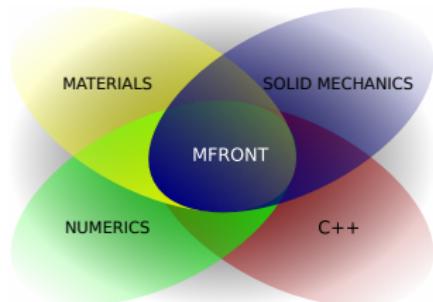


DE LA RECHERCHE À L'INDUSTRIE



Material knowledge management with the MFront code generator.



Séminaire Centrale Lille | THOMAS HELFER

21 JUNE 2018

Context

- Context
- About myself
- Material knowledge management in nuclear simulations
- The TFEL project and the MFront code generator
- Mechanical behaviours
- Performances
- Conclusions

Goal of the seminar

- Introduce MFront and the associated tools.

Goal of the seminar

- Introduce MFront and the associated tools.
- Gain new users and, hopefully, **contributors**.
 - See "How to contribute" below

Goal of the seminar

- Introduce MFront and the associated tools.
- Gain new users and, hopefully, **contributors**.
 - See "How to contribute" below
- Get feed-backs on the code and increase its robustness.

Goal of the seminar

- Introduce MFront and the associated tools.
- Gain new users and, hopefully, **contributors**.
 - See "How to contribute" below
- Get feed-backs on the code and increase its robustness.
- Meet academics for further works (PhD-thesis, etc.)
 - BTW, we are still looking for two candidates...

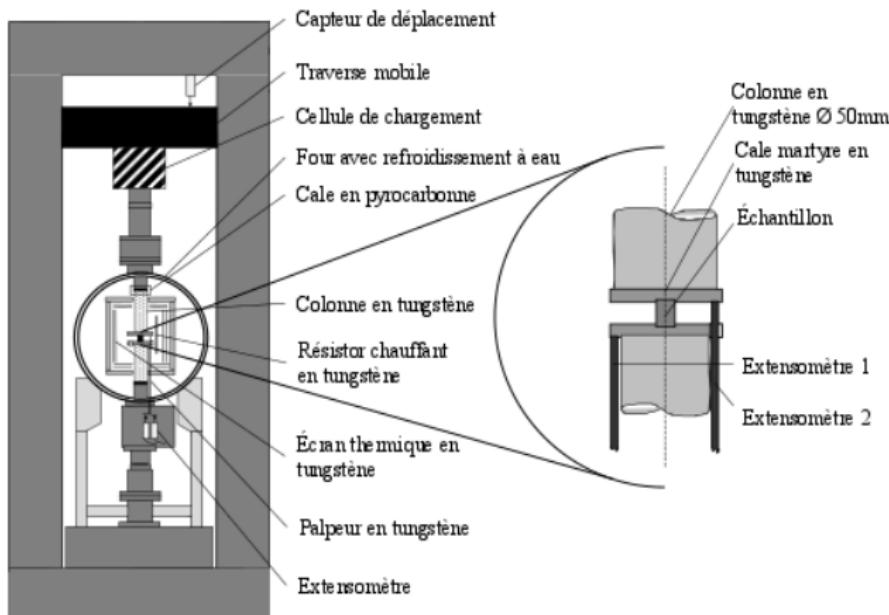
About myself

Identification of UO_2 viscoplastic behaviour : the experimental device



- Work in collaboration with the DEC/SFER/LCU :
 - 2 internships in parallel, followed by a PhD.
- Instron 1185 (max load 50 kN).

Identification of UO_2 viscoplastic behaviour : the experimental device



- Work in collaboration with the DEC/SFER/LCU :
 - 2 internships in parallel, followed by a PhD.
- Instron 1185 (max load 50 kN).

Identification of UO_2 viscoplastic behaviour : MFront implementation (version 3.2)

```
@DSL Implicit;  
@Behaviour FirstExample;  
  
@ModellingHypotheses {"."};  
@Epsilon 1.e-16;  
@IterMax 100;  
@Theta 1;  
  
@Brick "StandardElastoViscoPlasticity" {  
    stress_potential : "Hooke" {  
        young_modulus : ...,  
        poisson_ratio : ...,  
        thermal_expansion : ...,  
        thermal_expansion_reference_temperature : 293.15  
    },  
    inelastic_flow : "Norton" {  
        criterion : "Mises",  
        kinematic_hardening : "Armstrong-Frederick" {  
            C : ...,  
            D : ...  
        },  
        K : ...,  
        n : ...  
    }  
};
```

