ftrillaudp@pumas.ii.unam.mx

INSTITUTO DE INGENIERÍA UNAM

Open-source FEA software in applied superconductivity

- "Free software" means software that respects users' freedom and community... the users have the freedom to run, copy, distribute, study, change and improve the software (source: GNU project)

- Open-source software (OSS) is computer software with its source code made available with a license in which the copyright holder provides the rights to study, change, and distribute the software to anyone and for any purpose. (source: Wikipedia)

Acknowlegments

- Some of the examples were provided by:
  - Ana Neri from MISTI program (Massachusetts Institute of Technology) - GetFem++
  - Edgar Berrospe Juaréz from the Posgrado en Ingeniería (National Autonomous University of Mexico) and Dr. Víctor Manuel Rodriguez Zermeño from Karlsruhe Institute of Technology – COMSOL Multiphysics®

INSTITUTO DE INGENIERÍA UNAM

Contents

Introduction

- Within the past 10 to 15 years, the number of free codes and software have largely increased following the Internet boom.
- The tendency is to facilitate the user experience. Nevertheless, a certain level of coding skills is still required.
- Python is vastly used to interface underlying libraries often written in C++.
- **The following slides will present a few free FEA software for which the speaker has some level of experience from the perspective of the end user, not the developer. Beware! it is not a thorough list of all available software.**

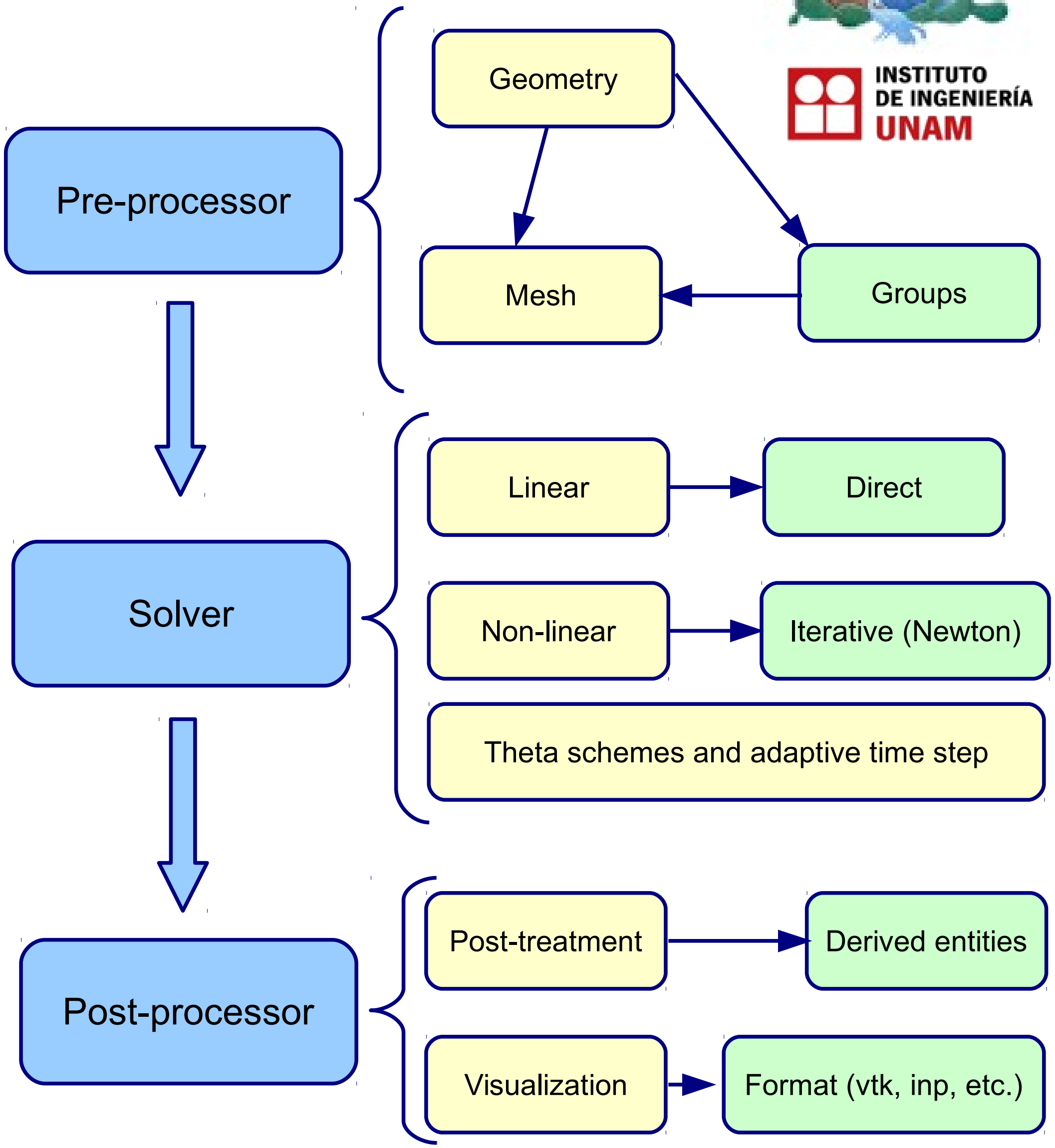
Quick overview

- Geometry and mesh generators: **Salome platform**, **FreeCAD**, **Gmsh**
- FEA Software:
  - Mechanics: **Code_ASTER**, **Cast3M**
  - Fluid mechanics: Code_SATURNE, openFOAM
  - Heat transfer (conduction, radiation): **SYRTHES**
  - Electromagnetism: FEMM, **GetDP**
  - Multi-physics: Cast3M, GetDP, FreeFem++, **GetFem++**, **CSC-Elmer**
- Visualization: **Paraview**, **opendx**, Gmsh, Salome
- Plotting tools: **gnuplot**, Grace, **matplotlib/python**
- Some libraries:
  - Numerical: MUMPS, PETSC, blas, lapack, gsl
  - Geometry and CAD: opencascade
  - Meshing: netgen, tetgen
  - General: **scipy**, **numpy**, open MPI, **GNU Octave**

General structure of FEA software

- Pre-processors:
  - CAD modelers
  - Mesh generators
- Solvers:
  - Linear / Non-linear or iterative
  - Parallelized / sequential
- Post-processors:
  - Data visualization (2D and 3D)
  - Plotting
- Typical operating system: Linux
- Main languages: Python and C++

Fig. 1: Overview of the general structure of FEA software.



