.
  - Save the strain measure and the dual stress (Ticket #236)
  - Computation of the speed of sound.
  - `API` versionning for the `generic` interface.

TFEL 3.4.2

```
import mtest
young_modulus = mtest.MaterialProperty(
      'src/libCastemVanadium.so', 'VanadiumAlloy_YoungModulus_SRMA')
# One may set each variable individually:
young_modulus.setVariableValue('Temperature', 562)
E = young_modulus.getValue()
# or use a dictionary:
E = young_modulus.getValue({'Temperature': 562})
# The call operator is available for material properties with only one argument::
E = young_modulus.getValue(562)
# The setParameter method can be used to change the value of a parameter:
young_modulus.setParameter('E0', 78e9)
```

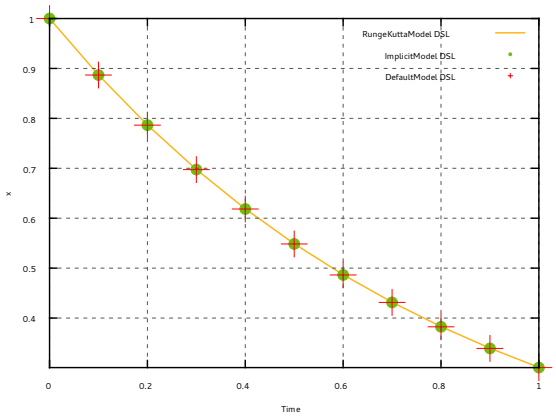
- Release notes :
  - `http://tfel.sourceforge.net/release-notes-3.4.2.html`
  - This version was released on September 21, 2021 along with Versions 3.0.9, 3.1.9, 3.2.6, 3.3.2, and inherits from the issues solved in those versions.
- Main new features :
  - Ability to compute the interaction matrix manually.
  - *Basic support for material properties in* `MTest`

```
@DSL RungeKuttaModelDSL;
@Model ode_rk54;
@Author Thomas Helfer;
@Date 21 / 09 / 2021;

@Epsilon 1.e-11;

@StateVariable real x;
@Parameter frequency A = 1.2;

@Derivative {
  dx = -A * x;
}
```

MFront

```
@Author Thomas Helfer;
@Date   21/09/2021;
@ModellingHypothesis "Tridimensional";
@Model<generic> "src/libBehaviour.so" "
        ode_rk54";
// internal state variable initialisations
@InternalStateVariable "x" 1;
// external state variable
@ExternalStateVariable "Temperature"
        293.15;
@Times {0, 1 in 10};
```

MTest



Results

- Ability to implement point-wise models as a particular case of generic behaviours and test in `MTest` :
  - No gradient, thermodynamic force nor tangent operator.
  - Description of swelling due to irradiation products, phase transition, etc..
  - Three new DSLs : `DefaultModel`, `RungeKuttaModel`, `ImplicitModel`

New features in Version TFEL-4.0

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

TFEL-4.0

- Release notes :
  - `http://tfel.sourceforge.net/release-notes-4.0.html`
  - First version based on the C++ -17 standard.
  - 18 227 test cases (18 208 in `TFEL/3.4.3`, 16 316 in `TFEL/3.4.0`)
  - Many tests cases have been enriched in `TFEL/4.0`.
  - Tested on `Linux`, `Mac OS` and `FreeBSD` with `gcc`, `clang` and `Intel` compilers.
- New or updated library documentations (TFEL/Math, TFEL/Material, TFEL/PhysicalConstants).
- New entry in the `MFront` gallery :
  - The `Burger_EDF_CIWAP_2021` constitutive law for concrete creep and shrinkage
- Highlights :
  - Rejuvenation of the `TFEL/Math` library.
  - Initial support for quantities in `MFront`.

