Gmsh mini-workshop: Interfacing solvers with ONELAB

C. Geuzaine, F. Henrotte, A. Jacques and R. Sabariego

In this workshop we will learn how to use the new ONELAB solver interface, for both existing and "home-made" solvers, in C++ and in Python.

Before proceeding any further, make sure to install a recent build of Gmsh. For your convenience, the "gmsh-getdp-*.zip" bundles from the ONELAB website contain both Gmsh, the GetDP solver and the "onelab.py" module.

1 Use the ONELAB functionalities within Gmsh

Gmsh is both the ONELAB server and a native C++ ONELAB client. We will first discover how you can use ONELAB within Gmsh itself:

- Open "indheat.geo" (in the "indheat" subdirectory) with Gmsh;
- Change the parameters and see how it affects the model (e.g. change "Terminals on the left?");
- Deselect the "Check model after each change" option in the little gear menu (bottom right of the tree menu) and examine its effect;
- Edit "indheat.geo" with a text editor and try to understand how parameters are defined;
- Modify one of your own geometries to bring some of your parameters into the GUI.

2 Use a local native C++ ONELAB client: GetDP

Let's now learn how to use another native C++ ONELAB client, installed locally on your computer: the GetDP solver (`http://geuz.org/getdp`).

- If GetDP is not installed on your computer, please install a recent build (if you installed the "gmsh-getdp-*.zip" bundle, GetDP is already installed in the same directory as the Gmsh executable);

- Open "pmsm.pro" (in the "machines" subdirectory) with Gmsh. This is a 2D model of a three-phase permanent magnet synchronous electric generator. Note that you can directly open the ".pro" file with Gmsh;
- Click on "Run";
- Display the current produced by this generator on a resistive load of 200 Ω in the phase A by clicking on the little graph icon next to "IA" and selecting "Top Left / Y". Use time as the ascissa on this graph by clicking on the graph icon next to the "Time" output in the "GetDP" node, and selecting "Top Left / X";
- Display other quantities of interest, e.g., the mechanical torque on the rotor "T_rotor";
- Change the Remanent induction of the magnets to 2.5 Tesla;
- Re-run the computation, first using a linear magnetic law, then a nonlinear law, and compare the results;
- Simulate the machine in motor mode, by setting a current source in the stator;
- Open the "pmsm.pro" file and check how the variables are defined. Do you recognize something?

Now that you are familiar with the basics of the native ONELAB interface in Gmsh and GetDP, it's time to customize your own mutli-physics problem:

- Open "magnetometer.pro" (in the "magnetometer" subdirectory) with Gmsh. This model describes a micro-magnetometer: current is flowing in a micro-beam, plunged in a static magnetic field; the Lorentz force is making the beam deform;
- Run the model and display the results using different formulations (Electrokinetic, Electrokinetic + Elasticity, Electrokinetic + Thermal);
- Modify the model to display interesting parameters and results in the GUI.

3 Use remote interfaced ONELAB clients: Elmer and Code_Aster

Interfaced clients are those clients that could not be (or were not yet...) modified to include ONELAB directly. We call them "interfaced" clients, by opposition to "native" clients.

Three reference open-source scientific calculations solvers will be considered here: Elmer (Multiphysics, `http://www.csc.fi/english/pages/elmer`) and Code_Aster (Structural

mechanics, `http://www.code-aster.org`). The three solvers are pre-installed on the virtual machine *amzyan* (remember this name) and are used by the ONELAB metamodel as *remote interfaced clients*.

For *interfaced* clients, ONELAB proceeds by pre-processing "instrumented" input files (with the ".ol" extension), to generate updated (i.e. reflecting the last changes made by the user via the GUI interface) input files for the client-solver before calling it. During this workshop, you are asked to analyze this input file instrumentation (which is done with a rather intuitive syntax), and to modify it. For *remote* clients, ONELAB operates the clients solvers through `ssh` and `rsync` technology.

3.1 How to work with the virtual machine amzyan

Here are the main commands to work with the virtual machine (vm) **amzyan**.

