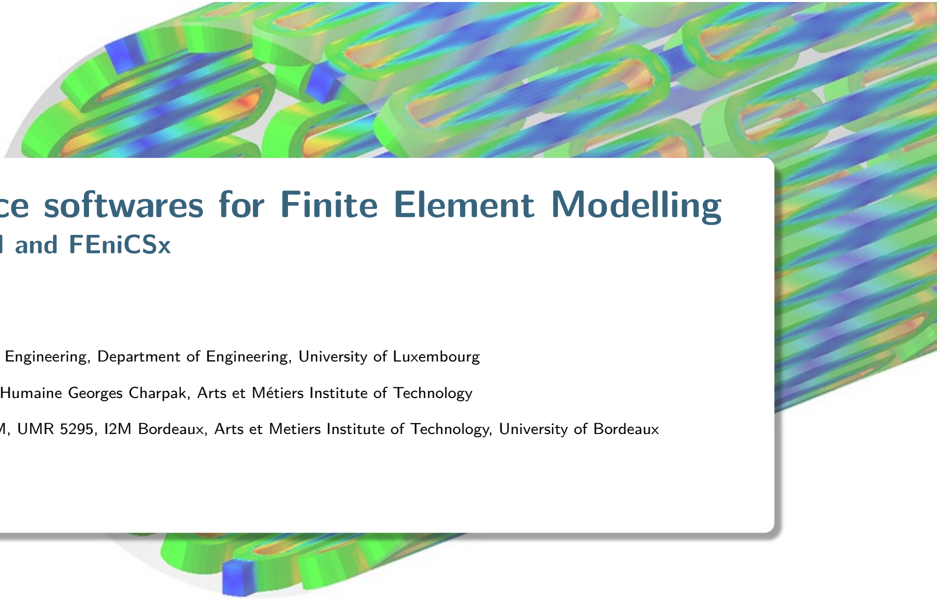


Open-Source softwares for Finite Element Modelling

Focus on GMSH and FEniCSx

T. Lavigne[1,2,3]

[1] Institute of Computational Engineering, Department of Engineering, University of Luxembourg

[2] Institut de Biomécanique Humaine Georges Charpak, Arts et Métiers Institute of Technology

[3] CNRS, Bordeaux INP, I2M, UMR 5295, I2M Bordeaux, Arts et Metiers Institute of Technology, University of Bordeaux

09-13/09/2024

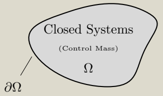
Github link

Finite Element Modelling:

Reminders for the method of Finite Element Method.

Closed and Open Systems

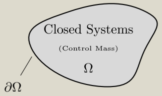
A system is a quantity of mass. The complement of a system, *i.e.* the mass or region outside the system, is the surrounding. [Holzapfel, 2002]



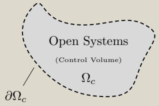
Closed system: fixed amount of mass in Ω which depends on time t. No mass can cross its boundary, but energy can cross the boundary, $\partial\Omega$.

Closed and Open Systems

A system is a quantity of mass. The complement of a system, *i.e.* the mass or region outside the system, is the surrounding. [Holzapfel, 2002]



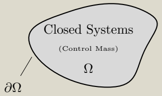
Closed system: fixed amount of mass in Ω which depends on time t. No mass can cross its boundary, but energy can cross the boundary, $\partial\Omega$.



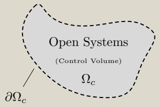
Open system: fixed amount of volume in Ω_c which is independent of time t. Mass and energy can cross the enclosing boundary (control surface), $\partial\Omega_c$.

Closed and Open Systems

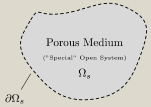
A system is a quantity of mass. The complement of a system, *i.e.* the mass or region outside the system, is the surrounding. [Holzapfel, 2002]



Closed system: fixed amount of mass in Ω which depends on time t. No mass can cross its boundary, but energy can cross the boundary, $\partial\Omega$.



Open system: fixed amount of volume in Ω_c which is independent of time t. Mass and energy can cross the enclosing boundary (control surface), $\partial\Omega_c$.



A porous medium: special Open System in which the control volume, Ω_s , is that occupied by the solid scaffold. Being the solid deformable Ω_s and $\partial\Omega_s$ depends on time t.

Strong form

The objective is to identify an unknown in the domain (temperature, displacement, *etc.*). To describe the behaviour of the system, continuous conservation equations, also referred as strong forms, have been introduced:

- ➢ Conservation equations of mass,
- ➢ Conservation equations of momentum,
- ➢ Conservation equations of energy.

Strong form

The objective is to identify an unknown in the domain (temperature, displacement, *etc.*). To describe the behaviour of the system, continuous conservation equations, also referred as strong forms, have been introduced:

- Conservation equations of mass,
- Conservation equations of momentum,
- Conservation equations of energy.

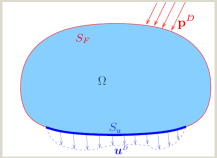
Boundary conditions can be of three types:
Dirichlet ($u = uD$ on S_U), Neumann ($\sigma \cdot n = p^D$ on S_F), Robin ($au + b\nabla u \cdot n = g$ on $\partial\Omega$).

Strong form

The objective is to identify an unknown in the domain (temperature, displacement, *etc.*). To describe the behaviour of the system, continuous conservation equations, also referred as strong forms, have been introduced:

- ➢ Conservation equations of mass,
- ➢ Conservation equations of momentum,
- ➢ Conservation equations of energy.

Boundary conditions can be of three types: Dirichlet ($u = uD$ on S_U), Neumann ($\sigma \cdot n = p^D$ on S_F), Robin ($au + b\nabla u \cdot n = g$ on $\partial\Omega$).



The conservation equations of momentum of a mechanical material would be:

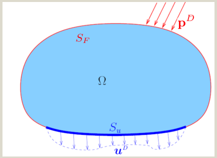
$$\nabla \cdot \underline{\sigma}(\underline{u}) + \underline{f} = \rho\underline{\gamma}$$

Strong form

The objective is to identify an unknown in the domain (temperature, displacement, *etc.*). To describe the behaviour of the system, continuous conservation equations, also referred as strong forms, have been introduced:

The fields describing the behaviour in the domain are continuous in their local form. As a consequence, an infinity of solutions must be computed to exactly solve the PDEs and analytical solutions are not always available (discontiinuities, singularities...): **Approximations are required.**

Two main methods have been developed: the discrete element method (DEM) and the finite element method (FEM). We focus on the second one.