Rejuvenation of the TFEL/Math library

```
namespace internals {

  template <unsigned short N>
  struct SigmaEqImpl;

  template <>
  struct SigmaEqImpl<1u> : public SigmaEqImplBase {
    template <class T>
    static typename std::enable_if<
        tfel::meta::Implements<T, StensorConcept>::cond,
        StensorNumType<T>>::type
    exe(const T& s) {
      typedef StensorNumType<T> NumType;
      typedef tfel::typetraits::base_type<NumType> base;
      constexpr auto cste = base{3} / base{2};
      const auto tr = trace(s) / 3;
      return std::sqrt(cste * (SigmaEqImplBase::square(s(0) - tr) +
                               SigmaEqImplBase::square(s(1) - tr) +
                               SigmaEqImplBase::square(s(2) - tr)));
    }
  };

  ...

} // end of namespace internals

template <typename T>
typename std::enable_if<tfel::meta::Implements<T, StensorConcept>::cond,
                        StensorNumType<T>>::type
sigmaeq(const T& s) {
  typedef tfel::math::internals::SigmaEqImpl<StensorTraits<T>::dime> Impl;
  return Impl::exe(s);
}
```

```
template <typename StensorType>
std::enable_if_t<implementsStensorConcept<StensorType>(),
                 numeric_type<StensorType>>
sigmaeq(const StensorType& s) {
  using real = base_type<numeric_type<StensorType>>;
  constexpr auto N = getSpaceDimension<StensorType>();
  constexpr auto one_third = real(1) / 3;
  constexpr auto cste = real(3) / 2;
  static_assert((N == 1) || (N == 2) || (N == 3), "invalid space dimension");
  auto square = [](const auto x) { return x * x; };
  const auto tr = one_third * trace(s);
  if constexpr (N == 1u) {
    return std::sqrt(cste * (square(s(0) - tr) +  //
                             square(s(1) - tr) +  //
                             square(s(2) - tr)));
  } else if constexpr (N == 2u) {
    return std::sqrt(cste * (square(s(0) - tr) +  //
                             square(s(1) - tr) +  //
                             square(s(2) - tr) + square(s(3))));
  } else if constexpr (N == 3u) {
    return std::sqrt(cste * (square(s(0) - tr) +  //
                             square(s(1) - tr) +  //
                             square(s(2) - tr) +  //
                             square(s(3)) + square(s(4)) + square(s(5))));
  }
} // end of sigmaeq
```

- Updated documentation

  `https://thelfer.github.io/tfel/web/tfel-math.html`
- Overall improvement of the code of the library :
  - `constexpr` and `noexcept` correctness.
  - Almost 25 % reduction of the number of line of codes.
- Better support for quantities.
- Support for higher order tensors.
- General framework to build non linear solvers.

Implementation of quantities in TFEL/Math

```
using namespace tfel::math;
// scalars
constexpr qt<Mass> m1(100.);
constexpr qt<Mass> m2(100.);
constexpr qt<Mass> m3 = m1 + 0.5 * m2;
constexpr qt<Acceleration, unsigned short> a(2);
constexpr qt<Force> f = m1 * a;
// stensors and views
auto sig = stensor<3u, stress>{stress{0}, stress{1}, stress{2},
                               stress{3}, stress{4}, stress{5}};
tvector<6u, double> stress_values;
map<stensor<3u, qt<Stress, double>>>(stress_values) = sig;
```

- Quantities are mathematical objects associated with units :
  - The compiler performs a dimensional analysis on every operations.
- The `map` and `map_derivative` allows to map memory from a vector and a matrix respectively of basic numeric types to tensorial types based on quantities.
  - Basic mechanism on which `MFront`' implicit schemes are built.

Support for quantities in MFront

```
@Parameter strainrate A = 8.e−67;
@Parameter real E = 8.2;
@Parameter stress K = 1;
@Parameter stress R0 = 20e6;
@Parameter stress Rinf = 40e6;
@Parameter real bvp = 10;

@Integrator {
  ...
  const auto seq = sigmaeq(sig);                    // seq has the unit of a stress
  const auto iseq = 1 / max(seps, seq);             // iseq has the unit of the inverse of a stress
  const auto n = 3 * deviator(sig) * (iseq / 2);    // normal has no unit
  const auto exp_bvp = exp(−bvp * (p + theta * dp)); // exp_bvp has no unit
  Rvp = R0 + (Rinf − R0) * (1 − exp_bvp);           // Rvp has the unit of a stress
  if (seq > Rvp) {
    const auto vp = A * pow((seq − Rvp) / K, E);    // vp has the unit of a strainrate
    fp −= vp * dt;                                  // fp has the unit a a strain
    // fp −= pow((seq − Rvp) / K, E) * dt;          // This would not compile !
    // fp −= A * pow(seq − Rvp, E) * dt;            // This would not compile !
  }
  feel += dp * n;
}
```

- The `UseQt` keyword actives the use of quantities.
- Dedicated documentation of the declaration of variables in `MFront`
- Quantities are supported in DSLs associated with material properties and behaviours.

