POLITECNICO DI TORINO

Master's Degree Programm in Electrical Engineer

Master's Thesis

e-Machine Export and Simulation from SyR-e to Ansys via Python

Supervisor:

- Prof. Gianmario Pellegrino

.....

Co-supervisor:

- Eng. Paolo Ragazzo

.....

Candidate:

Aimo Riccardo

.....

Academic Year 2020/2021

Acknowledgements

Firstly, I would like to express my deepest appreciation to Professor Gianmario Pellegrino for giving me the opportunity to have such a valuable thesis experience and for taking me closer to the world of electric machine design.
I would also like to give special thanks to engineer Paolo Ragazzo for his crucial support.
And so I would like to acknowledge everyone I have feelings for because they have been essential in getting through these five years, especially my parents Angela and Enrico that always believed in me.

Abstract

SyR-e is an open-source software developed by the PEIC group at PoliTO for the design of synchronous motors, allowing accurate electromagnetics analysis and trade-offs evaluation, but with limited multi-physic simulation capability in the thermal and mechanical environments. A first step in moving toward a multi-physics approach, which is key to achieving complete electrical machine design, is to couple software capable of performing such a study with SyR-e. The Ansys software family includes an electromagnetic field solver, a structural analysis, and a computational fluid dynamics (CFD) tool for accurate thermal simulations.

The present thesis aims to make the two software compatible from the electromagnetic point of view to give the future possibility to analyze the e-machine at 360 degrees.

The first goal was to implement the model export from Synchronous Reluctance Evolution (SyR-e) to **Ansys Maxwell** valid for any machine feasible in SyR-e and then launching a Finite Element Analysis (**FEA**) in Maxwell through the SyR-e interface.

Firstly the results obtained were validated with those obtained with **FEMM** a client for magnetostatic finite-element analysis that is already integrated in SyR-e and then the analysis was validated referring to an IPM machine for vehicular traction, focusing on the FEA evaluation of iron losses. The results are validated against existing data of the benchmark, provided by the partner company leader in the automotive industry.

Contents

List of Figures

List of Tables

List of Acronyms

FEA Finite Element Analysis

FEMM Finite Element Method Magnetics

GUI Graphical User Interface

IPM Interior Permanent Magnet

MMM Magnetic Model Manipulation

NDA Non-Disclosure Agreement

PM Permanent Magnet

SPM Surface Permanent Magnet

SyR Synchronous Reluctance

SyR-e Synchronous Reluctance Evolution

PMASR Permanent Magnet Assisted Synchronous Reluctance

1 Introduction

1.1 Objectives

The purpose of this work was to include additional functionalities beyond the ones already present in the open-source **SyR-e** (Chap.1.2) open source software. The first goal was to implement the model export from SyR-e to **Ansys Maxwell** (Chap.1.3) in order to enhance the design process. The second goal was to launch (Chap.**Finite Element Analysis (FEA)**) in Maxwell through the SyR-e interface. The results obtained from this new interface are validated with those obtained with **Finite Element Method Magnetics (FEMM)**, a client for magnetostatic finite-element analysis that is already integrated in SyR-e.
The analysis is validated referring to an interior permanent magnet machine for vehicular traction, focusing on the FEA evaluation of iron losses. The results are validated against existing data of the benchmark, provided by the partner company. In conclusion, the SyR-e to Ansys export and simulation feature was tested valid for any IPM machine created in SyR-e, as well as for machine of the Synchronous Reluctance and Surface-Mounted PM Synchronous types. Upon validation, the contributions from the thesis are now part of the open-source repository of the SyR-e project.

1.2 SyR-e - Synchronous Reluctance Evolution

SyR-e, acronym of Synchronous Reluctance - evolution, is an open source code born from the collaboration between the Politecnico di Torino and the Politecnico di Bari in 2014. The program was born thanks to the OctaveFEMM Matlab toolbox of FEMM [5] with the aim of designing Synchronous Reluctance (SyR) motors. SyR-e is used to design and model synchronous reluctance machines by using FEA associated with design equations and multi-objective optimization algorithms.

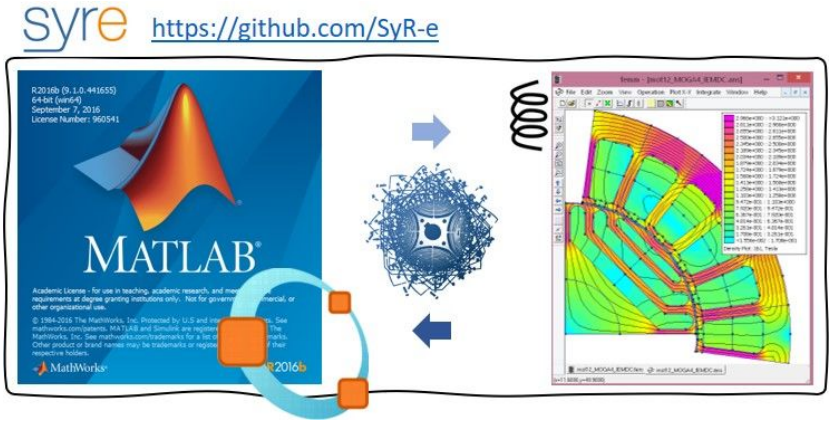


Fig. 1: SyR-e schematics

The software is developed in the Matlab environment and relies on the free FEMM client for magnetic simulations. Over time other CAD software suites such as Motor-CAD [4] and Simcenter Magnet [6] have been coupled to SyR-e. In all cases, the results obtained through the finite element software are returned to Matlab and reprocessed. The Parallel Computing toolbox of Matlab permits to execute several FEA simulations simultaneously and thanks to this function it is possible to reduce considerably the calculation time of some simulations.

SyR-e has two graphical user interfaces (GUI): GUI_Syre (Fig.2), for machine design and FEA simulation and GUI_Syre_MMM (Fig.3) for the manipulation of the machine data towards efficiency maps evaluation and simulation of the e-drive control.

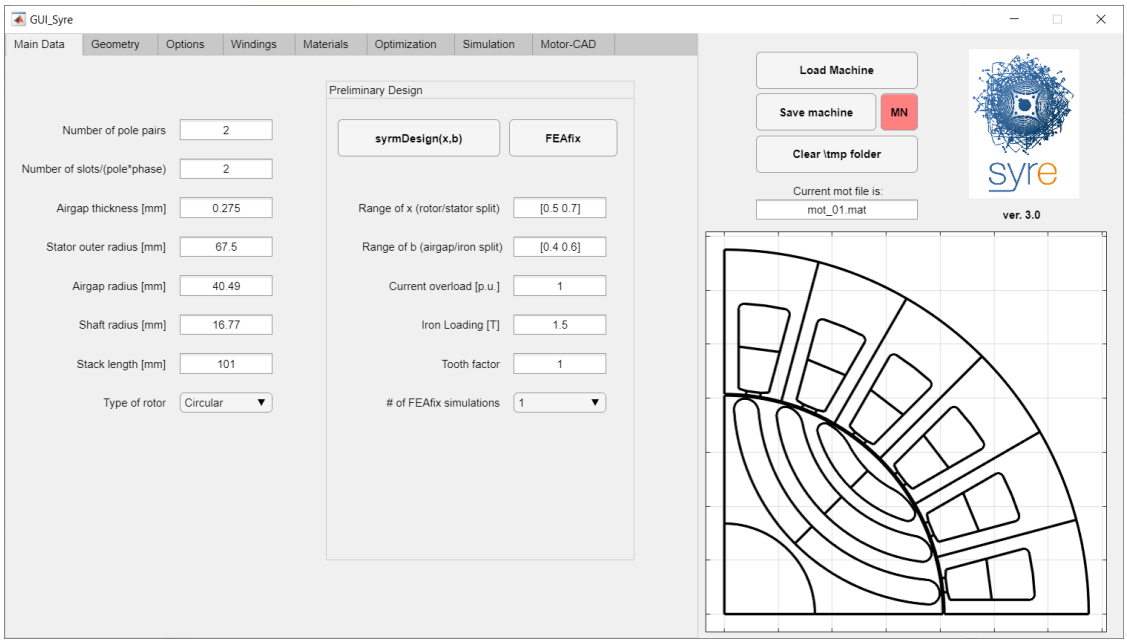


Fig. 2: Main data tab of SyR-e GUI

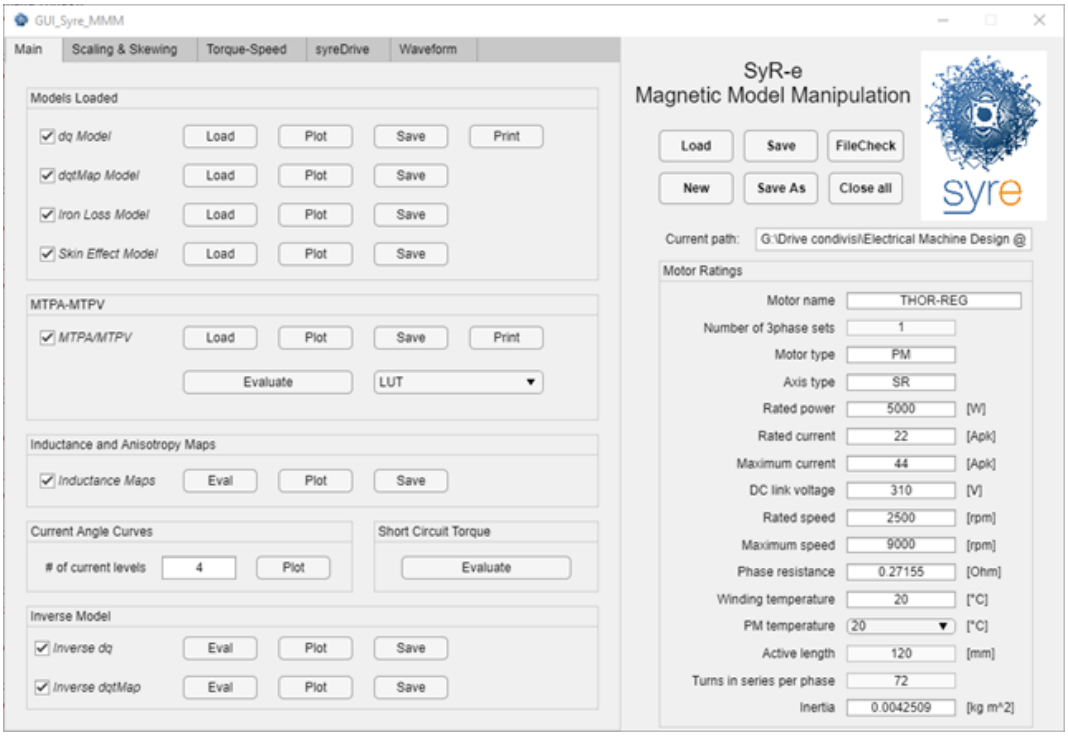
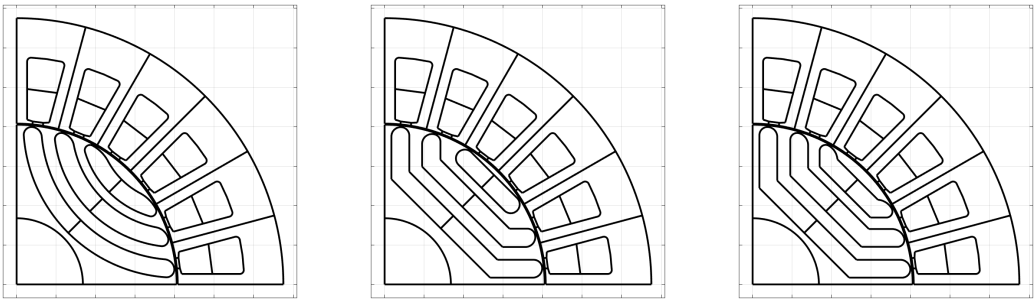


