



Muesli

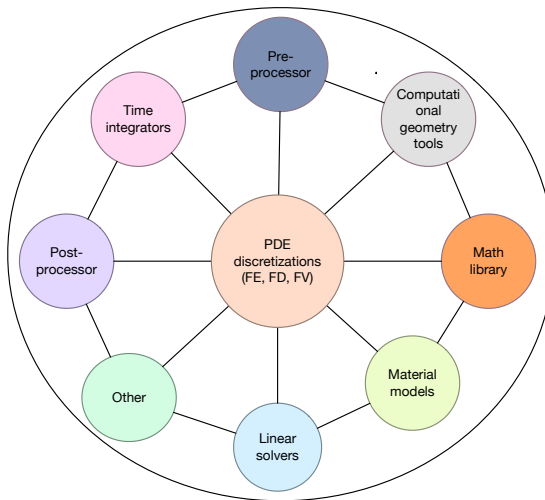
a Material UnivErSal Library

I. Romero, D. Portillo, D. del Pozo, D. Rodríguez, J. Segurado
`ignacio.romero@imdea.org`

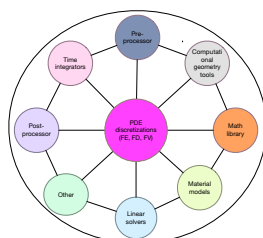
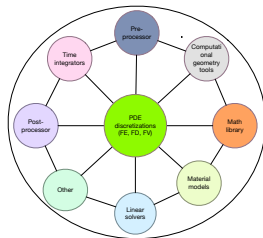
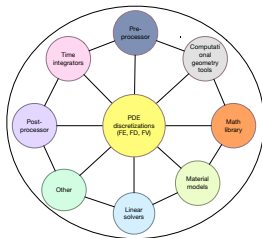
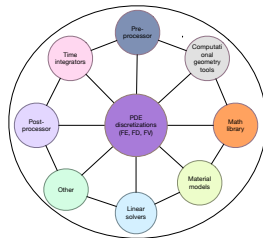
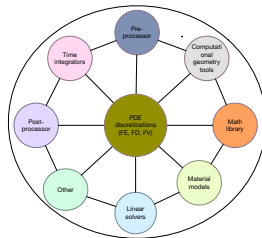
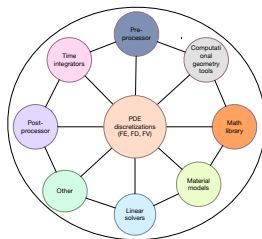
Technical University of Madrid, Spain
IMDEA Materials Institute, Getafe, Spain

ICME 2017
May 21-25, 2017. Ypsilanti, Michigan

Observation 1: The structure of a simulation code



How we build simulation codes



Open-source, collaborative projects

Linear Solvers	Pre/post	Computational geometry
SuperLU	Gmsh	CGAL
TAUCS	Paraview	LEDA
Paradiso	Salome	NESL
WSMP	[GiD]	
Non-linear solvers	Math libraries	Time integration
Petsc	Eigen	Petsc
TAO	TAO	[Matlab] / Octave
[Matlab] / Octave	BLAS/LAPACK	Intel ODE solver
Dlib	GSL	

Open-source, collaborative projects

Linear Solvers	Pre/post	Computational geometry
SuperLU	Gmsh	CGAL
TAUCS	Paraview	LEDA
Paradiso	Salome	NESL
WSMP	[GiD]	
Non-linear solvers	Math libraries	Time integration
Petsc	Eigen	Petsc
TAO	TAO	[Matlab] / Octave
[Matlab] / Octave	BLAS/LAPACK	Intel ODE solver
Dlib	GSL	

What about libraries for continuum material models?

Z-mat, Digimat, PolyUMod (closed), MFront (open) ...

