Pyleecan: an open-source Python object-oriented software for the multiphysic design optimization of electrical machines

P. Bonneel, J. Le Besnerais, R. Pile, E. Devillers

***Abstract* -- This paper presents the first open-source development project for the electromagnetic design optimization of electrical machines and drives named Pyleecan – PYthon Library for Electrical Engineering Computational ANalysis.**

This paper first details the objectives of Pyleecan open-source development project, and the object-oriented architecture of the software that has been developed and released. Then it reviews and compares the available free and open-source software used during the multiphysic design of electrical machines including electromagnetics, heat transfer and vibro-acoustic analysis.

***Index Terms*—Simulation software, Open source, Electrical machines, Design optimization, Multiphysics**

I. Introduction

Many researchers, R&D engineers and PhD students in electrical engineering develop their own design software of electrical machines, often using Matlab [23] language and Femm [7] - the 2D open source finite element software for electromagnetics which has been downloaded more than one million times over 19 years [11]. A large part of this scripting work carried by electrical engineers consists in developing drawing functions of a parametrized magnetic circuit geometry (e.g. magnets, slot shapes, winding), writing post-processing scripts of electromagnetic results (e.g. calculation of back emf, losses, magnetic forces, efficiency maps, inductance look-up tables), coupling this electromagnetic model to sensitivity study or optimization tools, and developing data visualization tools.

This scripting work is rarely shared with the electrical engineering community. Such developments can suffer from quality flaws due to missing peer reviewing, and they are sometimes hardly maintainable due to missing documentation and lack of knowledge transfer in the R&D department, especially when PhD students leave their laboratory.

If this scripting effort was shared and unified in a common environment, research in electrical engineering could be more efficient, and researchers could spend more time on what may be more important in their daily work: physics and creativity. The scripting work would then only focus on the development of innovative architectures of electrical machines.

The key motivation of this paper is therefore to lay the foundations of the first open-source, collaborative development project of a simulation software of electrical machines and drives: Pyleecan, standing for "PYthon Library for Electrical Engineering Computational Analysis". Fig. 1 presents the logo of the project.

P. Bonneel, J. Le Besnerais, E. Devillers and R. Pile are with EOMYS ENGINEERING, 121 rue de Chanzy, Lille-Hellemmes, France (website: www.eomys.com, e-mail: contact@eomys.com).

PYLEECAN
Python Library for Electrical
Engineering Computational Analysis

Fig. 1. Pyleecan logo

This paper first details the Pyleecan (PYthon Library for Electrical Engineering Computational ANalysis) open-source project in terms of objectives, community rules, licensing and development roadmap. Then, it reviews and compares the most common free software or open source software used by electrical engineers when designing electrical machines. Note that at the time of writing the public repository of Pyleecan software is not yet accessible: this article aims at communicating a first vision of the project, in order to gather potential contributors and update priorities in the development roadmap according to the feedback of the electrical engineering community. It is then planned to open Pyleecan 1.0 repository in September 2018.

II. Pyleecan Open Source Project

A. *Objectives*

Pyleecan objective is to provide a user-friendly, unified, flexible simulation framework for the multiphysic design and optimization of electrical machines and drives based on open-source software. Its key features include:

- Python-based open-source software
- Graphical User Interface
- Object-oriented modeling of electrical machines and drives
- Multi-physic simulation of electrical machines
- Different levels of modeling accuracy (e.g. analytic and numeric)
- Multi-objective optimization loops including parallelization
- Output data visualization scripts
- Online detailed documentation

On a long-term basis, Pyleecan will include five physics: Electrical, Electromagnetics, Heat Transfer, Structural Mechanics and Acoustics. These physics are sequential: each one takes the results of the previous as inputs. The software enables to choose the model (analytical, numerical, calling an external software...) to use for each physic. It is also possible to run only the structural and acoustics computation by enforcing the magnetic flux distribution computed by another software for instance.

Pyleecan is open-source to encourage the research in electrical engineering. It provides guidelines to help any user to modify, improve or adapt the topologies and models. It also gives access to any computation step to analyze its output or workflow which is important for teaching and for software validation.

B. *Object-Oriented Modeling of Electrical Machines*

1) *Principle*

Object-Oriented Programming (OOP) is a programming paradigm based on the concept of "object". The point of using OOP is to gain in abstraction by using high level concepts (aka the objects). These entities are defined by their attributes (values that define the object, for instance the number of slot of a stator) and their methods (functions that can interact with the object attributes, for instance a method to compute the surface of a lamination).