Fig. 3: Main tab of SyR-e MMM GUI

1.2.1 GUI_Syre: Machine Design and Simulation GUI

The design experience starts with the GUI_Syre interface, shown in Fig.2. The GUI allows the user to quickly modify the different parameters relative to the to the design and simulation.

Multiple rotor types are selectable, as shown in Fig.4: the fluid barriers are dedicated to SyR machines, Circular and Seg are for SyR and PM-assisted SyR machines. The Seg geometry includes the V-type Interior PM case.



(a) Circular (b) ISeg (c) Seg

(d) Fluid (e) Surface Permanent Magnet (SPM) (f) Vtype

Fig. 4: Type of rotor present in SyR-e

In Fig. it is possible to recognize six different windows:

- Main Data Tab

In this section it is possible to set the number of pole (p), the number of stator slots per

pole per phase (q), air gap length (g), the outer radius of the stator, the outer radius of the rotor, the radius of the shaft, the shaft length and the type of rotor geometry. In this window there is also a preliminary design tool that is very useful for the initial sizing of the machine. Using the command syrm-Design(x,b), the parametric design plan (x; b) is displayed with the torque and power factor contour lines as shown in Fig.5. The parameters on the axes of the design plane are:

- $x = \frac{r}{R}$ ratio of outer radius of rotor to stator
- $b = \frac{B_g}{B_{fe}}$ ratio of the peak flux density in the air gap to the peak flux density in the stator yoke.

It is easy to imagine that it is possible to obtain torque and power factor as (x,b) functions so that stator and rotor geometries are parametrized by means of few variables. The FEAfix button complements the preliminary design feature, correcting the output figures of the x,b plane via fast dedicated FEA simulations.

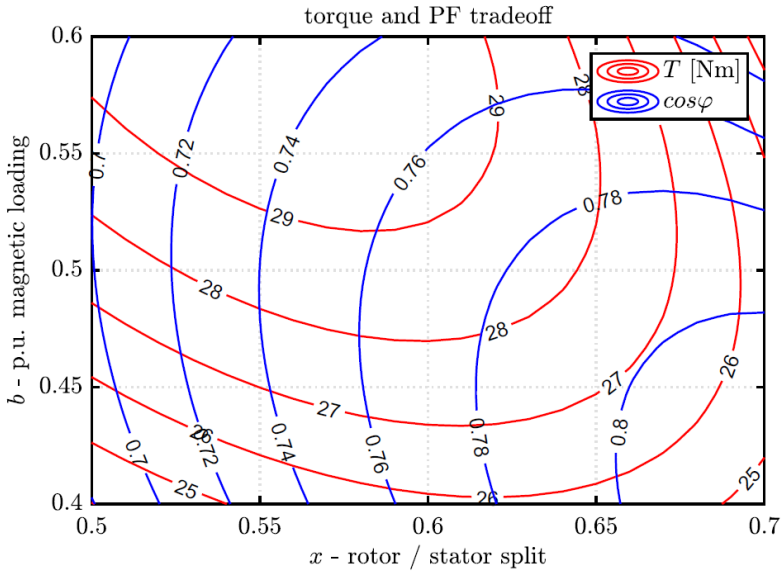


Fig. 5: SyR-e parametric design plan.

- Geometry Tab

The Geometry Tab (Fig6) is divided into two sections dedicated to the stator and rotor parameters respectively. As far as the stator is concerned, the main parameters are the length and width of the tooth, the type of slot, the slot opening et cetera. In the Rotor parameters section it is possible to define the number and geometry of the flow barriers by, the number of flow barriers, width of radial bridges,the width of flux barriers,the radial offset of the flow barriers and the angular position of the barriers points at the ari gap.

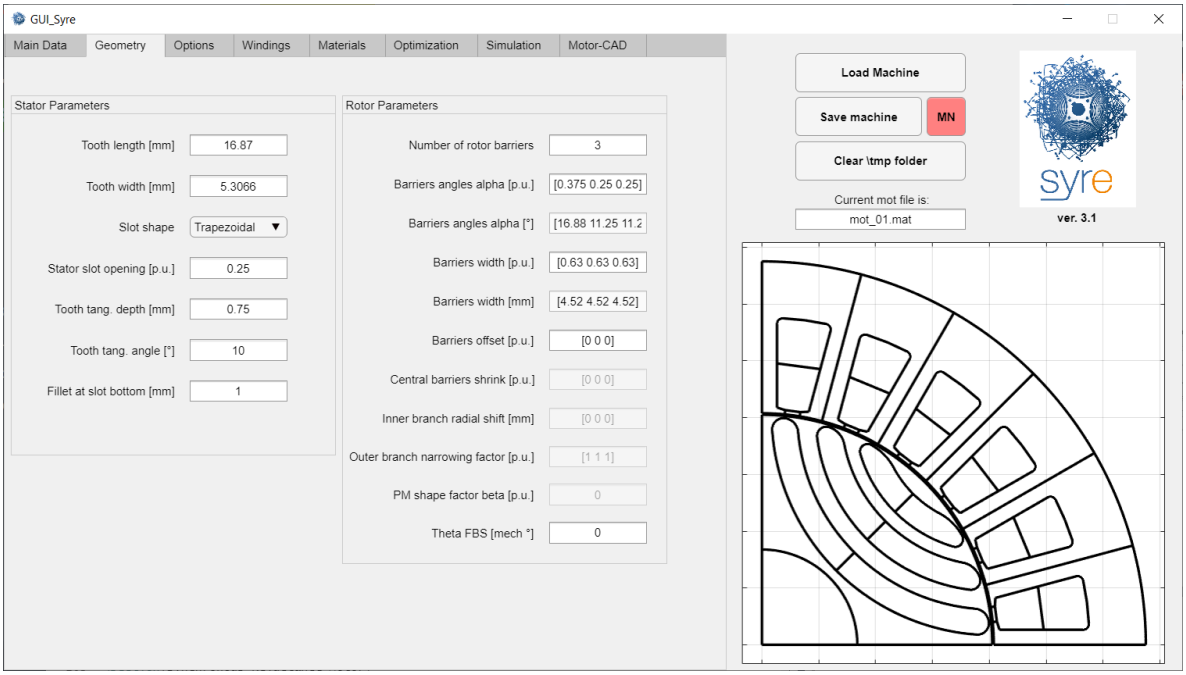


Fig. 6: Geometry tab of SyR-e GUI

- Options Tab

This window is dedicated to the thermal loading, the mesh settings, the overspeed limit and the rotor ribs settings. By defining the thermal load factor $k_j[\frac{W}{m^2}]$, loss per outer stator surface unit,the program determines the stator copper losses at standstill P_{Cu}. The evaluation of the rated current requires the estimation of the phase resistance, based on the geometric dimensions of the stator and the arrangement of the windings. The phase resistance (R_s) is reported at the temperature indicated in the target copper

temperature box. The rated current is calculate as $i_0 = \sqrt{\frac{P_{CU}}{3R}}$. A lumped-parameter slot thermal model allows to estimate the temperature of the windings, known the temperature of the housing.

The maximum rotation speed, indicated in the overspeed box, is used to size the radial ribs taking into account only the centrifugal stresses. The Minimum mech. tolerance parameter is used to set the thickness of tangential ribs.

Finally, through the Mesh and mesh MOOA parameters, it is possible to specify respectively the mesh resolution during optimization and post processing. There is also a 'split barriers' command that uses two separate ribs per barrier, in place of a single radial rib.

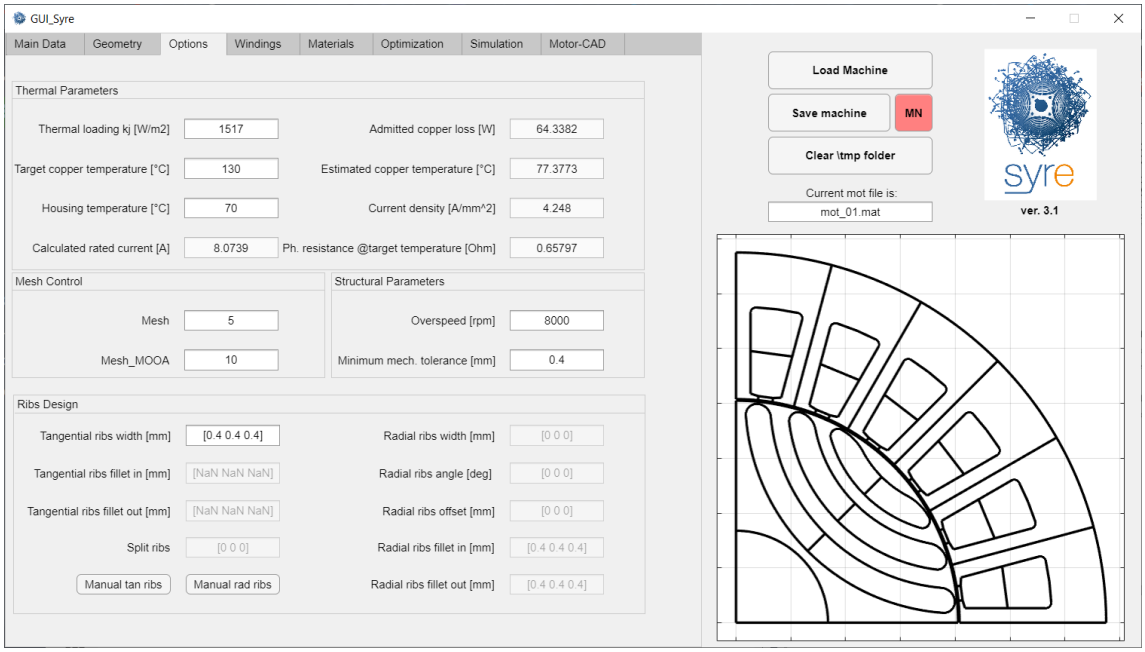


Fig. 7: Options tab of SyR-e GUI

- Windings Tab

In the Windings Tab (Fig. 8) the winding of the machine is defined via the filling factor, number of turns in series per phase, shortening factor (K_{racc}), slot division and the number of slots to be simulated. The topology of the winding is shown to the user by means of a table of size $2 \cdot n$ slots to simulate, an example is shown in Fig.44. SyR-e

supports single and double layer windings, but the table always shows two because if a single layer is used the two layers are the same.
The position of the conductors in the three phases is displayed using the numbers (-3,-2,-1,1,2,3), the sign represents the direction of the phase current in the windings.
The winding sequence is calculated automatically each time the q and Kracc parameters are changed. However, it is still possible to change the winding arrangement manually.
The Slot Model section simulates the conductors in a single slot to evaluate the skin and proximity effect as a function of the excitation frequency (AC loss factor).