Observation 2: Commercial vs. research codes

- In Computational Mechanics, progress in **commercial** and **research** codes runs at a different pace.
- **Commercial codes** are more robust, have many more features, but are “black boxes” with narrow open doors \Rightarrow **not best for development**
- **Research codes** are open, but often lack features, and have less benchmarks \Rightarrow **not best for general purposes**

Observation 2: Commercial vs. research codes

- In Computational Mechanics, progress in **commercial** and **research** codes runs at a different pace.
- **Commercial codes** are more robust, have many more features, but are “black boxes” with narrow open doors \Rightarrow **not best for development**
- **Research codes** are open, but often lack features, and have less benchmarks \Rightarrow **not best for general purposes**

Why material libraries are not shared?

It takes time/effort to tailor the needs to each code.

An exception: UMATs

- Many codes open the door to user material models or UMATs
- They are often developed in Fortran 77 (Abaqus, LS-DYNA, ANSYS)
- They all have different interfaces
- Make no provisions for benchmarking and testing
- If developed in a research code, they must be rewritten.

Observation 3: Coding material models

- In research and commercial codes **the general purpose materials are always the same** (Neohookean, viscoplastic, Mooney-Rivlin, Fourier, Newtonian fluid, ...)
- Most of them are **not that difficult** to implement, but **still need to be verified**.
- Thousands (?) of coding hours are spent in implementing/testing material models that are standard, and programmed elsewhere.
- As a result, each code ends with its own (non-transferable) implementation of Neohookean, viscoplastic, Mooney-Rivlin, etc.

Summary

Observations:

1. There is no freely available library of continuum constitutive models of materials, with the standard ones, and easily extensible.
2. Software development done in research codes is not automatically transferred to commercial codes.
3. Lack of sharing is not good for reliability.

Summary

Observations:

1. There is no freely available library of continuum constitutive models of materials, with the standard ones, and easily extensible.
2. Software development done in research codes is not automatically transferred to commercial codes.
3. Lack of sharing is not good for reliability.

There is an opportunity to develop a library of continuum constitutive models that is:

- Open-source
- Easily extensible
- Plug-able
- Reliable

Outline

- 1 Motivation
- 2 Structure
- 3 Interfacing
- 4 Reliability
- 5 Collaboration

Class structure

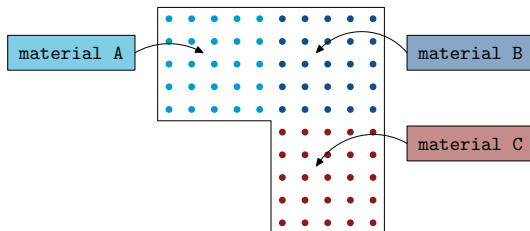
- MUESLI is an **object-oriented** library, written in C++.
- The class structure reflects the design philosophy of the library.
- At the highest level, there are **material families**, associated with the BVP they are supposed to be used for (solid mechanics, fluid mechanics, thermal problems, coupled thermomechanical, ...).
- The functionality and design of the material families are similar, but they independent.
- More families can be added, and more materials to each family as well (**fully extensible**).

Main concepts: material and material point

- Building blocks of MUESLI: `materials` and `materialPoints`
- `material` is the abstraction corresponding to a “material type”. Often, only a few per model.
- `materialPoint` corresponds to the abstraction of a specific physical point, of a given `material` and with a state that evolves in time. Often, thousands/millions per model.

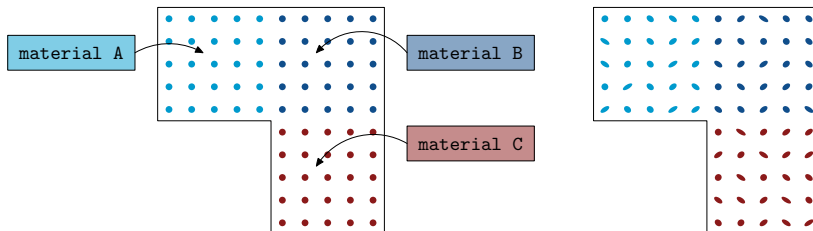
Main concepts: material and material point

- Building blocks of MUESLI: materials and materialPoints
- `material` is the abstraction corresponding to a “material type”. Often, only a few per model.
- `materialPoint` corresponds to the abstraction of a specific physical point, of a given `material` and with a state that evolves in time. Often, thousands/millions per model.