In Pyleecan OOP is used to define a simulation by combining several objects together. Each object is dedicated to a particular computation with a particular model/method. For instance, there are three Magnetic model objects in Pyleecan that compute the airgap flux in three different ways (Analytical, subdomain, FEA). It enables the user to know exactly how every step of the simulation is computed and to switch easily from one method to another.

2) *Evolutivity of OOP*

OOP enables to define "interfaces", defining parts of the code as black boxes with predefined input and output formats. This way, several different objects can be used to model this part of the software provided that they follow the interface standards.

As an example, the surface of an electrical machine lamination can be defined by the surface of the equivalent cylinder minus the slot surfaces. The "functional programming" paradigm approach – the most common scripting method used by engineers in languages such as Matlab – would create a switch case to call a different function according to the slot shape of the lamination. On the contrary, the OOP approach consists in defining a "SlotTypeA" class as a template for a specific slot shape with attributes that define its geometrical parameters (height, width…). A method *comp_surface()* is added to the class to compute the surface of the slot according to its schematics and its attributes.

In the software workflow, an instance of this class will be created with the actual geometrical values of the machine slots. To compute the lamination surface, the *comp_surface()* method is called to get the slot surface according to the current values of the attributes.

This process seems to be equivalent to the functional programming paradigm approach, but it provides an abstraction in the computation of the lamination surface. In this function, the method *comp_surface()* of the slot object is called regardless of what is the slot object. A "SlotTypeB" class can be defined as a template for another slot shape with different attributes and a different way to compute the surface (in a method *comp_surface()* with the same input and output features, called a "prototype" in OOP). Then, in the simulation workflow, the instance of Slot_Type_B will be handled exactly as the instance of Slot_Type_A without modifying the other functions.

In functional programming paradigm (i.e. embedding parts of the code within functions), adding a new slot shapes would require editing all the functions using slots (for instance by adding a new option to all the switch cases). This process is a significant source of errors and is very time consuming. With the OOP paradigm, adding a new slot only requires to define a new class with its attributes and its methods matching the slot interface requirements.

In Pyleecan, this feature is used to handle different machine topologies and different models within the same simulation workflow. It provides a convenient way to extend the software features. Moreover, with this architecture, contributors that are not familiar with computing science or OOP can contribute by following the class template and focus on the scientific part.

3) *Efficiency of OOP*

OOP enables to make a link between similar objects and reuse the scripting work of their methods by "inheritance": a "daughter class" that inherits from a "parent class" takes all its attributes and methods ; one can then define additional methods, or redefine these inherited methods in the daughter class.

introduces an example of inheritance between the Squirrel Cage Induction Machine (SCIM) and Doubly Fed Induction Machine (DFIM) classes.

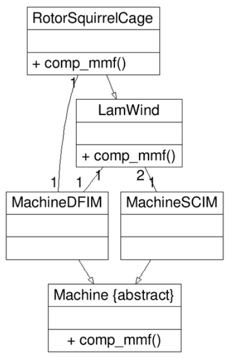

Fig. 2. Simplified Pyleecan UML class diagram

In this example, the combination of a "LamWind" (a lamination with winding) for the stator and a "RotorSquirrelCage" for the rotor defines a "MachineSCIM" class. "RotorSquirrelCage" class is defined as a particular case of "LamWind" class with inheritance, as the short circuited bars of an induction machine can be seen as a special winding case. This way, all the scripting work done on "LamWind" class methods can be reused in the "RotorSquirrelCage" class. For instance, the *comp_mmf()* method is defined in the "LamWind" class to compute the winding magnetomotive force. If this method is generic enough to be used in both classes, there is no need to define a *comp_mmf()* method in "RotorSquirrelCage". Otherwise, the inheritance enables to overwrite the *comp_mmf()* method of "LamWind" by a new method that could take into account the specificities of the squirrel cage mmf computation.

To compute the overall mmf of the machine, the corresponding method will call the respective functions *comp_mmf()* of the rotor and stator, and combine their results regardless of how they were computed thanks to the abstraction of the OOP approach.

Finally, to extend the SCIM class to a DFIM, the only necessary operation is to define a new machine type with a "LamWind" object for both rotor and stator; the same method *comp_mmf()* as for the "MachineSCIM" class can be used, so scripting is more efficient.