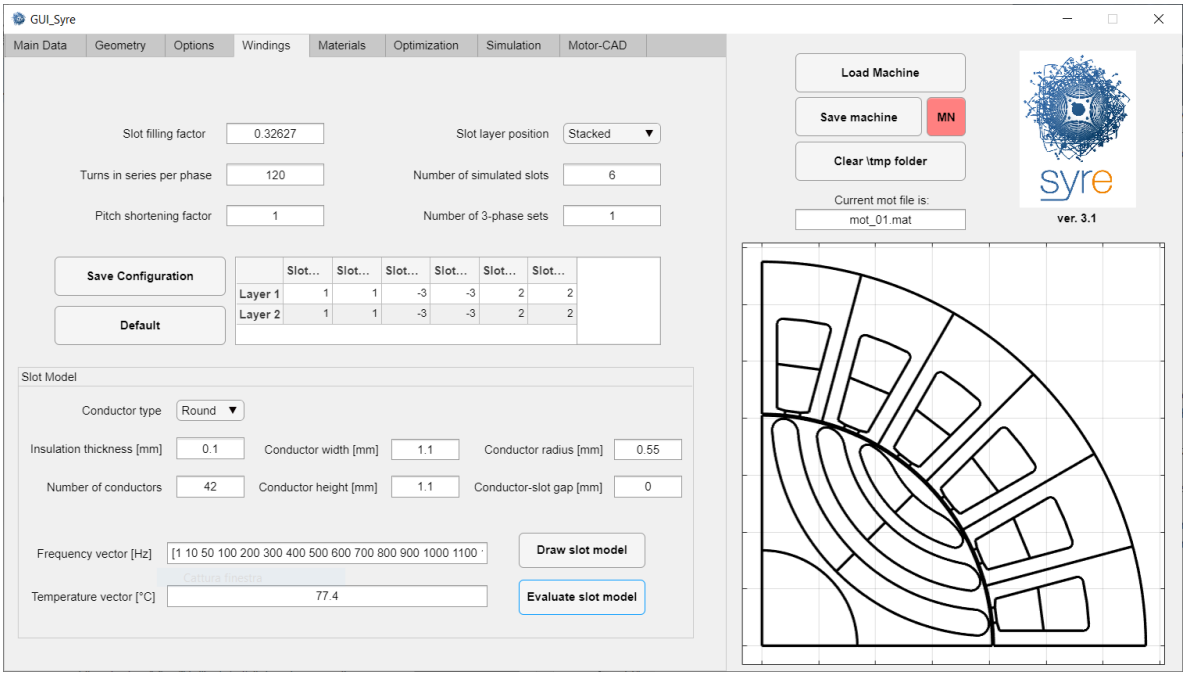


Fig. 8: Windings tab of SyR-e GUI

- Materials Tab

In the Materials Tab (Fig.9) you can set the materials of the stator core, rotor core, shaft, windings and magnets if any. There is also the possibility to add new materials. The software provides a library of pre-defined materials. New user defined materials can be added.

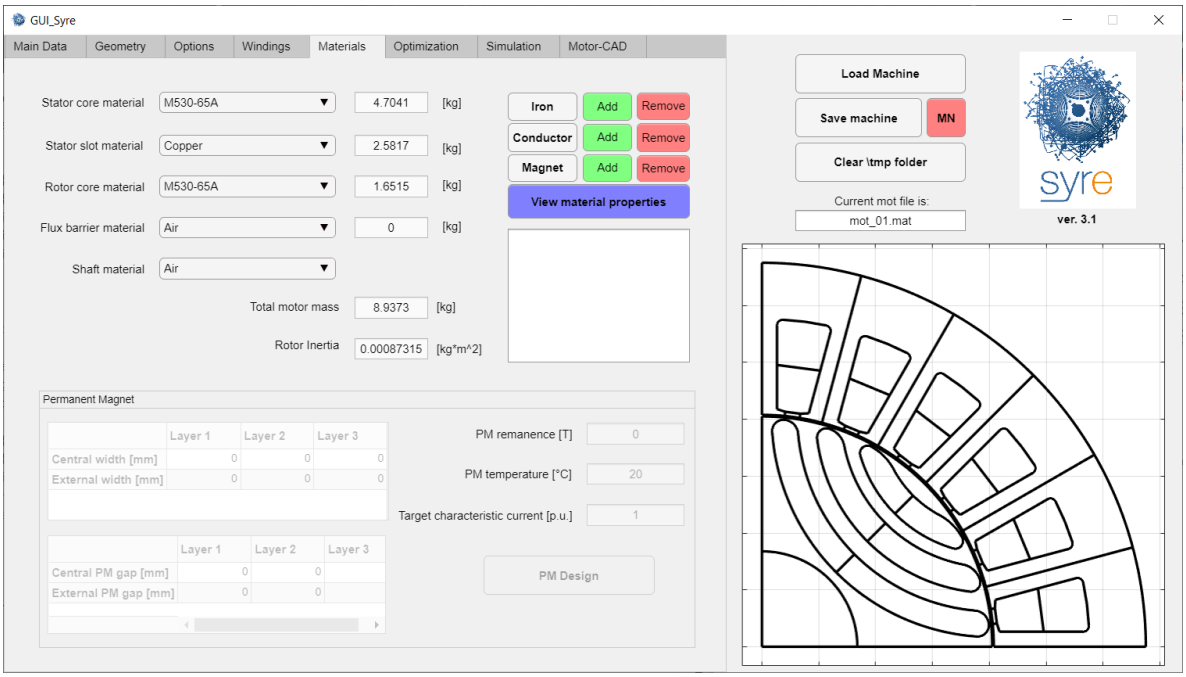


Fig. 9: Materials tab of SyR-e GUI

- Optimization Tab

The Optimization Tab (Fig.10) allows to perform the optimization of the examined machine using a multi-objective optimization algorithm, based on differential evolution. During the optimization procedure, some rotor and stator parameters are automatically modified in order to achieve the set objective.

To start the optimization it is necessary to set the parameters of the optimization algorithm, the variables to be optimized and the objectives.

The search space for the variables to be optimized must be entered in a reasonable manner, so as not to have unfeasible machines.

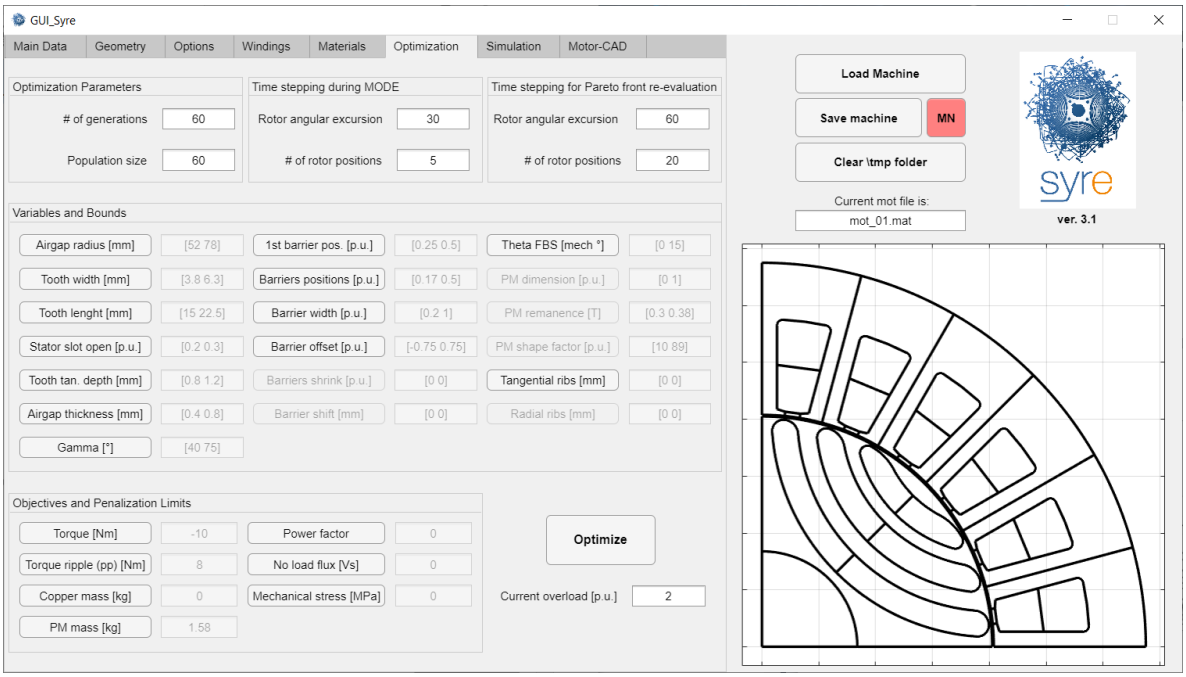


Fig. 10: Optimization tab of SyR-e GUI

It is possible to define the following optimization objectives: Maximum torque, Minimum torque ripple, which is the peak-to-peak torque ripple, Minimum copper mass, Minimum PM mass, Minimum no load flux, Target power factor within a $\pm 10\%$ tolerance of the inserted value. Fig.11 shows an example of a Pareto front obtained after an optimization process, with the aim of minimizing the torque ripple.