FEA software

Reference: COMSOL (ease = 10) and C++ (ease = 0)

Name	**Type**	**Usage**	**OS**	**GUI**	**TUI**	**Pre-compiled**	**Coupling**	**License**	**Ease (10 = easy, 0 = hard)**
Code_ASTER	Solver	Mechanics (2D, 3D)	Linux, Windows	partial	Python	yes	Saturne, syrthes, Salome, Gmsh	GPL	5
Code_Saturne	Solver	Fluid mechanics (2D, 3D)	Linux	partial	Python	yes	SYRTHES, ASTER, Salome	LGPL	5
SYRTHES	Solver	Conduction, radiation heat transfer (2D, 3D)	Linux, Windows	Full	FORTRAN	yes	Saturne, ASTER	GPL	8
Cast3M	*Solver, Mesher*	*Multi-physics (1D, 2D, 3D)*	*Linux, windows, Mac*	*No*	*Gibiane*	*yes*	*Salome*	*Free for research and teaching*	*6*
GetDP	*Solver*	*Multi-physics (2D, 3D)*	*Linux, Windows, Mac*	*No*	*C and own language*	*yes*	*Gmsh*	*GPL*	*4*
GetFem++	Solver, mesher	Multi-physics (1D, 2D, 3D)	Linux	No	C++, python, scilab	No	Gmsh	LGPL	3
FreeFem++	*Solver, mesher, viewer*	*Multi-physics (2D, 3D)*	*Linux, Windows, Mac*	*No*	*C++ idiom*	*Yes*	*Gmsh*	*LGPL*	*5*
FEMM	*Geometry, mesher, Solver, viewer*	*Electromagnetism (2D)*	*Windows, Linux*	*Yes*	*Lua*	*Yes*	*-*	*AFPL*	*8*
Elmer	Solver, viewer	Multi-physics (2D, 3D)	Windows, linux, Mac	Yes	Own language	Yes	Gmsh, salome	GPL	6

CAD Model, mesher, and viewer

Name	Type	OS	GUI	TUI	Pre-compiled	Coupling	License	Ease (10 = easy, 0 = hard)
Salome platform	CAD, mesher, viewer	Linux, windows	Yes	Python	Yes	ASTER, Saturne, SYRTHES, Cast3M	LGPL	8
Gmsh	*CAD, mesher, viewer*	*Linux, Windows, Mac*	*Yes*	*Own language*	*Yes*	*GetDP*	*GPL*	7
FreeCAD	CAD, mesher	Linux, Windows, Mac	Yes	Python	Yes	-	LGPL	7
Paraview	Viewer	Linux, Windows, Mac	Yes	Python	Yes	Through compatible format	BSD	8
Opendx	Viewer	Linux, Windows	Yes	-	Yes	Through compatible format	IBM public license	7

Other possible free 3D visualization software: Visit, MayaVi

Some examples

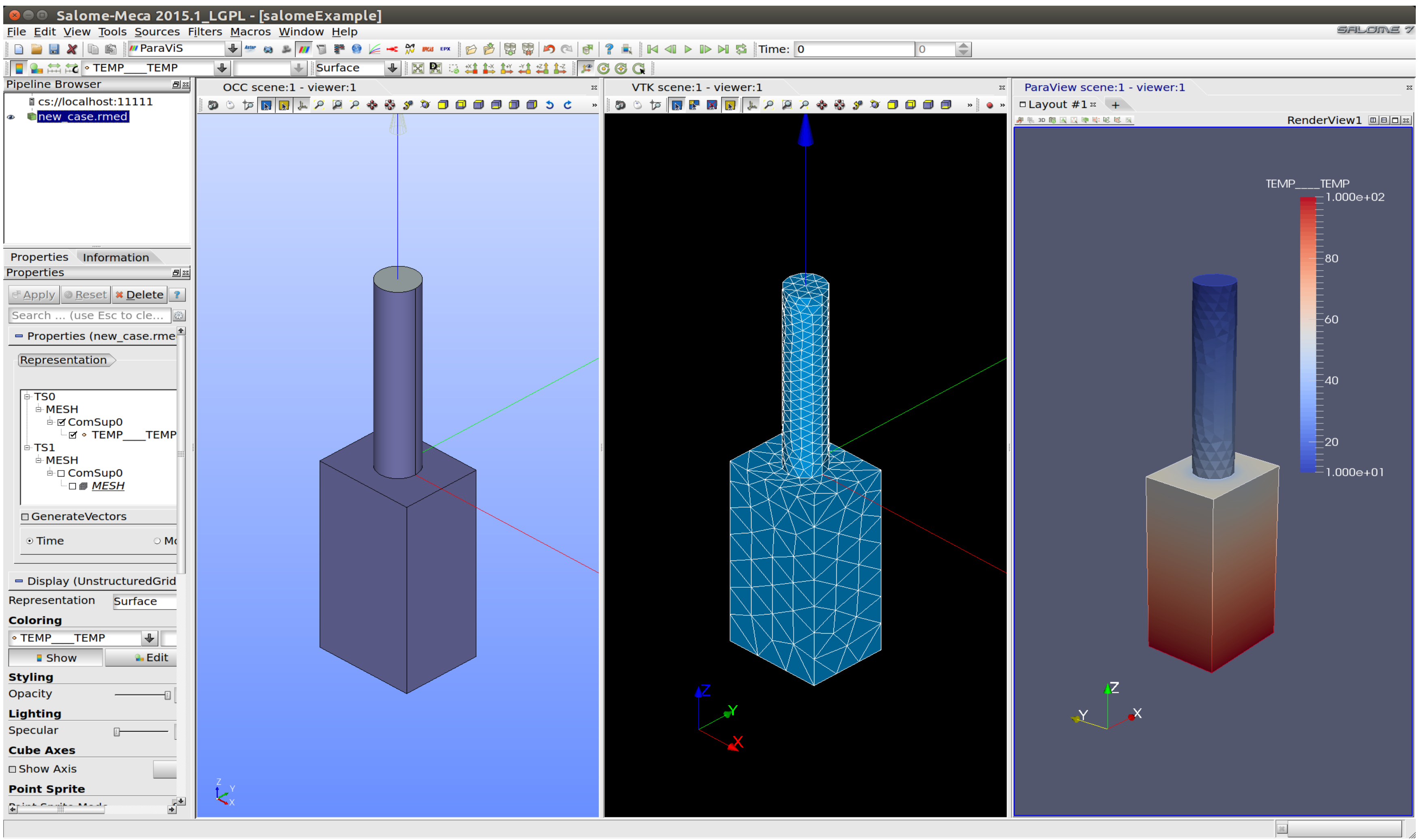
- Salome-Meca (Salome platform with Code_ASTER): creation of geometry and meshing, Linear thermal analysis.

