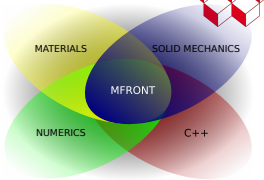


Overview of TFEL 4.2 and MGIS 2.2 and perspectives for TFEL 5.0

MFront User Meeting

06/11/2023

T. Helfer M. Wangermez

CEA, DES, IRESNE, DEC, SESC, LMCP, CADARACHE, FRANCE

Outline

Some highlights of 2023

Overview of `TFEL 4.2` and `MGIS 2.2`

Porting `TFEL` and `MGIS` on GPUs

What's going on for TFEL 5.0 and MGIS 3.0 ?

Conclusions

Addendum

Some highlights of 2023

2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.

2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:

2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:
    1. A basic introduction to `MFront` for material properties

2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:
    1. A basic introduction to `MFront` for material properties
    2. Implementing point-wise models with `MFront`

2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:
    1. A basic introduction to `MFront` for material properties
    2. Implementing point-wise models with `MFront`
    3. Domain specific languages dedicated to (visco-) plasticity of isotropic materials

2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:
    1. A basic introduction to `MFront` for material properties
    2. Implementing point-wise models with `MFront`
    3. Domain specific languages dedicated to (visco-) plasticity of isotropic materials
    4. Implementing behaviours using domain specific languages of the `Default` family

2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:
    1. A basic introduction to `MFront` for material properties
    2. Implementing point-wise models with `MFront`
    3. Domain specific languages dedicated to (visco-) plasticity of isotropic materials
    4. Implementing behaviours using domain specific languages of the `Default` family
  - See M. Wangermez's talk

2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:
    1. A basic introduction to `MFront` for material properties
    2. Implementing point-wise models with `MFront`
    3. Domain specific languages dedicated to (visco-) plasticity of isotropic materials
    4. Implementing behaviours using domain specific languages of the `Default` family
  - See M. Wangermez's talk
  - Ideal for newcomers or MFront's regular users.

2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:
    1. A basic introduction to `MFront` for material properties
    2. Implementing point-wise models with `MFront`
    3. Domain specific languages dedicated to (visco-) plasticity of isotropic materials
    4. Implementing behaviours using domain specific languages of the `Default` family
  - See M. Wangermez's talk
  - Ideal for newcomers or MFront's regular users.
  - Feed-backs are greatly appreciated, so do not hesitate to ask early drafts.

2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:
    1. A basic introduction to `MFront` for material properties
    2. Implementing point-wise models with `MFront`
    3. Domain specific languages dedicated to (visco-) plasticity of isotropic materials
    4. Implementing behaviours using domain specific languages of the `Default` family
  - See M. Wangermez's talk
  - Ideal for newcomers or MFront's regular users.
  - Feed-backs are greatly appreciated, so do not hesitate to ask early drafts.
- Ongoing collaboration with the Maison de la Simulation to port `TFEL`, and `MGIS` on GPUs:

2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:
    1. A basic introduction to `MFront` for material properties
    2. Implementing point-wise models with `MFront`
    3. Domain specific languages dedicated to (visco-) plasticity of isotropic materials
    4. Implementing behaviours using domain specific languages of the `Default` family
  - See M. Wangermez's talk
  - Ideal for newcomers or MFront's regular users.
  - Feed-backs are greatly appreciated, so do not hesitate to ask early drafts.
- Ongoing collaboration with the Maison de la Simulation to port `TFEL`, and `MGIS` on GPUs:
  - Founded by CEA's internal effort PTC-SIMU (Programme CEA Transversal de Compétences en Simulation).

2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:
    1. A basic introduction to `MFront` for material properties
    2. Implementing point-wise models with `MFront`
    3. Domain specific languages dedicated to (visco-) plasticity of isotropic materials
    4. Implementing behaviours using domain specific languages of the `Default` family
  - See M. Wangermez's talk
  - Ideal for newcomers or MFront's regular users.
  - Feed-backs are greatly appreciated, so do not hesitate to ask early drafts.
- Ongoing collaboration with the Maison de la Simulation to port `TFEL`, and `MGIS` on GPUs:
  - Founded by CEA's internal effort PTC-SIMU (Programme CEA Transversal de Compétences en Simulation).
  - See S. Khellal's talk for an overview of some first experiments.

2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:
    1. A basic introduction to `MFront` for material properties
    2. Implementing point-wise models with `MFront`
    3. Domain specific languages dedicated to (visco-) plasticity of isotropic materials
    4. Implementing behaviours using domain specific languages of the `Default` family
  - See M. Wangermez's talk
  - Ideal for newcomers or MFront's regular users.
  - Feed-backs are greatly appreciated, so do not hesitate to ask early drafts.
- Ongoing collaboration with the Maison de la Simulation to port `TFEL`, and `MGIS` on GPUs:
  - Founded by CEA's internal effort PTC-SIMU (Programme CEA Transversal de Compétences en Simulation).
  - See S. Khellal's talk for an overview of some first experiments.
  - See the second and third parts of this talk for the impact on `TFEL/Math`, `TFEL/Material` and `MGIS`.

2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:
    1. A basic introduction to `MFront` for material properties
    2. Implementing point-wise models with `MFront`
    3. Domain specific languages dedicated to (visco-) plasticity of isotropic materials
    4. Implementing behaviours using domain specific languages of the `Default` family
  - See M. Wangermez's talk
  - Ideal for newcomers or MFront's regular users.
  - Feed-backs are greatly appreciated, so do not hesitate to ask early drafts.
- Ongoing collaboration with the Maison de la Simulation to port `TFEL`, and `MGIS` on GPUs:
  - Founded by CEA's internal effort PTC-SIMU (Programme CEA Transversal de Compétences en Simulation).
  - See S. Khellal's talk for an overview of some first experiments.
  - See the second and third parts of this talk for the impact on `TFEL/Math`, `TFEL/Material` and `MGIS`.
- The `ANHONA` (Advanced NOnlinear HOmogenization for structural aNAlysis) project has been selected by the french National Agency for Research (ANR):

2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:
    1. A basic introduction to `MFront` for material properties
    2. Implementing point-wise models with `MFront`
    3. Domain specific languages dedicated to (visco-) plasticity of isotropic materials
    4. Implementing behaviours using domain specific languages of the `Default` family
  - See M. Wangermez's talk
  - Ideal for newcomers or MFront's regular users.
  - Feed-backs are greatly appreciated, so do not hesitate to ask early drafts.
- Ongoing collaboration with the Maison de la Simulation to port `TFEL`, and `MGIS` on GPUs:
  - Founded by CEA's internal effort PTC-SIMU (Programme CEA Transversal de Compétences en Simulation).
  - See S. Khellal's talk for an overview of some first experiments.
  - See the second and third parts of this talk for the impact on `TFEL/Math`, `TFEL/Material` and `MGIS`.
- The `ANHONA` (Advanced NOnlinear HOmogenization for structural aNAlysis) project has been selected by the french National Agency for Research (ANR):
  - A project geared by the most prominent french experts in nonlinear homogenization.

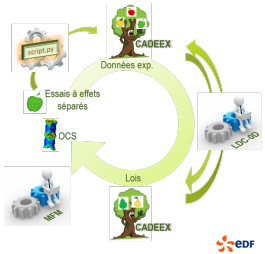
2023 highlights - I

- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:
    1. A basic introduction to `MFront` for material properties
    2. Implementing point-wise models with `MFront`
    3. Domain specific languages dedicated to (visco-) plasticity of isotropic materials
    4. Implementing behaviours using domain specific languages of the `Default` family
  - See M. Wangermez's talk
  - Ideal for newcomers or MFront's regular users.
  - Feed-backs are greatly appreciated, so do not hesitate to ask early drafts.
- Ongoing collaboration with the Maison de la Simulation to port `TFEL`, and `MGIS` on GPUs:
  - Founded by CEA's internal effort PTC-SIMU (Programme CEA Transversal de Compétences en Simulation).
  - See S. Khellal's talk for an overview of some first experiments.
  - See the second and third parts of this talk for the impact on `TFEL/Math`, `TFEL/Material` and `MGIS`.
- The `ANHONA` (Advanced NOnlinear HOmogenization for structural aNAlysis) project has been selected by the french National Agency for Research (ANR):
  - A project geared by the most prominent french experts in nonlinear homogenization.
  - `MFront` has been selected as the software backbone of this effort.