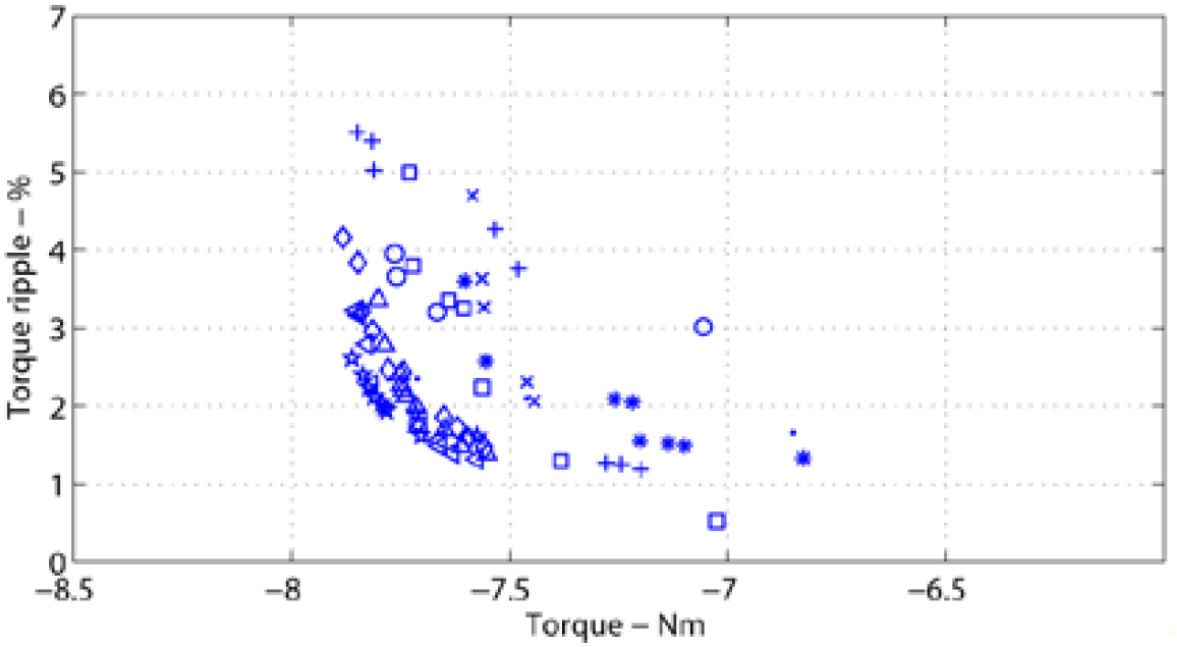


Fig. 11: Summary of the Pareto fronts of ten optimization runs, obtained using MOGA (Multi-Objective Differential Evolution). The runs were stopped at 1200 function calls. The markers indicate the different runs.[13]

- Simulation Tab

The post processing is executed using the SyR-e window (Fig.43). The main parameters to enter are the "rotor angular excursion" that is the angular span of the static time stepping in electrical degrees, the "current phase angle" in dq coordinates and the "number of rotor positions" that is the number of equally spaced rotor positions that will be simulated over the previously defined angular excursion. Moreover, the "current load" is the per unit current level used in the analysis, being the current i0 defined on the basis of the admissible joule losses at stall the base current value. Permanent magnets having residual flux density specified by "Br" can be included in the rotor barriers (Br can be extracted from the temperature characteristic too). Note that the radial ribs are not re-calculated considering permanent magnet mass during the post processing because the analysis will be executed starting from an existing .fem file.

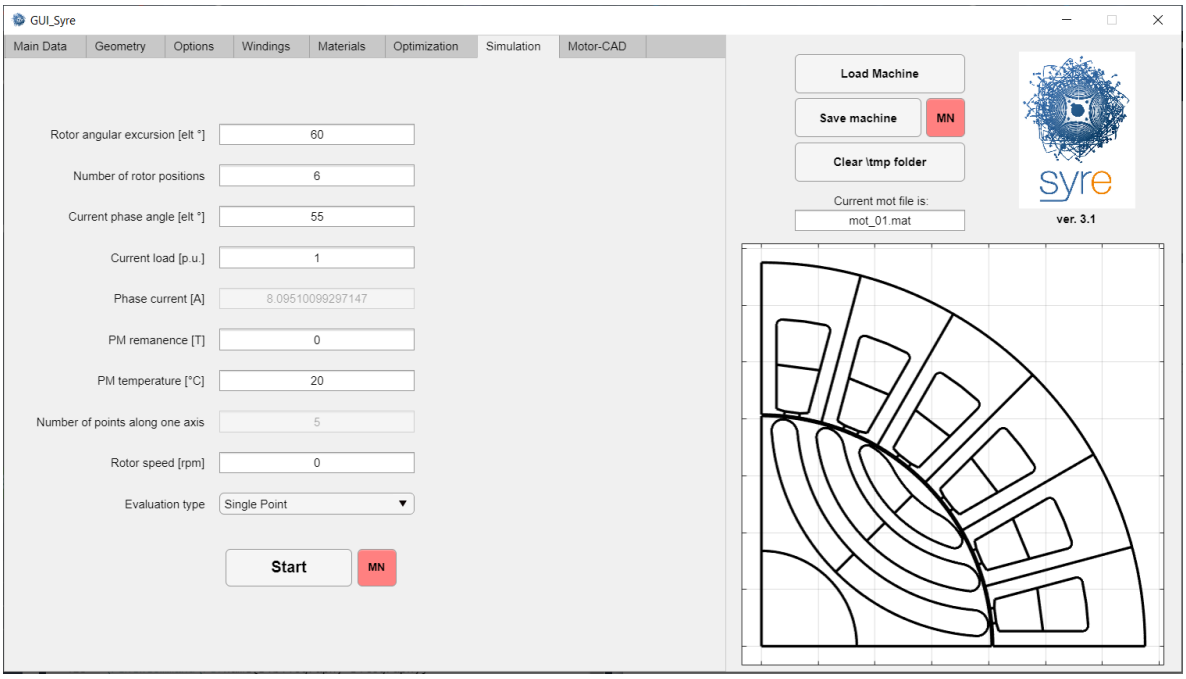


Fig. 12: Simulation tab of SyR-e GUI

There are seven types of post-processing which SyR-e can perform, selectable from the Evaluation Type menu:

- Single point evaluation/sensitivity analysis
- Flux map
- Demagnetization analysis
- Demagnetization curve
- Characteristic current computation
- Flux density analysis
- Current offset simulation
- Airgap force computation

- Structural analysis

Fig.11 shows an example of results of a single point and Flux map simulations:

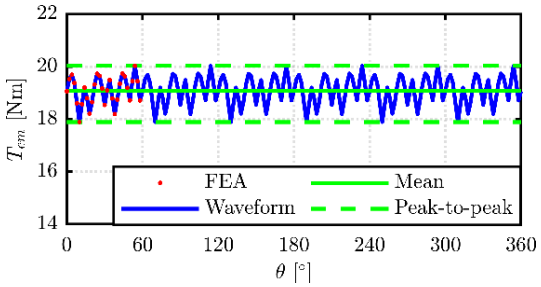


Fig. 13: Torque waveform obtained with 30 rotor positions simulated on 60 electrical degrees.[14]

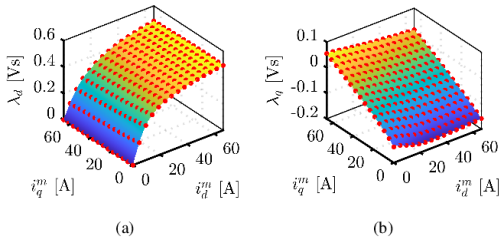


Fig. 14: Flux maps: d-axis flux linkage (a) and q-axis flux linkage (b). Red points represents the FEA simulations.[14]

- Motor-CAD Tab

A dedicated section is reserved to Motor-CAD post-processing. Here is possible to perform electromagnetic and thermal simulations by simply entering the required parameters. In Fig.15 is shown an example of geometry export from SyR-e to Motor-CAD:

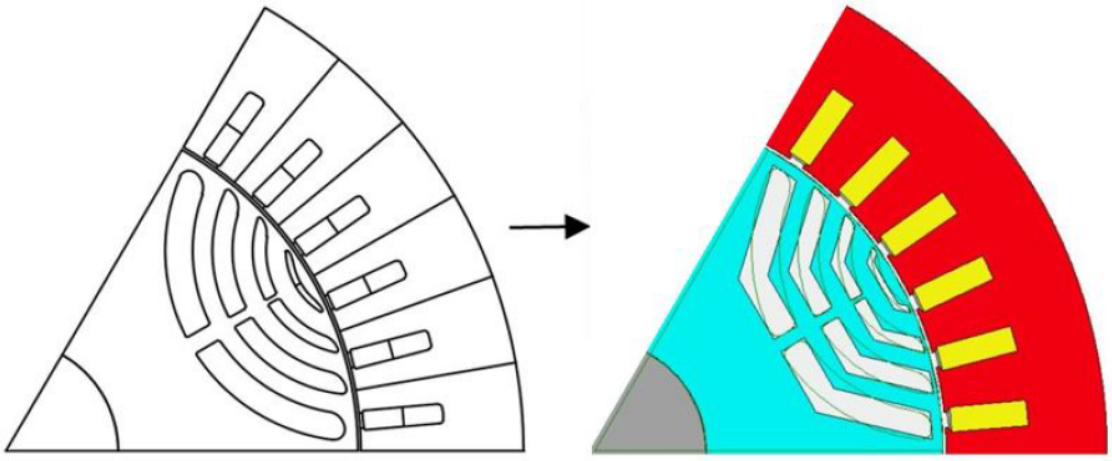


Fig. 15: Geometry model export from SyR-e to Motor-CAD [15]

1.2.2 Magnetic Model Manipulation GUI

Recently, a new graphical user interface called MMM GUI (Magnetic Model Manipulation) has been introduced and it is composed from five tabs with their respective functions:

- Main: contains the command to load single models and do a quick elaboration of the flux maps
- Scaling&Skewing: it is possible to scale the magnetic model (change stack length, number of turns and 3D inductances) and obtain the magnetic model of the skewed motor
- Torque-Speed: allows the evaluation of performance on the torque-speed plane
- syreDrive: allows the interface with Simulink model (under development)
- Waveform: at the moment, allows the creation of waveforms of single point (from dqtMap) and transient short-circuit behavior (under development)