C. *Why Python?*

Pyleecan is coded in Python for three main reasons.

Firstly, it is an interpreted language. It is not as efficient (computation time wise) as compiled language (like C++ for instance) but it provides some enhanced scripting features that are useful when running scientific calculations. For instance, several pre/post processing scripts can be included in the simulation workflow to customize it. If necessary, some packages are available to precompile Python code and speed-up computation time to be almost as efficient as C (Cython for instance).

Secondly, Python is free and has a very active scientific community. Most functionalities of a commercial language such as Matlab (numerical calculations, signal processing, data visualization) are available as open source packages, which can be reused or customized to speed up the development of Pyleecan project. For instance, presents an electrical machine plot with Matplotlib package. Other useful packages include Numpy for matrix calculations or Sympy for symbolic computation.

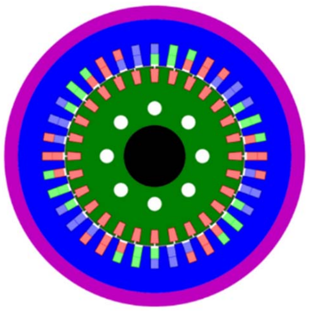
Fig. 3. Electrical machine plot with Matplotlib

Finally, Pyleecan aims at gathering a community of electrical engineers who may not be used to computing science. The complexity of C++ (pointers, compilation…) could discourage them to contribute to the project, and one of Python core rules is that "readability counts". Python language is therefore straightforward, simple to read [20], and close from Matlab already widely used by engineers. Besides, some development environments as user-friendly as Matlab can be found in Python (e.g. PythonXY).

D. *Python in Electrical Machines Design Tools*

Several Python packages or Python software already exist and may be coupled to Pyleecan:

- PySimulator [26]: GUI to define and run OpenModelica [27] simulation models, for use in AC drive simulation
- Dolomites [28] (formerly Koil): winding design tool
- Pyfemm [30]: API for coupling with Femm
- Femagtools [29]: API for coupling with FEMAG
- Scipy.optimize: optimization tools

Femagtools is a Python-API for Femag to transparently execute parameter studies and multi-objective optimization either locally on a server or using multiple servers in a network or a cloud infrastructure.

More generally there exists a large number of Python packages for design of experiments, multiobjective optimization, parallel computing, data mining and data visualization.

All commercial finite element electromagnetic software now propose high level Python scripting (e.g. PyFlux in Flux®) to drive their simulation process. Pyleecan will therefore be able to easily drive third party electromagnetic FEA software when necessary.

E. *Position of EOMYS*

EOMYS, a privately-owned company, has initially developed an object-oriented model of electrical machines based on Python within its MANATEE commercial software [1] dedicated to the fast electromagnetic and vibroacoustic design of electrical machines.

MANATEE software handles various topologies of radial flux electrical machines (e.g. inner and outer rotor surface, inset or interior permanent magnet synchronous machines, squirrel cage and doubly fed induction machines, wound rotor synchronous machines, synchro-reluctant machines) and several geometrical overlays (e.g. slot shapes, magnet shapes). MANATEE is already coupled to Femm/XFemm, Femag and GetDP. Its object-oriented model of electrical machines including their AC drive is a natural basis to bring together the key open-source initiatives such as Femm/XFemm, OneLab and Dolomites.

The initial contribution of EOMYS to Pyleecan open-source project therefore includes:

- the free release of its object-oriented model of electrical machines (WRSM, SPMSM, IPMSM, SCIM, DFIM)

- the share of documentation on geometry overlays used in MANATEE (see eomys.com)

- the share of electromagnetic validation cases (see eomys.com)

EOMYS will be the first maintainer of Pyleecan and intends to provide full time developers to the project. In return, EOMYS aims at integrating Pyleecan to its MANATEE software, which could then benefit from some user contributions (e.g. new overlays, extension to linear motors or axial flux machines).

F. *Licensing*

Pyleecan is under an Apache license. This license is permissive (with no copyleft) which allows to use Pyleecan even in a commercial close-source software. The Apache license was chosen to open the community to as many contributors as possible.

One of the key feature of Pyleecan is it flexibility, and in particular the possibility to be coupled with almost any software. With this license, private companies can include Pyleecan in their electrical engineering design workflow without the need to provide the resulting software code. Companies may keep part of their work with Pyleecan confidential, but they should be more prone to contribute to the open-source project to increase the robustness of their developments.

