



# **The MFEM-MGIS project: coupling MFEM and MGIS for high performance nonlinear simulations**

DE LA RECHERCHE À L'INDUSTRIE

## MFfront User Meeting

21/10/2021

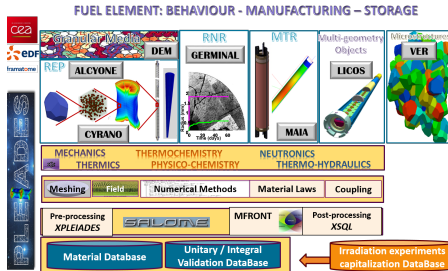
T. Helfer, G. Latu

- ▶ **Context and goals**
- ▶ **A small tutorial**
- ▶ **Design and features of the MFEM-MGIS project**
- ▶ **Feed-backs on some issues using MFEM**
- ▶ **Examples**
- ▶ **Conclusions and perspectives**

## Context and goals

- ▶ French Atomic Energy Commission (CEA), public institution.
- ▶ Our department: modelling and simulation of nuclear fuel.

The Pleiades Platform



Existing mechanical solver  
not well parallelized

- ▶ 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.

- ▶ Build a HPC general purpose non linear multi-physics library, *development began end of 2020*.
  - Primary focus is **non linear solid mechanics** and **heat transfer**.
  - Expected modelling (long-term): Shells, Beams, Phase-field approaches of brittle fracture, micromorphic models, Cosserat plasticity, strongly coupled thermo-chemical-mechanical or thermo-hydro-mechanical phenomena.
  
- ▶ A two-pillar library with *opensource* commitment (MFEM-MGIS *LGPL 3.0*):
  - MFEM: HPC finite element solver. (*LGPL 2.1*)
  - MGIS/MFront: constitutive laws, material modelling. (*LGPL 3.0 / GPL 3.0*)

## A small tutorial

# What are we talking about ?

## Example of end-user API

```
// loading the mesh and building the non linear problem
mfem_mgis::NonLinearEvolutionProblem problem(
    {"MeshFileName", mesh_file}, {"FiniteElementFamily", "H1"},
    {"FiniteElementOrder", order}, {"UnknownsSize", dim},
    {"Hypothesis", "PlaneStrain"}, {"Parallel", parallel});
// associating names to element and boundary attributes (could be automated for some mesh formats)
problem.setMaterialsNames({{1, "NotchedBeam"}});
problem.setBoundariesNames({{3, "LowerBoundary"}, {4, "SymmetryAxis"}, {2, "UpperBoundary"}});
// declaring behaviour integrators
problem.addBehaviourIntegrator("Mechanics", "NotchedBeam", library, behaviour);
// setting the initial state of the materials
auto& m1 = problem.getMaterial("NotchedBeam");
mgis::behaviour::setExternalStateVariable(m1.s0, "Temperature", 293.15);
...
// defining boundary conditions, postprocessings and solver parameters
problem.addUniformDirichletBoundaryCondition({"Boundary", "LowerBoundary"}, {"Component", 1}));
problem.addPostProcessing("ParaviewExportResults", {"OutputFileName", "ssna303-displacements"});
auto& solver = problem.getSolver();
...
// loop over time step
for (mfem_mgis::size_type i = 0; i != nsteps; ++i) {
    // updating the boundary values and resolution
    ...
    problem.solve(dt);
    problem.update();
    t += dt;
}
```

- Instantiating `NonLinearEvolutionProblem` class is the main entry point.
- **Behaviour integrator** is the main new concept of MFEM/MGIS .

# **Design and features of the MFEM-MGIS project**

Commissariat à l'énergie atomique et aux énergies alternatives - [www.cea.fr](http://www.cea.fr)

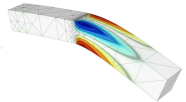
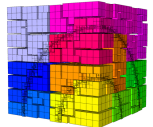


► Main features

- C++, opensource FEM library. Large community. <https://mfem.org/>.
- High-order finite element meshes and spaces.