The Main page of the GUI_Syre_MMM interface is reported in Fig.3 above and here in Fig.16 an example of a *syreDrive* simulation:

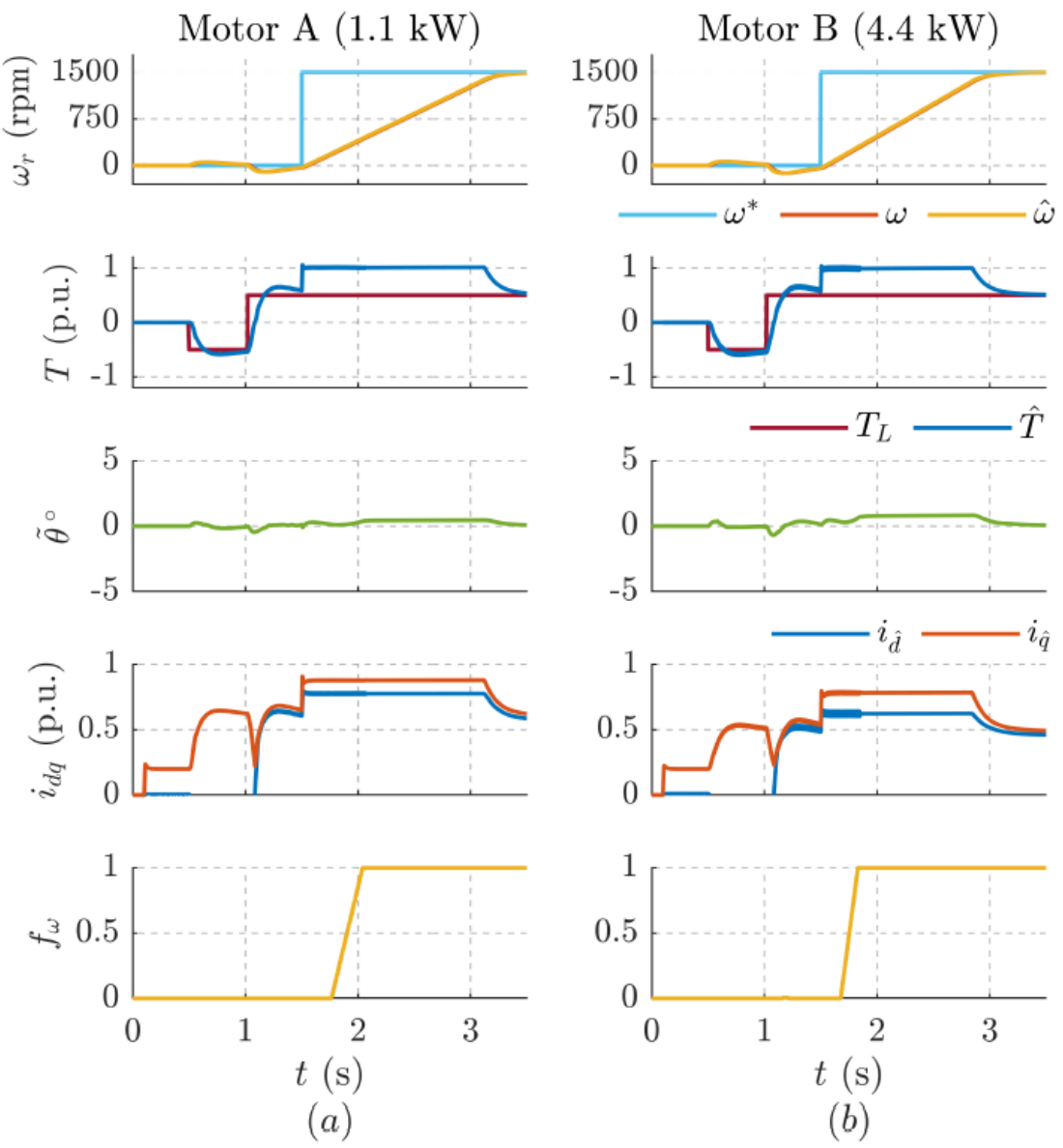


Fig. 16: Simulation result of syreDrive automated sensorless control generation using square-wave voltage injection with current-model flux demodulation at low speeds and APP at high speeds region: (a) Motor A (1.1 kW); (b) Motor B (4.4 kW). [17]

1.3 Ansys Maxwell

ANSYS, Inc. is an American Computer-aided engineering software developer headquartered south of Pittsburgh in Pennsylvania, United States. Ansys publishes engineering simulation software across a range of disciplines, structural analysis, computational fluid dynamics, explicit/implicit methods, and heat transfer.

Fig. 17: Ansys Logo

Ansys Maxwell is a high-performance software package that uses FEA to solve electromagnetic problems and can be used for electric machines, transformers, wireless charging, actuators and other electromechanical devices. It solves static, frequency-domain and time-varying magnetic and electric fields. Maxwell also offers specialized design interfaces for electric machines and power converters.
With Maxwell, is possible to precisely characterize the nonlinear, transient motion of electromechanical components and their effects on the drive circuit and control system design and offers trusted simulation of low-frequency electromagnetic fields in industrial components. It includes 3-D/2-D magnetic transient, AC electromagnetic, magnetostatic, electrostatic, DC conduction and electric transient solvers to accurately solve for field parameters including force, torque, capacitance, inductance, resistance and impedance. [3].

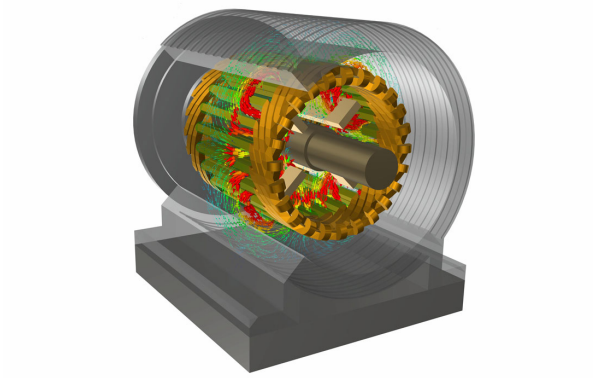


Fig. 18: Ansys Maxwell 3D geometry example [3]

The Ansys Maxwell scripting interface is compatible with Python and Visual Basic. Unfortunately, there is no direct interface to and from Matlab. Python will be used instead to embed the use of Ansys Maxwell into SyR-e.

1.4 Python as Scripting Interface

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Fig. 19: Python Logo

Python played a key role in this project because it allowed communication between Ansys Maxwell and Matlab. This was possible because Maxwell implements a version of IronPython that allows the software FEA to be controlled. IronPython is simply an implementation of the Python programming language that runs on implementations of the .NET Framework.

2 Interior PM and Reluctance Synchronous Motors

Interior Permanent Magnet (IPM) have been wide adopted in many industries application due to their inherent ruggedness, high efficiency and relatively low manufacturing cost. From a market perspective, most vehicle manufacturers and suppliers choose permanent magnet machines for the electric traction motors (Fig.20), either as an interior permanent magnet motor (IPM) or a Permanent Magnet Assisted Synchronous Reluctance (PMASR).[16].

The main difference between an **IPM** and a **PMASR** is that the first one is basically an SPM where reluctance has been added by inserting magnets inside, so alignment torque component is dominant. While the PMASR is a reluctance motor to which magnets are added inside the barriers to increase its performance by adding the contribution of PMs, but the major part of the torque comes from the reluctance.

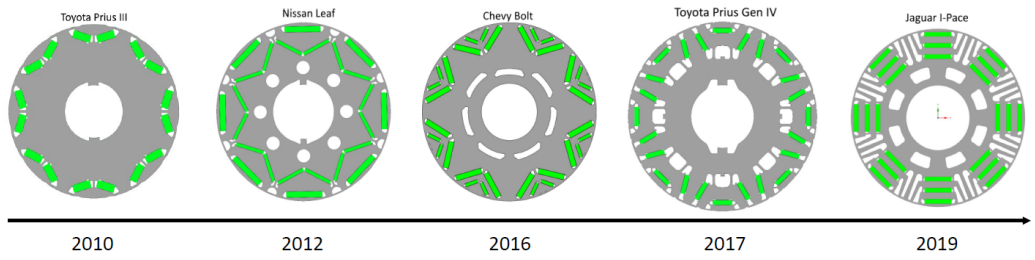


Fig. 20: Rotor design geometries of traction motors; Permanent magnets in green independent on magnetization direction

Among the various IPM topologies, **V-shape** IPMs exhibit higher power densities, higher efficiency, lower manufacturing cost and wider constant power operating range. As with SPM machines, IPMs take advantage of magnetic torque produced by both **PM**s and **reluctance** torque due to unequal reluctance between the d and q axes, similar to PMASR engines.

Unlike SPMs, however, designing an IPM has proven challenging due to rotor structure complexity and magnetic saturation.

One of IPM's important advantages over SPM is the simplicity of precisely manufacturing the IPM rotor lamination's outer shape as compared with the magnet's outer

shape in SPMs. This advantage also creates additional desirable characteristics in IPM machines[18]:

- IPMs use simple rectangular-shape magnets with parallel magnetization which reduces both magnet price and manufacturing cost;
- Magnets are mechanically captured within the rotor lamination, making them suitable for high speed operation without protective rings or retaining sleeves on the rotor;
- Presence of a flux bridge in provides better protection against demagnetization, offering a higher overloading capability.
- Sinusoidal air gap flux density distribution minimizes cogging torque, providing superior low speed velocity regulation.

2.1 Overview and Working Principle

The stator has a distributed three-phase winding similar to induction motor, in fact in Fig.21 it is possible to recognize the slots that contain the conductors. IPM machines are easily recognizable by the rotor geometry which has the magnets inside.
A machine with a V-shape rotor design has the poles formed by a couple of symmetrical magnets placed to form a V. An example always in Fig. 21, where a 6-pole V-shape structure can be seen. This structure is often used because of its inclination to optimization, as it is very easy to adjust the machine performance by varying some parameter of the barrier geometry.
Moreover it is particularly used in the **automotive** field thanks to the wide range at constant power that can provide due to the good possibility of defluxing. This is possible because, compared to the SPM that do not allow the flux to move tangentially inside the magnets, the IPM allow it. The configuration of V-shape barriers allows to obtain high values of L_q (minimum reluctance axes) so as not to have problems in keeping L_d (direction of magnetization of PMs) too low,important from a structural mechanics perspective.