- Launch the vm:
  `VBoxHeadless -s "amzyan-jeos_0.3" -v off &`
- Check whether the vm is running:
  `ping -c 5 amzyan.local`
- If the latter fails (Linux users) try:
  `sudo service avahi-deamon start`
- Log in on the vm (add the `-Y` option if you want a graphic display):
  `ssh vm@amzyan.local` (passwd: vm)
- Upload a file or a directory on the vm:
  `rsync -av -e 'ssh' localdir/file vm@amzyan.local:remotedir`
- Export your id_rsa key on the vm to log in without being prompted for a password:
  `cd`
  `ssh vm@amzyan.local mkdir -p .ssh`
  `cat .ssh/id_rsa.pub | ssh vm@amzyan.local 'cat >> .ssh/authorized_keys'`
  `ssh vm@amzyan.local` (should no longer ask for a password)
- Shutdown the vm (from a host terminal):
  `VBoxManage controlvm "amzyan-jeos_0.3" acpipowerbutton`
- Shutdown the vm (from a vm terminal):
  `sudo shutdown -h now`

3.2 3D beam modelling with Elmer

The metamodel "beam.ol" is a rather self-contained metamodel that has been developed for didactical purposes at the UCL. It uses the ".ol" ONELAB syntax and gives a quite exhaustive overview of it. This syntax serves not only to instrument the input files of the solvers, but also to describe the metamodel itself. For the latter purpose, it is thus an alternative to the more recently developed python-based ONELAB syntax (See section 4).

- Launch the vm **amzyan** (see above);
- Go to the subdirectory `BEAM`;
- Type `gmsh ./beam.ol` at prompt or open the file `beam.ol` with Gmsh;
- Click on "Run";
- Modify parameters (loads, boundary conditions), re-run;
- Check how the material library works, note that there are parameters editable by the user, and other that are not editable (read-only);
- Take a look to the metamodel description (file "beam.ol");
- Look at how the instrumentation of the input file works (file "beam.sif.ol").

3.3 Trusses with Code_Aster

- If not yet done, launch the vm **amzyan** and check it is running (see above);
- Go to the subdirectory `ASTER`;
- Type `gmsh ./demo004a.ol` at prompt or open the file `demo004a.ol` with Gmsh;
- Click on "Run";
- Modify parameters, re-run;
- Add a geometrical ONELAB parameter in "demo004a.geo" for the width of the structure;
- Add a ONELAB parameter in "demo004a.comm.ol" for the horizontal component of the applied force.

4 Interface your own native Python ONELAB client

The file "pend.py" (in the "pendulum" subdirectory) contains a Python solver for the double pendulum problem. The Python script solves the governing mechanical equations over a prescribed interval of time and exports a solution file "pend.msh", which is automatically displayed by Gmsh at the end of the calculation. Additionally, this rather didactical metamodel illustrates how to call a native Python subclient "sub.py", and (if you have Gnuplot installed on your system), an interfaced subclient "gnuplot":

- Check that the module "onelab.py" is in the same directory as your Gmsh executable, or that your $PYTHONPATH contains the path to "onelab.py";
- You can uncomment the line "os.system('gnuplot pend.plt')" if you have Gnuplot on your system. Linux users might have to add the option "-persist" in this call, i.e. "os.system('gnuplot pend.plt -persist')";
- Type "gmsh ./pend.py" at prompt in the "pendulum" subdirectory, or open the file "pend.py" directly with Gmsh;
- Click on "Run";
- Click on the "Play" icon in the status bar at the bottom the Gmsh window;
- Play around with the available parameters, re-run, and see what happens;
- Make a Gmsh graph with the x- and y-coordinates of the tip of the pendulum;
- Take a look at the commented out examples of parameter decorations at the end of the file "pend.py" and make the GUI interface prettier;
- Create new ONELAB parameters to allow the user to interactively set the length of the arms of the pendulum individually;
- Look at how parameters are got and set; Develop the simulation summary in "sub.py".