- First approach of MFront :
 - usage of build-in bricks

Identification of UO_2 viscoplastic behaviour : 1D Finite strain modelling (Imposed F_{zz})

```
@Author th202608@pleiades098.intra.cea.fr;
@Date 28 fevr. 2017;
@Description{
};

@AccelerationAlgorithm 'UAnderson';
@PredictionPolicy 'LinearPrediction';
@MaximumNumberOfSubSteps 10;

@ModellingHypothesis 'AxisymmetricalGeneralisedPlaneStrain';
@Behaviour<castem> 'src/libUmatBehaviour.so' 'umatisotropicviscoplasticityamstrongfrederickinematichardening';
@MaterialProperty<constant> 'YoungModulus' 150e9;
@MaterialProperty<constant> 'PoissonRatio' 0.3;

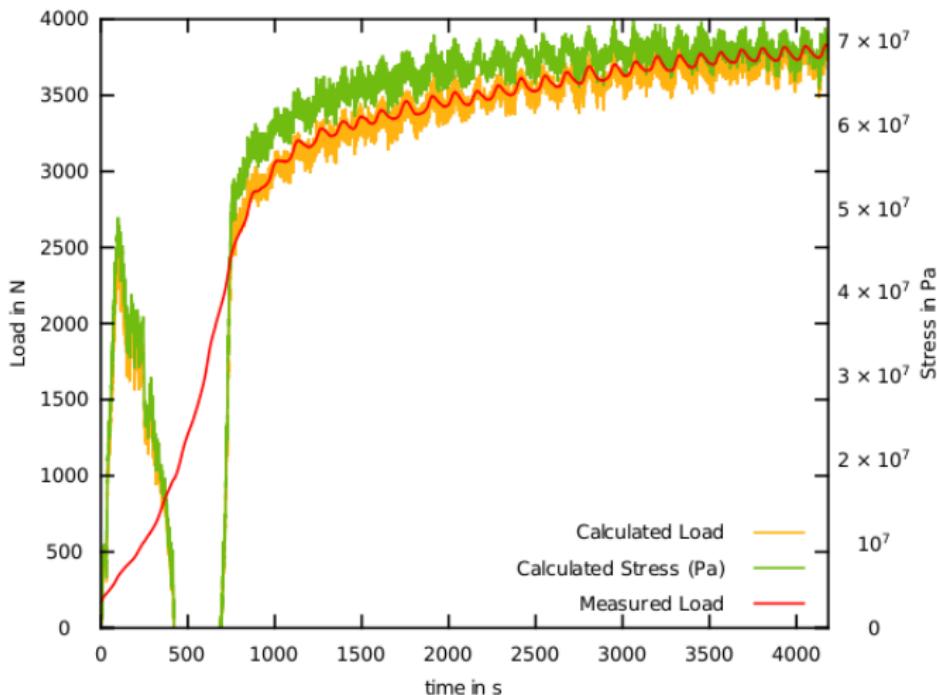
// Calculation of test 868_P4
@Real 'alpha' '20.882*1.e-6/60.'; //alpha is the constant displacement rate
@Real 'h0' '16.520*1e-3'; //h0 is the height of the pellet at the beginning of test
@Real 'timemax' 8000.; //timemax is the duration of test

@ExternalStateVariable 'Temperature' 1773.15;
@ExternalStateVariable 'Porosity' 0.02;

@ImposedDeformationGradient<data> 'FRR' 'charg_868_P4.prn' using '$1*60.':'1.-$3*1.e-6/h0';

@Times<data> 'charg_868_P4.prn' using '$1*60';
```

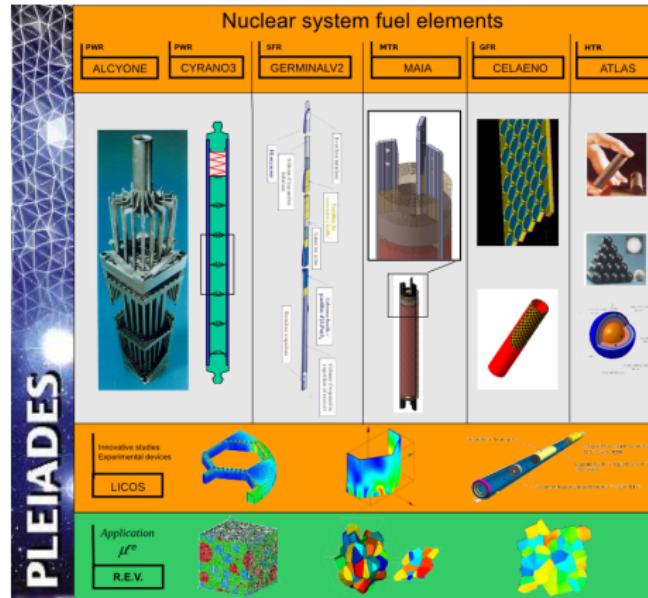
Identification of UO_2 viscoplastic behaviour : first results



■ Norton behaviour with non-linear kinematic hardening.

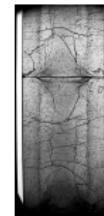
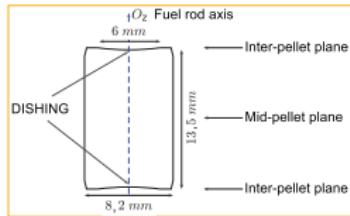
Material knowledge management in nuclear simulations

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.

