ONELAB: Bringing Open-Source Simulation Tools to Industry R&D and Education

C. Geuzaine[1], F. Henrotte[2], J.-F. Remacle[2], R. V. Sabariego[3]

[1]Dept. of Electrical Engineering and Computer Science, University of Liège, Belgium, cgeuzaine@ulg.ac.be
[2]iMMC - MEMA, Université catholique de Louvain, Belgium, francois.henrotte@uclouvain.be
[3] Dept. of Electrical Engineering (ESAT), EnergyVille, KU Leuven, ruth.sabariego@esat.kuleuven.be

We present the ONELAB software, a lightweight open source toolkit to interface finite elements and related solvers used in a variety of engineering disciplines, and to construct multi-code models with maximum flexibility, efficiency and user-friendliness. ONELAB is freely available at http://onelab.info.

***Index Terms*—Open source software, modeling, finite element analysis, education.**

I. Introduction

The industrial and the academic world share a global need for scientific computation software, in many domains from mechanical and electrical engineering to chemistry and biomedicine. While licensing costs for commercial tools are justified for large companies that use them extensively, we have witnessed first hand that smaller, more occasional users cannot afford the costs. Open source software constitutes an alternative; for scientific computing, professional quality codes of high scientific value are available in various engineering disciplines since the early 2000's: OpenFOAM [1] for computational fluid dynamics, Code_Aster [2] for structural analysis, GetDP [3], [4] for electromagnetics... These codes are competitive when compared with their commercial counterparts, with regard to both their capabilities and their performance [5], [6].

However, these tools still have a marginal impact in small- and medium-size businesses and in education. We think that the main reason is their lack of a common easy-to-use interface (for pre- and post-processing as well as for parameter input), together with scarce (nonexistent) documentation and examples—at least for the codes originating from academia. Also, we believe that industry is still reluctant to adopt open source tools due to the ongoing confusion between "open source" and "limited", or "unprofessional" freeware.

This techno-economical analysis coalesces with the fact that industrial product developers need system-level simulation tools. This means tools with significant multi-physics capabilities, whereas specialized codes like OpenFOAM and Code_Aster remain essentially mono-physics. Existing platforms for multiphysics simulations offer solutions, both commercial (e.g. ANSYS Workbench [7] or COMSOL [8]) and open source (e.g. SALOME [9] or Elmer [10]). However, the former are again expensive and the latter either lack the sought-after nimbleness and user-friendliness due to a "heavy-weight" top-down design, or lack the ability to interactively interface multiple specialized codes.

This tailored our design goal for the ONELAB (Open Numerical Engineering LABoratory) software library, directly inspired by (and based upon) the design of the open source CAD modeler, mesh generator and post-processor Gmsh [11], [12]: create a fast, light and user-friendly interface to popular open source solvers in order to construct multi-code models with maximum flexibility and efficiency.

II. Multi-code Simulations

Literature and the authors' experience show that the standard approach to multiphysics modelling by addition of extra functionality to a reference solver has severe limitations. The additional modules must be (at least partially) rewritten, and they remain therefore usually at a rather low level of sophistication when compared to their equivalents in specialized codes.

The alternative is to proceed by directly interfacing the specialized software rather than by implementing new functionalities in an existing code. This multi-code approach, which is also that of a platform like SALOME, allows using specialized simulation codes always with their latest and most advanced functionalities. The difference between SALOME and ONELAB resides in that the latter is designed as a lightweight toolkit rather than an integrated platform.

III. Working Principles

To deal with multi-code models, which are basically series of interrelated and logically organised calls to simulation solvers, ONELAB provides the following elements:

- a scripting language to describe the succession of calls and tests that make up the multi-code models,
- a pooled parameter space acting as a persistent server for the called solver-clients,
- tools for inter-code communication on local or distant machines,
- tools for error detection and diagnosis, to check on whether the simulation runs smoothly and on the overall coherence of the model.

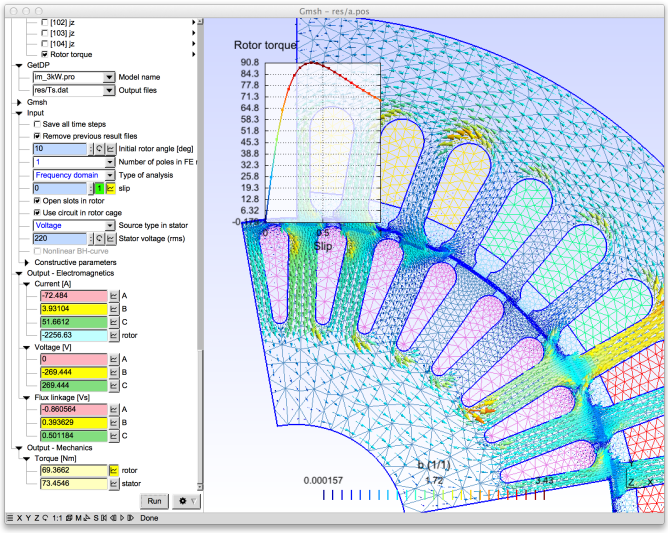

Fig. 1. Example of the graphical ONELAB front-end for a parametric induction motor model.