The conservation equations of momentum of a mechanical material would be:

$$\nabla \cdot \underline{\underline{\sigma}}(\underline{u}) + \underline{f} = \rho \underline{\gamma}$$

Variational (weak) form

To assess the strong form in a whole domain, the variational/weak forms are often used in continuum mechanics. Several admissible function spaces must be defined.

➢ The displacement field must be **sufficiently regular** (*i.e.* of bounded energy):

$$\mathcal{C} = \left\{ \underline{v} | \underline{v} \text{ continuous over } \Omega, \int_\Omega \underline{\underline{\sigma}}(\underline{v}) : \underline{\underline{\varepsilon}}(\underline{v}) \mathrm{d}V < +\infty \right\}$$

Variational (weak) form

To assess the strong form in a whole domain, the variational/weak forms are often used in continuum mechanics. Several admissible function spaces must be defined.

- The displacement field must be **sufficiently regular** (*i.e.* of bounded energy):

$$\mathcal{C} = \left\{ \underline{v} | \underline{v} \text{ continuous over } \Omega, \int_\Omega \underline{\underline{\sigma}}(\underline{v}) : \underline{\underline{\varepsilon}}(\underline{v}) \mathrm{d}V < +\infty \right\}$$

- **Kinematic compatibility** space:

$$\mathcal{C}(\underline{u}^D) = \left\{ \underline{v} | \underline{v} \in \mathcal{C} \text{ and } \underline{v} = \underline{u}^D \text{ on } S_U \right\}$$

Variational (weak) form

To assess the strong form in a whole domain, the variational/weak forms are often used in continuum mechanics. Several admissible function spaces must be defined.

- ➢ The displacement field must be **sufficiently regular** (*i.e.* of bounded energy):

$$\mathcal{C} = \left\{ \underline{v} | \underline{v} \text{ continuous over } \Omega, \int_\Omega \underline{\underline{\sigma}}(\underline{v}) : \underline{\underline{\varepsilon}}(\underline{v}) \mathrm{d}V < +\infty \right\}$$

- ➢ **Kinematic compatibility** space:

$$\mathcal{C}(\underline{u}^D) = \left\{ \underline{v} | \underline{v} \in \mathcal{C} \text{ and } \underline{v} = \underline{u}^D \text{ on } S_U \right\}$$

- ➢ **Static compatibility** space:

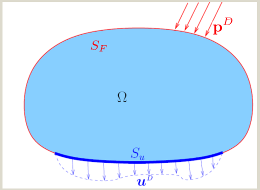
$$\mathcal{S}(\underline{p}^D, \underline{f}) = \left\{ \underline{\underline{\tau}} | \nabla \cdot \underline{\underline{\tau}} + \underline{f} = \rho \underline{\gamma} \text{ in } \Omega \text{ and } \underline{\underline{\tau}} \cdot \underline{n} = \underline{p}^D \text{ on } S_F \right\}$$

Variational (weak) form

To construct the weak form, the main idea is to **integrate the equilibrium equations multiplying it by a test function**. Let $\underline{w}$ be this function. For the sake of simplicity, we take $\underline{\gamma} = \underline{0}$.

$$\int_\Omega \underline{w}^T \cdot \left(\nabla \cdot \underline{\underline{\sigma}}(\underline{u}) + \underline{f}\right) \mathrm{d}V = 0, \ \forall \underline{w} \in \mathcal{C}$$

$$\underset{\text{part int}}{\Longrightarrow} -\int_\Omega \underline{\underline{\sigma}}(\underline{u}) : \underline{\underline{\varepsilon}}(\underline{w}) \mathrm{d}V + \int_\Omega \underline{w}^T \cdot \underline{f} \mathrm{d}V = 0, \ \forall \underline{w} \in \mathcal{C}$$



Strong Form:

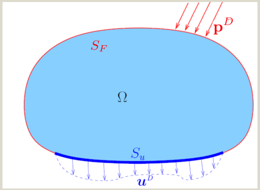
$$\nabla \cdot \underline{\underline{\sigma}}(\underline{u}) + \underline{f} = \rho \underline{\gamma}$$

Variational (weak) form

To construct the weak form, the main idea is to **integrate the equilibrium equations multiplying it by a test function**. Let $\underline{w}$ be this function. For the sake of simplicity, we take $\underline{\gamma} = \underline{0}$.

Adding the boundary conditions, it becomes:

$$\int_\Omega \underline{\underline{\sigma}}(\underline{u}) : \underline{\underline{\varepsilon}}(\underline{w})\mathrm{d}V - \int_\Omega \underline{w}^T \cdot \underline{f}\mathrm{d}V - \int_{S_U} \underline{w}^T \cdot \underline{p}\mathrm{d}S - \int_{S_F} \underline{w}^T \cdot \underline{p^D}\mathrm{d}S = 0, \forall \underline{w} \in \mathcal{C}$$



Strong Form:

$$\nabla \cdot \underline{\underline{\sigma}}(\underline{u}) + \underline{f} = \rho\underline{\gamma}$$

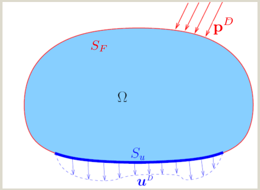
Variational (weak) form

To construct the weak form, the main idea is to **integrate the equilibrium equations multiplying it by a test function**. Let $\underline{w}$ be this function. For the sake of simplicity, we take $\underline{\gamma} = \underline{0}$.

Adding the boundary conditions, it becomes:

$$\int_\Omega \underline{\underline{\sigma}}(\underline{u}) : \underline{\underline{\varepsilon}}(\underline{w}) \mathrm{d}V - \int_\Omega \underline{w}^T \cdot \underline{f} \mathrm{d}V$$
$$- \int_{S_U} \underline{w}^T \cdot \underline{p} \mathrm{d}S - \int_{S_F} \underline{w}^T \cdot \underline{p}^D \mathrm{d}S = 0, \forall \underline{w} \in \mathcal{C}$$

There is no explicit reference to the displacement boundary condition. The pressure on S_U is unknown. It is the reaction force due to the imposed displacement.