- Gmsh and GetDP (Onelab): Creation of geometry and meshing. HTS bulk and flux trapping. Visualization with Gmsh and Gnuplot.

- GetFem++: AC losses in HTS thin layer comparison with analytical formula and COMSOL Multiphysics®. Post-processing and visualization with ParaView.

Salome-Meca

- Creation of geometries and meshes through TUI or GUI
- Crtl+n to open a new study, ctrl+t to launch a script
- Parametric study with the creation of a notebook
- Main modules: GEOM (modeler), SMESH (mesher), HOMARD (adaptive mesh), ASTER (solver) ...

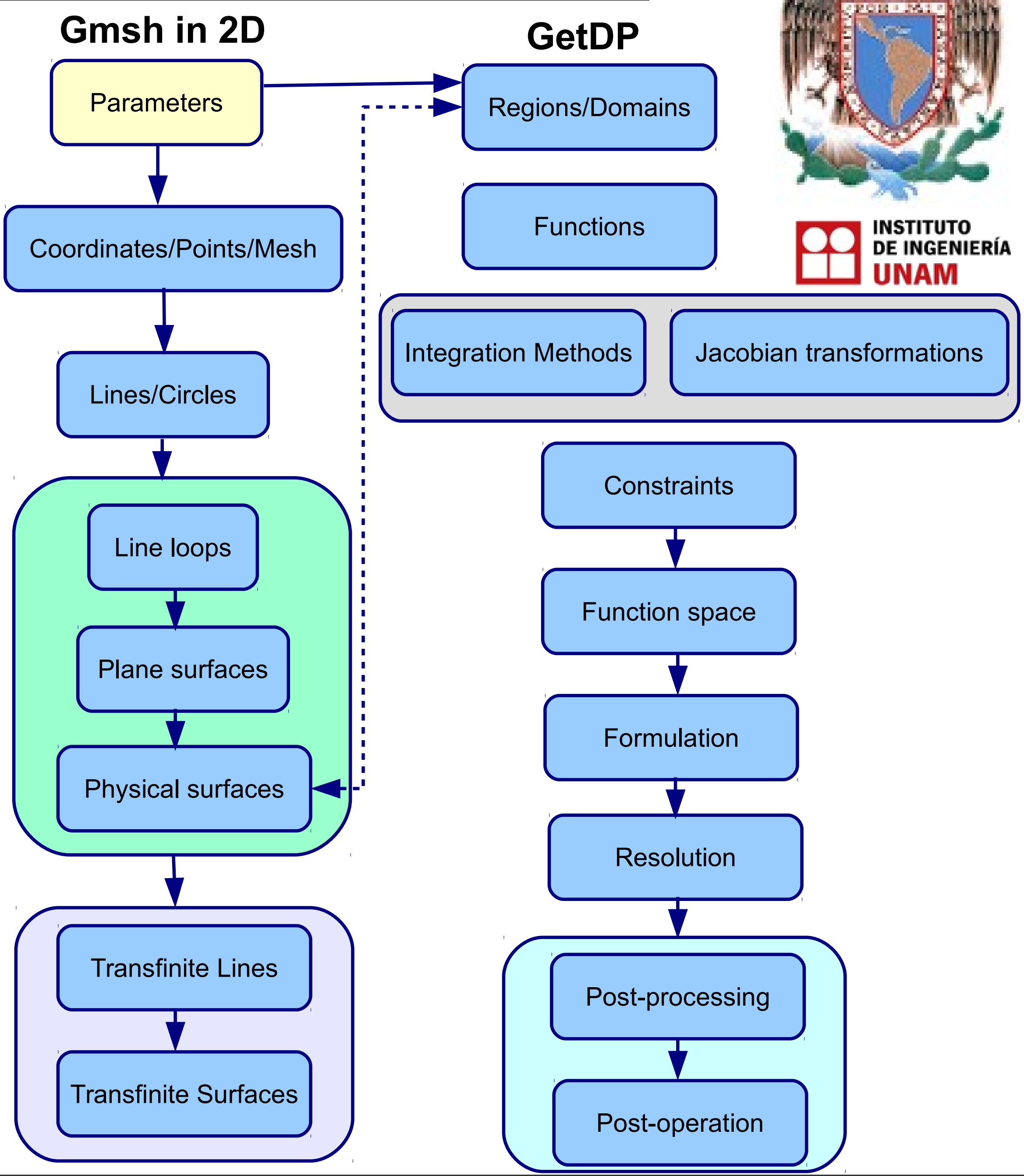


Onelab (Gmsh+GetDP) – part 1/4

- Gmsh can be used in TUI or GUI mode.
- Creation of a basic geometry and meshing (*.geo):

  >>gmsh -2 example.geo -o example.msh

- Output mesh in *example.msh*. The groups on meshes are identified in *.geo as Physical Line/Surface/Volume.
- Generation of quadrangle mesh through Transfinite Line and Transfinite Surface in 2D.
- The name of the solver file *.pro should be the same as the mesh: *example.pro*. *Then,* this file can be easily run in Gmsh.
- Post-processing as *.pos and table format files (gnuplot).