2023 highlights - I

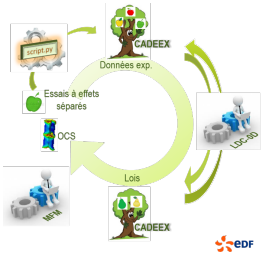
- The `MFront` Book is growing with 4 chapters already written.
  - Current outline:
    1. A basic introduction to `MFront` for material properties
    2. Implementing point-wise models with `MFront`
    3. Domain specific languages dedicated to (visco-) plasticity of isotropic materials
    4. Implementing behaviours using domain specific languages of the `Default` family
  - See M. Wangermez's talk
  - Ideal for newcomers or MFront's regular users.
  - Feed-backs are greatly appreciated, so do not hesitate to ask early drafts.
- Ongoing collaboration with the Maison de la Simulation to port `TFEL`, and `MGIS` on GPUs:
  - Founded by CEA's internal effort PTC-SIMU (Programme CEA Transversal de Compétences en Simulation).
  - See S. Khellal's talk for an overview of some first experiments.
  - See the second and third parts of this talk for the impact on `TFEL/Math`, `TFEL/Material` and `MGIS`.
- The `ANHONA` (Advanced NOnlinear HOmogenization for structural aNAlysis) project has been selected by the french National Agency for Research (ANR):
  - A project geared by the most prominent french experts in nonlinear homogenization.
  - `MFront` has been selected as the software backbone of this effort.
  - A post-doctoral position is currently open to strenghten `MFront`'s abilities in nonlinear homogenization with many intesting challenges [6].

2023 highlights - II



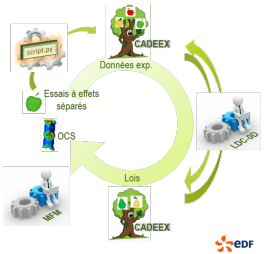
- End of the working group "Separated effects validation" by EDF, Framatome and EDF which aims at defining at standardized and consistent process to identify mechanical behaviours and properties for their usage in safety critical studies.

2023 highlights - II



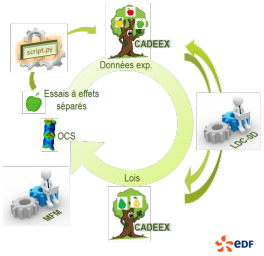
- End of the working group "Separated effects validation" by EDF, Framatome and EDF which aims at defining at standardized and consistent process to identify mechanical behaviours and properties for their usage in safety critical studies.
- The work group defined `MFront` as the standard format for implementing mechanical behaviours and properties.

2023 highlights - II



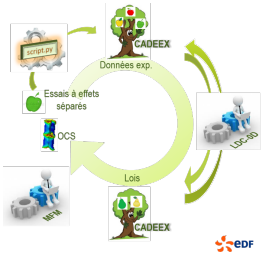
- End of the working group "Separated effects validation" by EDF, Framatome and EDF which aims at defining at standardized and consistent process to identify mechanical behaviours and properties for their usage in safety critical studies.
- The work group defined `MFront` as the standard format for implementing mechanical behaviours and properties.
- Several tools were extended, including `MFrontGallery`:

2023 highlights - II



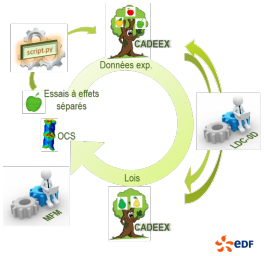
- End of the working group "Separated effects validation" by EDF, Framatome and EDF which aims at defining at standardized and consistent process to identify mechanical behaviours and properties for their usage in safety critical studies.
- The work group defined `MFront` as the standard format for implementing mechanical behaviours and properties.
- Several tools were extended, including `MFrontGallery`:
  - `https://github.com/thelfer/MFrontGallery`

2023 highlights - II



- End of the working group "Separated effects validation" by EDF, Framatome and EDF which aims at defining at standardized and consistent process to identify mechanical behaviours and properties for their usage in safety critical studies.
- The work group defined `MFront` as the standard format for implementing mechanical behaviours and properties.
- Several tools were extended, including `MFrontGallery`:
  - `https://github.com/thelfer/MFrontGallery`
- Those tools are promoted in the MECANUM project funded by the "France Relance" plan:

2023 highlights - II



- End of the working group "Separated effects validation" by EDF, Framatome and EDF which aims at defining at standardized and consistent process to identify mechanical behaviours and properties for their usage in safety critical studies.
- The work group defined `MFront` as the standard format for implementing mechanical behaviours and properties.
- Several tools were extended, including `MFrontGallery`:
  - `https://github.com/thelfer/MFrontGallery`
- Those tools are promoted in the MECANUM project funded by the "France Relance" plan:
  - `https://www.economie.gouv.fr/plan-de-relance`

2023 highlights - IV

CODE aster

- The unstable version of `code_aster` is now based on `MGIS`:

2023 highlights - IV

CODE aster

- The unstable version of `code_aster` is now based on `MGIS`:
  - The `aster` interface is to be removed in `TFEL 5.x`

2023 highlights - IV

- The unstable version of `code_aster` is now based on `MGIS`:
  - The `aster` interface is to be removed in `TFEL 5.x`
- Test of `MGIS` and `MFront` in `MEF++` at Michelin:

2023 highlights - IV

- The unstable version of `code_aster` is now based on `MGIS`:
  - The `aster` interface is to be removed in `TFEL 5.x`
- Test of `MGIS` and `MFront` in `MEF++` at Michelin:
  - See D. Siedel's talk.

2023 highlights - IV

- The unstable version of `code_aster` is now based on `MGIS`:
  - The `aster` interface is to be removed in `TFEL 5.x`
- Test of `MGIS` and `MFront` in `MEF++` at Michelin:
  - See D. Siedel's talk.
  - Required a change of the TFEL licence which is now GPLv3 with runtime exceptions to be usable in any commercial software.

2023 highlights - IV

- The unstable version of `code_aster` is now based on `MGIS`:
  - The `aster` interface is to be removed in `TFEL 5.x`
- Test of `MGIS` and `MFront` in `MEF++` at Michelin:
  - See D. Siedel's talk.
  - Required a change of the TFEL licence which is now GPLv3 with runtime exceptions to be usable in any commercial software.
- Intensive development of the new CEA's thermomechanical solver named `Manta`.

2023 highlights - IV

- The unstable version of `code_aster` is now based on `MGIS`:
  - The `aster` interface is to be removed in `TFEL 5.x`
- Test of `MGIS` and `MFront` in `MEF++` at Michelin:
  - See D. Siedel's talk.
  - Required a change of the TFEL licence which is now GPLv3 with runtime exceptions to be usable in any commercial software.
- Intensive development of the new CEA's thermomechanical solver named `Manta`.
  - See the dedicated talk.

2023 highlights - IV

- The unstable version of `code_aster` is now based on `MGIS`:
  - The `aster` interface is to be removed in `TFEL 5.x`
- Test of `MGIS` and `MFront` in `MEF++` at Michelin:
  - See D. Siedel's talk.
  - Required a change of the TFEL licence which is now GPLv3 with runtime exceptions to be usable in any commercial software.
- Intensive development of the new CEA's thermomechanical solver named `Manta`.
  - See the dedicated talk.
- TFEL is currently being audited by `Code Reckons`:

2023 highlights - IV

- The unstable version of `code_aster` is now based on `MGIS`:
  - The `aster` interface is to be removed in `TFEL 5.x`
- Test of `MGIS` and `MFront` in `MEF++` at Michelin:
  - See D. Siedel's talk.
  - Required a change of the TFEL licence which is now GPLv3 with runtime exceptions to be usable in any commercial software.
- Intensive development of the new CEA's thermomechanical solver named `Manta`.
  - See the dedicated talk.
- TFEL is currently being audited by `Code Reckons`:
  - `https://www.codereckons.com`

2023 highlights - IV

- The unstable version of `code_aster` is now based on `MGIS`:
  - The `aster` interface is to be removed in `TFEL 5.x`
- Test of `MGIS` and `MFront` in `MEF++` at Michelin:
  - See D. Siedel's talk.
  - Required a change of the TFEL licence which is now GPLv3 with runtime exceptions to be usable in any commercial software.
- Intensive development of the new CEA's thermomechanical solver named `Manta`.
  - See the dedicated talk.
- TFEL is currently being audited by `Code Reckons`:
  - `https://www.codereckons.com`
  - Several training sessions to improve `TFEL` with the `C++` and HPC expert Joel Falcou.