► HPC oriented

- Scales to hundreds of thousands of cores.
- Many hardware devices are supported, such as GPUs.
- Minimal changes to switch from serial to parallel version.
- Top-notch optimizations for speeding up computations.
- Able to call many standard linear solvers (Hypre, PETSC,

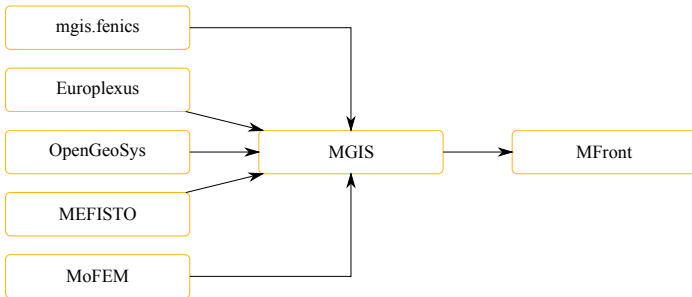


► Mesh features

- Triangular, quadrilateral, tetrahedral, wedge, and hexahedral elements.
- Periodic meshes.
- Conforming and nonconforming adaptive mesh refinement.
- Parallel load balancing (several methods).

► Non linear PDE's resolution

- handled by **non linear form integrators** which describes how to build the element contribution to the *residual* and its *jacobian*.  
Glue with MGIS is here.

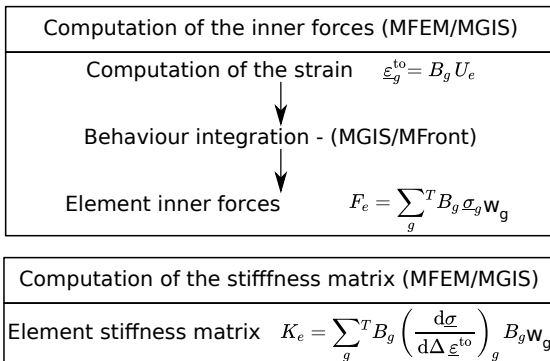


- The MGIS project provides classes on the solver side to retrieve **metadata** from an MFront behaviour and call the behaviour integration over a time step.
- Written in C++. Bindings exists for C, Fortran2003, python, Julia.
- Used/tested in `mgis.fenics`, `OpenGeoSys`, `Manta`, `XPer`, `MoFEM`, `Disk++`, `Kratos Multiphysics`, `JuliaFEM`, `NairnMPM`, `esys.escript`, `DUNE`, `00FEM`, and more...

- ▶ Statement:
  - *Poor* support of multiple materials in MFEM for solid mechanics.
  - *No* support for functions on integration points for a given material (element attribute).
- ▶ Proposed improvements:
  - Add support for non linear behaviour integrators based on MFront.
  - Add functions on integration points that depends on material identifier (element attribute).
  - Simplified High-level API for end users (mechanics)

- ▶ Statement:
  - *Poor* support of multiple materials in MFEM for solid mechanics.
  - *No* support for functions on integration points for a given material (element attribute).
- ▶ Proposed improvements:
  - Add support for non linear behaviour integrators based on MFront.
  - Add functions on integration points that depends on material identifier (element attribute).
  - Simplified High-level API for end users (mechanics)
- ▶ In practice:
  - MFEM-MGIS is a library made up of
    1. Classes that inherit from MFEM:  
loops on elements, linear & non-linear solvers, mesh management...
    2. Classes that inherit from MGIS: materials management, internal variables, fill matrix entries...
    3. Procedures for the end user to interact with MFEM & MGIS