G. *Community Rules*

One of Pyleecan goal is to provide a universal framework for the development of new electrical engineering models. It will provide code guidelines (such as naming convention or documentation templates) to ensure the consistency of the code. OOP approach provides a natural structure for scripting work.

To organize the community, EOMYS plan to schedule 4 meetings per year (every 3 months) to coordinate development efforts and synchronize resources of interns, Master's thesis and PhD students. These meetings can be held at conferences like EuroSciPy (European conference for scientific Python usage) or scientific conferences on electrical machines (e.g. ICEM, COMPUMAG, ISEF).

H. *Code of Ethics and Code of Conduct*

An Open Source software should theoretically comply with the statement *"No discrimination against fields of endeavor"* [15]. The annotation says "*the major intention of this clause is to prohibit license traps that prevent open source from being used commercially*". Pyleecan aims at being an open source project, but EOMYS as founder and main contributor would like to avoid the use of Pyleecan for the design, development or production of electric actuators intended to be used in military weapons or in defense applications.

One aim of Pyleecan project is to become a catalyst for research in electrical machines applied to sustainable mobility and energy production, towards more efficient, environmental-friendly electrical drives.

Pyleecan also follows a code of conduct for its development community inspired by the Django code of conduct (https://www.djangoproject.com/conduct/).

I. *Resources*

Although the initial sources of Pyleecan fully come from EOMYS company, the core development team should contain developers both from private companies and public laboratories. The core development team should reflect Pyleecan user community. It consists now of

- Pierre Bonneel, software engineer at EOMYS (Manatee)
- Emile Devillers, R&D engineer at EOMYS (Manatee)
- Christophe Geuzaine, Professor at ULg (GetDP)
- Johan Gyselinck, Professor at ULB (GetDP)
- Ronald Tanner, software architect at Semafor (Femagtools).

EOMYS commits to contribute to Pyleecan with at least one equivalent full-time employee in 2018-2019.

J. *Development Roadmap*

The initial development roadmap includes the following tasks, their priority will be ordered according to the contributors needs

- Generic geometry templates (to be used in all the model and to provide guidelines to extend the software)
- Coupling with Femm/XFemm (2D) for PMSM magnetostatics
- Coupling with Syr-E
- Coupling with Dolomites
- Coupling with Gmsh/GetDP (2D) for PMSM magnetostatics
- Coupling with Femag (2D) for PMSM magnetostatics.
- Coupling with GetDP (2D) for IM magneto-harmonic
- Flexible simulation workflow (multisimulation, possibility to import data from any software, support to cloud computing)

K. *How to Join?*

You can subscribe to Pyleecan newsletter at the following link: http://eepurl.com/dov3PH. You can also subscribe to our repository on Github to be warned of the changes.

III. REVIEW OF OPEN SOURCE / FREE SOFTWARE USED IN ELECTRICAL MACHINE DESIGN

A. Why a new open source project?

Before creating a new open source project, it is important to list the existing ones to know if it doesn't already exist. It is better to contribute to an existing project rather than creating an equivalent one. Too many open source projects died because of lack of contributors. This part presents the existing electrical engineering open source project and what differentiates them from the Pyleecan project.

B. SWOT Analysis of Open-Source Software

Strengths:

- Productivity: researchers can work more efficiently, as it avoids "reinventing the wheel"
- Flexibility: open access to the code allows customization
- Reliability: "many eyes see every bug"
- Cost: anyone can use the software freely, even on clusters

Weaknesses:

- Durability: without funding, the project can lose its contributors and stop being maintained
- Management: the community needs to be well organized to drive the software development
- Reliability: technical support and validations are not mandatory

Opportunities:

- The market of electrical machine design software is shared by a few companies
- The challenges of sustainable energy and mobility require innovative software tools

Threats:

- The community may want to develop a software different from the one intended by the original maintainers
- Some modifications of the code made by private companies may never be provided to the community

C. Software Licenses

Before reviewing the different software generally used by electrical engineers when designing electrical machines, it is important to clarify the licensing systems and the differences between open source and free software.

The terms "free software" and "open source" software historically refers to two different movements initiated respectively by the Free Software Foundation (FSF, 1985) and by the Open Source Initiative (OSI, 1998). For FSF "free software" meant "free as in free speech, not as in free beer". One example of FSF developments is the GNU General Public License (GPL) which guarantees the rights of end-users to run, view, and share source code freely. Founded after the FSF, the OSI decided not to use the word "free" to avoid confusion and was initiated to emphasize the benefits of community-driven collaborative development.