Training sessions

- 3 training sessions are actually planned:
  - 2 training sessions associated in the context of the `MECANUM` project at EDF Lab Saclay:
    - November 2023 and early 2024 (Februar or March).
    - Mostly for engineers at EDF, CEA and Naval Group
  - 1 training session at Cadarache in the context of the `Opera HPC` European project:
    - Mostly targest nuclear engineers from EPFL, VTT, CEA, etc.
- Training sessions are organized **on demand**: typically spanned over two days by groups of around 12 people. The requester must provide rooms, computers, and ~~**beer**~~ meals, etc...

Overview of TFEL 4.2 and MGIS 2.2

Current state of the TFEL 4.x branch

- The TFEL 4.x branch is maturing and starts being used in production.
- TFEL 4.1.x is delivered with:
  - unstable version of code_aster.
  - the upcoming Version 2024 of Cast3M (maybe TFEL 4.2.x).
- Currently, 17 publications based on MFront in 2023 [14, 5, 4, 10, 7, 16, 1, 12, 11, 2, 8, 20, 18, 12, 19, 15, 17].

Issues fixed

- A new release comes with new features and bug fixes.

Issues fixed

- A new release comes with new features and bug fixes.
- Bugs are fixed in the branch corresponding to the release where the bug was introduced and the fix is then spread to newer branches.

Issues fixed

- A new release comes with new features and bug fixes.
- Bugs are fixed in the branch corresponding to the release where the bug was introduced and the fix is then spread to newer branches.
  - For instance, if a bug has been introduced in `TFEL 3.2`, the bug is fixed in the `rliv-3.2` branches and spread in the `rliv-3.3`, `rliv-3.4`, `rliv-4.0`, `rliv-4.1` and `master` branches.

Issues fixed

- A new release comes with new features and bug fixes.
- Bugs are fixed in the branch corresponding to the release where the bug was introduced and the fix is then spread to newer branches.
  - For instance, if a bug has been introduced in `TFEL 3.2`, the bug is fixed in the `rliv-3.2` branches and spread in the `rliv-3.3`, `rliv-3.4`, `rliv-4.0`, `rliv-4.1` and `master` branches.
- `TFEL 4.2` will be thus be released along with several minor releases: `4.1.1`, `3.4.5`, `3.2.9`, `3.1.11`, `3.0.12`.

Issues fixed

- A new release comes with new features and bug fixes.
- Bugs are fixed in the branch corresponding to the release where the bug was introduced and the fix is then spread to newer branches.
  - For instance, if a bug has been introduced in `TFEL 3.2`, the bug is fixed in the `rliv-3.2` branches and spread in the `rliv-3.3`, `rliv-3.4`, `rliv-4.0`, `rliv-4.1` and `master` branches.
- `TFEL 4.2` will be thus be released along with several minor releases: `4.1.1`, `3.4.5`, `3.2.9`, `3.1.11`, `3.0.12`.
- `TFEL 4.2` is mostly a bug-fixed release:

Issues fixed

- A new release comes with new features and bug fixes.
- Bugs are fixed in the branch corresponding to the release where the bug was introduced and the fix is then spread to newer branches.
  - For instance, if a bug has been introduced in `TFEL 3.2`, the bug is fixed in the `rliv-3.2` branches and spread in the `rliv-3.3`, `rliv-3.4`, `rliv-4.0`, `rliv-4.1` and `master` branches.
- `TFEL 4.2` will be thus be released along with several minor releases: `4.1.1`, `3.4.5`, `3.2.9`, `3.1.11`, `3.0.12`.
- `TFEL 4.2` is mostly a bug-fixed release:
  - `4.2`: 13 issues fixed.

Issues fixed

- A new release comes with new features and bug fixes.
- Bugs are fixed in the branch corresponding to the release where the bug was introduced and the fix is then spread to newer branches.
  - For instance, if a bug has been introduced in `TFEL 3.2`, the bug is fixed in the `rliv-3.2` branches and spread in the `rliv-3.3`, `rliv-3.4`, `rliv-4.0`, `rliv-4.1` and `master` branches.
- `TFEL 4.2` will be thus be released along with several minor releases: `4.1.1`, `3.4.5`, `3.2.9`, `3.1.11`, `3.0.12`.
- `TFEL 4.2` is mostly a bug-fixed release:
  - `4.2`: 13 issues fixed.
  - `4.1.1`: 7 issues fixed.

Issues fixed

- A new release comes with new features and bug fixes.
- Bugs are fixed in the branch corresponding to the release where the bug was introduced and the fix is then spread to newer branches.
  - For instance, if a bug has been introduced in `TFEL 3.2`, the bug is fixed in the `rliv-3.2` branches and spread in the `rliv-3.3`, `rliv-3.4`, `rliv-4.0`, `rliv-4.1` and `master` branches.
- `TFEL 4.2` will be thus be released along with several minor releases: `4.1.1`, `3.4.5`, `3.2.9`, `3.1.11`, `3.0.12`.
- `TFEL 4.2` is mostly a bug-fixed release:
  - `4.2`: 13 issues fixed.
  - `4.1.1`: 7 issues fixed.
  - `3.4.5`: 4 issues fixed.

Issues fixed

- A new release comes with new features and bug fixes.
- Bugs are fixed in the branch corresponding to the release where the bug was introduced and the fix is then spread to newer branches.
  - For instance, if a bug has been introduced in `TFEL 3.2`, the bug is fixed in the `rliv-3.2` branches and spread in the `rliv-3.3`, `rliv-3.4`, `rliv-4.0`, `rliv-4.1` and `master` branches.
- `TFEL 4.2` will be thus be released along with several minor releases: `4.1.1`, `3.4.5`, `3.2.9`, `3.1.11`, `3.0.12`.
- `TFEL 4.2` is mostly a bug-fixed release:
  - `4.2`: 13 issues fixed.
  - `4.1.1`: 7 issues fixed.
  - `3.4.5`: 4 issues fixed.
  - `3.2.9`: 2 issues fixed.

Issues fixed

- A new release comes with new features and bug fixes.
- Bugs are fixed in the branch corresponding to the release where the bug was introduced and the fix is then spread to newer branches.
  - For instance, if a bug has been introduced in `TFEL 3.2`, the bug is fixed in the `rliv-3.2` branches and spread in the `rliv-3.3`, `rliv-3.4`, `rliv-4.0`, `rliv-4.1` and `master` branches.
- `TFEL 4.2` will be thus be released along with several minor releases: `4.1.1`, `3.4.5`, `3.2.9`, `3.1.11`, `3.0.12`.
- `TFEL 4.2` is mostly a bug-fixed release:
  - `4.2`: 13 issues fixed.
  - `4.1.1`: 7 issues fixed.
  - `3.4.5`: 4 issues fixed.
  - `3.2.9`: 2 issues fixed.
  - `3.1.11`: 6 issues fixed.

Issues fixed

- A new release comes with new features and bug fixes.
- Bugs are fixed in the branch corresponding to the release where the bug was introduced and the fix is then spread to newer branches.
  - For instance, if a bug has been introduced in `TFEL 3.2`, the bug is fixed in the `rliv-3.2` branches and spread in the `rliv-3.3`, `rliv-3.4`, `rliv-4.0`, `rliv-4.1` and `master` branches.
- `TFEL 4.2` will be thus be released along with several minor releases: `4.1.1`, `3.4.5`, `3.2.9`, `3.1.11`, `3.0.12`.
- `TFEL 4.2` is mostly a bug-fixed release:
  - `4.2`: 13 issues fixed.
  - `4.1.1`: 7 issues fixed.
  - `3.4.5`: 4 issues fixed.
  - `3.2.9`: 2 issues fixed.
  - `3.1.11`: 6 issues fixed.
  - `3.0.12`: 6 issues fixed.