Strong Form:

$$\nabla \cdot \underline{\underline{\sigma}}(\underline{u}) + \underline{f} = \rho \underline{\gamma}$$

Remove the unknown

This is the classical frame of the finite element method. The test field is constrained to kinematic fields admissible to 0 such that:

$$\mathcal{C}(\underline{0}) = \{\underline{w} | \underline{w} \in \mathcal{C} \text{ and } \underline{w} = \underline{0} \text{ on } S_U\}$$

$\underline{u}$ still verifies $\underline{u} = \underline{u}^D$ on S_U. The problem can therefore be re-written as:

Find $\underline{u} \in \mathcal{C}(\underline{u}^D)$ such that

$$\int_\Omega \underline{\underline{\sigma}}(\underline{u}) : \underline{\underline{\varepsilon}}(\underline{w}) \mathrm{d}V - \int_\Omega \underline{w}^T \cdot \underline{f} \mathrm{d}V - \int_{S_F} \underline{w}^T \cdot \underline{p}^D \mathrm{d}S = 0, \forall \underline{w} \in \mathcal{C}(\underline{0})$$

Weak imposed Displacement

Another method is to impose the displacement in a weak equation:

$$\int_{S_U} (\underline{u} - \underline{u}^D)^T \cdot \underline{t} \mathrm{d}S = 0, \; \forall \underline{t} \in \mathcal{C}'(S_U)$$

where $\mathcal{C}'(S_U) = \left\{ \underline{t} | \int_{S_U} \underline{w}^T \underline{t} \mathrm{d}S < +\infty, \; \forall w \in \mathcal{C} \right\}$.

The mixed problem to be solved is then:

Find $(\underline{u}, \underline{t}) \in \mathcal{C} \times \mathcal{C}'(S_U)(\underline{u}^D)$ such that

$$\int_\Omega \underline{\underline{\sigma}}(\underline{u}) : \underline{\underline{\varepsilon}}(\underline{w}) \mathrm{d}V - \int_\Omega \underline{w}^T \cdot \underline{f} \mathrm{d}V - \int_{S_U} \underline{w}^T \cdot \underline{p} \mathrm{d}S - \int_{S_F} \underline{w}^T \cdot \underline{p}^D \mathrm{d}S = 0, \; \forall \underline{w} \in \mathcal{C}$$

$$\int_{S_U} \underline{u}^T \cdot \underline{t} \mathrm{d}S = \int_{S_U} [\underline{u}^D]^T \cdot \underline{t} \mathrm{d}S, \; \forall \underline{t} \in \mathcal{C}'(S_U)$$

Construction of an approximation subspace

To obtain the numerical solution, one first need to construct an approximation subspace. For example:

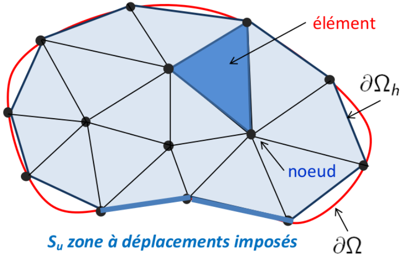
$$\underline{v}(\underline{x}) = \underline{u}^{(D)}(\underline{x}) + \sum_{k=1}^{N} \underline{\alpha}^{k} \varphi_k(\underline{x}),\ \underline{u}^{(D)} \in \mathcal{C}(\underline{u}^{D}),\ \varphi_k \in \mathcal{C}(\underline{0})$$

$\underline{\alpha}^k$ being the generalised displacements (nodal displacements).

Construction of an approximation subspace

$$\underline{v}(\underline{x}) = \underline{u}^{(D)}(\underline{x}) + \sum_{k=1}^{N} \alpha^k \varphi_k(\underline{x}),\ \underline{u}^{(D)} \in \mathcal{C}(\underline{u}^D),\ \varphi_k \in \mathcal{C}(\underline{0})$$

The approximation lives in a discretized space represented by a **mesh** composed of **nodes** and **elements**.



- ➢ The union of the elements gives the domain,
- ➢ the intersection of the elements can be null,
- ➢ Continuous solutions are obtained by interpolating the nodal solutions,
- ➢ $\Omega_h \to \Omega$ when $h \to 0$

Good practice for coding:

Reminders of good practice when developing codes.

Before Starting

Dealing with FEM: You need to be careful, an FEM code can solve many equations but you need to ensure solving the right equations and to solve the equations right. To do so:

- ➢ Begin with a scheme on which you place all the boundary conditions,
- ➢ Write your model with rigour on a paper,
- ➢ Implement it,
- ➢ "Make it right then make it clean, a code that run properly is only 50% of the work": check the results with a critical look (mesh size [convergence], simplicity of the model, existing analytical solution, *etc.*).

Before Starting

Dealing with FEM: You need to be careful, an FEM code can solve many equations but you need to ensure solving the right equations and to solve the equations right. To do so:

- ➢ Begin with a scheme on which you place all the boundary conditions,
- ➢ Write your model with rigour on a paper,
- ➢ Implement it,
- ➢ "Make it right then make it clean, a code that run properly is only 50% of the work": check the results with a critical look (mesh size [convergence], simplicity of the model, existing analytical solution, *etc.*).

Before Starting

Dealing with FEM: You need to be careful, an FEM code can solve many equations but you need to ensure solving the right equations and to solve the equations right. To do so:

- ➢ Begin with a scheme on which you place all the boundary conditions,
- ➢ Write your model with rigour on a paper,
- ➢ Implement it,
- ➢ "Make it right then make it clean, a code that run properly is only 50% of the work": check the results with a critical look (mesh size [convergence], simplicity of the model, existing analytical solution, *etc.*).

Before Starting

Dealing with FEM: You need to be careful, an FEM code can solve many equations but you need to ensure solving the right equations and to solve the equations right. To do so:

- Begin with a scheme on which you place all the boundary conditions,
- Write your model with rigour on a paper,
- Implement it,
- "Make it right then make it clean, a code that run properly is only 50% of the work": check the results with a critical look (mesh size [convergence], simplicity of the model, existing analytical solution, *etc.*).

Before Starting

Here are few recommendations if you start a **new language**:

- ➢ Start with simple examples,
- ➢ Find tutorials if you can,
- ➢ Structure properly your codes and identify the difficulties.

