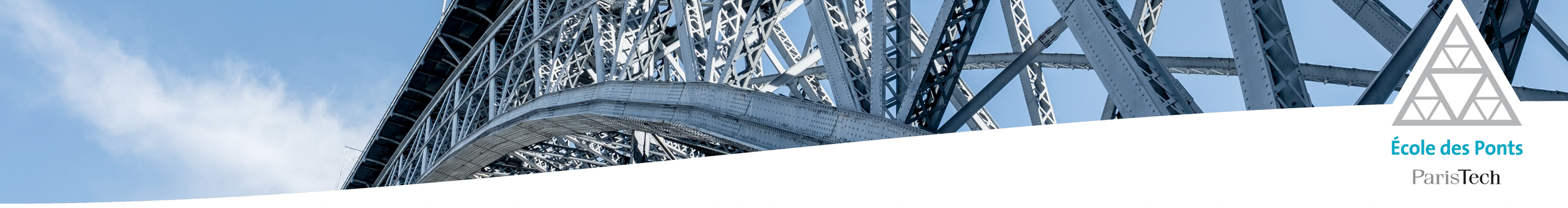


Combining FEniCS and MFront Code Generation Tools for Nonlinear Mechanics

Jérémy Bleyer[a], Thomas Helfer[b]

[a] *Laboratoire Navier, ENPC, Univ Gustave Eiffel, CNRS, Marne-La-Vallée, France*
[b] *CEA, DES, IRESNE, DEC, SESC, LSC, Cadarache, France*

Binding two code-generation tools

FEniCS : a code-generation software package for automated solution of **partial differential equations**

$$\int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} \, \mathrm{dx} - \int_\Omega p \, \mathrm{div}\, \boldsymbol{v} \, \mathrm{dx} + \int_\Omega q \, \mathrm{div}\, \boldsymbol{u} \, \mathrm{dx} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, \mathrm{dx}$$

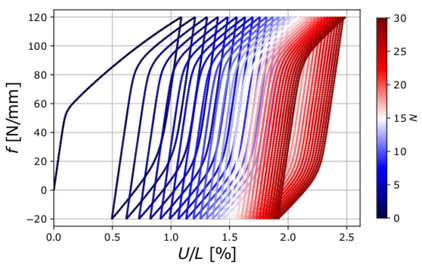
MFront : a code-generation tool for complex **material constitutive laws**

$$\Delta\varepsilon, \boldsymbol{\sigma}_n, \boldsymbol{Y}_n \rightarrow \boxed{\text{MFront}} \rightarrow \boldsymbol{\sigma}_{n+1}, \boldsymbol{Y}_{n+1}, \frac{\partial \boldsymbol{\sigma}}{\partial \Delta\varepsilon}$$

Non-linear material constitutive laws

e.g. elasto-viscoplasticity, damage, ductile fracture
usually formulated as a set of **non-linear equations** at the **material point level** =>
implicit stress/strain relation,
history-dependent

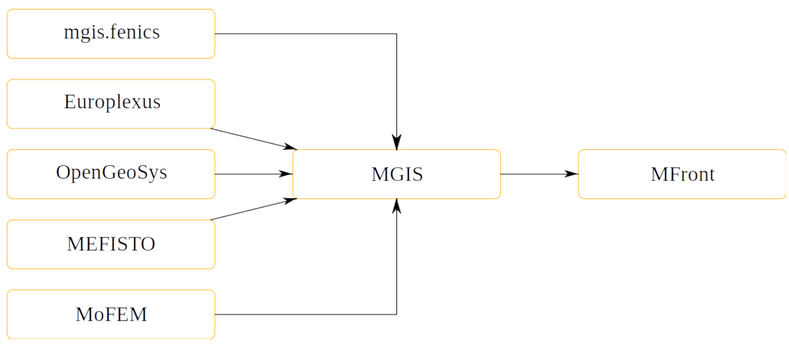
Implementation should also return the **tangent operator**