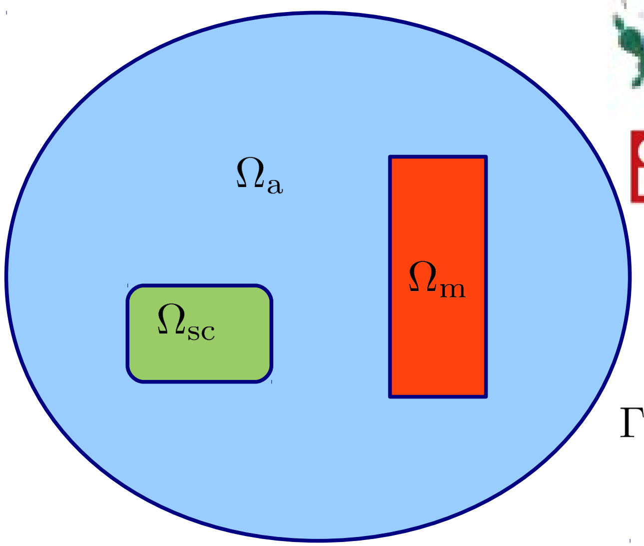


INSTITUTO DE INGENIERÍA UNAM

Onelab (Gmsh+GetDP) – part 2/4

- YBCO bulk magnetization under field cooling conditions: comparison with experimental data
- Axisymmetry, homogeneous and isotropic material properties
- *A*-formulation and power law:

$$\begin{cases} \nabla \times [\nu_0 \nabla \times \mathbf{A}] + \sigma \dfrac{\partial \mathbf{A}}{\partial t} - \mathbf{J}_\mathrm{a} = 0 \\ \sigma = J_\mathrm{c} E_\mathrm{c}^{-\frac{1}{n}} \left(\left\| \dfrac{\partial \mathbf{A}}{\partial t} \right\| + \epsilon \right)^{\frac{1-n}{n}} \end{cases}$$



- Green's formula to extremize into the weak form:

$$\int_\Omega \nu_0 (\nabla \times \mathbf{A}) . (\nabla \times \mathbf{A}^\star)\, d\Omega + \int_\Gamma (\mathbf{A}^\star \times \mathbf{n}) . (\nabla \times \mathbf{A})\, d\Gamma + \int_\Omega \frac{\partial \mathbf{A}}{\partial t} . \mathbf{A}^\star d\Omega - \int_\Omega \mathbf{J}_\mathrm{a} . \mathbf{A}^\star d\Omega = 0$$