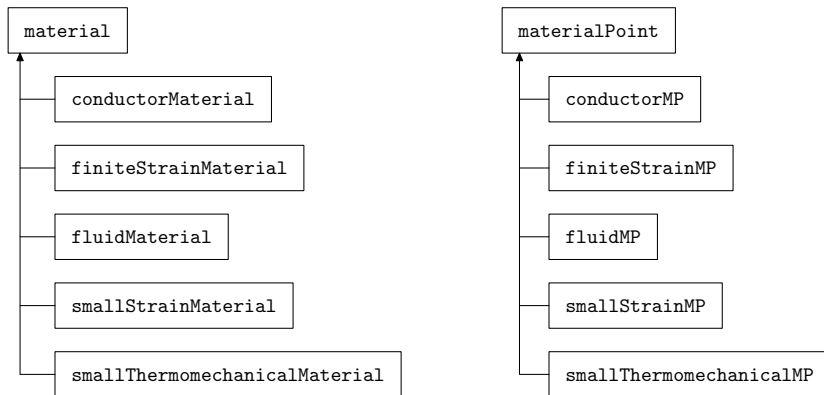
Main concepts: material and material point

- Building blocks of MUESLI: materials and materialPoints
- `material` is the abstraction corresponding to a “material type”. Often, only a few per model.
- `materialPoint` corresponds to the abstraction of a specific physical point, of a given `material` and with a state that evolves in time. Often, thousands/millions per model.

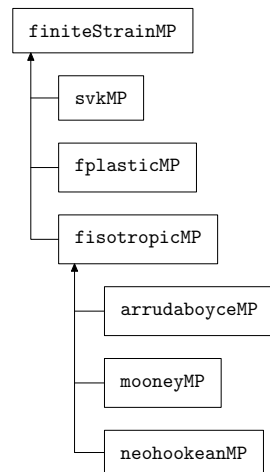
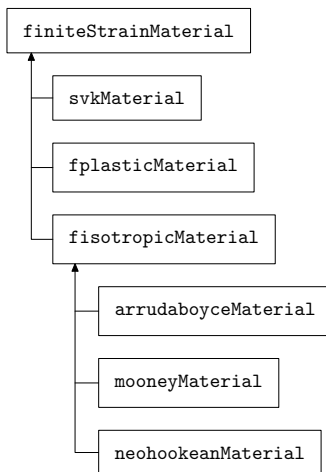


Material families

- MUESLI defines pairs of material/materialPoint. Currently:



Example: finite strain materials



High level abstraction

- MUESLI includes tensor classes to simplify and speed up the implementation of new constitutive models
- Classes for vectors, tensors (2nd and 4th order, symmetric, skew) and rotations with operator overload and dozens of utility functions.
- **Code resembles equations!** Example:
 - ▷ $\boldsymbol{\sigma} = 2\mu \boldsymbol{\varepsilon} + \lambda \text{tr}(\boldsymbol{\varepsilon}) \mathbf{I}$
 - ▷ `sigma = 2.0*mu*eps + lambda*eps.trace()*Id;`
- High-performance library EIGEN can be used.

Adding materials to MUESLI

- Each material family has its own requirements. Some functions are mandatory, other optional (library provides default).
- **Example:** `finiteStrainMaterial` class

finiteStrainMaterial

```
set/update/commit State  
compute Energy, compute Dissipation  
compute Cauchy Stress  
compute material Tangent
```

```
First Piola-Kirchhoff stress  
Second Piola-Kirchhoff stress  
Kirchhoff stress  
Stress vector (Voigt)  
Spatial tangent  
Tangent contractions...  
Tangent matrices (Voigt)
```

Extending/interfacing MUESLI

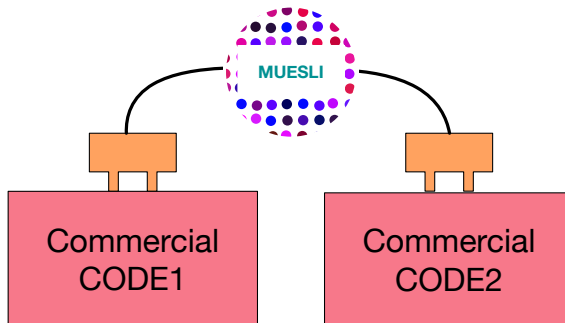
Extending:

- ▷ MUESLI is designed so that it “easy” to add new material models.
- ▷ Only a few key functions need to be programmed (pure virtual). The library provides many extra ones.