The PWR fuel element

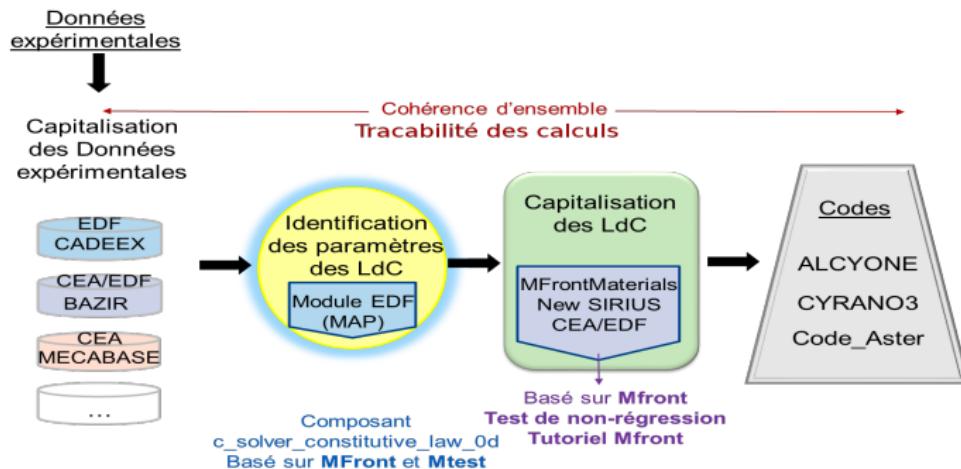


- Brittle fracture.
- Porosity growth.
- Viscoplastic behaviour.

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**

Current situation and perspectives



- **Rationalisation and Standardisation is required**, in the Nuclear industry (EDF, AREVA, CEA) :
 - Each partner has built its own strategy.
 - **A working group** has recently been set up to tackle this issue.
- However, we are willing to share this effort with the whole (at least french) community of the non-linear mechanics.

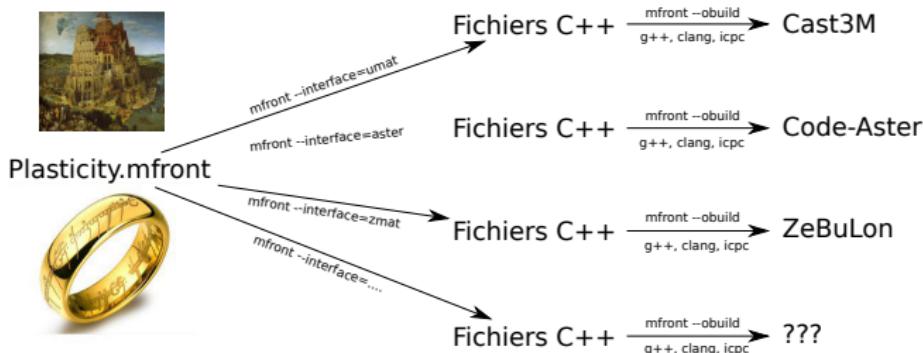
The TFEL project and the MFront code generator

Definitions

- **State variables** : the (thermodynamical) state of a material is assumed to be fully characterized by a set of state variables defined at each point of the material.
- **Model** : description of how the state variables of a set of materials evolves due to one or several physical phenomena :
 - Point-wise models
 - Equilibrium based models (heat transfer, mechanics, etc...)
- **Behaviours** : in equilibrium based models, a specific material is described by a behaviour which relates a gradient to the associated flux and makes the state variables evolves.
- **Material properties** : functions of the current state variables of the material used by some generic models or behaviours to describe a specific material.

- **State variables** : the (thermodynamical) state of a material is assumed to be fully characterized by a set of state variables defined at each point of the material.
- **Model** : description of how the state variables of a set of materials evolves due to one or several physical phenomena :
 - Point-wise models
 - Equilibrium based models (heat transfer, mechanics, etc...)
- **Behaviours** : in equilibrium based models, a specific material is described by a behaviour which relates a gradient to the associated flux and makes the state variables evolves.
- **Material properties** : functions of the current state variables of the material used by some generic models or behaviours to describe a specific material.

MFront goals



- TFEL is a project various libraries/softwares, including MFront.
- MFront is a code generation tool dedicated to material knowledge (material properties, mechanical behaviours, point-wise models) with focus on :
 - Numerical efficiency (see various benchmarks).
 - Portability (Cast3M, Cyrano, Code_Aster, Europlexus, TMFTT, AMITEX_FFTP, Abaqus, CalculiX, MTest).
 - Ease of use : *Longum iter est per pracepta, breve et efficax per exempla* (It's a long way by the rules, but short and efficient with examples).