- ▶ Statement:
  - *Poor* support of multiple materials in MFEM for solid mechanics.
  - *No* support for functions on integration points for a given material (element attribute).
- ▶ Proposed improvements:
  - Add support for non linear behaviour integrators based on MFront.
  - Add functions on integration points that depends on material identifier (element attribute).
  - Simplified High-level API for end users (mechanics)
- ▶ In practice:
  - MFEM-MGIS is a library made up of
    1. Classes that inherit from MFEM:  
loops on elements, linear & non-linear solvers, mesh management...
    2. Classes that inherit from MGIS: materials management, internal variables, fill matrix entries...
    3. Procedures for the end user to interact with MFEM & MGIS
- ▶ MFEM-MGIS is a C++17 opensource library:
  - <https://github.com/thelfer/mfem-mgis>
  - <https://github.com/latug0/mfem-mgis-examples>



- Behaviour integrators are associated with a material identifier (element attribute).
- Behaviour integrators are called in the assembly loop over the elements for:
  - the residual (contribution of the inner forces)
  - the jacobian matrix (i.e. the tangent stiffness matrix)

- ▶ Behaviour integrators are meant to:
  - Compute the gradients from unknowns (using the  $B$  matrix).
  - Handle the state variables.
  - Call the behaviour integration.
  - Compute the inner forces from the thermodynamic forces.
  - Compute the stiffness matrix from the consistent tangent operator blocks.
- ▶ All those steps depends on:
  - Kind of problem described (mechanics, heat transfer).
  - Symmetry of the material (isotropic or orthotropic).
  - Modelling hypotheses: 3D, plane strain, plane stress, axisymmetry ...
- ▶ The writing of behaviour integrators is tedious and error-prone and shall be done with care.
  - However, specific behaviour integrators gives access to pieces of physics.
  - Code generation to the rescue !

- ▶ Aim of this code-generator
  - Generate behaviour integrators from definition of the gradients.
  - Automated code factorisation/optimisation (no sparse matrix multiply).
  - Avoid coding errors due to tedious formula.
- ▶ Machinery
  - Based on the GiNaC for symbolic computations in C++
  - Current scope: isotropic and orthotropic, small and finite strain behaviours in plane strain, plane stress and tridimensional hypotheses.
- ▶ Extensions (to come)
  - Support of axisymmetry
  - Non linear heat transfer, non linear diffusion.
  - Can be extended to other non-linear models, wide range of phenomena.



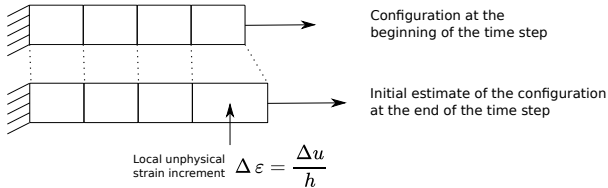
## Feed-backs on some issues using MFEM

# Issues related to the behaviour integration step

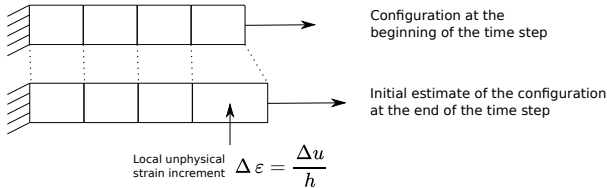
- ▶ The integration of the mechanical behaviour is a **complex** kernel solving a system of ordinary differential equations which can **fail**.
- ▶ Problem: the `NonlinearFormIntegrator::AssembleElementVector` and `NonlinearFormIntegrator::AssembleElementGrad` does not handle failures.

# Issues related to the behaviour integration step

- ▶ The integration of the mechanical behaviour is a **complex** kernel solving a system of ordinary differential equations which can **fail**.
- ▶ Problem: the `NonlinearFormIntegrator::AssembleElementVector` and `NonlinearFormIntegrator::AssembleElementGrad` does not handle failures.
- ▶ As a work-around, we derived our own Newton algorithm which separates the behaviour integration step from residual and stiffness assemblies.
  - This solution seems compatible with partial assembly (untested).
  - This work-around **does not work with** PETSc.
- ▶ More generally, we started a discussion on how to improve the robustness of operators in MFEM:
  - <https://github.com/mfem/mfem/issues/2139>



- Imposing the boundary Dirichlet in MFEM leads to unphysically strain increments on elements near the boundary at the first iteration:
  - The problem becomes more and more important as the mesh is refined.
  - In finite strain, this leads to severe divergence of the Newton algorithm due to the geometric stiffness matrix.