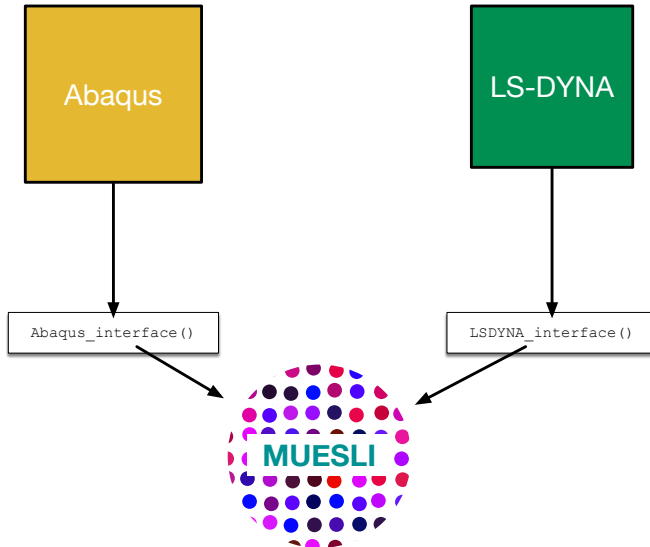
Interfacing:

- ▷ Code is exposed: including it in existing codes is simple.
- ▷ **The same material models** can be employed with commercial codes by designing the appropriate interfaces.

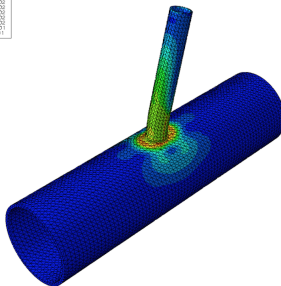
Two ways of using MUESLI



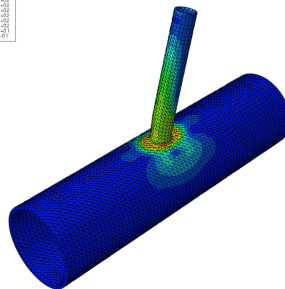
Current interfaces



Example: Abaqus



Solution with Abaqus' material



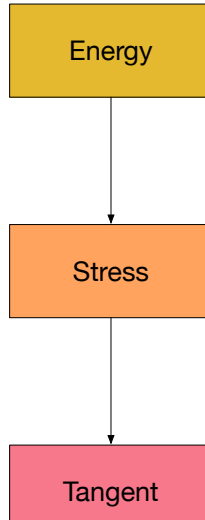
Solution with MUESLI's material

Automatic testing

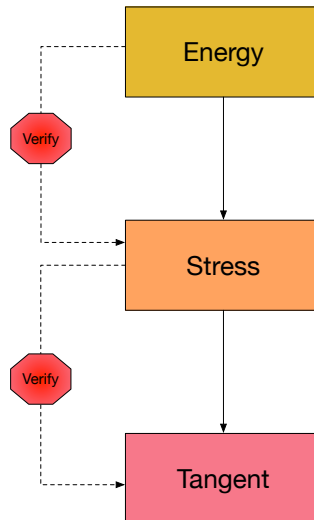
Goal: to make MUESLI as reliable as possible:

- Include **automatic checks** for every material model.
 - ▷ Each family has different tests.
 - ▷ Check the **consistency** of all the implemented functions.
- Extremely useful to debug consistent tangents in complex nonlinear behavior (e.g., finite strain plasticity)
- Every user material added to the library can be tested.