Ease of use



mcc
@mcclure111

Suivre



In C++ we don't say "Missing asterisk" we say "error C2664: 'void std::vector<block,std::allocator<_Ty>>::push _back(const block &)' cannot convert argument 1 from 'std::_Vector_iterator<std::_Vector_val<std::_Simple_types<block>>>' to 'block &&'" and i think that's beautiful

 Traduire le Tweet

13:30 - 1 juin 2018

- Hopefully, only a very small subset of C/C++ is required to implement very complex behaviours.

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-developed by CEA and ÉDF.
 - Implementing an interface to ZMAT.
- 2015 : Officially part of Code_Aster/Salomé-Méca.
- 2016 : TFEL-3.0 is released :
 - Support of Europlexus, Abaqus/Standard, Abaqus/Explicit
 - StandardElasticity brick
 - ≈ 250 000 lines of codes
- 2017 : TFEL-3.1 is released :
 - Experimental support of Ansys
- 2018 : TFEL-3.2 will be released :
 - Full support of Ansys and LS-DYNA, generic 'MFront' behaviour.
 - The StandardElastoViscoplasticity brick

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 developed 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 platform.

- TFEL 3.x is based on the C++ 11 standard.
- TFEL 3.x can be compiled with various C++ compilers :
 - gcc, from version 4.7 to version 8.1
 - clang, from version 3.5 to version 5.0
 - icc, version 2016, 2017 and 2018.
 - Visual Studio, version 15 and 17.
- TFEL is mainly developed on LiNuX.
- ports have been made to various POSIX systems, Mac Os, FreeBSD, OpenSolaris, cygwin, Windows Subsystem for LiNuX, Haiku etc... and Windows !

- Very stringent compilers warnings :

- g++ -Wall -Wextra -pedantic
 - Wdisabled-optimization
 - Wlong-long
 - Winline
 - Wswitch
 - Wsequence-point
 - Wignored-qualifiers
 - Wzero-as-null-pointer-constant
 - Wvector-operation-performance
 - Wtrampolines
 - Wstrict-null-sentinel
 - Wsign-promo
 - Wsign-conversion
 - Wold-style-cast
 - Wnoexcept
 - Wmissing/include-dirs
 - Wmissing-declarations
 - Wlogical-op
 - Winit-self
 - ...

- Documentation.

- Continuous integration based on Jenkins

- More than 10 000 tests :

- Most of them are based on mtest

- More tests inside PLEIADES applications.

- mtest :
 - material point simulation
 - finite strain 1D pipe simulation
- MFrontGallery :
 - <https://github.com/thelfer/MFrontGallery>
 - A project managing compilation of MFront files and execution of units-tests.
- MAP identification tools :
 - not (yet) released as open-source.
- TPlot :
 - <https://github.com/thelfer/TPlot>
 - a simple Qt based tool to draw 2D graphs.
- QEmacs :
 - <https://github.com/thelfer/QEmacs>
 - a simple texteditor with special support for MFront and mtest

News

Overview

Getting started

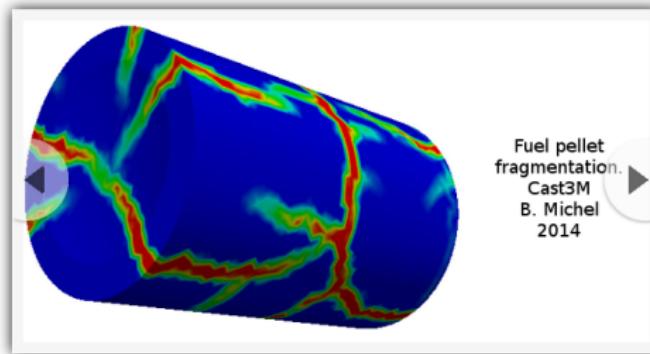
Documentation

Contributing

Getting Help



MFront: a code generation tool dedicated to material knowledge



<http://tfel.sourceforge.net>

<http://tfel.sourceforge.net/about.html>

<http://tfel.sourceforge.net/gallery.html>

Mechanical behaviours

- Mechanical equilibrium : find $\Delta \vec{U}$ such as :

$$\vec{R}(\Delta \vec{U}) = \vec{0} \quad \text{avec} \quad \vec{R}(\Delta \vec{U}) = \vec{F}_i(\Delta \vec{U}) - \vec{F}_e$$

- Mechanical equilibrium : find $\Delta \vec{U}$ such as :

$$\vec{R}(\Delta \vec{U}) = \vec{0} \quad \text{avec} \quad \vec{R}(\Delta \vec{U}) = \vec{F}_i(\Delta \vec{U}) - \vec{F}_e$$

- element contribution to inner forces :

$$\begin{aligned}\vec{F}_i^e &= \int_{V^e} \underline{\sigma}_{t+\Delta t}(\Delta \underline{\varepsilon}^{to}, \Delta t) : \underline{\mathbf{B}} dV \\ &= \sum_{i=1}^{N^G} (\underline{\sigma}_{t+\Delta t}(\Delta \underline{\varepsilon}^{to}(\vec{\eta}_i), \Delta t) : \underline{\mathbf{B}}(\vec{\eta}_i)) w_i\end{aligned}$$

where \mathbf{B} gives the relationship between $\Delta \underline{\varepsilon}^{to}$ and $\Delta \vec{U}$