Practically, ONELAB is thus based on a client-server structure, with a server-side database and graphical front-end on the one hand, and local or remote clients communicating in-memory or through TCP/IP sockets on the other hand. A peculiarity of ONELAB lies in that it has no *a priori* knowledge about any specifics (input file format, syntax, ...) of the simulation codes it calls. In practice, this is made possible by having the simulation preceded by a analysis phase, asking the clients to upload their parameter set to the server and update their values. The issues of completeness and consistency of the parameter sets are thus completely dealt with on the client side: the role of ONELAB is limited to data centralization, interactive modification by the user and re-dispatching. The same philosophy applies to the CAD and post-processing layers, which are handled as regular clients.

There exist two ways of interfacing simulation codes with ONELAB. Native clients (e.g. GetDP, Gmsh or any C++ or Python code) use ONELAB as a library and implement the communication with the server of parameters directly from within the solver-client's code. Non-native clients, on the other hand, work by adding specific ONELAB commands directly into their input files. They are then interpreted by ONELAB, very much like a preprocessor would do, in order to generate valid updated input files for the solver, just before calling it. Native interfacing is more transparent but also more demanding. It requires code access (the solver needs be yours or open source) and an excellent knowledge of the code's internal architecture. Non-native interfacing, on the other hand, is ready-to-use as it uses the client solver through its natural input channels (input files and comand line options).

IV. Open Source vs. Commercial

The objective of ONELAB is to eventually offer to engineers and teachers a service similar to that of integrated commercial multiphysics packages (e.g. COMSOL, or Ansys-Workbench), but relying on open-source software. There are two main obstacles to this.

First, most potential users of ONELAB use computers with proprietary (Windows or Mac) operating systems, whereas many open-source high-level scientific codes are preferably distributed for Linux platforms. Multi-code modelling thus also often implies multi-platform compatibility issues, which ONELAB solves by providing tools for calling clients on distant machines (with other operating systems), or by distributing virtual machines with pre-installed solvers.

Second, software integration as realized in commercial packages allows extensive *a priori* testing and validation. Guaranteed consistency and stability makes up the added value of the product and justifies its cost. Upstream validation is however impossible with ONELAB, which is called to articulate heterogeneous solvers of various origins to build specific models. Consistency and stability need thus be ensured *a posteriori* by the design and the validation of multi-code template models. The added-value of the ONELAB approach lies in the distributed models, and not in the software itself. Fig. 1 shows an example of the graphical ONELAB front-end for an induction motor template model simulated with Gmsh [11], [12] and GetDP [3], [4]. All relevant parameters (geometrical but also those related with the power supply and the simulation) are made available to the user in the left panel. This template model can be solved as is, or serve as a model to be adapted to one's needs. This and many other examples can be found on the project's website: http://onelab.info.

Aknowledgments

This work has been supported in part by the Walloon Region (WIST3 No 1017086 "ONELAB", No 1217703 "FEDO") and by the Belgian Science Policy (IAP P7/02). The authors also wish to thank Patrick Dular, Johan Gyselinck, Maxime Graulich and Emilie Marchandise for their valuable contributions.

References

[1] OpenFOAM, the open source CFD toolbox: http://www.opencfd.co.uk/openfoam/
[2] Code_Aster: http://www.code-aster.org
[3] P. Dular, C. Geuzaine, F. Henrotte and W. Legros. *A General Environment for the Treatment of Discrete Problems and its Application to the Finite Element Method*. IEEE Trans. Mag. 34(5), p 3395–3398, 1998.
[4] GetDP: http://getdp.info
[5] G. Fernández, M. Meis and F. Varas. *Free software for numerical simulation: industrial applications*. Departamento de Matemática Aplicada II, Universidad de Vigo, Spain. XIII Spanish-French School Jacques-Louis Lions on Numerical Simulation in Physics & Engineering, Valladolid, 15-19 september 2008.
[6] A. Nilsson. *Some experiences on the accuracy and parallel performance of OpenFOAM for CFD in water turbines*. Lecture Notes in Computer Science, Volume 4699, p 168-176, 2009.
[7] Ansys Workbench, Platform for Advanced Engineering Simulation: http://www.ansys.com/Products
[8] COMSOL: http://www.comsol.com
[9] SALOME, the Open Source Integration Platform for Numerical Simulation: http://www.salome-platform.org
[10] Elmer, Open Source Finite Element Software for Multiphysical Problems: http://www.elmerfem.org
[11] C. Geuzaine and J.-F. Remacle, *Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities*. Int. J. Numer. Eng, 79(11), pp. 1309–1331, 2009.
[12] Gmsh: http://gmsh.info