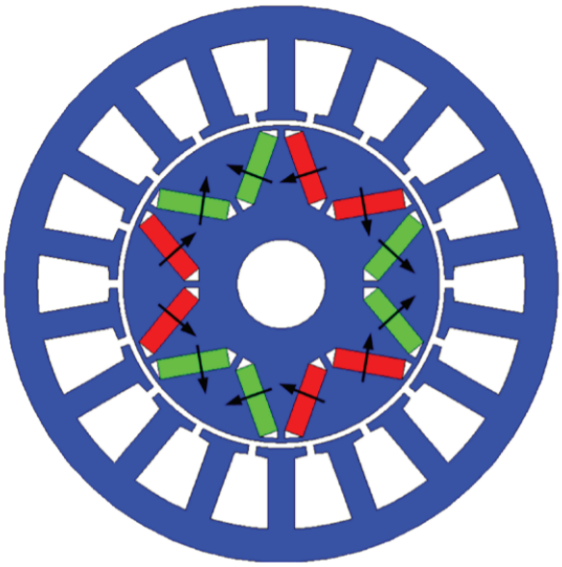


Fig. 21: Example of a IPM motor with a V-shape rotor design

In general, IPM motors operate due to a dual nature, one is the torque generated by the permanent magnets as the SPM motors from which they are from, and the other an additional effect due to the insertion of the internal magnets which is the reluctance torque.

The first torque term therefore is due to the interaction of the rotating magnetic field created at the stator with that generated by the magnets. The production due to reluctance arises precisely from the presence in the rotor of different reluctance magnetic paths that provides preferential paths to the stator-generated flux. The operation of this machine also involves the magnetic saturation of the laminations especially in the areas of the bridges.

2.2 Dynamic dq Model

The study of three-phase machines, particularly synchronous ones, is realized on a rotating cartesian system dq anchored to the machine rotor. This representation is obtained through the Clark and Park transforms from the three-phase model.

There are two main conventions that can be used in orienting these axes:

- First is PMs torque component dominant group, which includes IPM and SPM machines where PM magnetization direction is aligned with positive d axis (Fig. 22a-d).

- For the predominant reluctance component machine the d-axis is the one in which the flux lines are not obstructed as shown in Fig. 22e-h. The q-axis, on the other hand, is oriented in the direction in which the machine exhibits maximum reluctance.

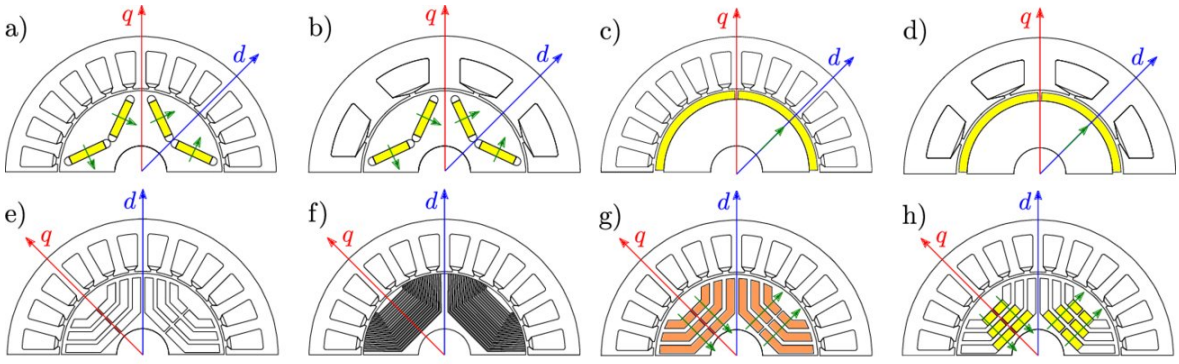


Fig. 22: Example of four pole machine families with corresponding dq axes: IPM with a) distributed and b) concentrated winding stator; SPM with c) distributed and d) concentrated winding stator; e) SyRM; f) AxLam SyRM; g) "weak" PM material PMASR; and h) "strong" PM material PMASR. Green arrows indicate PM magnetization direction.[8]

IPM machines are more relevant in this work because they are the object of study and therefore the first convention is used.
In the dq-axes representation the torque expression can be derived from a simple energy

balance for a SyR motor as:

$$\begin{aligned} P_{elt} &= \frac{3}{2}(v_d \cdot i_d + v_q \cdot i_q) + \cancelto{0}{v_r \cdot i_r} = \\ &= \frac{3}{2}[(R_s \cdot i_d^2 + R_s \cdot i_q^2) + \rightarrow Stator\ Joule\ Losses \\ &\quad +(i_d \cdot \dot{\lambda}_d + i_q \cdot \dot{\lambda}_q) + \rightarrow Magnetic\ Power \\ &\quad (p\omega_r \lambda_d \cdot i_q - p\omega_r \lambda_q \cdot i_d)] \rightarrow Mechanical\ Power \end{aligned} \tag{1}$$

Where:

$v_d, v_q, i_d, i_q, \lambda_d, \lambda_q$: Stator voltages, currents and fluxes dq projections

R_s :Stator Winding Resistance

ω_r Rotor Angular Velocity

p : pole pair

The general expression of the dynamic torque generated by a SyR machine is then:

$$T = \frac{3}{2}p(\lambda_d \cdot i_q - \lambda_q \cdot i_d) \tag{2}$$

Knowing that $\lambda_d = L_d(i_d) \cdot i_d + \lambda_{md}$ and if the magnet are present and aligned with the q-axis $\lambda_d = Lq \cdot i_q$:

$$T = \frac{3}{2}p[\lambda_{md} \cdot i_q + (Ld - Lq) i_d i_q] \tag{3}$$

L_d and L_q are the direct axis synchronous inductance and quadrature synchronous inductance and remembering that the machine is magnetically asymmetric it follows that $L_d < L_q$.

Remarkably, the torque is composed of two components one due to the presence of PMs T_M alignment torque) and one because of the reluctance (T_R). Considering varying the current angle γ (constant module) on the dq plane, it can be said that T_M is proportional to $sin(\gamma)$ while T_R is proportional to $sin(2\gamma)$. Then the resulting torque T waveform will be (Fig.23):

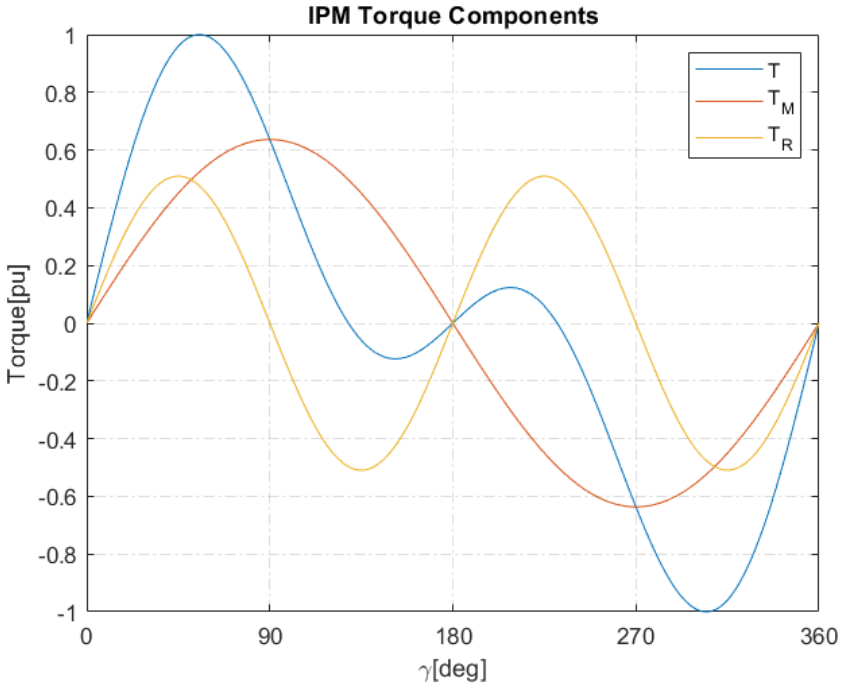


Fig. 23: Torque components waveform of a IPM motor for constant module current on 360° angle on dq : Blue- total torque Red- alignment torque Yellow - reluctance torque

Therefore it can be said that if it is intended to use a range that includes the maximum torque, it is necessary that the control of current work in the first quadrant (dq-axes) for positive torques and in the fourth for negative torques.

2.3 Permanent Magnets

In the study of a permanent magnet machine, the magnet can be replaced by an equivalent electrical circuit, which simulates its behavior under both stationary and dynamic conditions.

Equivalence can be deduced by starting magnetic circuits such as the one in Fig. 24. The circuit is equipped with an air gap of section St and length lt , by a section of magnetic material of high permeability and by a permanent magnet of length lm and section Sm , equipped with the magnetic characteristic shown on the side of the structure. If we linearize the characteristic of the magnet around the possible point of work,

the relation between induction and field inside the magnet has the following expression (Eq.4) [11]:

$$B_m = B_0 + \mu_1 \cdot H_m \tag{4}$$

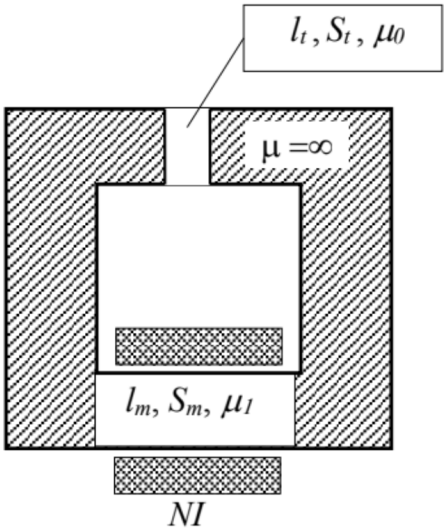


Fig. 24: Diagram of an electromagnet excited by a PM [11]

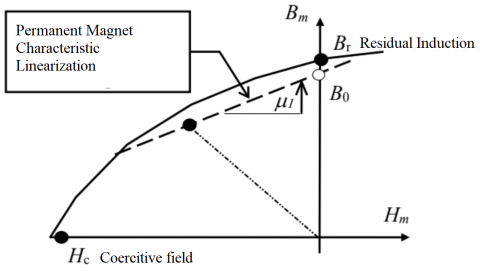


Fig. 25: Characteristic of the magnet (linearized around the working point) [11]

The magnetic field on the air gap H_t can be obtained from the Flux law $B_m \cdot S_m = B_t \cdot S_t$ and Ampere's law $H_m \cdot l_m + H_t \cdot l_t = 0$. Working out the two expressions by substituting the Eq.4 and the relation $B_t = \mu_0 \cdot H_t$ is obtained:

$$H_t = \frac{B_0}{\mu_0} \cdot \frac{l'_m}{l_t + \frac{S_t}{S_m} l'_m} \; where \; l'_m = l_m \cdot \frac{\mu_0}{\mu_1} \tag{5}$$

Considering replacing the magnet with a coil for which the following relation $NI = \frac{B_0}{\mu_0} \cdot l'_m$ is valid, the same air gap behavior is obtained. With this, the equivalence between a magnet and a coil through which a current flows is thus demonstrated.