While coding

Dos

- ☞ Structure your code,
- ☞ Use meaningful variable names,
- ☞ Put safeguards / tests,
- ☞ Create a readme file alongside,
- ☞ Search for help in forums and manual.

Dont's

- ✗ Directly start on the computer,
- ✗ Start with a complex problem,
- ✗ Feel like you have to memorise every method or every way to do something. Google is your friend,
- ✗ Hesitate to ask for help, one might have already faced the problem and find the solution.

While coding

Dos

- ☞ Structure your code,
- ☞ Use meaningful variable names,
- ☞ Put safeguards / tests,
- ☞ Create a readme file alongside,
- ☞ Search for help in forums and manual.

Dont's

- ✗ Directly start on the computer,
- ✗ Start with a complex problem,
- ✗ Feel like you have to memorise every method or every way to do something. Google is your friend,
- ✗ Hesitate to ask for help, one might have already faced the problem and find the solution.

Remark: Remember that someone might use your code one day. Make it user-friendly, it should not need any comments within the code but a readme aside if you respect the here-above recommendations. You can further have a look to this conference.

Collaborative Tools

Collaborative tools and platform exist for coding. Do not hesitate to use them and download "cheat sheets" when you start using them. This is for instance the case of **GitHub** for code sharing and development, **Docker** to use images of a specific machine configuration, **Doxygen** for automatic generation of code descriptions, *etc.* All the links can be found here.

Finite Element Software:

Commercial and Open Source Software.

Commercial and open-source software

Both commercial and open-source softwares have been developed for the mesh generation and the FE computation.

Commercial

- ☞ Pre-programmed common tools and laws,
- ☞ User-friendly design,
- ☞ Robust and stable versions that are often compatible across different releases,
- ✗ Expensive & license-based $\Longrightarrow$ reduce direct reproducibility,
- ✗ Black-Boxes & custom extensions and subroutines represent a complex task.

Commercial and open-source software

Both commercial and open-source softwares have been developed for the mesh generation and the FE computation.

Commercial

- ☞ Pre-programmed common tools and laws,
- ☞ User-friendly design,
- ☞ Robust and stable versions that are often compatible across different releases,
- ✗ Expensive & license-based $\implies$ reduce direct reproducibility,
- ✗ Black-Boxes & custom extensions and subroutines represent a complex task.

Commercial and open-source software

Both commercial and open-source softwares have been developed for the mesh generation and the FE computation.

Commercial

- ☞ Pre-programmed common tools and laws,
- ☞ User-friendly design,
- ☞ Robust and stable versions that are often compatible across different releases,
- ✗ Expensive & license-based $\Longrightarrow$ reduce direct reproducibility,
- ✗ Black-Boxes & custom extensions and subroutines represent a complex task.

Commercial and open-source software

Both commercial and open-source softwares have been developed for the mesh generation and the FE computation.

Commercial

- ☞ Pre-programmed common tools and laws,
- ☞ User-friendly design,
- ☞ Robust and stable versions that are often compatible across different releases,
- ✗ Expensive & license-based $\Longrightarrow$ reduce direct reproducibility,
- ✗ Black-Boxes & custom extensions and subroutines represent a complex task.

Commercial and open-source software

Both commercial and open-source softwares have been developed for the mesh generation and the FE computation.

Commercial

- ☞ Pre-programmed common tools and laws,
- ☞ User-friendly design,
- ☞ Robust and stable versions that are often compatible across different releases,
- ✗ Expensive & license-based $\Longrightarrow$ reduce direct reproducibility,
- ✗ Black-Boxes & custom extensions and subroutines represent a complex task.

Commercial and open-source software

Both commercial and open-source softwares have been developed for the mesh generation and the FE computation.

Commercial

☞ Pre-programmed common tools and laws,

☞ User-friendly design,

☞ Robust and stable versions that are often compatible across different releases,

✗ Expensive & license-based $\Longrightarrow$ reduce direct reproducibility,

✗ Black-Boxes & custom extensions and subroutines represent a complex task.

Open-source

☞ Packages available to everyone,

☞ Promote transparency and flexibility,

☞ Relies on the community and have new routines (collaborative nature),

✗ Sometimes lack comprehensive documentation,

✗ Issues with maintaining updates and compatibility.

Commercial and open-source software

Both commercial and open-source softwares have been developed for the mesh generation and the FE computation.

Commercial

- ☞ Pre-programmed common tools and laws,
- ☞ User-friendly design,
- ☞ Robust and stable versions that are often compatible across different releases,
- ✗ Expensive & license-based $\implies$ reduce direct reproducibility,
- ✗ Black-Boxes & custom extensions and subroutines represent a complex task.

Open-source

- ☞ Packages available to everyone,
- ☞ Promote transparency and flexibility,
- ☞ Relies on the community and have new routines (collaborative nature),
- ✗ Sometimes lack comprehensive documentation,
- ✗ Issues with maintaining updates and compatibility.

Commercial and open-source software

Both commercial and open-source softwares have been developed for the mesh generation and the FE computation.

Commercial

- ☞ Pre-programmed common tools and laws,
- ☞ User-friendly design,
- ☞ Robust and stable versions that are often compatible across different releases,
- ✗ Expensive & license-based $\Longrightarrow$ reduce direct reproducibility,
- ✗ Black-Boxes & custom extensions and subroutines represent a complex task.

Open-source

- ☞ Packages available to everyone,
- ☞ Promote transparency and flexibility,
- ☞ Relies on the community and have new routines (collaborative nature),
- ✗ Sometimes lack comprehensive documentation,
- ✗ Issues with maintaining updates and compatibility.

Commercial and open-source software

Both commercial and open-source softwares have been developed for the mesh generation and the FE computation.

Commercial

- ☞ Pre-programmed common tools and laws,
- ☞ User-friendly design,
- ☞ Robust and stable versions that are often compatible across different releases,
- ✗ Expensive & license-based $\Longrightarrow$ reduce direct reproducibility,
- ✗ Black-Boxes & custom extensions and subroutines represent a complex task.

Open-source

- ☞ Packages available to everyone,
- ☞ Promote transparency and flexibility,
- ☞ Relies on the community and have new routines (collaborative nature),
- ✗ Sometimes lack comprehensive documentation,
- ✗ Issues with maintaining updates and compatibility.