- Mechanical equilibrium : find $\Delta \vec{U}$ such as :

$$\vec{\mathbb{R}}(\Delta \vec{U}) = \vec{0} \quad \text{avec} \quad \vec{\mathbb{R}}(\Delta \vec{U}) = \vec{\mathbb{F}}_i(\Delta \vec{U}) - \vec{\mathbb{F}}_e$$

- element contribution to inner forces :

$$\vec{\mathbb{F}}_i^e = \sum_{i=1}^{N^G} (\underline{\sigma}_{t+\Delta t}(\Delta \underline{\epsilon}^{to}(\vec{\eta}_i), \Delta t) : \underline{\underline{\mathbf{B}}}(\vec{\eta}_i)) w_i$$

- Resolution using the Newton-Raphson algorithm :

$$\Delta \vec{U}^{n+1} = \Delta \vec{U}^n - \left(\frac{\partial \vec{\mathbb{R}}}{\partial \Delta \vec{U}} \Big|_{\Delta \vec{U}^n} \right)^{-1} \cdot \vec{\mathbb{R}}(\Delta \vec{U}^n) = \Delta \vec{U}^n - \underline{\underline{\mathbb{K}}}^{-1} \cdot \vec{\mathbb{R}}(\Delta \vec{U}^n)$$

- Mechanical equilibrium : find $\Delta \vec{U}$ such as :

$$\vec{R}(\Delta \vec{U}) = \vec{0} \quad \text{avec} \quad \vec{R}(\Delta \vec{U}) = \vec{F}_i(\Delta \vec{U}) - \vec{F}_e$$

- element contribution to inner forces :

$$\vec{F}_i^e = \sum_{i=1}^{N^G} (\underline{\sigma}_{t+\Delta t}(\Delta \underline{\epsilon}^{to}(\vec{\eta}_i), \Delta t) : \underline{\underline{B}}(\vec{\eta}_i)) w_i$$

- Resolution using the Newton-Raphson algorithm :

$$\Delta \vec{U}^{n+1} = \Delta \vec{U}^n - \underline{\underline{\mathbb{K}}}^{-1} \cdot \vec{R}(\Delta \vec{U}^n)$$

- element contribution to the stiffness :

$$\underline{\underline{\mathbb{K}}}^e = \sum_{i=1}^{N^G} t \underline{\underline{B}}(\vec{\eta}_i) : \frac{\partial \Delta \underline{\sigma}}{\partial \Delta \underline{\epsilon}^{to}}(\vec{\eta}_i) : \underline{\underline{B}}(\vec{\eta}_i) w_i$$

$\frac{\partial \Delta \underline{\sigma}}{\partial \Delta \underline{\epsilon}^{to}}$ is the **consistent tangent operator**

Main functions of the mechanical behaviour

$$\left(\underline{\epsilon}^{to}|_t, \vec{Y}|_t, \Delta \underline{\epsilon}^{to}, \Delta t \right) \xrightarrow[\text{behaviour}]{} \left(\underline{\sigma}|_{t+\Delta t}, \vec{Y}|_{t+\Delta t}, \frac{\partial \Delta \underline{\sigma}}{\partial \Delta \underline{\epsilon}^{to}} \right)$$

- Given a strain increment $\Delta \underline{\epsilon}^{to}$ over a time step Δt , the mechanical behaviour must compute :
 - The value of the stress $\underline{\sigma}|_{t+\Delta t}$ at the end of the time step.
 - The value of internal state variables, noted $\vec{Y}|_{t+\Delta t}$ at the end of the time step.
 - The consistent tangent operator : $\frac{\partial \Delta \underline{\sigma}}{\partial \Delta \underline{\epsilon}^{to}}$
- For specific cases, the mechanical behaviour shall also provide :
 - a prediction operator
 - the elastic operator (Abaqus-Explicit, Europlexus)
 - estimation of the stored and dissipated energies (Abaqus-Explicit)

An very simple example

```

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

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

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

@Integrator{
    const real eps = 1.e-12;
    const auto mu = computeMu(young,nu);
    const auto seq = sigmaeq(sig);
    const auto tmp = A*pow(seq,E-1.);
    const auto df_dseq = E*tmp;
    const auto iseq = 1/max(seq,eps*young);
    const Stensor n = 3*deviator(sig)*iseq/2;
    // implicit system
    // the StandardElasticity already set feel to deel-det
    feel += dp*n;
    // by default, fp is initialized to dp
    fp -= tmp*seq*dt;
    // jacobian
    dfeel_ddeel += 2*mu*theta*dp*iseq*(Stensor4::M()-(n^n));
    dfeel_ddp   = n;
    dfp_ddeel  = -2*mu*theta*df_dseq*dt*n;
} // end of @Integrator