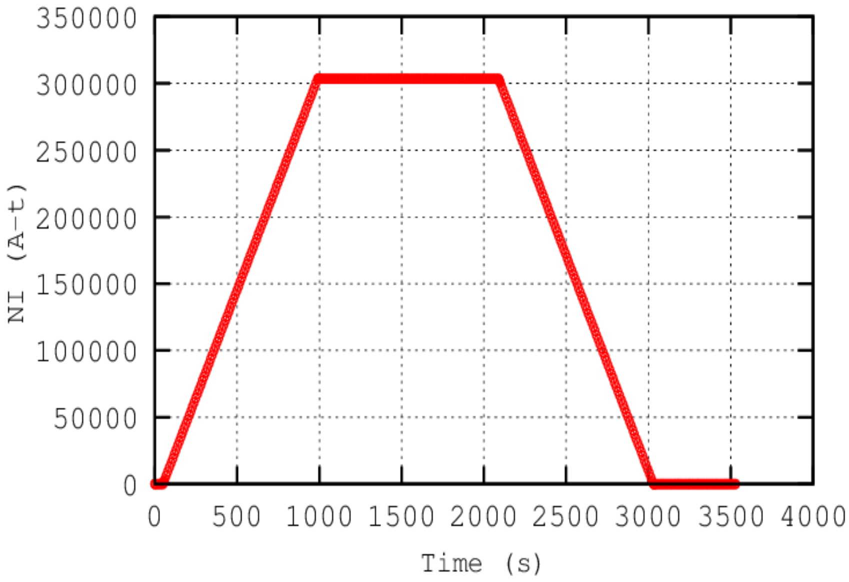
Integration by parts

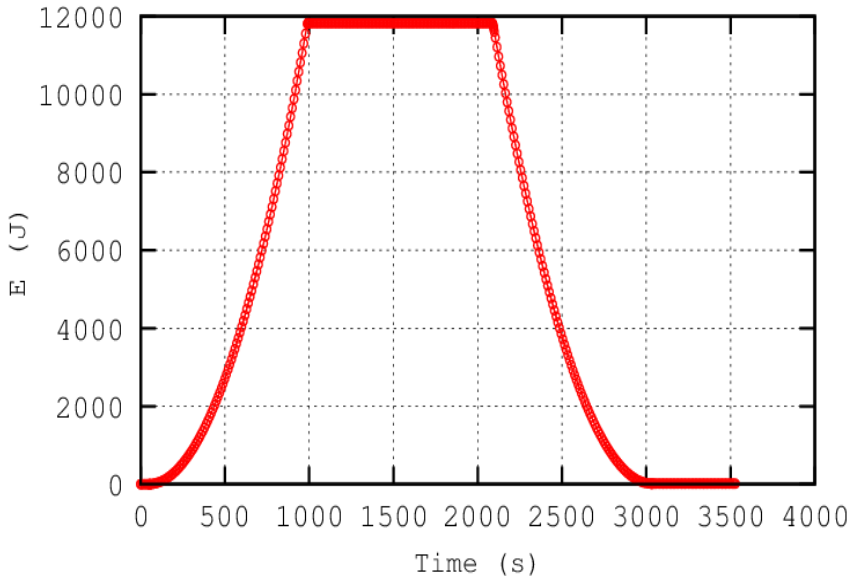
Boundary Γ

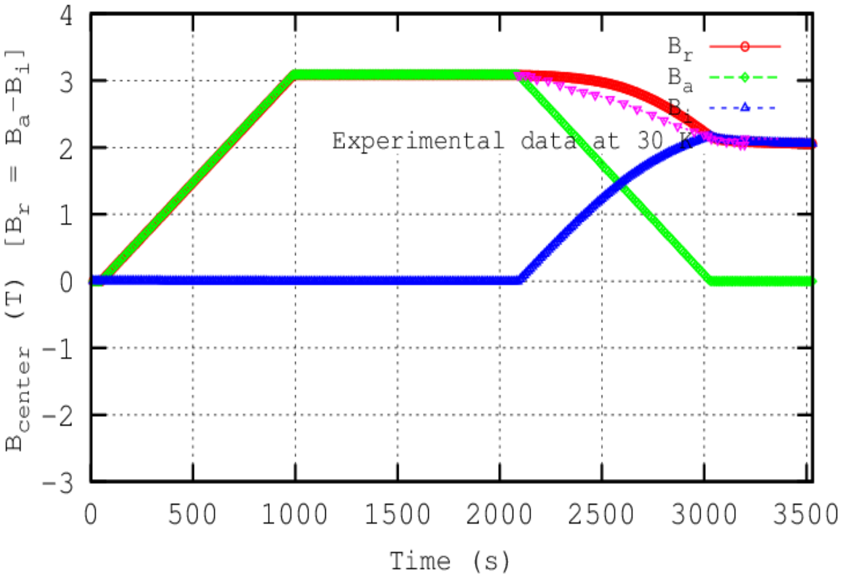
$$\Omega = \Omega_\mathrm{sc} \cup \Omega_\mathrm{m} \cup \Omega_\mathrm{a}$$
$$\Omega_\mathrm{sc} \cap \Omega_\mathrm{m} = 0$$
$$\Omega_\mathrm{sc} \cap \Omega_\mathrm{a} = 0$$
$$\Omega_\mathrm{m} \cap \Omega_\mathrm{a} = 0$$

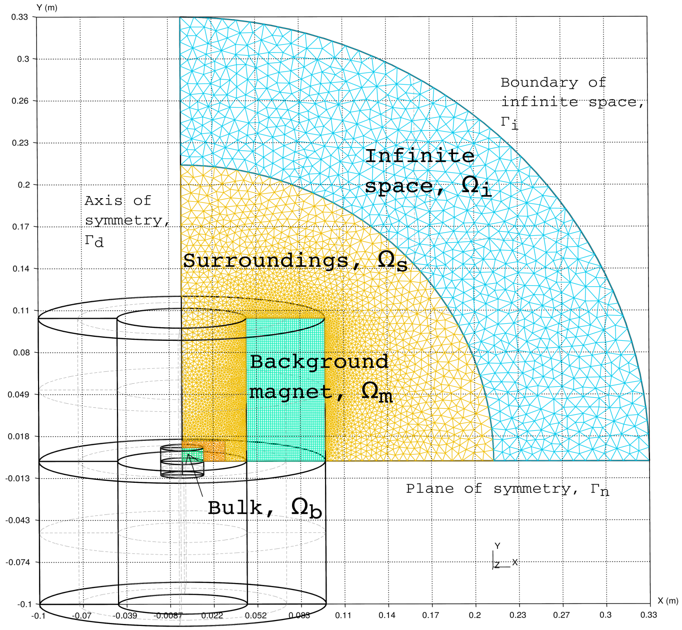
```
Equation{
Galerkin{[nu0*Dof{Curl  A},  {Curl  A}];  In  Omega;  Jacobian  axiJacobian;  Integration GaussPoint;}
Galerkin{DtDof[sigma[{Curl   A},   -Dt[{A}]]*Dof{A},   {A}];   In   Omega_sc;   Jacobian axiJacobian; Integration gaussPoint;}
Galerkin{-Dof{Ja}, {A}]; In Omega_m; Jacobian axiJacobian; Integration GaussPoint;}
}
```

Onelab (Gmsh+GetDP) – part 3/4



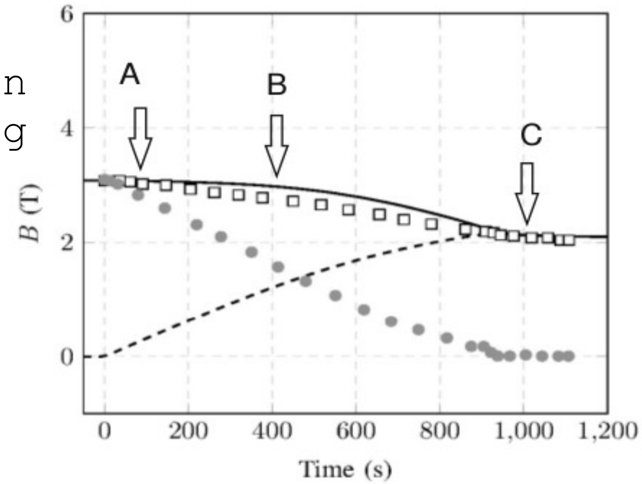




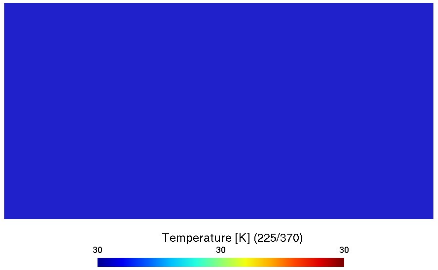


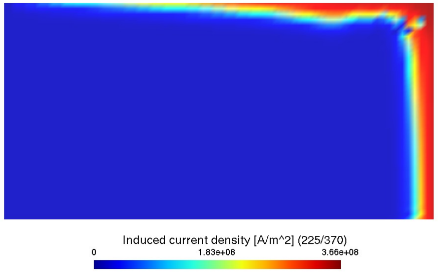
Onelab (Gmsh+GetDP) – part 4/4

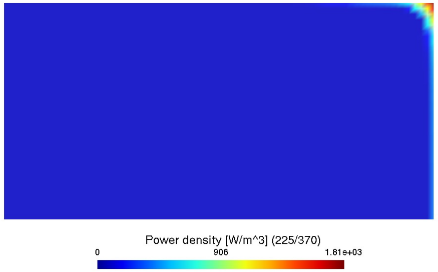
Sample is magnetized in field cooling conditions. Three time steps are marked A, B and C.