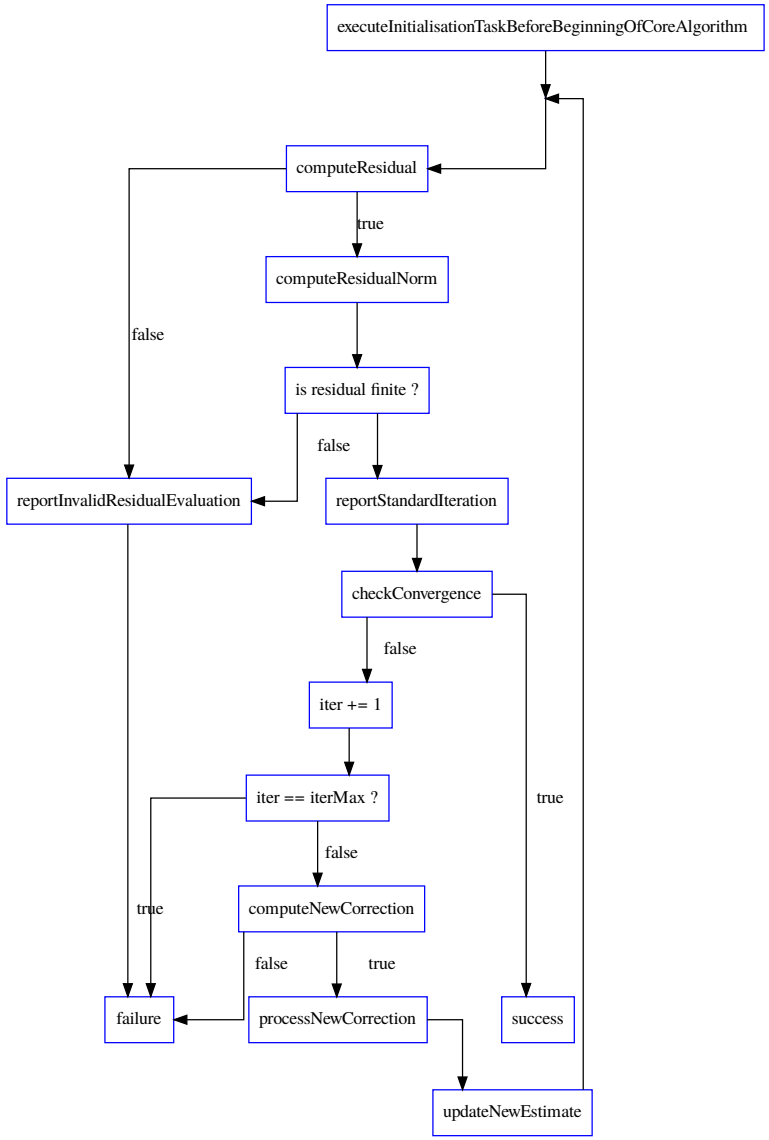
The `derivative_type` metafunction

```
derivative_type<qt<double,Stress>, double> a;                    // a is a qt<double, Stress>
derivative_type<stensor<3u,double>, stensor<3u,double>> b;  // b is a st2tost2<3u,real>
derivative_type<st2tost2<3u,double>, stensor<3u,double>> c; // c is a 6 order tensor !
// Numerical derivative of a scalar to scalar function
const auto df = tfel::math::computeNumericalDerivative(
    [](const auto& v) { return v * v; }, 0.5, 1.e-2);
// Numerical derivative of a symmetric tensor to scalar function
const auto trace_derivative = computeNumericalDerivative(
    [](const auto& s) { return trace(s); }, Stensor::Id(), 1.e-2);
// Numerical derivative of a symmetric tensor to symmetric tensor function
const auto deviator_derivative = computeNumericalDerivative(
    [](const auto& s) { return deviator(s); }, Stensor::Id(), 1.e-2);
```

- `derivative_type<A,B>` is an alias to the type of the derivative of an object of type `A` with respect to an object of type `B`.
  - The very pratical way to define an higher order tensor.
  - Works well with quantities.
- The `computeNumericalDerivative` can be used to differentiate a (λ) function with respect to its argument using a center finite difference schemes.
  - Limited to scalars and tensorial objects for arity 1 (vectors, symmetric tensors, non symmetric tensors).