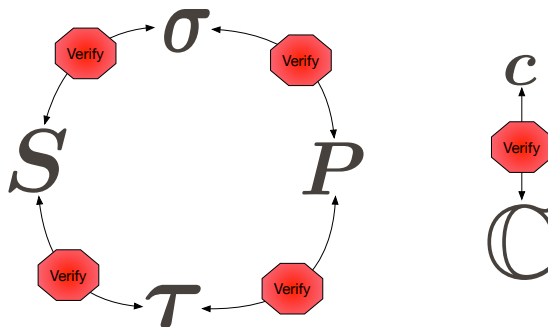
Example: checks for solid mechanics



Example: checks for solid mechanics



Example: checks for solid mechanics



Test driver output

```

MUESLI  TESTS

Test done on : Mon Jun 6 18:47:58 2016
Username      : ignacio
Hostname      : mafalda.local
OS            : UNKNOWN OS

-----
Testing smallstrain elastic material
-----
1. Comparing stress with DWeff. Test passed.
2. Comparing tensor C with DStress. Test passed.
3. Comparing stress tensor and Voigt stress. Test passed.
4. Comparing contract tangent with C_ijkl v_j w_l. Test passed.
5. Comparing volumetric tangent with (1/9) C_iiij. Test passed.
6. Comparing specific and generic dev tangent contraction Test passed.
6. Comparing tangent tensor and Voigt matrix. Test passed.
7. Comparing uniaxial stress and derivative of energy (under development). Test passed.
8. Comparing uniaxial stiffness and derivative of stress (under development) Test passed.

-----
Testing smallstrain visco elastic material
-----
1. Comparing stress with DWeff. Test passed.
2. Comparing tensor C with DStress. Test passed.
3. Comparing stress tensor and Voigt stress. Test passed.
4. Comparing contract tangent with C_ijkl v_j w_l. Test passed.
5. Comparing volumetric tangent with (1/9) C_iiij. Test passed.
6. Comparing specific and generic dev tangent contraction Test passed.
6. Comparing tangent tensor and Voigt matrix. Test passed.
7. Comparing uniaxial stress and derivative of energy (under development). Test passed.
8. Comparing uniaxial stiffness and derivative of stress (under development) Test passed.

-----
Testing smallstrain elasto plastic material
-----
Drucker-Prager type
1. Comparing stress with DWeff. Test passed.
2. Comparing tensor C with DStress. Test passed.
3. Comparing stress tensor and Voigt stress. Test passed.
4. Comparing contract tangent with C_ijkl v_j w_l. Test passed.
5. Comparing volumetric tangent with (1/9) C_iiij. Test passed.
6. Comparing specific and generic dev tangent contraction Test passed.
:

```

MUESLI as a collaborative project

- MUESLI is distributed under GPL3 licence
- Contributions to the project of new materials are welcome (credits will be given)
- Contribution of new interfaces (to other commercial codes) are also welcome.

Current status

● Version 1.2 of MUESLI has been released in 2017. **Current work:**

- **More:**

- ▷ Constitutive models: damage, coupled finite strain, more plasticity criteria, crystal plasticity, complex fluids, ...)
- ▷ Interfaces: Ansys, Nastran, ...
- ▷ 1D / 2D relations

- **Better:**

- ▷ Documentation
- ▷ Collaborative environment

Current status

● Version 1.2 of MUESLI has been released in 2017. **Current work:**

- **More:**

- ▷ Constitutive models: damage, coupled finite strain, more plasticity criteria, crystal plasticity, complex fluids, ...)
- ▷ Interfaces: Ansys, Nastran, ...
- ▷ 1D / 2D relations

- **Better:**

- ▷ Documentation
- ▷ Collaborative environment

Contributions will be added with due credit!

Summary: MUESLI's features

Proprietary code	➡	Open source, GPL 3.0
Fortran 77	➡	Object oriented, C++ (with abstract operators)
Code specific	➡	Easy to plug into existing codes
Research vs. commercial	➡	Same routines for both
Untested	➡	Automatic checks (also debugging tools)

<http://www.materials.imdea.org/Muesli>



Muesli

a Material UnivErSal Library

I. Romero, D. Portillo, D. del Pozo, D. Rodríguez, J. Segurado
`ignacio.romero@imdea.org`

Technical University of Madrid, Spain
IMDEA Materials Institute, Getafe, Spain

ICME 2017
May 21-25, 2017. Ypsilanti, Michigan