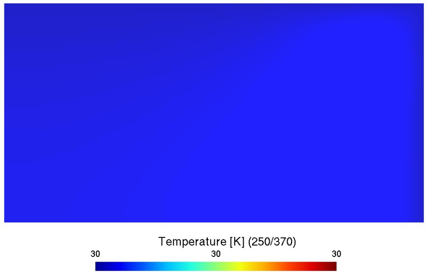
A: sample at operating temperature, the background magnet is ramped down

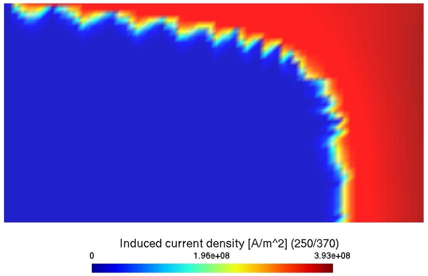


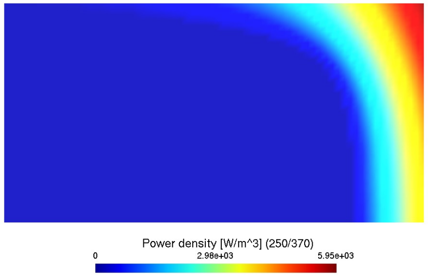




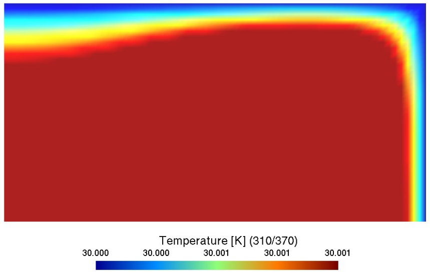
B: ramping down at a rate of 0.003 T/s

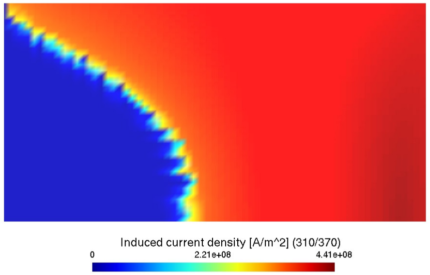


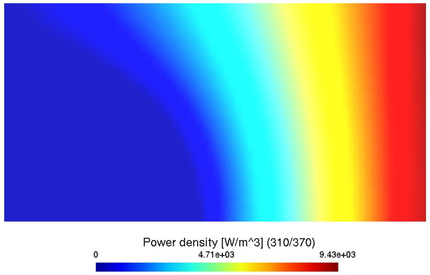




C: trapping flux, barely any temperature change over the magnetization







GetFem++ – part 1/3

- GetFem++ is a C++ library to be compiled and linked.
- On ubuntu 14.04:

```
>>sudo mkdir /opt/GetFempp
>>cd ~ /Downloads
>>wget http://download.gna.org/getfem/stable/getfem-5.0.tar.gz
>>tar -xvzf getfem-5.0.tar.gz; cd getfem-5.0
>>cd getfem-5.0; ./configure --prefix=/opt/GetFempp --with-pic -enable-
python --enable-qhull --enable-experimental
>>make; make check; sudo make install; make clean
```

- *.cpp text file to be compiled:

```
>>g++ example.cpp -o example $(getfem-config --libs) -lboost_system;
```

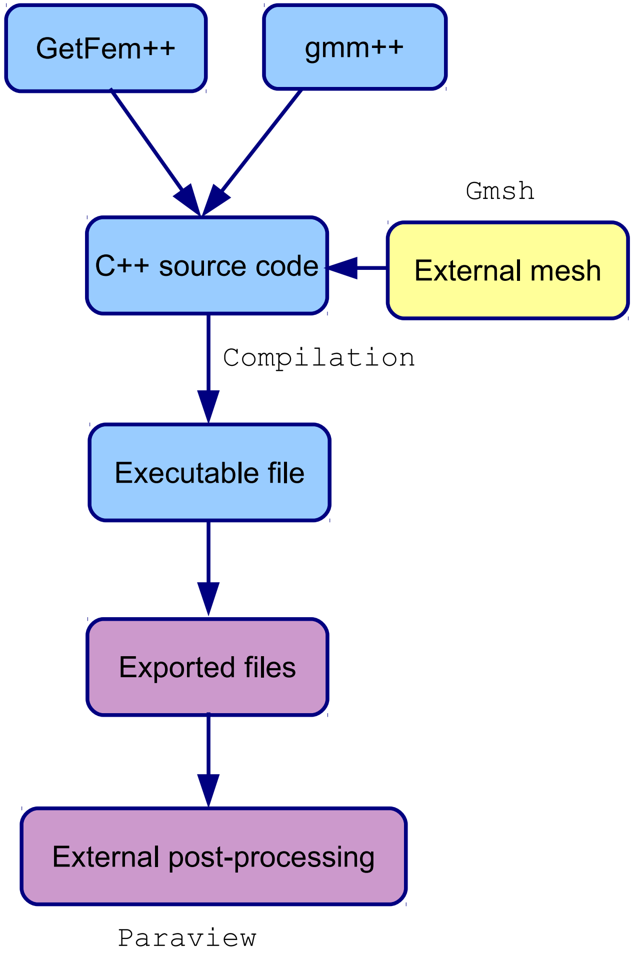
- Read in Gmsh mesh:

```
getfem::mesh mesh;
string meshname("gmsh:./" + filename + ".msh");
getfem::import_mesh(meshname, mesh);
mesh.optimize_structure();
```