Issues fixed

- A new release comes with new features and bug fixes.
- Bugs are fixed in the branch corresponding to the release where the bug was introduced and the fix is then spread to newer branches.
  - For instance, if a bug has been introduced in `TFEL 3.2`, the bug is fixed in the `rliv-3.2` branches and spread in the `rliv-3.3`, `rliv-3.4`, `rliv-4.0`, `rliv-4.1` and `master` branches.
- `TFEL 4.2` will be thus be released along with several minor releases: `4.1.1`, `3.4.5`, `3.2.9`, `3.1.11`, `3.0.12`.
- `TFEL 4.2` is mostly a bug-fixed release:
  - `4.2`: 13 issues fixed.
  - `4.1.1`: 7 issues fixed.
  - `3.4.5`: 4 issues fixed.
  - `3.2.9`: 2 issues fixed.
  - `3.1.11`: 6 issues fixed.
  - `3.0.12`: 6 issues fixed.
- See the release notes for a description of each issues fixed.

Issues fixed

- A new release comes with new features and bug fixes.
- Bugs are fixed in the branch corresponding to the release where the bug was introduced and the fix is then spread to newer branches.
  - For instance, if a bug has been introduced in `TFEL 3.2`, the bug is fixed in the `rliv-3.2` branches and spread in the `rliv-3.3`, `rliv-3.4`, `rliv-4.0`, `rliv-4.1` and `master` branches.
- `TFEL 4.2` will be thus be released along with several minor releases: `4.1.1`, `3.4.5`, `3.2.9`, `3.1.11`, `3.0.12`.
- `TFEL 4.2` is mostly a bug-fixed release:
  - `4.2`: 13 issues fixed.
  - `4.1.1`: 7 issues fixed.
  - `3.4.5`: 4 issues fixed.
  - `3.2.9`: 2 issues fixed.
  - `3.1.11`: 6 issues fixed.
  - `3.0.12`: 6 issues fixed.
- See the release notes for a description of each issues fixed.
- Creating a release from any of the `rliv` branches is a matter of minutes.

Generalized Gurson-Tvergaard-Needleman yield surface

- The Gurson-Tvergaard-Needleman yield surface can be generalized by replacing the von Mises equivalent stress by another equivalent stress (Hill, Hosford, etc..):

$$S = \left(\frac{\sigma_{eq}}{\sigma_\star}\right)^2 + 2q_1 f_\star \cosh\left(\frac{3}{2} q_2 \frac{\sigma_m}{\sigma_\star}\right) - 1 - q_3 f_\star^2 = 0$$

Generalized Gurson-Tvergaard-Needleman yield surface

- The Gurson-Tvergaard-Needleman yield surface can be generalized by replacing the von Mises equivalent stress by another equivalent stress (Hill, Hosford, etc..):

$$S = \left(\frac{\sigma_{eq}}{\sigma_\star}\right)^2 + 2q_1 f_\star \cosh\left(\frac{3}{2} q_2 \frac{\sigma_m}{\sigma_\star}\right) - 1 - q_3 f_\star^2 = 0$$

- The envisaged syntax is:

```
@Brick "StandardElastoViscoPlasticity" {
    stress_potential : "Hooke" {
        young_modulus : 200e9, poisson_ratio : 0.3
    },
    inelastic_flow : "Plastic" {
        criterion : "GursonTvergaardNeedleman" {
            q1 : 1.5, q2 : 1.0, q3 : 2.2, fc : 0.01, fr : 0.1
        },
        isotropic_hardening : "Linear" {
            R0 : 150e6
        },
        matrix_equivalent_stress: "Hosford"{
            a: 8
        }
    },
    porosity_evolution : {
        growth_model : "StandardPlasticModel"
};
```

Stress potentials for damage gradient models

- Several stress potentials will be introduced for coupling damage gradients models:

Stress potentials for damage gradient models

- Several stress potentials will be introduced for coupling damage gradients models:
  - hydrostatic-deviatoric [3], spectral split [9], etc..

Stress potentials for damage gradient models

- Several stress potentials will be introduced for coupling damage gradients models:
  - hydrostatic-deviatoric [3], spectral split [9], etc..
- Just like the `Hooke` stress potential is the basis of the `StandardElasticity` brick, those new stress potentials will be used to introduce new bricks.

Stress potentials for damage gradient models

- Several stress potentials will be introduced for coupling damage gradients models:
  - hydrostatic-deviatoric [3], spectral split [9], etc..
- Just like the `Hooke` stress potential is the basis of the `StandardElasticity` brick, those new stress potentials will be used to introduce new bricks.
- The envisaged syntax is:

```
@Brick "StandardElastoViscoPlasticity" {
    stress_potential : "SpectralSplit" {
        young_modulus : 200e9, poisson_ratio : 0.3,
        elastic_energy_release_rate: true // default
    }
};
```

Stress potentials for damage gradient models

- Several stress potentials will be introduced for coupling damage gradients models:
  - hydrostatic-deviatoric [3], spectral split [9], etc..
- Just like the `Hooke` stress potential is the basis of the `StandardElasticity` brick, those new stress potentials will be used to introduce new bricks.
- The envisaged syntax is:

```
@Brick "StandardElastoViscoPlasticity" {
    stress_potential : "SpectralSplit" {
        young_modulus : 200e9, poisson_ratio : 0.3,
        elastic_energy_release_rate: true // default
    }
};
```

- The damage will be defined as an external state variable named `d`.

Stress potentials for damage gradient models

- Several stress potentials will be introduced for coupling damage gradients models:
  - hydrostatic-deviatoric [3], spectral split [9], etc..
- Just like the `Hooke` stress potential is the basis of the `StandardElasticity` brick, those new stress potentials will be used to introduce new bricks.
- The envisaged syntax is:

```
@Brick "StandardElastoViscoPlasticity" {
    stress_potential : "SpectralSplit" {
        young_modulus : 200e9, poisson_ratio : 0.3,
        elastic_energy_release_rate: true // default
    }
};
```

- The damage will be defined as an external state variable named `d`.
- By default, the quadratic degradation function is used:

$$g(d) = (1-d)^2$$

Stress potentials for damage gradient models

- Several stress potentials will be introduced for coupling damage gradients models:
  - hydrostatic-deviatoric [3], spectral split [9], etc..
- Just like the `Hooke` stress potential is the basis of the `StandardElasticity` brick, those new stress potentials will be used to introduce new bricks.
- The envisaged syntax is:

```
@Brick "StandardElastoViscoPlasticity" {
    stress_potential : "SpectralSplit" {
        young_modulus : 200e9, poisson_ratio : 0.3,
        elastic_energy_release_rate: true // default
    }
};
```

- The damage will be defined as an external state variable named `d`.
- By default, the quadratic degradation function is used:

$$g(d) = (1-d)^2$$

- The elastic energy release variable, defined as the partial derivative of the free energy with respect to damage, is computed as an auxiliary state variable.

External workspaces for nonlinear solvers

```
template<unsigned short N, typename NumericType>
struct ExternallyAllocatedWorkspace{
  ExternallyAllocatedWorkspace(NumericType* const v)
      : fzeros(v), zeros(v + N), delta_zeros(v + 2 * N), jacobian(v + 3 * N){};
  //! \brief residual vector
  tfel::math::View<tfel::math::tvector<N, NumericType>> fzeros;
  //! \brief current estimate of the unknowns
  tfel::math::View<tfel::math::tvector<N, NumericType>> zeros;
  //! \brief current correction
  tfel::math::View<tfel::math::tvector<N, NumericType>> delta_zeros;
  //! \brief jacobian matrix
  tfel::math::View<tfel::math::tmatrix<N, N, NumericType>> jacobian;
};
```

- In `TFEL/Math`, nonlinear solvers can now be parametrized by a class, refered to as a workspace. This class must provide the required data members for the solvers to work (jacobian matrix, residual, etc..)