- Imposing the boundary Dirichlet in MFEM leads to unphysically strain increments on elements near the boundary at the first iteration:
  - The problem becomes more and more important as the mesh is refined.
  - In finite strain, this leads to severe divergence of the Newton algorithm due to the geometric stiffness matrix.
- In most **mechanical** solvers, a **prediction** of the solution based on the tangent problem is performed.
  - We were not able to implement this prediction so far.
  - <https://github.com/mfem/mfem/issues/2174>

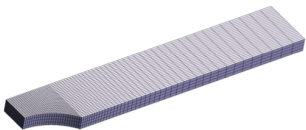
# Exemples

# Non linear case

## Elastoplastic modelling, finite strain

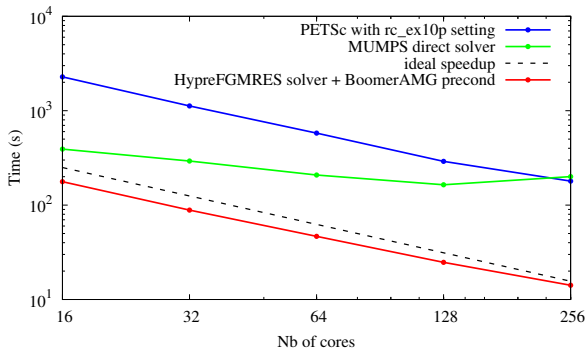
### Case description

- Uniaxial tensile test on a 3D notched beam
- Imposed displacement: right of the beam
- Symmetry conditions at: left position, down position



Geometry of the notched beam

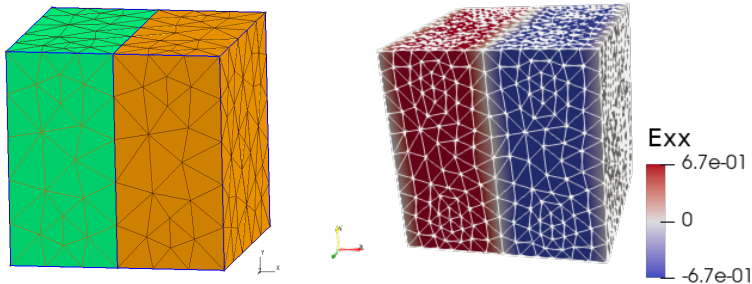
Notched beam case, 1.5M unknowns  
Strong scaling, Execution time (8 time steps)  
on mesocentre Aix-Marseille Univ.



## Test case setting - elastic model

- ▶ Two isotropic materials, each one filling half of a cube
- ▶ Aim : compare against MFEM-MGIS results against **analytical** results
  - considering 3 cases with uniaxial strain
  - considering 3 cases with shear test

## Mesh and result





## Properties

- ▶ 14 million unknowns, elastic modelling
- ▶ Ref. case on 32 cores - skylake @ mesocentre Aix-Marseille University
- ▶ Main linear solver: Conj. Gradient (iterative), no preconditioner used

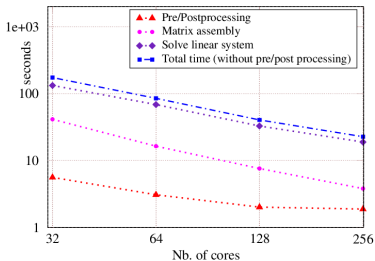
## Timings

- ▶ MFEM linear solve
  - total **166s**, matrix assembly 39s, CGsolver 120s
- ▶ MFEM non-linear solve - residual form, a single newton iteration
  - total **179s**, matrix assembly 53s, CGsolver 117s
- ▶ MFEM-MGIS non-linear solve - residual form, a single newton iteration
  - total **188s**, matrix assembly 42s, CGsolver 133s
- ▶ Overheads due to MFEM-MGIS are low
  - Due to: read/write to memory buffers, function calls for assembly