- Brick architecture (basic blocks):

```
std::string curl_H("(Grad_H(2,1)-Grad_H(1,2))");
std::string curl_Test_H("(Grad_Test_H(2,1)-Grad_Test_H(1,2))");
transient_bricks.add(
getfem::add_nonlinear_generic_assembly_brick(
model, mim, "rho_Cu."+curl_H+"."+curl_Test_H, wire_ID, true));
transient_bricks.add(
getfem::add_nonlinear_generic_assembly_brick(model, mim,
"rho_air."+curl_H+"."+curl_Test_H, surrounding_ID,true));
```

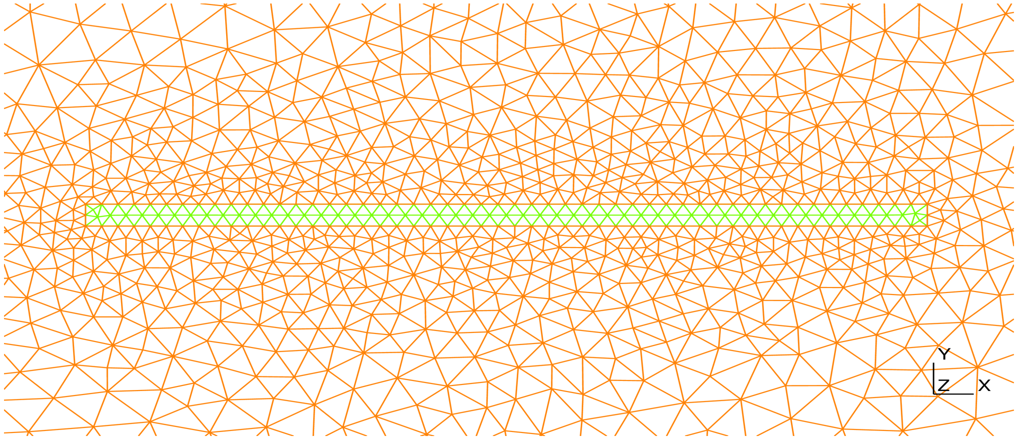
- The library gmm++ provides: basic types of sparse, dense matrices and vectors, and their operations.



GetFem++ – part 2/3

- AC losses in thin HTS layer.
- Plane problem with homogeneous and isotropic material properties
- *H*-formulation:

$$\begin{cases} \nabla \times [\rho \nabla \times \mathbf{H}] + \mu_0 \dfrac{\partial \mathbf{H}}{\partial t} = 0 \\ \rho = \dfrac{E_c}{J_c} \left\| \dfrac{\nabla \times \mathbf{H}}{J_c} \right\|^{n-1} + \epsilon \end{cases}$$

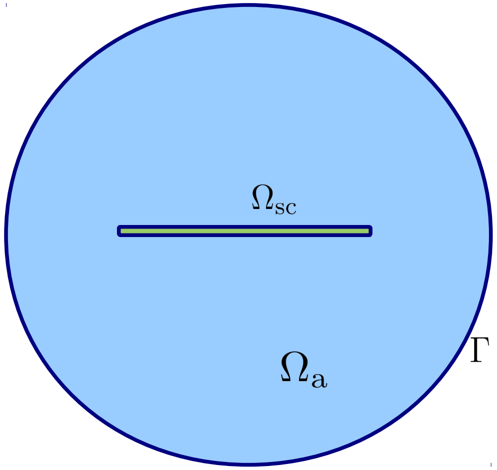


Zoom on the thin HTS layer .

- Boundary value:

$$\mathbf{H}|_{t=0} = 0$$

$$\mathbf{H}|_{\Gamma} = \frac{I_0 \sin(\omega t)}{2\pi (x^2 + y^2)^{\frac{3}{2}}} (y\mathbf{e}_x + x\mathbf{e}_y)$$