This paper does not only discuss about free (as in beer) software use in electrical engineering, because the important question is more about how flexible software are rather than how cheap they are when dealing with advanced R&D work: applied research in electrical machines is expected to regularly push the conventional models to their limits, requiring regular updates of the geometrical layouts, material libraries, numerical solvers, post-processings, documentation, etc. This requires the software to be open-source, allowing the end-users to modify the software so that it fulfil new needs.

Some examples of licenses which can be found among software used by electrical engineers are presented:

- GNU GPL: everyone is free to use the software and redistribute it on a free basis. It is a copyleft license which means that derivative work can only be distributed under the same license terms (i.e. open source).
- BSD: it allows to modify and distribute the software's code in the source or binary format as long as a copy of the copyright notice, list of conditions, and the disclaimer are retained. It is not a copyleft license: it is permissive for the redistribution, even in commercial close-source software.
- Apache: similar to BSD license, plus more explicit patent rights and the obligation to explicitly list all the modifications done to the original software in any redistribution.
- Aladdin Free Public License: the results of the program can be used for any purpose (including commercial), but a special license is needed to resell the program itself or to include parts of the source code in a new commercial product.

D. Review of Electromagnetic Design Software

This part reviews free and open-source electromagnetic design software. It presents the limitation of each software that justified the creation of the Pyleecan project. Table 1 summarizes the different studied software.

iMOOSE (Innovative Modern Object-Oriented Solving Environment) has been initially developed in the Department of Electrical Machines of Aachen University, Germany. It is no longer maintained, the last release was on the 03/11/2003 [17]. Femag3D project [22] has been stopped in 2013 and the software currently belongs to the Institute of Electrical Energy Conversion of TU Darmstadt, Germany - no open-source license has been released.

Femag is a software package for the 2D-FE-simulation of electric machines (static magnetic fields, temporally sinusoidal currents or fields, mechanical deformations and stresses, temperature fields, etc) initially developed by the Institute for Electrical Machines of ETH-Zurich. It is available under a commercial license but may be used freely for non-commercial projects and academia. It can model both induction motors and permanent magnet synchronous machines, but without circuit coupling. Femag can be driven with a Python library called Femagtools.

Similarly, Femm [7] is limited to 2D and cannot be used for coupled circuit simulation. Femm core is in C but the simulation models can be defined and solved using Lua higher level programming language. Femm is coupled to Scilab, Octave/Matlab. XFemm [4] allows to call Femm without the use of ActiveX, enabling parallel execution.

OneLAB based on GetDP [25] and Gmsh [24] is a 2D/3D computational tool for multi-physics problems which has been applied to electrical machines [1] ; it can simulate strong circuit coupling. If the user community is smaller compared to Femag or Femm, a library of electrical machine examples has been started.

Elmer is also a 2D/3D computational tool for multi-physics problems, its solver Elmer FEM is open-source. Started in Finland in 1995, it has been recently applied to electrical machines design in several works [1][2].

SMEKlib is an open source 2D FEA library for electrical machines developed in Matlab (object oriented) and released in 2017 [1].

MotorAnalysis is a free software for 2D FEA of induction machines and PM synchronous machines. The release date and licensing are unknown to the authors.

Released in 2014, Syr-e [11] is a Matlab-based simulation open source software initially developed at Politecnico di Bari and Politecnico di Torino Universities (Italy) for the design of synchronous reluctance machines, based on a coupling to Femm. It is compatible with Octave under GNU GPL as an alternative to the proprietary language Matlab. It is coupled with a multiobjective optimization tool.

E. Review of Other CAE Software for Electrical Machine Design

Winding design

Several tools for the design of the winding of AC electrical machines have been developed and released.

Dolomites project [28], released in 2017, is a Python-based GUI for winding design which was formerly developed in C++ under the name Koil at the Electric Drives Lab (EDLab) of the University of Padova, Italy. An export to GetDP of the winding circuit definition is now included.

Anfractus tool [23] is a Matlab-based free software for the design of electrical machines winding released in 2018. It is free but not open source contrary to Dolomites.

Control & electronics

Some specific modelling tools are used when looking at the electrical machine as a system for control purposes. Scilab/Scicos can be used for Simulink-typed dynamic modelling of electrical machines control [12] based on free software.