Commercial and open-source software

Both commercial and open-source softwares have been developed for the mesh generation and the FE computation.

Commercial

- ☞ Pre-programmed common tools and laws,
- ☞ User-friendly design,
- ☞ Robust and stable versions that are often compatible across different releases,
- ✗ Expensive & license-based $\Longrightarrow$ reduce direct reproducibility,
- ✗ Black-Boxes & custom extensions and subroutines represent a complex task.

Open-source

- ☞ Packages available to everyone,
- ☞ Promote transparency and flexibility,
- ☞ Relies on the community and have new routines (collaborative nature),
- ✗ Sometimes lack comprehensive documentation,
- ✗ Issues with maintaining updates and compatibility.

Geometrical Modelling

In the context of finite element computations, commercial and open-source software solutions are complemented by specialised tools for Computer Aided Design (CAD) modelling, mesh generation, forming an integrated ecosystem for engineering simulations and research.

Commercial

- ➢ CATIA (Dassault Systèmes),
- ➢ Fusion 360 (Autodesk),
- ➢ Abaqus (Dassault Systèmes).

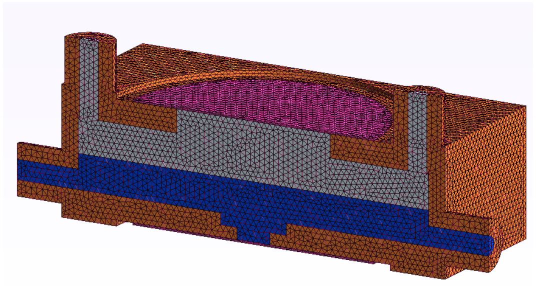
Open-source

- ➢ FreeCAD (parametric 3D modeller),
- ➢ **GMSH** (CAD + mesh generator),
- ➢ NETGEN (mesh generator),
- ➢ TETGEN (mesh generator).

GMSH

GMSH is a versatile 3D finite element **mesh generator** (structured, unstructured and hybrid) with an integrated **CAD** engine and an OpenCasCade (OCC) kernel available for **elementary entities** and **boolean operations**. It also has a python front-end.

Complex Multipart Geometries can also be created as presented by Tomo2FE article and github codes.



Finite Element computation

Common softwares found in the computation of FE solutions are the following:

Commercial

- COMSOL Multiphysics,
- ANSYS,
- ACEFEM/ACEGen,
- RADIOSS,
- ABAQUS.

Open-source

- **DEAL.ii** (C++),
- **FEniCSx** (C++ and python),
- NGSOLVE (C++ and python),
- CAST3M (FORTRAN).

Deal.ii and FEniCSx

These are open-source software libraries designed for **solving partial differential equations** (the variationnal form) using finite element methods (FEM).

DEAL.ii provides a highly flexible environment with **adaptive mesh refinement, parallel computing, and various finite element types**, making it valuable for academic research and large simulation solutions of complex PDEs.

FEniCSx, the next-generation version of the **FEniCS project**, presents **common advantages** (except dynamic adaptive mesh refinement) with a python front-end available.

Deal.ii and FEniCSx

These are open-source software libraries designed for **solving partial differential equations** (the variationnal form) using finite element methods (FEM).

DEAL.ii provides a highly flexible environment with **adaptive mesh refinement, parallel computing, and various finite element types**, making it valuable for academic research and large simulation solutions of complex PDEs.

FEniCSx, the next-generation version of the **FEniCS project**, presents **common advantages** (except dynamic adaptive mesh refinement) with a python front-end available.

Deal.ii and FEniCSx

These are open-source software libraries designed for **solving partial differential equations** (the variationnal form) using finite element methods (FEM).

DEAL.ii provides a highly flexible environment with **adaptive mesh refinement, parallel computing, and various finite element types**, making it valuable for academic research and large simulation solutions of complex PDEs.

FEniCSx, the next-generation version of the **FEniCS project**, presents **common advantages** (except dynamic adaptive mesh refinement) with a python front-end available.

Workshop Contents:

Overview of the example codes.

Github Resources

A bench of simple example is provided as part of a Github repository. It focuses on the use of FEniCSx (version 0.8.0) and GMSH (version >4.11).
The following elements are required to be able to run the examples:

- ➢ Docker or Singularity with super-user rights (or a local installation of the softwares),
- ➢ GMSH software,
- ➢ Paraview software.

All the tutorials are made available in an interactive version in the .zip arxiv file.

Github Resources

In case FEniCSx is not installed on your machine, you can run:

```
    # docker run -ti -v $(pwd):/home/fenicsx/shared -w /home/fenicsx/shared
th0maslavigne/dolfinx:v0.8.0
```

or, if Docker is not configured as a non-root user

```
    # sudo docker run -ti -v $(pwd):/home/fenicsx/shared -w /home/fenicsx/shared
th0maslavigne/dolfinx:v0.8.0
```

Each case has the python file which can be run with :

```
    # python3 filename.py
```

For parallel computation, run with <N> the number of cores:

```
    # mpirun -n <N> python3 filename.py
```

Github Resources

To work with an interactive Jupyter environment, you can once create the docker environment with:

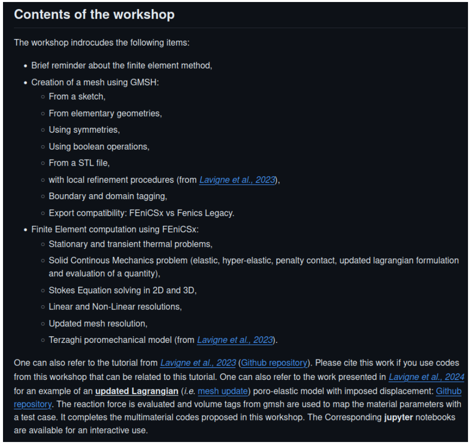
```
   # docker run -init -p 8888:8888 -v "$(pwd)":/root/shared -name=jupyter_dolfinx
dolfinx/lab:v0.8.0
```

Then to run it just consider using:

```
   # docker container start -i jupyter_dolfinx
```

All the commands and installation procedures are recalled in the Github repository.