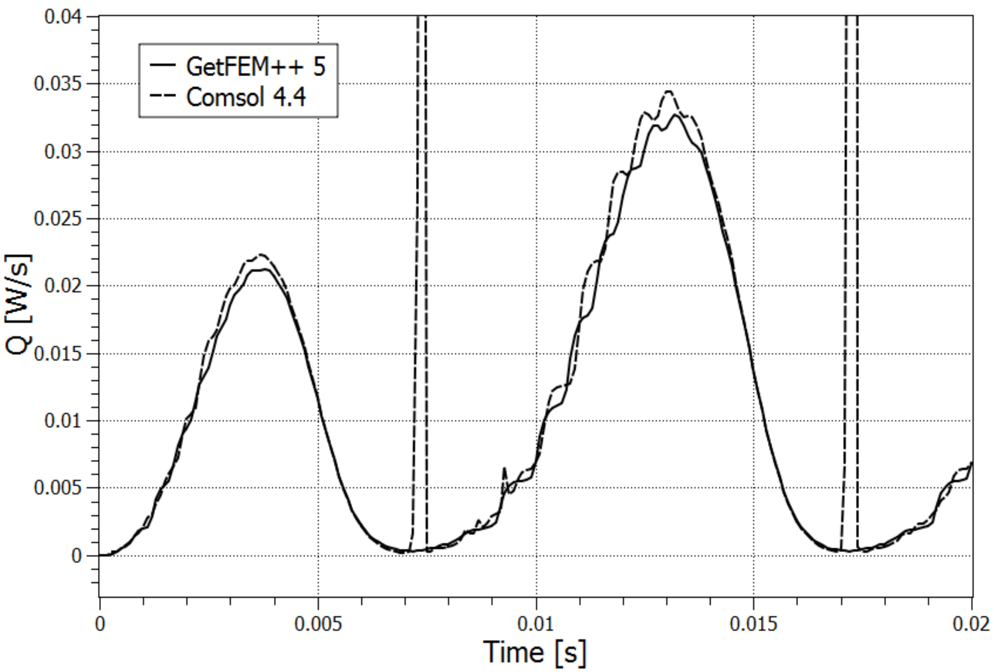
GetFem++ – Comparison - part 3/3

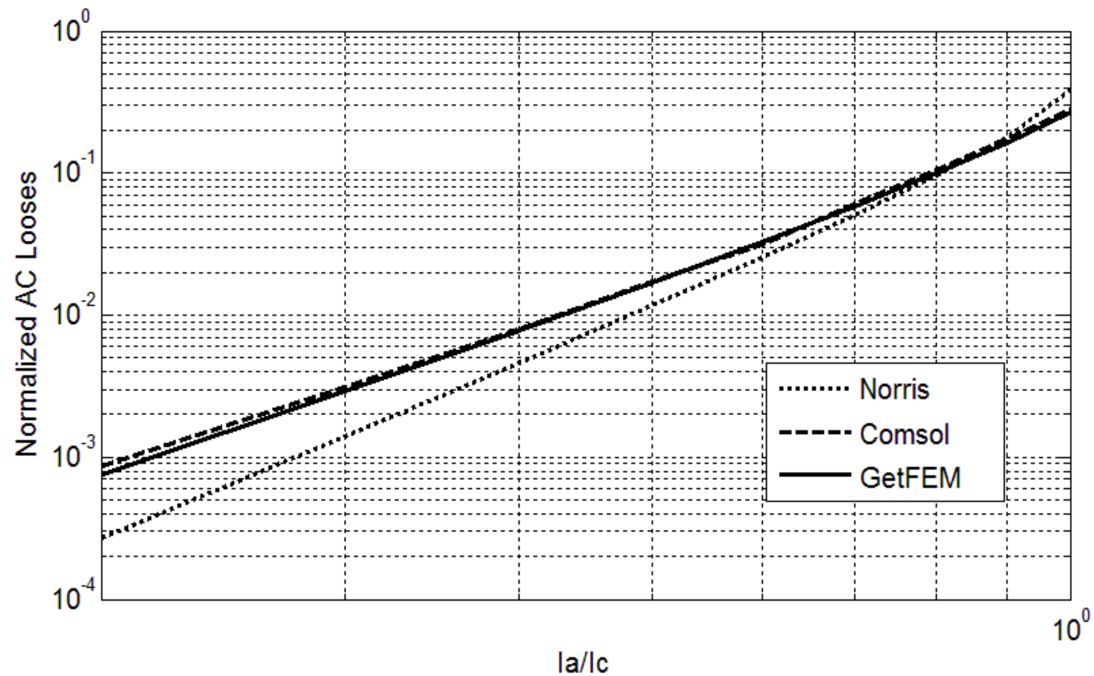
Frequency : 50 Hz
Tape thickness: 0.1 mm
Tape width: 1.5 mm

Computation not optimized!!

INSTITUTO DE INGENIERÍA UNAM

J_c (A/mm^2)	E_c $(\mu V/cm)$	n	ε	ρ air (Ω-m)
270	0.1	19	1e-14	1





Result @ $0.8I_c$

Values of parameters from: R. Brambilla *et al*, Supercond. Sci. Technol. 20 (2007) 16-24
Normalization factor: $f*I_c^2/(\mu_0*\pi)$

A few additional examples

- Modeling with GNU Octave and FreeFem++ was presented by Dr. Víctor Manuel Rodriguez Zermeño at EUCAS 2015, Lyon France and are available on arXiv:

  Víctor M. R. Zermeño, Salman Quaiyum, Francesco Grilli, "Open Source Codes for Computing the Critical Current of Superconducting Devices", arXiv:1509.01856

- Gmsh/GetDP (onelab), example on the website:

  http://onelab.info/wiki/Superconducting_wire

- *M. Krasl, R. Vlk, J. Rybar, "Losses in Windings of Superconducting Traction Transformer, 2D and 3D Model", Proceedings of the 6th WSEAS/IASME, 2006.*

- Gmsh and Code_aster for fusion:

  http://www.fusionvic.org/LastResults/LastResults_FE-stress-analysis-of-the-case/FE-stress-analysis-of-the-case.htm

Recommendations

- Some level of programming knowledge is required. Python is widely used as binding or interfacing language. Most of the free software are developed for Linux.
- For productivity, it may be recommended to use commercial software or you have to be fairly proficient if time is crucial.
- Cross-check results as benchmarking: experimental data, analytical formulae, comparison between different software/codes, etc..
- **Register to the forums and mailing lists.** Answers are not always provided but you have it for free.
- Try out different software. Some may fit better your need than others.

References

- Code_ASTER: http://www.code-aster.org/V2/spip.php?rubrique2
- Code_Saturne: http://code-saturne.org/cms/
- FEMM: http://www.femm.info/wiki/HomePage
- FreeCAD: http://www.freecadweb.org/
- FreeFem++: http://www.freefem.org/ff++/
- GetFem++: http://download.gna.org/getfem/html/homepage/
- GetDP (stand alone): http://www.geuz.org/getdp/
- Gmsh/GetDP bundled under onelab: http://onelab.info/wiki/ONELAB
- Gmsh (stand alone): http://geuz.org/gmsh/
- Gnuplot: http://www.gnuplot.info/
- Grace: http://plasma-gate.weizmann.ac.il/Grace/
- Matplotlib: http://matplotlib.org/
- MayaVi: http://mayavi.sourceforge.net/
- MUMPS: http://mumps.enseeiht.fr/
- Numpy: http://www.numpy.org/
- Opendx: http://www.opendx.org/index2.php
- Open MPI: http://www.open-mpi.org/
- Paraview: http://www.paraview.org/

References

- PETsc: http://www.mcs.anl.gov/petsc/
- Salome platform: http://www.salome-platform.org/
- Salome-Meca (including Code_ASTER): http://www.code-aster.org/V2/spip.php?article303
- Scipy: http://www.scipy.org/
- SYRTHES: http://researchers.edf.com/software/syrthes-44340.html
- Visit: https://wci.llnl.gov/simulation/computer-codes/visit/