OpenModelica (license OSMC-PL 1.2) can be also used to make dynamic simulation of AC drives [14][13] as illustrated in Fig. 2.

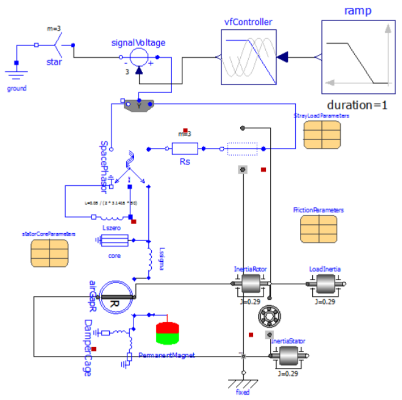

Fig. 4: Example of PMSM inverter drive model in OpenModelica [1][13]

Others

Other open source or free works in electrical engineering can be found on the internet (SourceForge, GitHub, Matlab FielExchanche repositories) but they are not backed by scientific publications and/or are no longer maintained.

F. Review of Multiphysic Design Software

III. The electrical machine designer must include heat transfer analysis during the first electromagnetic design loops. Similarly, acoustic noise and vibrations due to magnetic forces which depend in particular on slot/pole combination and lamination dimensions should also be assessed in pre-sizing phase. It is therefore important to account for multiphysic simulation capabilities, especially heat transfer, structural mechanics and acoustics. Usually all electromagnetic FEA software can solve heat diffusion equation and can then handle simple conducto-convective heat transfer problems.

Table 2 summarizes the different studied multiphysic software.

TABLE 1
ELECTRICAL ENGINEERING FREE / OPEN-SOURCE SOFTWARE

Name	License	Description	Language
Femm [7]	Aladdin Free Public License	2d non linear time-harmonic electromagnetic FEA	C++ Lua
OneLab (Gmsh & GetDP) [1]	GNU GPL	2d or 3d non-linear magnetoharmonic, multiharmonic or time stepping electromagnetic FEA	C++ Python
Elmer [2]	GNU GPL v2	2d or 3d [1]	Fortran 90, C and C++
Femag2d	Proprietary license but free for academic work	2D static or temporally sinusoidal changing electromagnetic fields	Fortran-2003, C/C++, Lua
code_Carmel	Proprietary license (L2EP/EDF) but free for academic work	3D non-linear time stepping electromagnetic FEA, including circuit coupling	Fortran 90
Syr-E	Apache 2.0	2d non linear magnetostatics (Femm based)	Matlab/Scilab
SMEKlib	MIT	2D-FEA Library for Electrical Machines in Matlab, non linear time stepping or time harmonic	Matlab/Scilab
MotorAnalysis	?	2D-FEA software for induction motor and PM machines	Matlab
Femag3d [22]	?	3d non linear magnetostatic FEA (GetDP based)	Lua
iMOOSE [17]	GNU GPL v2	2d or 3d static, time–harmonic and transient electromagnetic	C++

TABLE 2
MULTIPHYSIC ELECTRICAL ENGINEERING SOFTWARE

Name	License	Description
Elmer	GNU GPL v2	Multiphysics
Code_Aster	GNU GPL v3	Structural mechanics Acoustics
OpenFoam	GNU GPL v3	Fluid mechanics
Kratos Multiphysics	BSD	Heat transfer Fluid dynamics
Femm	*Aladdin Free Public License*	Heat transfer diffusion
GetDP	GNU GPL v2	Multiphysics
Agros2d	GNU GPL	Multiphysics

G. Review of Pre/Post Processing Software

Some specific software are used for pre-processing, meshing, and post processing / data visualization.

Gmsh [24] is a mesh generator released under the GNU GPL containing modules for geometry description, meshing, solving and post-processing. Gmsh supports parametric input and since version 3.0 supports constructive solid geometry features, based on Open Cascade technology.

Salome [31] is an open-source software providing a generic platform for pre/post processing of numerical simulations. Its cross-platform solution distributed under the GNU LGPL license also uses Open Cascade. Code-Aster and Salome platform are both based on the scripting language Python.

ParaView is an open source cross-platform application for interactive, scientific data visualization under BSD license.