Contents of the workshop



Contents of the workshop

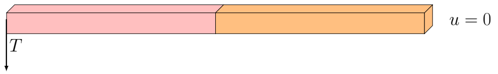
The workshop indrocudes the following items:

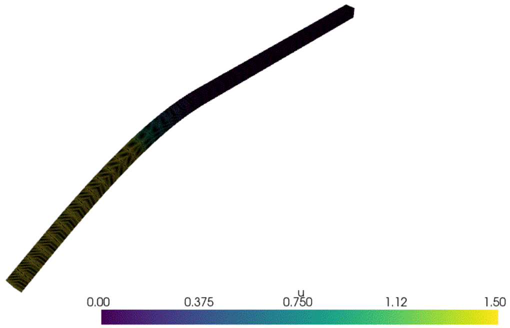
- Brief reminder about the finite element method,
- Creation of a mesh using GMSH:
  - From a sketch,
  - From elementary geometries,
  - Using symmetries,
  - Using boolean operations,
  - From a STL file,
  - with local refinement procedures (from *Lavigne et al., 2023*),
  - Boundary and domain tagging,
  - Export compatibility: FEniCSx vs Fenics Legacy.
- Finite Element computation using FEniCSx:
  - Stationary and transient thermal problems,
  - Solid Continous Mechanics problem (elastic, hyper-elastic, penalty contact, updated lagrangian formulation and evaluation of a quantity),
  - Stokes Equation solving in 2D and 3D,
  - Linear and Non-Linear resolutions,
  - Updated mesh resolution,
  - Terzaghi poromechanical model (from *Lavigne et al., 2023*).

One can also refer to the tutorial from *Lavigne et al., 2023* (Github repository). Please cite this work if you use codes from this workshop that can be related to this tutorial. One can also refer to the work presented in *Lavigne et al., 2024* for an example of an **updated Lagrangian** (*i.e.* mesh update) poro-elastic model with imposed displacement: Github repository. The reaction force is evaluated and volume tags from gmsh are used to map the material parameters with a test case. It completes the multimaterial codes proposed in this workshop. The Corresponding **jupyter** notebooks are available for an interactive use.

Elastic Multi material Beam

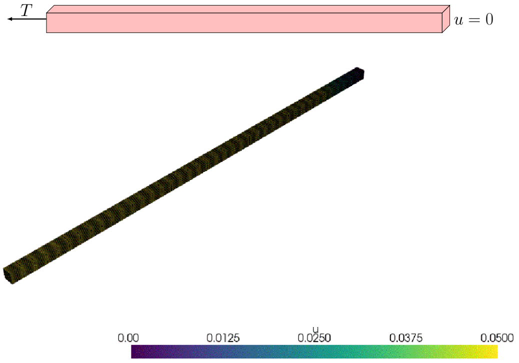
- ➢ Mesh & Tags with FEniCSx,
- ➢ The beam is subdivided into two subdomains,
- ➢ Both sides respect a same constitutive law,
- ➢ Material properties are mapped,
- ➢ Evaluation of the displacement on a surface,
- ➢ Non-linear solver and incremental load.





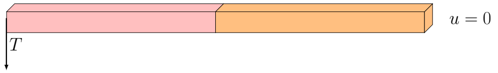
Elastic Updated Lagrange Beam

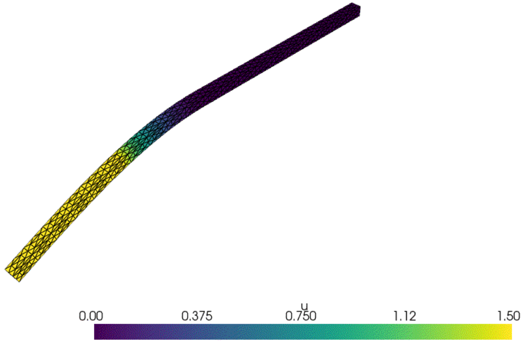
- ➢ Mesh & Tags with FEniCSx,
- ➢ Update of the mesh between two time steps,
- ➢ Initial solution setting
- ➢ Evaluation of the displacement on a surface,
- ➢ Non-linear solver and incremental load.



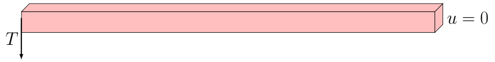
Hyper-elastic and Elastic multi-material beam

- ➢ Mesh & Tags with GMSH,
- ➢ The beam is subdivided into two subdomains,
- ➢ The two subdomains have a different constitutive law,
- ➢ Evaluation of the displacement on a surface,
- ➢ Non-linear solver and incremental load.

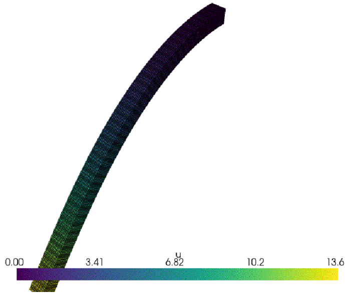




Hyper-elastic beam

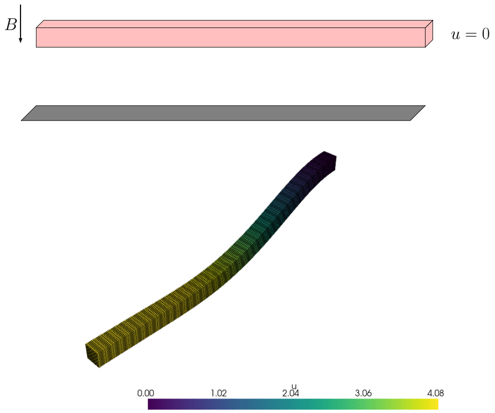


- ➢ Mesh & Tags with FEniCSx,
- ➢ Neo-Hookean constitutive law,
- ➢ Evaluation of the displacement on a surface,
- ➢ Non-linear solver and incremental load.



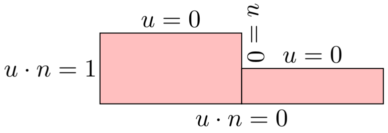
Penalty Contact beam

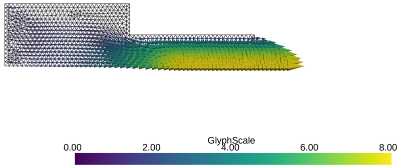
- Mesh & Tags with FEniCSx,
- Neo-Hookean constitutive law,
- introducing penalty contact with a plane of equation $z = -4$,
- Evaluation of the displacement on a surface,
- Non-linear solver and incremental load.