```

- Implicit integration.
- Implicit system :

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

- Jacobian :

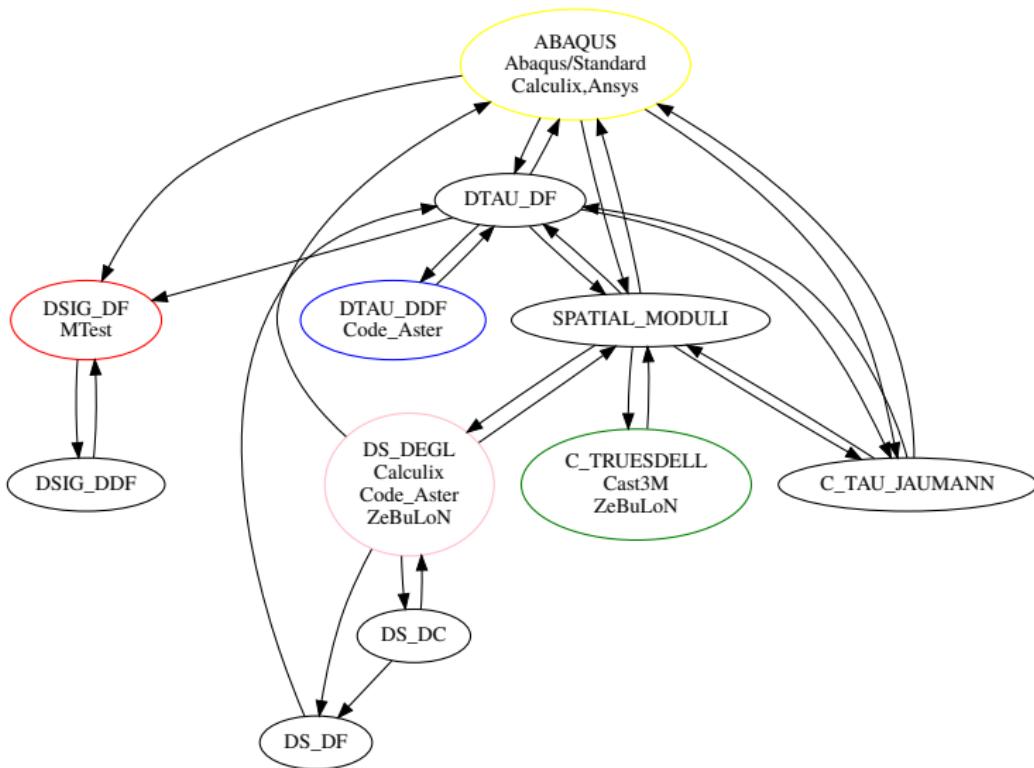
$$\begin{cases} \frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta \underline{\epsilon}^{el}} = \underline{I} + \frac{2 \mu \theta \Delta p}{\sigma_{eq}} (\underline{M} - \underline{n} \otimes \underline{n}) \\ \frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta p} = \underline{n}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

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

Other functions of the mechanical behaviour

- Provide a estimation of the next time step for time step automatic adaptation
- Check bounds :
 - Physical bounds
 - Standard bounds
- Clear error messages
- Parameters
 - It is all about AQ !
 - Parametric studies, identification, etc...
- Generate 'MTest' files on integration failures
- Generated example of usage :
 - Generation of MODELISER/MATERIAU instructions (Cast3M)
 - Input file for Abaqus, Ansys
- Provide information for dynamic resolution of inputs (MTest/Aster/Europlexus) :
 - Numbers Types (scalar, tensors, symmetric tensors)
 - Entry names /Glossary names...

Example of portability issue : tangent operator for finite strain behaviours



Performances

- Expression templates :

- `const auto a=b+c;`
 - `a` is the operation of adding `b` and `c`.
 - the operation is only performed when `a` is assigned to a concrete object (lazy evaluation).
 - `a` **shall be eliminated** by the optimizer.

- Loop unrolling.

- Concepts and partial specialisations.

- Views.

- Compile-time dimensional analysis.

- Expression templates.

- Loop unrolling :

- Consider `const stensor<1u> a=b+c+2*d;`.
 - This operation is equivalent to :

```
a[0]=b[0]+c[0]+2*d[0];  
a[1]=b[1]+c[1]+2*d[1];  
a[2]=b[2]+c[2]+2*d[2];
```

- This requires to compile a specialisation of the behaviour several times for every supported modelling hypotheses.

- Concepts and partial specialisations.

- Views.

- Compile-time dimensional analysis.