It is interesting seeing how the demagnetization curve of a magnet is affected by temperature. In this regard an example is shown in Fig.26:

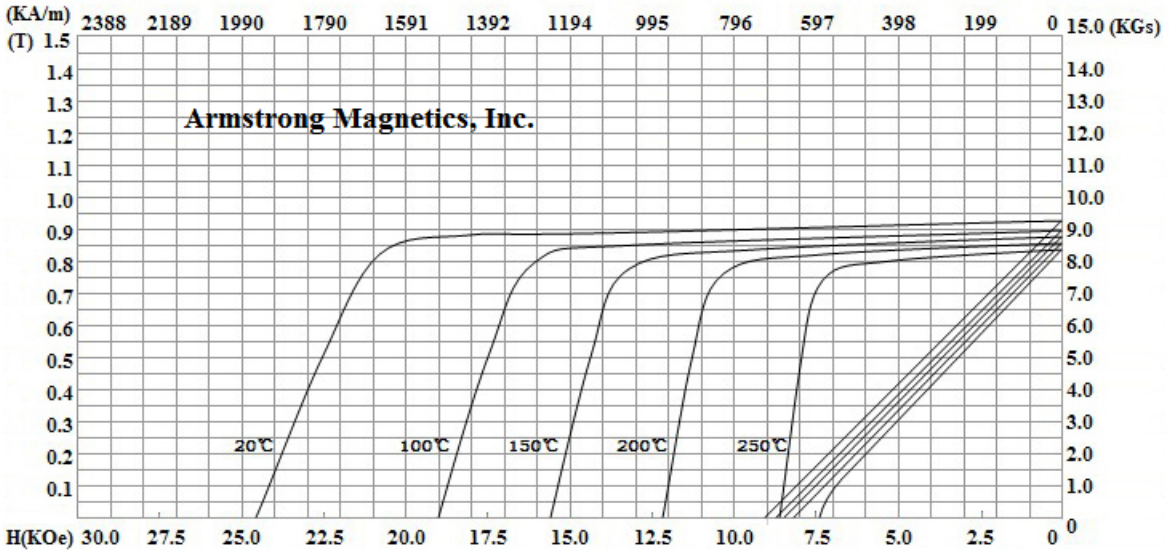


Fig. 26: Demagnetization curve for sintered magnet SmCo5 grade 20s at different temperature

It is very important to consider the effect of temperature because as can be seen in the example this shifts considerably the demagnetization value of the magnet, at the knee of the curve. Therefore when the magnet is inserted in a magnet circuit, as an electric machine, it is always necessary to verify the working point for the maximum expected temperature.

2.4 Iron Losses

As is well known, electrical machines are usually composed of ferromagnetic plates. During the normal operation of these devices two main phenomena cause losses inside the plates [12]:

- Hysteresis Losses
- Eddy Current

Hysteresis Losses

Recalling that the magnetic energy on unit volume can be calculated as:

$$\frac{dW_m}{V_{fe}} = HdB \tag{6}$$

Consider a typical B-H saturation curve of a ferromagnetic material placed in a hysteresis loop:

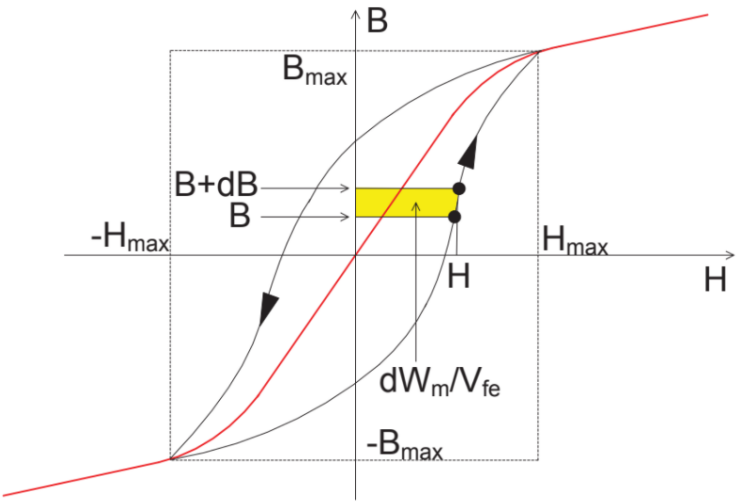


Fig. 27: Hysteresis loop of a ferromagnetic material

The yellow area in Fig.27 represents the change in magnetic energy caused by a dB variation in induction. This change will be positive when dB is positive and vice versa. Assuming the system is alternating and the induction goes from $+B_{max}$ to $-B_{max}$ the working point will follow the edge of the hysteresis at each cycle. It can be inferred graphically that the magnetic energy per unit volume that occurs on the way up is less than on the way down, and the difference coincides with the area of hysteresis

It can be shown experimentally that the cycle area is proportional to the maximum induction ($A_{cycle} \propto B_{max}^{\gamma}$). Usually γ takes values between 1.6 and 2. After saying that and introducing a constant of proportionality (K_{ist}) it can be said that the losses are:

$$p_{fe,hys} = k_{hys} B_{max}^{\gamma} f \tag{7}$$

Eddy Current Losses

These losses are due to the currents in the laminations which tend to oppose variations

in the flow through the laminated core according to the Lenz principle. The most widely used practical formula for quantifying specific eddy current losses is the following:

$$p_{fe,eddy} = k_{eddy}\delta^2 B_{max}^2 f^2 \tag{8}$$

Where δ is the thickness of the plates.

Steinmetz Equation

The Steinmez formula for total losses in iron can be obtained by assembling the Eq.7-8 :

$$p_{fe} = k_{hys}B_{max}^{\gamma} f + k_{eddy}\delta^2 B_{max}^2 f^2 \tag{9}$$

An additional term corresponding to additional losses is often considered in the formula. Most additional losses are due to the action of time-varying magnetic fluxes on conductive metal parts, such as structural parts of machines, mechanical shafts, and so on.

The additional term can be merged into one of the two main terms by changing the exponents of the formula, or the following equation is often used:

$$p_{fe,add} = k_{add} \cdot (B_{max} f)^{1.5} \tag{10}$$

3 Export Machine Models from SyR-e to Ansys Maxwell

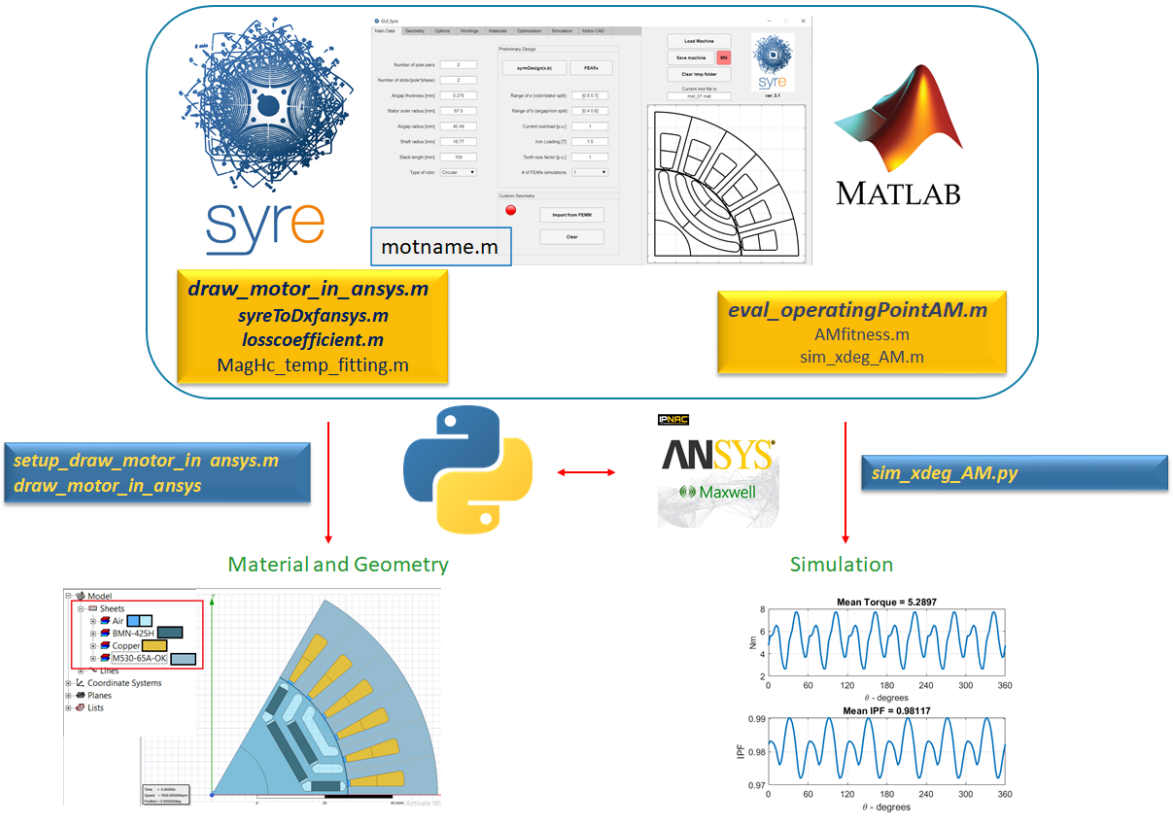


Fig. 28: Export and Simulation process diagram

In order to build a joint design procedure with **SyR-e** and **Ansys Maxwell**, it was employed the **IronPython** language implemented in the Ansys software. It was possible to connect the two software by taking advantage of the possibility to automate Ansys through the IronPython language implemented in the software itself.

The interfacing consists in a first part in which the machine geometry and materials are imported from SyR-e to Maxwell. Subsequently, the Magnetic simulation is executed after assigning its parameters and then the results are imported in Matlab.