External workspaces for nonlinear solvers

```
template<unsigned short N, typename NumericType>
struct ExternallyAllocatedWorkspace{
  ExternallyAllocatedWorkspace(NumericType* const v)
      : fzeros(v), zeros(v + N), delta_zeros(v + 2 * N), jacobian(v + 3 * N){};
  //! \brief residual vector
  tfel::math::View<tfel::math::tvector<N, NumericType>> fzeros;
  //! \brief current estimate of the unknowns
  tfel::math::View<tfel::math::tvector<N, NumericType>> zeros;
  //! \brief current correction
  tfel::math::View<tfel::math::tvector<N, NumericType>> delta_zeros;
  //! \brief jacobian matrix
  tfel::math::View<tfel::math::tmatrix<N, N, NumericType>> jacobian;
};
```

- In `TFEL/Math`, nonlinear solvers can now be parametrized by a class, refered to as a workspace. This class must provide the required data members for the solvers to work (jacobian matrix, residual, etc..)
- The default workspace still allocates those data members on the stack.

External workspaces for nonlinear solvers

```
template<unsigned short N, typename NumericType>
struct ExternallyAllocatedWorkspace{
  ExternallyAllocatedWorkspace(NumericType* const v)
      : fzeros(v), zeros(v + N), delta_zeros(v + 2 * N), jacobian(v + 3 * N){};
  //! \brief residual vector
  tfel::math::View<tfel::math::tvector<N, NumericType>> fzeros;
  //! \brief current estimate of the unknowns
  tfel::math::View<tfel::math::tvector<N, NumericType>> zeros;
  //! \brief current correction
  tfel::math::View<tfel::math::tvector<N, NumericType>> delta_zeros;
  //! \brief jacobian matrix
  tfel::math::View<tfel::math::tmatrix<N, N, NumericType>> jacobian;
};
```

- In `TFEL/Math`, nonlinear solvers can now be parametrized by a class, refered to as a workspace. This class must provide the required data members for the solvers to work (jacobian matrix, residual, etc..)
- The default workspace still allocates those data members on the stack.
- This feature has been introduced to reduce the number of registers used on GPUs.

What's left to end the TFEL 4.x development cycle ?

- Finish support for initialize function (mostly on the MGIS side).
- Finish support for post-processing function (mostly on the MGIS side).
- Support for JIT in MGIS.
- New interfaces:
  - Plaxis: https://github.com/thelfer/tfel/issues/272
  - Flac3D: https://github.com/thelfer/tfel/issues/271

Porting TFEL and MGIS on GPUs

Motivation and challenges

- HPC is moving towards GPUs computing:

Motivation and challenges

- HPC is moving towards GPUs computing:
  - More raw power.

Motivation and challenges

- HPC is moving towards GPUs computing:
  - More raw power.
  - Less watts per flops.

Motivation and challenges

- HPC is moving towards GPUs computing:
  - More raw power.
  - Less watts per flops.
- Porting to the `GPUs` is beneficial for TFEL.

Motivation and challenges

- HPC is moving towards GPUs computing:
  - More raw power.
  - Less watts per flops.
- Porting to the GPUs is beneficial for TFEL.
- Mechanical behaviours are a priori bad candidates with lots of logics and lot of memory transfer.

Motivation and challenges

- HPC is moving towards GPUs computing:
  - More raw power.
  - Less watts per flops.
- Porting to the `GPUs` is beneficial for TFEL.
- Mechanical behaviours are a priori bad candidates with lots of logics and lot of memory transfer.
  - The work load is generally very unbalanced (localized damage and/or plasticity).

Motivation and challenges

- HPC is moving towards GPUs computing:
  - More raw power.
  - Less watts per flops.
- Porting to the `GPUs` is beneficial for TFEL.
- Mechanical behaviours are a priori bad candidates with lots of logics and lot of memory transfer.
  - The work load is generally very unbalanced (localized damage and/or plasticity).
  - Registers may be a bottleneck that would limit the number of cores that can be adressed. One may only use a fraction $1/2^n$ of the available cores.

Motivation and challenges

- HPC is moving towards GPUs computing:
  - More raw power.
  - Less watts per flops.
- Porting to the `GPUs` is beneficial for TFEL.
- Mechanical behaviours are a priori bad candidates with lots of logics and lot of memory transfer.
  - The work load is generally very unbalanced (localized damage and/or plasticity).
  - Registers may be a bottleneck that would limit the number of cores that can be adressed. One may only use a fraction $1/2^n$ of the available cores.
- Treating one integration point only is meaningless: the behaviour integration must be performed of a significant number of integration points.

Motivation and challenges

- HPC is moving towards GPUs computing:
  - More raw power.
  - Less watts per flops.
- Porting to the `GPUs` is beneficial for TFEL.
- Mechanical behaviours are a priori bad candidates with lots of logics and lot of memory transfer.
  - The work load is generally very unbalanced (localized damage and/or plasticity).
  - Registers may be a bottleneck that would limit the number of cores that can be adressed. One may only use a fraction $1/2^n$ of the available cores.
- Treating one integration point only is meaningless: the behaviour integration must be performed of a significant number of integration points.
  - Considerable impact on the current architecture and frontier between `MFront` and `MGIS`.

Motivation and challenges

- HPC is moving towards GPUs computing:
  - More raw power.
  - Less watts per flops.
- Porting to the `GPUs` is beneficial for TFEL.
- Mechanical behaviours are a priori bad candidates with lots of logics and lot of memory transfer.
  - The work load is generally very unbalanced (localized damage and/or plasticity).
  - Registers may be a bottleneck that would limit the number of cores that can be adressed. One may only use a fraction $1/2^n$ of the available cores.
- Treating one integration point only is meaningless: the behaviour integration must be performed of a significant number of integration points.
  - Considerable impact on the current architecture and frontier between `MFront` and `MGIS`.
  - This is the reaons why we focus on a specific use case (FFT solvers) in our early experiments.

Early experiments on GPUs

- Salem Khellal's talk.

Challenges for TFEL

- The TFEL/Math and TFEL/Material libraries have been partially ported to GPUs using differents programming models: CUDA (with the nvcc compiler or the clang compiler), SYCL, Kokkos).

Challenges for TFEL

- The TFEL/Math and TFEL/Material libraries have been partially ported to GPUs using differents programming models: CUDA (with the nvcc compiler or the clang compiler), SYCL, Kokkos).
  - As those libraries mostly introduces template functions and does not use virtual polymorphism, the port boils down to adding special flags at the beginning of the function declaration and implementation, which is a bit tedious but not difficult.

Challenges for TFEL

- The TFEL/Math and TFEL/Material libraries have been partially ported to GPUs using differents programming models: CUDA (with the nvcc compiler or the clang compiler), SYCL, Kokkos).
  - As those libraries mostly introduces template functions and does not use virtual polymorphism, the port boils down to adding special flags at the beginning of the function declaration and implementation, which is a bit tedious but not difficult.
  - Some functions still uses exceptions, which are not supported on GPUs.

Challenges for TFEL

- The TFEL/Math and TFEL/Material libraries have been partially ported to GPUs using differents programming models: CUDA (with the nvcc compiler or the clang compiler), SYCL, Kokkos).
  - As those libraries mostly introduces template functions and does not use virtual polymorphism, the port boils down to adding special flags at the beginning of the function declaration and implementation, which is a bit tedious but not difficult.
  - Some functions still uses exceptions, which are not supported on GPUs.
  - Some functions have been designed to heavily use the stack, which may lead to the usage of a huge number of registers. This is the motivation for introducing the external workspaces for the nonlinear solvers.

Challenges for TFEL

- The TFEL/Math and TFEL/Material libraries have been partially ported to GPUs using differents programming models: CUDA (with the nvcc compiler or the clang compiler), SYCL, Kokkos).
  - As those libraries mostly introduces template functions and does not use virtual polymorphism, the port boils down to adding special flags at the beginning of the function declaration and implementation, which is a bit tedious but not difficult.
  - Some functions still uses exceptions, which are not supported on GPUs.
  - Some functions have been designed to heavily use the stack, which may lead to the usage of a huge number of registers. This is the motivation for introducing the external workspaces for the nonlinear solvers.
- The most reliable eigen solver in TFEL is based on Jacobi's iterative algorithm. Its usage is strongly adviced when second derivatives of the eigen values are required. This is notably the case for the logarithmic strain framework. A closed formed alternative by Scherzinger et al. is worth investigating [13].

Challenges for MFront

- GPUs requires much more optimized algorithms.