2D Stokes problem

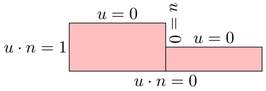
- Mesh & Tags with GMSH (from a sketch and from elementary entities),
- Stokes variational form,
- Post-processing stress/strain-rate,
- Linear solver.

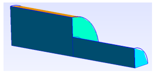


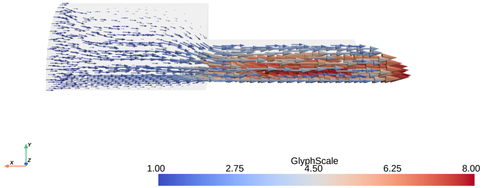


3D Stokes problem

- ➢ Mesh & Tags with GMSH (from a 2D geometry & from elementary entities),
- ➢ Stokes variational form,
- ➢ Post-processing stress/strain-rate,
- ➢ Non-linear solver.



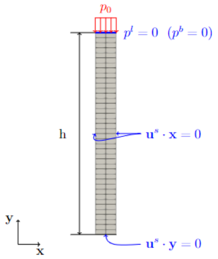


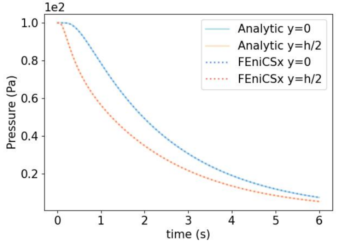


Terzaghi consolidation problem

From Lavigne et al. 2023:

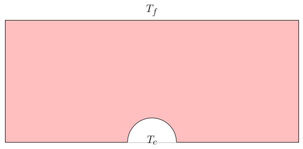
- ➢ Mesh & Tags with FEniCSx,
- ➢ Porous model and mixed space,
- ➢ L2 error function in pressure,
- ➢ Non-linear solver,
- ➢ csv export.

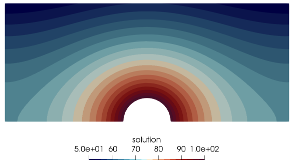

p_0
$p^l = 0 \quad (p^b = 0)$
h
$\mathbf{u}^s \cdot \mathbf{x} = 0$
$\mathbf{u}^s \cdot \mathbf{y} = 0$
y
x



Stationary thermal problem

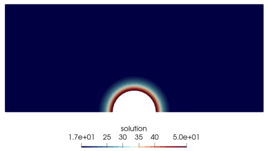
- Mesh & Tags with GMSH,
- Plate with a thermal source at its bottom center,
- Dirichlet and Robin boundary conditions,
- Linear solver.

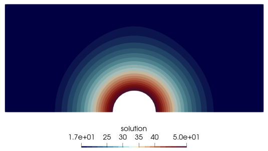


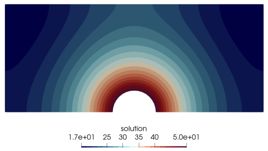


Transient thermal problem

- ➢ Mesh & Tags with GMSH,
- ➢ Plate with a thermal source at its bottom center,
- ➢ Dirichlet and Robin boundary conditions,
- ➢ Linear solver.





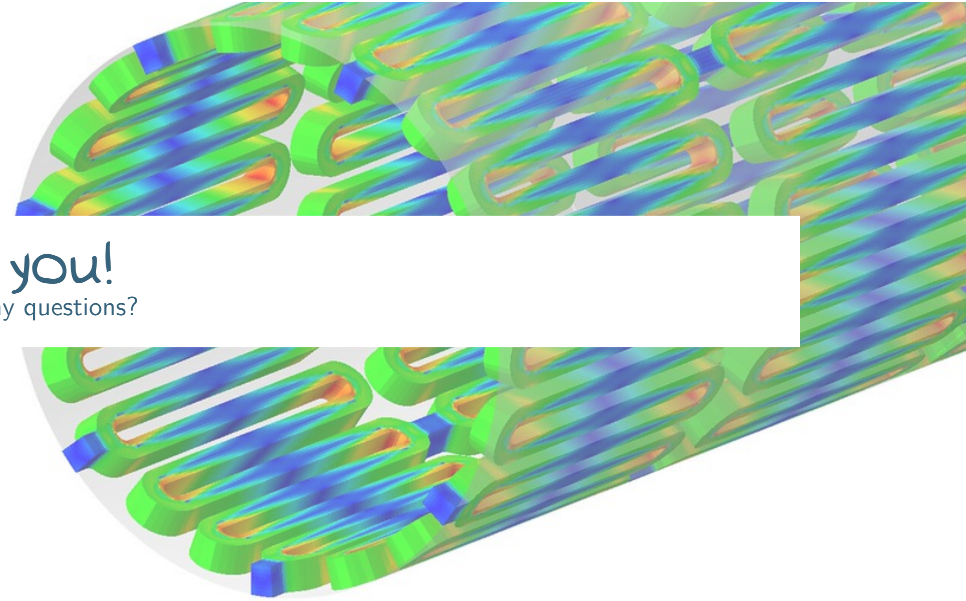


UNIVERSITY OF LUXEMBOURG

INSTITUT de BIOMÉCANIQUE HUMAINE GEORGES CHARPAK

I2M BORDEAUX INSTITUT DE MÉCANIQUE ET D'INGÉNIERIE

Luxembourg National Research Fund



Thank you!

Do you have any questions?

Acknowledgments

This repository is inspired from the work of Jørgen S. Dokken and Christophe Geuzaine. The author also wants to thank F. Daghia and Ludovic Chamoin for the quality of the finite element courses at the École Normale Supérieure Paris-Saclay, Jack Hale and Stéphane Urcun for their help in debugging the codes throughout the PhD work and Giuseppe Sciumè for the invite.
This activity is part of Thomas Lavigne PhD work. This research was funded in whole, or in part, by the Luxembourg National Research Fund (FNR), grant reference No. 17013182. For the purpose of open access, the author has applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission.

References

aliasLinux (20xx).
Create an alias for linux.
https://www.malekal.com/comment-creer-un-alias-linux/.

BasixDocs (20xx).
Basix documentation.
https://docs.fenicsproject.org/basix/v0.8.0/python/.