A Python package in the MGIS project

MFront Generic Interface Support project

A generic interface between Mfront and (FE, FFT...) solvers
Bindings : C, Fortan, Python, Julia



within FeniCS

$$F(\boldsymbol{v}) = \int_\Omega \boldsymbol{\sigma}(\boldsymbol{u}) \cdot \delta \boldsymbol{g}(\boldsymbol{v}) \, \mathrm{dx} - L(\boldsymbol{v}) = 0 \quad \forall \boldsymbol{v} \in V$$

$$a_{\text{tangent}}(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \delta \boldsymbol{g}(\boldsymbol{v}) \cdot \mathbb{T} \cdot \delta \boldsymbol{g}^j(\boldsymbol{u}) \, \mathrm{dx}$$

Stress and **tangent blocks** represented by “Quadrature” functions => not optimal but not intrusive

The only metadata **not provided** by MGIS is how the gradients (e.g. strain) relate to the unknown fields (e.g. displacement)

The user is required
to provide this link using
UFL expressions (**registration**)

```
mat_prop = {"YoungModulus": E, "PoissonRatio": nu, "Hardening":H, "YieldStrength": sig0}
material = mf.MFrontNonlinearMaterial("src/libBehaviour.so",
                "IsotropicLinearHardeningPlasticity", hypothesis="plane_strain",
                 material_properties=mat_prop)
u = Function(V, name="Displacement")
problem = mf.MFrontNonlinearProblem(u, material, quadrature_degree=2, bcs=bc)
problem.register_gradient("Strain", sym(grad(u)))
```

Monolithic thermo-elastoplasticity

Generalized behaviour

displacement + temperature

```
@DSL DefaultGenericBehaviour;
@Behaviour HeatTransferPhaseChange;

@Gradient TemperatureGradient ∇T;
∇T.setGlossaryName("TemperatureGradient");

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

@StateVariable real h;
h.setEntryName("Enthalpy"); //per unit of volume

@AdditionalTangentOperatorBlock ∂j/∂ΔT;
@AdditionalTangentOperatorBlock ∂h/∂ΔT;
```

$$\varepsilon^{\text{el}} = \varepsilon^{\text{to}} - \varepsilon^{\text{p}} - \varepsilon^{\text{th}}$$
$$\boldsymbol{\sigma} = \lambda \, \mathrm{tr}(\varepsilon^{\text{el}}) \boldsymbol{I} + 2\mu\varepsilon^{\text{el}}$$
$$s = \frac{C_\varepsilon}{T^{\text{ref}}}(T - T^{\text{ref}}) + \frac{\kappa}{\rho}\mathrm{tr}(\varepsilon^{\text{el}})$$
$$\boldsymbol{j} = -k \nabla T$$

$$\int_\Omega \boldsymbol{\sigma}_{n+1} : \nabla^s \widehat{\boldsymbol{u}} \, \mathrm{dx} = \int_{\partial\Omega} \boldsymbol{T} \cdot \widehat{\boldsymbol{u}} \, \mathrm{dS} \quad \forall \widehat{\boldsymbol{u}}$$
$$\int_\Omega (\rho T^{\text{ref}} \frac{s_{n+1} - s_n}{\Delta t} - \boldsymbol{j}_{n+1}) \widehat{T} \, \mathrm{dx} = -\int_{\partial\Omega} q \widehat{T} \, \mathrm{dS} \quad \forall \widehat{T}$$

```
Vue = VectorElement("CG", mesh.ufl_cell(), 2)  # displacement finite element
Vte = FiniteElement("CG", mesh.ufl_cell(), 1)  # temperature finite element
V = FunctionSpace(mesh, MixedElement([Vue, Vte]))

v = Function(V)
(u, Theta) = split(v)
problem = mf.MFrontNonlinearProblem(v, material, quadrature_degree=2, bcs=bcs)
problem.register_gradient("Strain", sym(grad(u)))
problem.register_gradient("TemperatureGradient", grad(Theta))
problem.register_external_state_variable("Temperature", Theta + Tref)
```

```
sig = problem.get_flux("Stress")
j = problem.get_flux("HeatFlux")
s = problem.get_state_variable("EntropyPerUnitOfMass")
problem.initialize()

s_old = s.copy(deepcopy=True)
v_ = TestFunction(V)
u_, T_ = split(v_)  # Displacement and temperature test functions
eps_ = problem.gradients["Strain"].variation(v_)
mech_residual = dot(sig, eps_)*problem.dx
thermal_residual = (rho*Tref*(s - s_old)/dt*T_ - dot(j, grad(T_)))*problem.dx
problem.residual = mech_residual + thermal_residual
problem.compute_tangent_form()
```