Challenges for MFront

- GPUs requires much more optimized algorithms.
- The first candidates for GPUs are small-strain isotropic (visco-)plastic behaviours whose integration can be reduced to finding the root of a scalar nonlinear function:

Challenges for MFront

- GPUs requires much more optimized algorithms.
- The first candidates for GPUs are small-strain isotropic (visco-)plastic behaviours whose integration can be reduced to finding the root of a scalar nonlinear function:
  - See M. Wangermez's talks.

Challenges for MFront

- GPUs requires much more optimized algorithms.
- The first candidates for GPUs are small-strain isotropic (visco-)plastic behaviours whose integration can be reduced to finding the root of a scalar nonlinear function:
  - See M. Wangermez's talks.
  - Those DSLs must be extended to take the plane stress hypothesis into account. The `MultipleIsotropicMisesFlows` must be extented to compute the consistent tangent operator.

Challenges for `MFront`

- GPUs requires much more optimized algorithms.
- The first candidates for GPUs are small-strain isotropic (visco-)plastic behaviours whose integration can be reduced to finding the root of a scalar nonlinear function:
  - See M. Wangermez's talks.
  - Those DSLs must be extended to take the plane stress hypothesis into account. The `MultipleIsotropicMisesFlows` must be extented to compute the consistent tangent operator.
  - If the elastic properties are known to be constant, the elastic strain can be removed from the state variables.

Challenges for MFront

- GPUs requires much more optimized algorithms.
- The first candidates for GPUs are small-strain isotropic (visco-)plastic behaviours whose integration can be reduced to finding the root of a scalar nonlinear function:
  - See M. Wangermez's talks.
  - Those DSLs must be extended to take the plane stress hypothesis into account. The `MultipleIsotropicMisesFlows` must be extented to compute the consistent tangent operator.
  - If the elastic properties are known to be constant, the elastic strain can be removed from the state variables.
  - New highly specialized DSLs can be introduced, taking into account the hydrostatic pressure and the von Mises stress.

Challenges for MFront

- GPUs requires much more optimized algorithms.
- The first candidates for GPUs are small-strain isotropic (visco-)plastic behaviours whose integration can be reduced to finding the root of a scalar nonlinear function:
  - See M. Wangermez's talks.
  - Those DSLs must be extended to take the plane stress hypothesis into account. The `MultipleIsotropicMisesFlows` must be extented to compute the consistent tangent operator.
  - If the elastic properties are known to be constant, the elastic strain can be removed from the state variables.
  - New highly specialized DSLs can be introduced, taking into account the hydrostatic pressure and the von Mises stress.
    - In this case, the behavior integration is reduced to finding the solution of a system of two nonlinear equations.

Challenges for MFront

- GPUs requires much more optimized algorithms.
- The first candidates for GPUs are small-strain isotropic (visco-)plastic behaviours whose integration can be reduced to finding the root of a scalar nonlinear function:
  - See M. Wangermez's talks.
  - Those DSLs must be extended to take the plane stress hypothesis into account. The `MultipleIsotropicMisesFlows` must be extented to compute the consistent tangent operator.
  - If the elastic properties are known to be constant, the elastic strain can be removed from the state variables.
  - New highly specialized DSLs can be introduced, taking into account the hydrostatic pressure and the von Mises stress.
    - In this case, the behavior integration is reduced to finding the solution of a system of two nonlinear equations.
- JIT compilation to fine-tune the implementation and reduce memory/transfer usage.

Challenges for MFront

- GPUs requires much more optimized algorithms.
- The first candidates for GPUs are small-strain isotropic (visco-)plastic behaviours whose integration can be reduced to finding the root of a scalar nonlinear function:
  - See M. Wangermez's talks.
  - Those DSLs must be extended to take the plane stress hypothesis into account. The `MultipleIsotropicMisesFlows` must be extented to compute the consistent tangent operator.
  - If the elastic properties are known to be constant, the elastic strain can be removed from the state variables.
  - New highly specialized DSLs can be introduced, taking into account the hydrostatic pressure and the von Mises stress.
    - In this case, the behavior integration is reduced to finding the solution of a system of two nonlinear equations.
- JIT compilation to fine-tune the implementation and reduce memory/transfer usage.
- Finite strain strategies consists in pre- and post-processing steps. In particular, the logarithmic strain framework requires the storage of temporary objects between those steps. An external workspace-like strategy is probably required.

Challenges for MGIS

- MGIS must support JIT compilation of MFront behaviours.

Challenges for MGIS

- MGIS must support JIT compilation of MFront behaviours.
- MGIS must hide details about the programming model, memory allocation, data access, etc...

Challenges for MGIS

- MGIS must support JIT compilation of MFront behaviours.
- MGIS must hide details about the programming model, memory allocation, data access, etc...
  - MGIS shall not impose any programming model. The latter shall be choosen at runtime in most cases.

Challenges for MGIS

- MGIS must support JIT compilation of MFront behaviours.
- MGIS must hide details about the programming model, memory allocation, data access, etc...
  - MGIS shall not impose any programming model. The latter shall be choosen at runtime in most cases.
- For crystal plasticity, it may be required to consider material zones to describe grains...

Challenges for MGIS

- MGIS must support JIT compilation of MFront behaviours.
- MGIS must hide details about the programming model, memory allocation, data access, etc...
  - MGIS shall not impose any programming model. The latter shall be choosen at runtime in most cases.
- For crystal plasticity, it may be required to consider material zones to describe grains...
- The loop over the integration point must be part of the generated code. That's fair for FFT solvers.

Challenges for MGIS

- MGIS must support JIT compilation of MFront behaviours.
- MGIS must hide details about the programming model, memory allocation, data access, etc...
  - MGIS shall not impose any programming model. The latter shall be choosen at runtime in most cases.
- For crystal plasticity, it may be required to consider material zones to describe grains...
- The loop over the integration point must be part of the generated code. That's fair for FFT solvers.
- There are still many open questions in FEM:

Challenges for MGIS

- MGIS must support JIT compilation of MFront behaviours.
- MGIS must hide details about the programming model, memory allocation, data access, etc...
  - MGIS shall not impose any programming model. The latter shall be choosen at runtime in most cases.
- For crystal plasticity, it may be required to consider material zones to describe grains...
- The loop over the integration point must be part of the generated code. That's fair for FFT solvers.
- There are still many open questions in FEM:
  - The memory to store 10^6 tangent operators in a 3D finite strain analysis is huge: $10^6 \cdots 81 \cdots 8 \approx 650\ Mb$.

Challenges for MGIS

- MGIS must support JIT compilation of MFront behaviours.
- MGIS must hide details about the programming model, memory allocation, data access, etc...
  - MGIS shall not impose any programming model. The latter shall be choosen at runtime in most cases.
- For crystal plasticity, it may be required to consider material zones to describe grains...
- The loop over the integration point must be part of the generated code. That's fair for FFT solvers.
- There are still many open questions in FEM:
  - The memory to store 10^6 tangent operators in a 3D finite strain analysis is huge: $10^6 \cdots 81 \cdots 8 \approx 650\ Mb$.
    - This may be required by matrix-free solvers...

Challenges for MGIS

- MGIS must support JIT compilation of MFront behaviours.
- MGIS must hide details about the programming model, memory allocation, data access, etc...
  - MGIS shall not impose any programming model. The latter shall be choosen at runtime in most cases.
- For crystal plasticity, it may be required to consider material zones to describe grains...
- The loop over the integration point must be part of the generated code. That's fair for FFT solvers.
- There are still many open questions in FEM:
  - The memory to store 10^6 tangent operators in a 3D finite strain analysis is huge: $10^6 \cdots 81 \cdots 8 \approx 650\ Mb$.
    - This may be required by matrix-free solvers...
    - Cast3M-like accelerated fixed-point algorithms are worth considering, in particular if we consider that the elastic matrix is just a preconditioner (incomplete simple precision decomposition may just also work).