Some optimisation techniques in TFEL/Math

- Expression templates.
- Loop unrolling.
- Concepts and partial specialisations :

```

template<-typename T,typename T2>
typename std::enable_if_c
(( tfel :: meta::Implements<T,StensorConcept>::cond) && (StensorTraits<T>::dime==3u)&&
 ( tfel :: meta::Implements<T2,StensorConcept>::cond) && (StensorTraits<T2>::dime==3u)&&
 ( tfel :: typetraits :: IsFundamentalNumericType<StensorNumType<T2>>::cond),
stensor<3u,StensorNumType<T2>>
>::type
convertCorotationalCauchyStressToSecondPiolaKirchhoffStress(const T& s, const T2& U)
using real = tfel :: typetraits :: base_type<StensorNumType<T2>>;
constexpr real cst = Cstcreal::sqr(2);
const auto J = det(U);
const auto IU = invert(U);
return {J*(s[2]*|U[4]|*|U[4]|+(cste*s[5]*|U[3]|*2*s[4]*|U[0]|*|U[4]|*s[1]*|U[3]|*|U[3]|*2*s[3]*|U[0]|*|U[3]|*2*s[0]*|U[0]|*|U[0]|)/2,
J*(s[2]*|U[5]|*|U[5]|*(cste*s[4]*|U[3]|*2*s[5]*|U[1]|*|U[5]|*s[0]*|U[3]|*|U[3]|*2*s[3]*|U[1]|*|U[3]|*2*s[1]*|U[1]|*|U[1]|)/2,
J*(s[1]*|U[5]|*|U[5]|*(cste*s[3]*|U[4]|*2*s[5]*|U[2]|*|U[5]|*s[0]*|U[4]|*|U[4]|*2*s[4]*|U[2]|*|U[4]|*2*s[2]*|U[2]|*|U[2]|)/2,
J*(cste*s[2]*|U[4]|*s[5]*|U[3]|*cste*s[4]*|U[0]|*|U[5]|*s[4]*|U[3]|*cste*s[5]*|U[1]|*|U[4]|*s[3]*|U[3]|*|U[3]|*2*s[1]*|U[1]|*2*s[0]*|U[0]|*|U[3]|*2*s[3]*|U[0]|*|U[1]|)/2,
J*((s[5]*|U[4]|*cste*s[1]*|U[3]|*cste*s[3]*|U[0]|*|U[5]|*s[4]*|U[4]|*|U[4]|*s[3]*|U[3]|*2*s[2]*|U[2]|*2*s[0]*|U[0]|*|U[4]|*cste*s[5]*|U[2]|*|U[3]|*2*s[4]*|U[0]|*|U[2]|)/2,
J*(s[5]*|U[5]|*|U[5]|*(s[4]*|U[4]|*s[3]*|U[3]|*2*s[2]*|U[2]|*2*s[1]*|U[1]|*|U[5]|*(cste*s[0]*|U[3]|*cste*s[3]*|U[1]|*|U[4]|*cste*s[4]*|U[2]|*|U[3]|*2*s[5]*|U[1]|*|U[2]|)/2;
}

```

- Views.
- Compile-time dimensional analysis.

- Expression templates.
- Loop unrolling.
- Concepts and partial specialisations.
- Views :

$$\text{— } J = \begin{pmatrix} \frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta \underline{\epsilon}^{el}} & \frac{\partial f_{\underline{\epsilon}^{el}}}{\partial \Delta p} \\ \frac{\partial f_p}{\partial \Delta \underline{\epsilon}^{el}} & \frac{\partial f_p}{\partial \Delta p} \end{pmatrix}$$

- dfeel_ddeel is view of J :
 - Implementing the ST2toST2Concept concept.
 - Direct modification of J .
 - Compile-time computation of the offset.

- Compile-time dimensional analysis.

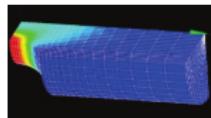
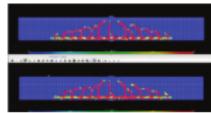
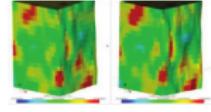
Some optimisation techniques in TFEL/Math

- Expression templates.
- Loop unrolling.
- Concepts and partial specialisations.
- Views.
- Compile-time dimensional analysis :
 - Adding a strain and a stress tensor leads to a **compile-time** error.
 - Available in **TFEL** for years.
 - Not yet available in **MFront**.
 - Feature planned for **TFEL 3.1**.

- Expression templates.
- Loop unrolling.
- Concepts and partial specialisations.
- Views.
- Compile-time dimensional analysis.
- **Those techniques far exceed the programming skills of most C++ professional programmers.**
 - We have anticipated the C++ language evolutions.
 - Some parts of TFEL are deprecated and removed when an equivalent features are introduced in the standard.
 - Concepts will have build-in language support in C++ -20.
 - It explains the great gap between TFEL-2.0 and TFEL-3.0, but « old » MFront files are unaffected ⇒ transparent for the end users.