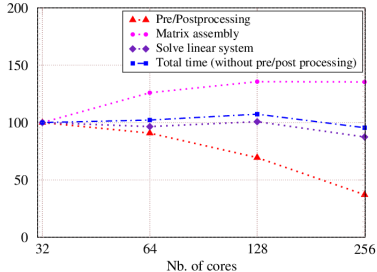
## Strong scaling

- Using MFEM-MGIS non-linear solve
- Strong scaling from 32 to 256 cores
- Scalable parallelism observed

Execution time, Skylake - Mesocentre AMU



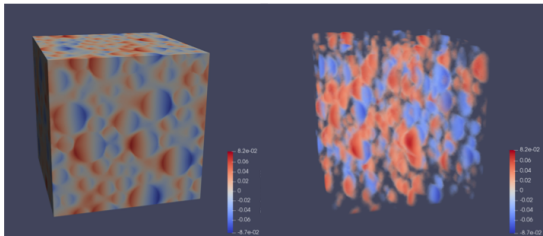
Relative efficiency, Skylake - Mesocentre AMU



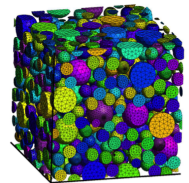
## MFEM contributions & coupling

- ▶ using as input : 3D periodic gms<sub>h</sub> meshes (public MFEM contrib)
- ▶ read/write MED files (private development)
- ▶ in progress : using netgen to get REV meshes (gms<sub>h</sub> format)

## Large runs

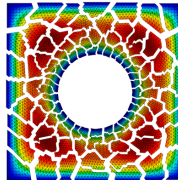


2000 inclusions (cut spheres), elastic modelling,  
displacement in x direction is shown,  
linear system with 500M unknowns



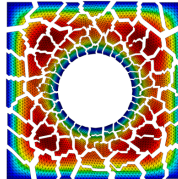
## Conclusions and perspectives

- ▶ What was done ?
  - High level declarative API suitable for engineering studies and upcoming integration in the PLEIADES platform.
    - MFEM' API still accessible user a lower level API (not shown here)
  - Multi-material support
  - Ability to handle arbitrary complex small and finite strain behaviours:
    - All behaviours are set at runtime (dynamically loaded libraries).



## ► What was done ?

- High level declarative API suitable for engineering studies and upcoming integration in the PLEIADES platform.
  - MFEM' API still accessible user a lower level API (not shown here)
- Multi-material support
- Ability to handle arbitrary complex small and finite strain behaviours:
  - All behaviours are set at runtime (dynamically loaded libraries).



## ► What comes next ?

- Overall robustness of the code (critical).
- Prediction and handling of the Dirichlet boundary conditions (critical).
- Provide more examples (urgent).
- Extension to other physical phenomena (non linear heat-transfer, diffusion, non local mechanics). *Should be easy* using code generation of behaviour integrators .
- Adaptive mesh refinements (requires new data structures in MGIS).
- Additional boundary conditions, including contact with friction.
- Port to GPUs (requires tremendous work on the MFront and MGIS side).
- Support for partial assembly (if possible ...), checkpoint/restart.

**Thank you for your attention.**

**Time for discussion !**

<https://github.com/thelfer/mfem-mgis>

<https://github.com/thelfer/>

MFrontGenericInterfaceSupport

<https://github.com/thelfer/mfem-mgis>

<https://tfel.sourceforge.net>

<https://www.researchgate.net/project/TFEL-MFront>

[https://twitter.com/TFEL\\_MFront](https://twitter.com/TFEL_MFront)

**tfel-contact@cea.fr**



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

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

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

- Resolution using the Newton-Raphson algorithm:

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

- Element contribution to inner forces:

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

- Element contribution to the stiffness:

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

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