H. Conclusions and Choice of Pyleecan Modules

Although several FEA open source electromagnetic software are available, none of them includes at the same time

- a library of electrical machines topologies and layouts
- a library of magnetic materials
- a Graphical User Interface
- an intuitive scripting mode based on object-oriented approach

Besides, several free software initiatives are carried in Matlab commercial software. Even if Matlab code can be compatible with Scilab or Octave, some of them use Matlab toolboxes and cannot be executed without a Matlab commercial license. The files are sometimes encrypted in pcode so that the scripts are not open source and cannot benefit of collaborative developments. The authors strongly believe that these initiatives have no future when developed in proprietary languages such as Matlab, and w the authors of these packages are encouraged to join Pyleecan project and translate their code to Python.

Several open-source-based initiatives have been given up due to lack of resources and technical support: this is one reason for which Pyleecan will be distributed under a software license that allows commercial use.

IV. Conclusions

Several software developments have been initiated in the community of researchers and R&D engineers to ease the design of innovative electrical machines using free or open source software. Pyleecan open-source project aims at unifying and coordinating development efforts starting from a Python-object-oriented model of electrical machines released by EOMYS. This object-oriented model provides an architecture that can be easily improved and extended even by engineers with few skills in computing science.

VI. Acknowledgements

We thank J. Gyselinck, C. Geuzaine and R. Tanner for useful discussions around this paper.

V. References

[1] P. Ponomarev, J. Keränen, M. Lyly, J. Westerlund and P. Råback, "Multi-slice 2.5D modelling and validation of skewed electrical machines using open-source tools," *2016 IEEE Conf. Electromagnetic Field Computation (CEFC)*, Miami, FL, pp. 1-1.

[2] P. Ponomarev, J. Keränen and P. Pasanen, "Electromagnetic transient finite element 3D modelling of electrical machines using open-source tools," *2016 XXII Int. Conf. Electrical Machines (ICEM)*, Lausanne, pp. 1657-1661.

[3] S. Sathyan, A. Belahcen, J. Kataja, T. Vaimann and J. Sobra, "Computation of stator vibration of an induction motor using nodal magnetic forces," *2016 XXII Int. Conf. Electrical Machines (ICEM)*, Lausanne, pp. 2198-2203.

[4] R. Crozier and M. Mueller, "A new MATLAB and octave interface to a popular magnetics finite element code," *2016 XXII Int. Conf. Electrical Machines (ICEM)*, Lausanne, pp. 1251-1256.

[5] E. A. Lomonova, J. J. H. Paulides, S. Wilkins, and J. Tegenbosch, "ADEPT: ADvanced electric powertrain technology - Virtual and hardware platforms," *2015 X International Conference on Ecological Vehicles and Renewable Energies (EVER)*, Monte Carlo, pp. 1-10.

[6] J. Keränen *et al.*, "Efficient Parallel 3-D Computation of Electrical Machines With Elmer," in *IEEE Trans. Magn.*, vol. 51, no. 3, pp. 1-4, 2015.

[7] FEMM (Finite Element Method Magnetics) by D. Meeker. [Online]. Available: http://www.femm.info

[8] MANATEE software (Magnetic Acoustic Noise Analysis Tool for Electrical Engineering) by EOMYS ENGINERING. [Online] Available: http://www.manatee-software.com

N.N. Zablodskii, V.E. Pliugin, and A.N. Petrenko, "Using object-oriented oriented design principles in electric machines development", *Electrical Engineering & Electromechanics, Kharkiv*, no. 1, p. 17–20, 2016.

[10] C. Kral and A. Haumer, "Object Oriented Modeling of Rotating Electrical Machines", *Advances in Computer Science and Engineering*. InTech, 2011. [Online]. Available: https://www.intechopen.com/books/advances-in-computer-science-and-engineering/object-oriented-modeling-of-rotating-electrical-machines

[11] D. Meeker *et al.*, "Electrical machine analysis using free software," *2017 IEEE Energy Conversion Congress and Exposition (ECCE)*, Cincinnati, OH, pp. 1-289.

[12] P. Aree, "Steady-state analysis of self-excited induction generator using scilab open-source software," *2014 2nd IEEE Conference on Power Engineering and Renewable Energy (ICPERE),* Bali, pp. 185-188.

[13] G. Lalovic, "AC drives modeling in Modelica", Bachelor's Thesis, BRNO University of Technology, 2014.

[14] M. Ceraolo, "A new Modelica Electric and Hybrid Power Trains library", *2015 11th Int. Modelica Conf.*, Versailles, France, pp. 785-794.