Challenges for MGIS

- MGIS must support JIT compilation of MFront behaviours.
- MGIS must hide details about the programming model, memory allocation, data access, etc...
  - MGIS shall not impose any programming model. The latter shall be choosen at runtime in most cases.
- For crystal plasticity, it may be required to consider material zones to describe grains...
- The loop over the integration point must be part of the generated code. That's fair for FFT solvers.
- There are still many open questions in FEM:
  - The memory to store 10^6 tangent operators in a 3D finite strain analysis is huge: $10^6 \cdots 81 \cdots 8 \approx 650\ Mb$.
    - This may be required by matrix-free solvers...
    - Cast3M-like accelerated fixed-point algorithms are worth considering, in particular if we consider that the elastic matrix is just a preconditioner (incomplete simple precision decomposition may just also work).
- See also Manta's talk for another point of view.

What's going on for TFEL 5.0 and MGIS 3.0 ?

Port to GPUs

- See the previous challenges !
- Extension of the JIT abilities introduced in TFEL 4.1.

Port to C++ -23

- Time for a great overhaul of `TFEL/Math` and `TFEL/Material` !
  - As always, backward compatibility must be preserved.
- Concepts to replace SFINAE pattern.
  - `trace` implementation in TFEL 4.2:

```
template <typename StensorType>
constexpr std::enable_if_t<implementsStensorConcept<StensorType>(),
                           numeric_type<StensorType>>
trace(const StensorType& s) {
  return s(0) + s(1) + s(2);
}
```

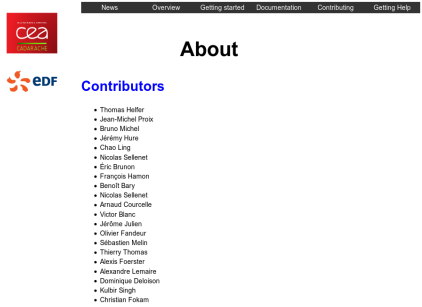
  - `trace` implementation in TFEL 5.0:

```
constexpr auto trace(const StensorConcept auto& s) {
  return s(0) + s(1) + s(2);
}
```

- Multidimensional `operator[]`

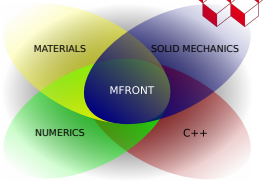
Conclusions

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)

cea iresne



Thanks for you attention ! Any questions?

Initialize functions for behaviours (experimental)

```
@InitializeFunction ElasticStrainFromInitialStress {
  const auto K = 2 / (3 * (1 - 2 * nu));
  const auto pr = trace(sig) / 3;
  const auto s = deviator(sig);
  eel = eval((pr / K) * Stensor::Id() + s / mu);
}
```

- The `@InitializeFunction` keyword introduces a code block that can be used to initialize internal state variables at the very beginning of the computation.
  - A behaviour can define many initialize functions that can be called individually by the calling solver.

Initialize functions for behaviours (experimental)

```
@InitializeFunction ElasticStrainFromInitialStress {
  const auto K = 2 / (3 * (1 - 2 * nu));
  const auto pr = trace(sig) / 3;
  const auto s = deviator(sig);
  eel = eval((pr / K) * Stensor::Id() + s / mu);
}
```

- The `@InitializeFunction` keyword introduces a code block that can be used to initialize internal state variables at the very beginning of the computation.
  - A behaviour can define many initialize functions that can be called individually by the calling solver.
- Initalize functions take sthe state at the beginning of the time step (all increments are null) and update the value of the state variables.

Initialize functions for behaviours (experimental)

```
@InitializeFunction ElasticStrainFromInitialStress {
  const auto K = 2 / (3 * (1 - 2 * nu));
  const auto pr = trace(sig) / 3;
  const auto s = deviator(sig);
  eel = eval((pr / K) * Stensor::Id() + s / mu);
}
```

- The `@InitializeFunction` keyword introduces a code block that can be used to initialize internal state variables at the very beginning of the computation.
  - A behaviour can define many initialize functions that can be called individually by the calling solver.
- Initalize functions take sthe state at the beginning of the time step (all increments are null) and update the value of the state variables.
- Initalize functions may also have dedicated intputs (called initialize function variables) introduced by the `@InitializeFunctionVariable`.
  - An initialize function variable can be common to several initialize functions.

Initialize functions for behaviours (experimental)

```
@InitializeFunction ElasticStrainFromInitialStress {
  const auto K = 2 / (3 * (1 - 2 * nu));
  const auto pr = trace(sig) / 3;
  const auto s = deviator(sig);
  eel = eval((pr / K) * Stensor::Id() + s / mu);
}
```

- The `@InitializeFunction` keyword introduces a code block that can be used to initialize internal state variables at the very beginning of the computation.
  - A behaviour can define many initialize functions that can be called individually by the calling solver.
- Initalize functions take sthe state at the beginning of the time step (all increments are null) and update the value of the state variables.
- Initalize functions may also have dedicated intputs (called initialize function variables) introduced by the `@InitializeFunctionVariable`.
  - An initialize function variable can be common to several initialize functions.
- Initialize functions are only supported by the `generic` interface.

Post-processings of behaviours (experimental)

```
// ! principal strains
@PostProcessingVariable tvector<3u,strain> ep;
ep.setEntryName("PrincipalStrain");
// ! compute the principal strain
@PostProcessing PrincipalStrain {
  ep = eto.computeEigenValues();
}
```

- The `@PostProcessing` keyword introduces a code block that can be used to perform computations in a separate step of the behaviour integration.

Post-processings of behaviours (experimental)

```
// ! principal strains
@PostProcessingVariable tvector<3u,strain> ep;
ep.setEntryName("PrincipalStrain");
// ! compute the principal strain
@PostProcessing PrincipalStrain {
  ep = eto.computeEigenValues();
}
```

- The `@PostProcessing` keyword introduces a code block that can be used to perform computations in a separate step of the behaviour integration.
- The outputs of post-processings are stored in so-called post-processing variables declared by the `@PostProcessingVariable`.

Options to domain specific languages

```
@DSL Default{parameters_as_static_variables : true};
```

- Options to domain specific languages modify their default behaviour:
  - The goal is inhibit some features (for instance, the modification of the parameters from a text file).

Options to domain specific languages

```
@DSL Default{parameters_as_static_variables : true};
```

- Options to domain specific languages modify their default behaviour:
  - The goal is inhibit some features (for instance, the modification of the parameters from a text file).
  - Customize the compilation for performances (for instance, treating parameters as static variables).

Options to domain specific languages

```
@DSL Default{parameters_as_static_variables : true};
```

- Options to domain specific languages modify their default behaviour:
  - The goal is inhibit some features (for instance, the modification of the parameters from a text file).
  - Customize the compilation for performances (for instance, treating parameters as static variables).
  - Paves the way toward computations on GPUs and on the fly compilation of behaviours.

Options to domain specific languages

```
@DSL Default{parameters_as_static_variables : true};
```

- Options to domain specific languages modify their default behaviour:
  - The goal is inhibit some features (for instance, the modification of the parameters from a text file).
  - Customize the compilation for performances (for instance, treating parameters as static variables).
  - Paves the way toward computations on GPUs and on the fly compilation of behaviours.
- The list of available options for a DSL can be retrieved as follows:

```
$ mfront --list-dsl-options=RungeKutta
```

Options to domain specific languages - II

```
$ mfront --obuild --interface=generic                                \
         --behaviour-dsl-option=parameters_as_static_variables:true \
         --behaviour-dsl-option='overriding_parameters:{T:293.15}'  \
         Plasticity.mfront
```

- DSL options can be specified in a block after the definition of the DSL or on the command line (see `MFrontGallery` project):
  - `--dsl-option`
  - `--material-property-dsl-option`
  - `--behaviour-dsl-option`
  - `--model-dsl-option`

Options to domain specific languages - II

```
$ mfront --obuild --interface=generic                                \
         --behaviour-dsl-option=parameters_as_static_variables:true \
         --behaviour-dsl-option='overriding_parameters:{T:293.15}'  \
         Plasticity.mfront
```

- DSL options can be specified in a block after the definition of the DSL or on the command line (see `MFrontGallery` project):
  - `--dsl-option`
  - `--material-property-dsl-option`
  - `--behaviour-dsl-option`
  - `--model-dsl-option`
- DSL Options can also gathered in an JSON-like file:

```
$ mfront --obuild --interface=generic                \
         --behaviour-dsl-options-file=options.json \
         Plasticity.mfront
```

where the file `options.json` file may look like:

```
overriding_parameters : {T : 293.15, dT : 0},
parameters_as_static_variables : true
```

References I

[1] Mohammad Abbas, Benoît Bary, and Ludovic Jason. “3D mesoscale analysis of the effects of steel bar ribs geometry on reinforced concrete bond strength”. In: Finite Elements in Analysis and Design 219 (July 1, 2023), p. 103928. ISSN: 0168-874X. DOI: 10.1016/j.finel.2023.103928. URL: https://www.sciencedirect.com/science/article/pii/S0168874X23000215 (visited on 04/13/2023).