Logarithmic strain plasticity

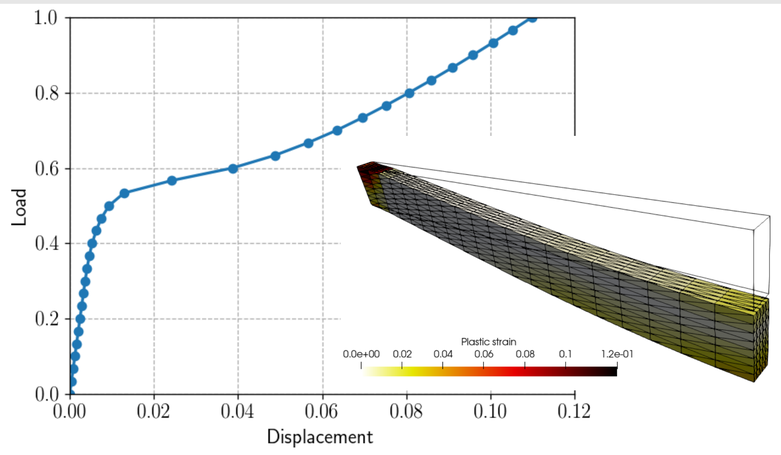
Hencky strain measure
+ elasto-plastic behaviour

$$\boldsymbol{H} = \frac{1}{2}\log(\boldsymbol{F}^{\mathsf{T}} \cdot \boldsymbol{F})$$

```
@DSL Implicit;

@Behaviour LogarithmicStrainPlasticity;

@StrainMeasure Hencky;

@Brick StandardElastoViscoPlasticity{
  stress_potential : "Hooke" {
        young_modulus : 210e9,
        poisson_ratio : 0.3
        },
  inelastic_flow : "Plastic" {
    criterion : "Mises",
    isotropic_hardening : "Linear" {H : 500e6,
                                    R0 : 250e6}
  }
};
```

$$\boldsymbol{H} = \boldsymbol{H}^e + \boldsymbol{H}^p$$
$$\boldsymbol{T} = \mathbb{C} : \boldsymbol{H}^e$$
$$f(\boldsymbol{T}) = \|\boldsymbol{T}\| - (\sigma_0 + Hp) \leq 0$$
$$\dot{\boldsymbol{H}}^p = \dot{p}\frac{\boldsymbol{T}}{\|\boldsymbol{T}\|}, \quad \dot{p} \geq 0$$

```
material = mf.MFrontNonlinearMaterial("./src/libBehaviour.so", "LogarithmicPlasticity")
problem = mf.MFrontNonlinearProblem(u, material, bcs=bc)
problem.set_loading(dot(selfweight, u)*dx)
problem.register_gradient("DeformationGradient", Identity(3)+grad(u))
```



Conclusion

Working binding between FeniCS and MFront
Complex behaviours with very limited syntax, extension to generalized behaviours

Perspectives: better solution with `dolfin-x`, `ExternalOperator` ?

FEniCS : https://fenicsproject.org/
TFEL/MFront : http://tfel.sourceforge.net/
MGIS : https://thelfer.github.io/mgis/web/index.html