[15] Open Source Initiative. [Online]. Available : https://opensource.org/osd-annotated

[16] Open Life Cycle Assessment (OpenLCA) software. [Online]. Available: http://www.openlca.org

[17] J. Keränen, A. Manninen, and J. Pippuri, "Multi-physics simulations for electrical machine development", VTT Research report, 2015. [Online] Available: http://www.vtt.fi/inf/julkaisut/muut/2015/VTT-R-04618-15.pdf.

[18] A. Lehikoinen, T. Davidsson, A. Arkkio, and A. Belahcen (in press), "A High-Performance Open-Source Finite Element Analysis Library for Magnetics in MATLAB," *2018 Int. Conf. Electrical Machines (ICEM),* 2018.

[19] D. van Riesen, C. Monzel, C. Kaehler, C. Schlensok, and G. Henneberger,"iMOOSE-an open-source environment for finite-element calculations," *IEEE Trans. on Magn.,* vol 40, no. 2, pp. 1390 - 1393, 2004.

[20] T. E. Oliphant, "Python for Scientific Computing," *Computing in Science & Engineering,* vol. 9, no. 3, pp. 10-20, 2007.

[21] J. M. Díaz-Chacón, C. A. Hernández, V. M. Brauer, A. N. Valle, R. B. B. Ovando-Martínez, and A. A. Adeniyi, "Development of a didactic set of 3D-FEM magnetostatic simulations by using a free software," *2017 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, Ixtapa, pp. 1-6.

[22] B. Funieru, "FEMAG 3d status", Proceedings of FEMAG Anwendertreffen, Zürich, October 2013

[23] D. Ouamara, F. Dubas, M. Nadjib Benallal, S. A. Randi, C. Espanet, « Automatic winding generation using matrix representation - Anfractus tool 1.0", Journal of Advanced Engineering, Acta Polytecnica, Vol 58, n°1, 2018

[24] C. Geuzaine and J.-F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. International Journal for Numerical Methods in Engineering 79(11), pp. 1309-1331, 2009.

[25] 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," in *IEEE Transactions on Magnetics*, vol. 34, no. 5, pp. 3395-3398, Sep 1998.

[26] Pfeiffer A., Hellerer M., Hartweg S., Otter M. and Reiner M.: PySimulator – A Simulation and Analysis Environment in Python with Plugin Infrastructure. In: Proceedings of 9th International Modelica Conference, Munich, Germany, Sept. 2012.

[27] Ganeson A. K., Fritzson P., Rogovchenko O., Asghar A., Sjölund M. and Pfeiffer A.: An OpenModelica Python Interface and its use in PySimulator. Accepted for publica-tion in the Proceedings of 9th International Modelica Conference, Munich, Germany, Sept. 2012

[28] Dolomites software, https://sourceforge.net/projects/dolomites/

[29] Femagtools Python library, https://pypi.python.org/pypi/femagtools

[30] Pyfemm Python library, https://pypi.python.org/pypi/pyfemm/0.1.0

[31] SALOME Platform, www.salome-platform.org

[32] Matlab, The MathWorks, Inc., Natick, Massachusetts, United States.

VII. Biographies

P. Bonneel graduated in 2014 from the "Ecole Nationale Supérieure des Sciences Appliquées et de Technologie" of Lannion (ENSSAT) in signal analysis, computing science and electronics. After a first experience in software development in speech synthesis at Voxygen, he currently works in EOMYS ENGINEERING as an R&D engineer in informatics. He is responsible of the development and support of MANATEE software (Magnetic Acoustic Noise Analysis Tool for Electrical Engineering), a Python-based simulation software of electrical machines specialized in the calculation of electromagnetic noise and vibrations.

J. Le Besnerais currently works in EOMYS ENGINEERING as an R&D engineer on the analysis and reduction of acoustic noise and vibrations in electrical systems.

Following a M.Sc. specialized in Applied Mathematics (Ecole Centrale Paris, France) in 2005, he made an industrial PhD thesis in Electrical Engineering at the L2EP laboratory of the Ecole Centrale de Lille, North of France, on the reduction of electromagnetic noise and vibrations in traction induction machines with ALSTOM Transport. He worked from 2008 to 2013 as an engineer in the railway and wind industries (Alstom, Siemens Wind Power, Nenuphar Wind) on some multiphysic design and optimization tasks at system level (heat transfer, acoustic noise and vibrations, electromagnetics, structural mechanics and aerodynamics). In 2013, he founded EOMYS ENGINEERING, a company providing applied research and development services including modeling and simulation, scientific software development and experimental measurements.