Generic framework for non linear solvers

- A CRTP-based generic framework to implement a wide range of non linear solvers for system of equations whose size is known at compile-time :
  - Newton-Raphson, Broyden, Levenberg-Marquardt, Powell' dog leg algorithm
- Each steps of the algorithm can be customized in base classes.
- In case of failure during the iterations, a basic line-search algorithm is performed by default.



Example of usage

```
struct NewtonRaphsonSolver
    : public tfel::math::
          TinyNewtonRaphsonSolver<2u, double, NewtonRaphsonSolver> {
  NewtonRaphsonSolver() {
    this->zeros = {0., 0.};
    this->epsilon = 1.e-14;
    this->iterMax = 20;
  }
  //
  bool computeResidual() noexcept {
    constexpr double a = 1.;
    constexpr double b = 10.;
    auto& f = this->fzeros;
    auto& x = this->zeros;
    auto& J = this->jacobian;
    f(0) = a * (1 - x(0));
    f(1) = b * (x(1) - x(0) * x(0));
    J(0, 0) = -a;
    J(0, 1) = 0.;
    J(1, 0) = -2 * b * x(0);
    J(1, 1) = b;
    return true;
  }  // end of computeResidual

};  // end of struct NewtonRaphsonSolver
```

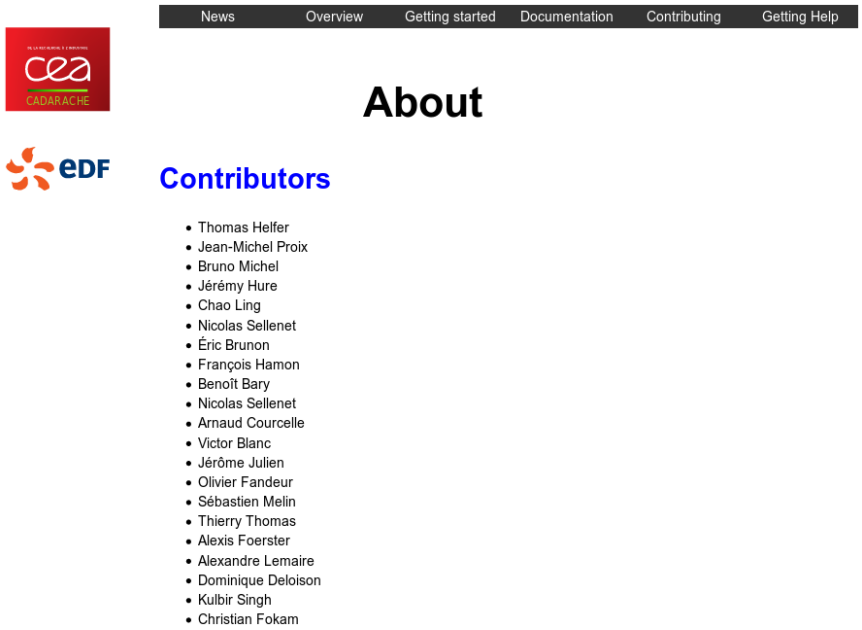
- Basically, this is a (very) simplified version of the code that `MFront` generates.

Conclusions

Current state and future works

- `MFront` is an ever improving code generation tool dedicated to material knowledge with one foot in the industrial world and one foot in the academic world.
- The development of `TFEL-4.0` has been driven by :
  - The port to the `C++-17` standard.
    - Complete rewrite of the `TFEL/Math` library.
    - Initial support for quantities.
    - `MFront` files from previous version shall be fully compatible.
  - Homogeneisation :
    - Cecilia Gicquel' talkd at the MFront User Meeting.
  - Data driven simulation :
    - See Marius Durvillard' talk at the MFront User Meeting.
  - Support of GPUs (experimental/in progress)
- Future works :
  - Additional tutorials.
  - Initialisation and post-processings support.
  - Support of GPUs and vectorisation.

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://www.researchgate.net/project/TFEL-MFront
https://twitter.com/TFEL_MFront
https://github.com/thelfer/

tfel-contact@cea.fr

The development of MFront is supported financially by CEA, EDF and Framatome in the framework of the PLEIADES project.