DE LA RECHERCHE À L'INDUSTRIE

# On the road to TFEL 3.*x*

— Thomas Helfer

Februar 2015

www.cea.fr

# A changing development model

# Objectives and financial support

- Up to now, `MFront` was developed to meet PLEIADES project needs :
  - clear financial support
  - clear objectives : fuel performances codes
- With open-source, things may change :
  - "open-sourcisation" was funded by EDF and CEA :
    - for 2015, support is limited to the web site maintenance and documentation (ex : mtest documentation)
  - interest of a broader community :
    - external contributions are welcomed !
    - search of additional financial support

# Planned new mfront features

# Missing features

- tangent consistent operator for `MultipleIsotropicMisesCreep`;
- plane stress (and generalised plane stress) support for specific behaviours

**Interesting for the Cyrano fuel performances codes : planned for 2015/2016**

# Various improvements

- the mechanical behaviour shall (to be developed) :
  - give an estimate of the next time step
  - warn if the given strain increment is too large
  - warn if the given strain increment leads to unreliable results :
    - an increment of 10 % of the equivalent plastic strain is **not** admissible whatever the integration scheme is used
- consistent tangent operators in `Cast3M`/`PLEIADES`

**Depends on the evolution of the PLEIADES/COPL operator (2015/2016)**

# Interesting evolutions of MFront

# Port to C++11

- TFEL heavily relies on advanced programming techniques :
  - this is a **serious** maintainability issue
- Most of those techniques are now part of the C++11 standard
  - this was anticipated ! !
  - we reduced the code size by more than 10 % only by removing some header and using sed !
  - generated code is more than 11 % smaller !
- There could be a performance win

# New algorithms

- the "true" Powell dogleg algorithm
- the Brent method (scalar non linear equation)
- introduce other trust region update algorithm :
    - in particular non monotonic ones

# Bricks

```
@DSL Implicit;
@Behaviour OrthotropicElasticity;

@OrthotropicBehaviour;
@RequireStiffnessTensor;
@Brick "StandardElasticity";
```

- making a portable behaviour requires *too much of boiler plate* code :
  - ≈ 50% of the `ImplicitNorton` example
  - most of them are related to elasticity !
- most behaviours are build by adding (uncoupled) plastic/viscoplastic flows and a (damaged) elasticity tensor
- We want to introduce *mechanical behaviour bricks* in `MFront` :
  - must be compatible/consistent with current `MFront` practices
    - some parts of the behaviour may come from bricks, other may come from standard user code
    - one must take care of evil details : variation of material properties with temperature, etc..
    - requires a dependency manager à la `licos`
  - one may provide robust and optimised implementations
  - we already have an experimental `Elasticity` brick

# Potential

- integration of some common mechanical behaviours can be written as a minimisation of this potential :

$$\rho\phi\left(\left.\vec{Y}\right|_{t+\Delta\,t},\underline{\epsilon}^{to}\right)+\Delta\,t\,D\left(\frac{\left.\vec{Y}\right|_{t+\Delta\,t}-\left.\vec{Y}\right|_{t}}{\Delta\,t}\right)$$

- formalism used in `matlib` ?
- could probably be extended to implicit standard materials (bipotential)...
- only requires on function definition :
  - (smart)-automatic differentiation ?