References II

[2] Dylan Agius et al. “An experimental and computational study into strain localisation in beta-annealed Ti-6Al-4V”. en. In: Procedia Structural Integrity. 17th Asia-Pacific Conference on Fracture and Strength and the 13th Conference on Structural Integrity and Failure (APCFS 2022 & SIF 2022) 45 (Jan. 2023), pp. 4–11. ISSN: 2452-3216. DOI: 10.1016/j.prostr.2023.05.002. URL: https://www.sciencedirect.com/science/article/pii/S2452321623003190 (visited on 06/24/2023).

References III

[3] Hanen Amor, Jean-Jacques Marigo, and Corrado Maurini. “Regularized formulation of the variational brittle fracture with unilateral contact: Numerical experiments”. In: Journal of the Mechanics and Physics of Solids 57.8 (Aug. 2009), pp. 1209–1229. ISSN: 0022-5096. DOI: 10.1016/j.jmps.2009.04.011. URL: http://www.sciencedirect.com/science/article/pii/S0022509609000659 (visited on 10/23/2014).

[4] G. Bacquaert et al. “A standard thermodynamic-based extension of the Modified Cam-Clay soil model and its applications”. In: European Journal of Mechanics - A/Solids (Sept. 1, 2023), p. 105122. ISSN: 0997-7538. DOI: 10.1016/j.euromechsol.2023.105122. URL: https://www.sciencedirect.com/science/article/pii/S0997753823002140 (visited on 09/03/2023).

References IV

[5] C. B. Fokam et al. “Implementation by an Implicit Approach of an Elastoplastic Behavior Law in the Finite Element Cast3M Code”. In: International Applied Mechanics 59.1 (Jan. 1, 2023), pp. 124–129. ISSN: 1573-8582. DOI: 10.1007/s10778-023-01206-0. URL: https://doi.org/10.1007/s10778-023-01206-0 (visited on 09/12/2023).

[6] Thomas Helfer et al. “Implicit integration of the constitutive equations of a polycrystal obtained by the Berveiller-Zaoui homogeneization scheme”. In: 15ème colloque national en calcul des structures. 83400 Hyères-les-Palmiers, France: Université Polytechnique Hauts-de-France [UPHF], May 2022. URL: https://hal.science/hal-03718089.

References V

[7] Karol Lewandowski et al. “Multifield finite strain plasticity: Theory and numerics”. In: Computer Methods in Applied Mechanics and Engineering 414 (Sept. 1, 2023), p. 116101. ISSN: 0045-7825. DOI: 10.1016/j.cma.2023.116101. URL: https://www.sciencedirect.com/science/article/pii/S0045782523002256 (visited on 08/02/2023).

[8] Ronan Madec et al. “Plastic anisotropy and composite slip: Application to uranium dioxide”. In: Acta Materialia (May 18, 2023), p. 119016. ISSN: 1359-6454. DOI: 10.1016/j.actamat.2023.119016. URL: https://www.sciencedirect.com/science/article/pii/S1359645423003476 (visited on 05/22/2023).

References VI

[9] Christian Miehe, Martina Hofacker, and Fabian Welschinger. “A phase field model for rate-independent crack propagation: Robust algorithmic implementation based on operator splits”. In: Computer Methods in Applied Mechanics and Engineering 199.45 (Nov. 15, 2010), pp. 2765–2778. ISSN: 0045-7825. DOI: 10.1016/j.cma.2010.04.011. URL: http://www.sciencedirect.com/science/article/pii/S0045782510001283 (visited on 07/01/2016).

[10] Panteleimon Rapanakis et al. “A three-dimensional numerical modelling of an underground gallery excavation considering the influence of sedimentary rock cross-anisotropy”. In: 10th European Conference on Numerical Methods in Geotechnical Engineering (NUMGE 2023). London, June 2023. DOI: 10.53243/NUMGE2023-78.

References VII

[11] L. Riparbelli et al. “Coupling numerical and experimental methods to characterise the mechanical behaviour of the Mona Lisa: a method to enhance the conservation of panel paintings”. In: Journal of Cultural Heritage 62 (July 1, 2023), pp. 376–386. ISSN: 1296-2074. DOI: 10.1016/j.culher.2023.06.013. URL: https://www.sciencedirect.com/science/article/pii/S1296207423001127 (visited on 07/01/2023).

[12] Lorenzo Riparbelli et al. “Hygromechanical behaviour of wooden panel paintings: classification of their deformation tendencies based on numerical modelling and experimental results”. In: Heritage Science 11.1 (Feb. 6, 2023), p. 25. ISSN: 2050-7445. DOI: 10.1186/s40494-022-00843-x. URL: https://doi.org/10.1186/s40494-022-00843-x (visited on 02/07/2023).

References VIII

[13] W. M. Scherzinger and C. R. Dohrmann. “A robust algorithm for finding the eigenvalues and eigenvectors of 3x3 symmetric matrices”. In: Computer Methods in Applied Mechanics and Engineering 197.45 (Aug. 15, 2008), pp. 4007–4015. ISSN: 0045-7825. DOI: 10.1016/j.cma.2008.03.031. URL: http://www.sciencedirect.com/science/article/pii/S0045782508001436.

[14] C. Sénac, J. Hure, and B. Tanguy. “Yield surface for void growth and coalescence of porous anisotropic materials under axisymmetric loading”. In: Journal of the Mechanics and Physics of Solids 179 (Oct. 1, 2023), p. 105365. ISSN: 0022-5096. DOI: 10.1016/j.jmps.2023.105365. URL: https://www.sciencedirect.com/science/article/pii/S0022509623001692 (visited on 11/03/2023).

References IX

[15] David Siedel. “Une approche numérique robuste pour la description de la rupture fragile et du comportement viscoplastique des crayons de combustible”. These en préparation. Université Paris sciences et lettres, 2023. URL: https://www.theses.fr/s239728 (visited on 06/23/2023).

[16] Adrien Socié et al. “A fully coupled Hydraulic Mechanical Chemical approach applied to cementitious material damage due to carbonation”. In: npj Materials Degradation 7.1 (July 27, 2023). Number: 1 Publisher: Nature Publishing Group, pp. 1–11. ISSN: 2397-2106. DOI: 10.1038/s41529-023-00378-x. URL: https://www.nature.com/articles/s41529-023-00378-x (visited on 07/29/2023).

References X

[17] François Soleilhet, Marc Quiertant, and Karim Benzarti. “Numerical Modelling of the Nonlinear Shear Creep Behavior of FRP-Concrete Bonded Joints”. In: Materials 16.2 (2023). ISSN: 1996-1944. DOI: 10.3390/ma16020801. URL: https://www.mdpi.com/1996-1944/16/2/801.

[18] Eyram Tsekpuia et al. “A microstructure-based three-scale homogenization model for predicting the elasto-viscoplastic behavior of duplex stainless steels”. In: International Journal of Plasticity (Mar. 5, 2023), p. 103575. ISSN: 0749-6419. DOI: 10.1016/j.ijplas.2023.103575. URL: https://www.sciencedirect.com/science/article/pii/S074964192300061X (visited on 03/07/2023).

References XI

[19] Lei Zhang et al. “Modeling the visco-elastoplastic behavior of deep coal based on conformable derivative”. In: Mechanics of Time-Dependent Materials (Jan. 27, 2023). ISSN: 1573-2738. DOI: 10.1007/s11043-023-09588-x. URL: https://doi.org/10.1007/s11043-023-09588-x (visited on 01/29/2023).

[20] Harald Ziegelwanger. “PCBA reliability simulation in the cloud”. In: 2023 24th International Conference on Thermal, Mechanical and Multi-Physics Sim 2023, pp. 1–7. DOI: 10.1109/EuroSimE56861.2023.10100837.