- Expression templates.
- Loop unrolling.
- Concepts and partial specialisations.
- Views.
- Compile-time dimensional analysis.
- **Those techniques far exceed the programming skills of most C++ professional programmers.**
 - We have anticipated the C++ language evolutions.
 - Some parts of TFEL are deprecated and removed when an equivalent features are introduced in the standard.
 - Concepts will have build-in language support in C++ -20.
 - It explains the great gap between TFEL-2.0 and TFEL-3.0, but « old »MFront files are unaffected ⇒ transparent for the end users.
- **A code generator is required for the library to be used by "standard" engineers ⇒ MFront.**

Benchmarks with Code_Aster

Description	Algorithme	Temps CPU total (Aster vs MFront)	Illustration
Visco-plastic and damaging for steel (Mustata and Hayhurst 2005; EDF 2012)	Implémenté	17mn43s vs 7mn58s	
Damaging for concrete (Mazars and Hamon; EDF 2013a)	Default	45mn vs 63mn	
Generic Single crystal viscoplasticity (Méric and Cailletaud 1991; EDF 2013b)	Implémenté	28mn vs 24mn	

- Benchmarks made in 2013 by the Code_Aster developpers when evaluating MFront.
- Calling external behaviours in Code_Aster has improved...

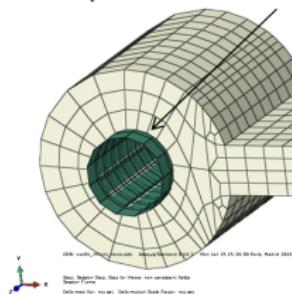
Benchmarks with Code_Aster

Description	Algorithme	Temps CPU	Illustration
FCC single crystal viscoplasticity (Monnet, Naamane, and Devincre 2011 EDF (2013b))	Inéplasticité	33m54s vs 29m30s	
FCC homogenized polycrystals 30 grains (Berveiller and Zaoui 1978; EDF 2013b)	Runge-Kutta 4/5	9s67 vs 8s22	
Anisotropic creep with phase transformation (EDF 2013c)	Inéplasticité	180s vs 171s	

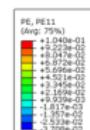
- Benchmarks made in 2013 by the Code_Aster developpers when evaluating MFront.
- Calling external behaviours in Code_Aster has improved...

Benchmarks with Abaqus/Standard

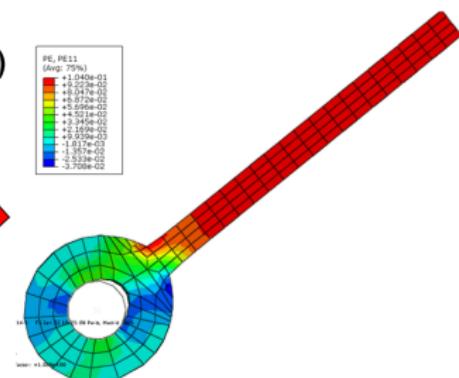
Contact with a rigid axis
(free rotation around z)



Pressure
(-500 MPa)



Imposed displacement
on this edge
(4 mm along y)



	CPU Time (s)	Difference	Number of increments	Difference
ABAQUS	192	-	52	-
ABAQUS/UMAT(*)	1200	525%	295	467%
ABAQUS/MFRONT	201	5%	52	0%

(*) Numerical Jacobian

MFRONT seems to be much more efficient
than a badly implemented UMAT!!!

Conclusions

On going efforts

- MFront is a building block for the material knowledge management strategy of EDF and the PLEIADES plateform :
 - Focused of software quality and numerical performances.
- There are ongoing efforts on defining open tools to ensure a fully-consistent strategy encompassing all the steps up to engineering studies :
 - Experimental data.
 - Identification procedures.
 - Behaviour implementations.
 - Unit testing.
 - Documentation.

Future works

- Documentation :
 - Better code documentation.
 - New entries in the gallery.
- Numerical stability tests based on the CADNA library :
 - www.lip6.fr/cadna
 - <https://github.com/thelfer/cadna>
- Support of units :
 - Compile-time checks without any runtime overhead.
- More work on implicit algorithms :
 - Trust-region algorithms.
 - Homotopy methods.
- Generalised behaviours.
- New interfaces (ANSYS,...).
- An official Debian/Ubuntu package.
- **The fourth MFront user days !**

A huge technical depth





How to contribute

[News](#) [Overview](#) [Getting started](#) [Documentation](#) [Contributing](#) [Getting Help](#)



Contributors

- Thomas Heffer
- Jean-Michel Proix
- Bruno Michel
- Jérémie Hure
- Chao Ling
- Nicolas Selenet
- Eric Brunon
- François Hamon
- Benoît Bary
- Nicolas Selenet
- Arnaud Courcelle
- Victor Blanc
- Jérôme Julien
- Olivier Fandeur
- Sébastien Melin
- Thierry Thomas
- Alexis Foerster
- Alexandre Lemaire
- Dominique Delaison
- Kubir Singh
- Christian Fokam

About

- 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://www.openhub.net/p/tfel>

[https://github.com/thelfer/
tfel-contact@cea.fr](https://github.com/thelfer/tfel-contact@cea.fr)