Chambert, J., Lihoreau, T., Joly, S., Chatelain, B., Sandoz, P., Humbert, P., Jacquet, E., and Rolin, G. (2019).
Multimodal investigation of a keloid scar by combining mechanical tests in vivo with diverse imaging techniques.
Journal of the Mechanical Behavior of Biomedical Materials, 99:206–215.

DealIIDocs (20xx).
Deal.ii documentation.
https://www.dealii.org/current/index.html.

DealIILibrary (20xx).
Deal.ii library.
https://www.dealii.org/current/doxygen/deal.II/index.html.

DealIITutorials (20xx).
Deal.ii tutorials.
https://www.dealii.org/current/doxygen/deal.II/Tutorial.html.

References

DealIIWebsite (20xx).
Deal.ii website.
https://www.dealii.org/.

DockerCheatSheet (20xx).
Docker cheat sheet.
https://docs.docker.com/get-started/docker_cheatsheet.pdf.

DockerCheatSheet2 (20xx).
Docker cheat sheet 2.
https://dockerlabs.collabnix.com/docker/cheatsheet/.

DockerHub (20xx).
Docker hub.
https://hub.docker.com/.

DockerWebsite (20xx).
Docker website.
https://www.docker.com/products/docker-desktop/.

DolfinxDocs (20xx).
Dolfinx documentation.
https://docs.fenicsproject.org/dolfinx/v0.8.0/python/.

References

Doxygen (20xx).
Doxygen documentation.
https://www.doxygen.nl/index.html.

FEniCSChangelog (20xx).
Fenicsx changelog.
https://github.com/FEniCS/dolfinx/releases.

FEniCSForum (20xx).
Fenicsx discourse forum.
https://fenicsproject.discourse.group/.

FEniCSGithub (20xx).
Fenicsx github repositories.
https://github.com/orgs/FEniCS/repositories.

FEniCSLegacyDocs (20xx).
Fenics legacy documentation.
https://fenicsproject.org/olddocs/.

FEniCSTutorial (20xx).
Fenicsx tutorial.
https://jsdokken.com/dolfinx-tutorial/.

References

FEniCSWebsite (20xx).
Fenics project website.
https://fenicsproject.org/.

FFCxDocs (20xx).
Ffcx documentation.
https://docs.fenicsproject.org/ffcx/main/.

GITDocs (20xx).
Git documentation.
https://docs.github.com/en/get-started.

GITReference (20xx).
Git reference.
https://git-scm.com/docs.

GITTutorial (20xx).
Git interactive tutorial.
https://learngitbranching.js.org/?locale=fr_FR.

GMSHAPITutorials (20xx).
Gmsh api tutorials.
https://bthierry.pages.math.cnrs.fr/tutorial/gmsh/api/.

References

GMSHDownload (20xx).
Gmsh download.
https://gmsh.info/.

GMSHGitLab (20xx).
Gmsh gitlab.
https://gitlab.onelab.info/gmsh/gmsh.

GMSHManual (20xx).
Gmsh manual.
https://gmsh.info/doc/texinfo/gmsh.html.

GMSHOverview (20xx).
Gmsh introductive presentation.
https://gmsh.info/doc/course/general_overview.pdf.

Holzapfel, G. A. (2002).
Nonlinear solid mechanics: a continuum approach for engineering science.

Lavigne, T. (2024).
Github: Mesh generation and finite element analysis with gmsh and fenicsx.
https://github.com/Th0masLavigne/FEniCSx_GMSH_tutorials.git.

References

Logg, A., Mardal, K.-A., and Wells, G. (2012).
Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book.
Springer.

MarkdownDocs (20xx).
Markdown documentation.
`https://markdown.frama.io/`.

Martin, R. (20xxa).
Clean code - conference.
`https://www.youtube.com/watch?v=7EmboKQH8lM&list=PLmmYSbUCWJ4x1GO839azG_BBw8rkh-zOj&index=1`.

Martin, R. (20xxb).
Clean code - uncle bob / lesson 1.
`https://www.youtube.com/watch?v=7EmboKQH8lM`.

MatplotlibDocs (20xx).
Matplotlib documentation.
`https://matplotlib.org/stable/users/index.html`.

Meshio (20xx).
Meshio documentation.
`https://pypi.org/project/meshio/`.

References

MeshioGithub (20xx).
Meshio github.
https://github.com/nschloe/meshio.

MeshUpdate (20xx).
Mesh update: Updated lagrangian.
https://fenicsproject.discourse.group/t/how-to-do-updated-lagrangian-when-the-displacement-lives-in-a-different-space-to-the-mesh-geometry/10760/2.

MKDocs (20xx).
Mkdocs documentation.
https://squidfunk.github.io/mkdocs-material/.

MKDocsGithub (20xx).
Mkdocs github.
https://github.com/squidfunk/mkdocs-material.

Neper (20xx).
Neper website.
https://neper.info/.

NumpyDocs (20xx).
Numpy documentation.
https://numpy.org/doc/stable/index.html.

References

OpenCascade (20xx).
Opencascade commands.
https://koehlerson.github.io/gmsh.jl/dev/occ/occ/.

ParaviewDownload (20xx).
Paraview download.
https://www.paraview.org/download/.

Pygmsh (20xx).
Pygmsh documentation.
https://pypi.org/project/pygmsh/.

PygmshGithub (20xx).
Pygmsh github.
https://github.com/nschloe/pygmsh.

PythonClick (20xx).
Click documentation.
https://click.palletsprojects.com/.

PythonDocs (20xx).
Python official documentation.
https://docs.python.org/3/.

References

PythonMultiprocessing (20xx).
Python multiprocessing documentation.
https://docs.python.org/fr/3/library/multiprocessing.html.

ScipyDocs (20xx).
Scipy documentation.
https://docs.scipy.org/doc/scipy/.

Singularity (20xx).
Singularity user guide.
https://docs.sylabs.io/guides/3.5/user-guide/introduction.html.

UFLDocs (20xx).
Ufl documentation.
https://fenics.readthedocs.io/projects/ufl/en/latest/.

VTKDocs (20xx).
Vtk documentation.
https://vtk.org/documentation/.

WindowsDockerCheatSheet (20xx).
Windows docker cheat sheet.
https://gist.github.com/danijeljw/a7a2553bd06742648172363ce3983a9a.