The most challenging part was the first one, due to the significantly different nature of reasoning on the machine drawing between **Maxwell** and **SyR-e**.

3.1 Geometry and Materials Export

3.1.1 Update Material Library

The export of the motor includes both geometry and material properties and it is performed by means of two main codes, both called "draw_motor_in_ansys", developed on Matlab and Python.
The Matlab-based code creates a Python code called "setup_draw_motor_in_ansys.py" using the fprintf function. This step is necessary to import the **BH** curves (fig.30) that require a command for each point of the curve. The best way to export the curve would be to use a vector or to repeat with a for loop the command that updates the library, but these two ways are incompatible with the Python commands provided by Maxwell, then this part of the code has been created using the fprintf command directly from Matlab. Thanks to this expedient the other data of the machine have been passed.
The code *"draw_motor_in_ansys.mat"* is therefore composed of:

- A first part where the variables are collected from the machine's **".mat"** file.
- The Python file creation and writing
- Launch Ansys and the **".py"** codes with dos commands

	Name	Location	Origin	Relative Permeability	Bulk Conductivity	
	Air	Project		1	0	0
	BMN-42SH	Project		1.05	666666.666667siemens/m	-1007981.306
	Copper	Project		1	58000000siemens/m	0
	M530-65A-OK	Project		B-H Curve...	0	0A_per_mete
	vacuum	Project	Materials	1	0	0

Fig. 29: Example of imported materials on the Ansys library from a simulated motor

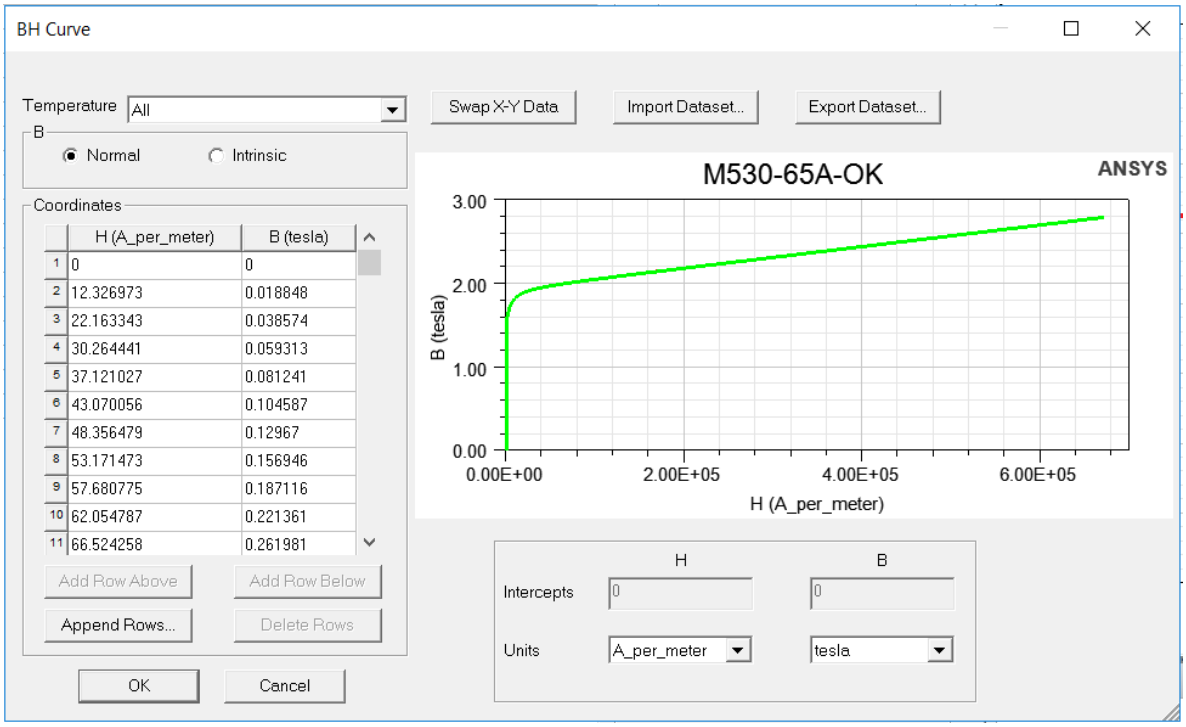


Fig. 30: Example of the imported BH curve of the core material M530-65A

Two functions were created to calculate the material properties of the machine as listed below:

- *"syreToDxfansys.m"*: Function already provided by SyR-e able to create a .dxf file containing the geometry of the selected machine.
- *"losscoefficients.m"*: calculates the loss coefficients of the model used in Ansys from those used in SyR-e through curve fitting, models discussed in Chap.3.2.1
- *"MagHc_temp_fitting.m"*: obtains the coefficient of temperature dependence of the coercive magnetic field of permanent magnets (PMs) (Chap.3.2.2)

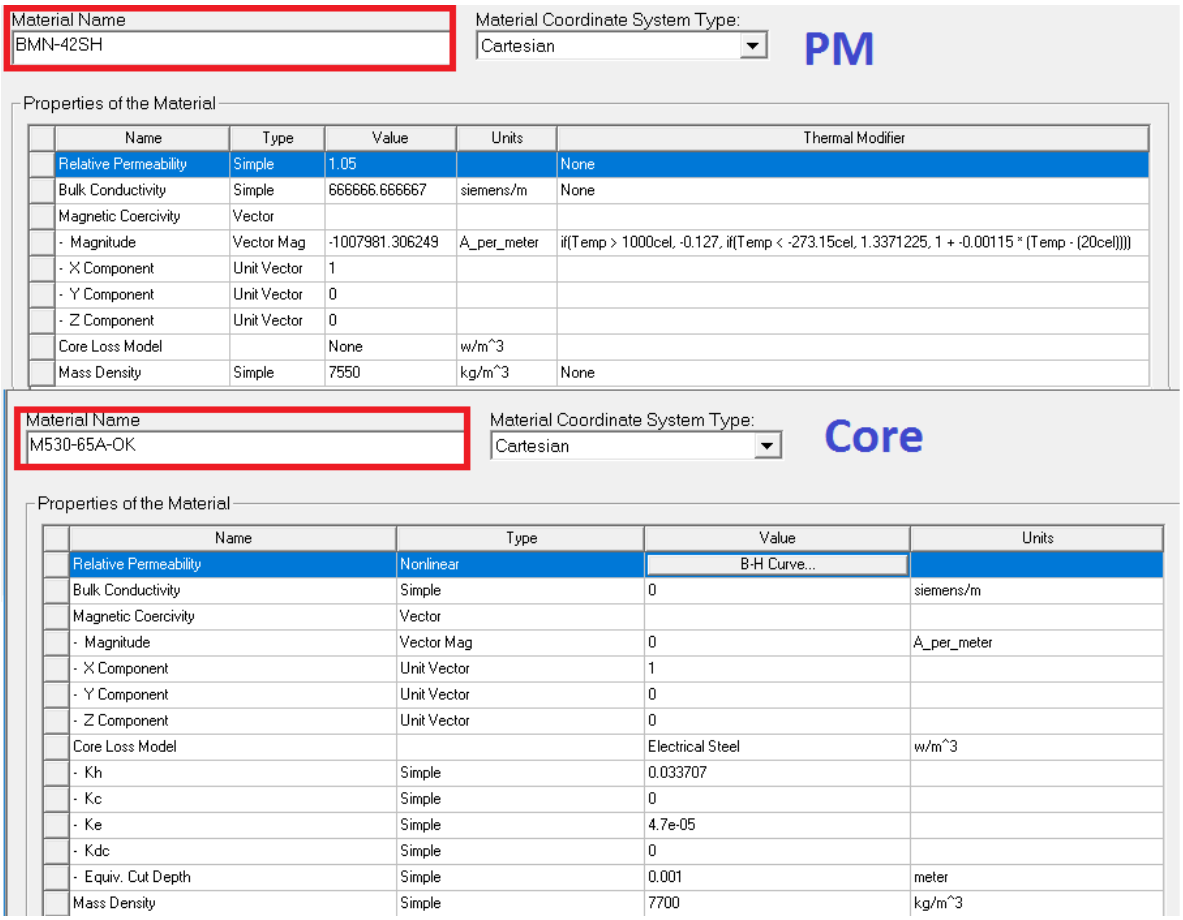


Fig. 31: Example of imported properties for the iron material M530-65A and for the PM material BMN-42SH

3.1.2 Geometry Export and Construction

Once the material library is updated, the machine geometry is exported to Maxwell by means of the code *"draw_motor_in_ansys.py"* .

Such a task is accomplished by exporting the SyR-e structures *geo* and *mat*,that contain the machines data, via fprintf by the function *"setup_draw_motor_in_ansys.py"* . Once this information is stored into **Python structures**, shown in Fig. 32, these are passed from the first python code to the second one with the help of a temporary file called ***pickle***. This action has been necessary because is not possible to install eventual libraries in the Iron Python inserted in Ansys and there is a restricted choice of usable commands.

```
geodata={
"R":67.500000, #stator radius
"r":40.490000, #rotor radius
"p" :3,   #pole pair
"g" :0.275000,   #air gap thickness
"nlay" :3,   #layer of rotor barrier
"PM_CS":[[25.3815,20.0857,19.3235,28.7565,24.9907,34.0505,],[18.5511,17.2603,28.1617,12.7055,8.7646,2.6537,],\
[0.86603,0.86603,0.86603,0.86603,0.86603,2.8328e-16,],[0.5,0.5,-0.5,0.5,0.5,1,],], #Coordinate system magnet: 0-x 1-y 2-xvect 3-yvect
"q":2,  #slot per cave per phase
"radial_ribs_split":2, #area added by split barrier
"n_PM":6, #number of permanent magnet (number of magnetic segments)
"l":101.000000, #stack length
"filepath":"C:/Users/tesi/Google Drive/2020 - Tesi LM - Riccardo Aimo/prove/mot_01seg/", #motor"s folder
"filename":"segsplittest.mat", #motor"s file mat name
}
#struct material names
material={
"rotor":"M530-65A-OK",
"stator":"M530-65A-OK",
"shaft":"M530-65A-OK",
"slotcond":"Copper",
"magnet":"BMN-42SH",
}
```

Fig. 32: Struct containig the motor data useful for the construction in Ansys

Once the needed data are collected the geometry is exported using the ***dxf*** file and invoking a import command. In order to obtain the machine drawing inside the FEA software the information of the dxf file path is stored.

Thus, the geometry is imported in Maxwell as lines and faces, subsequently the next step is to define the areas and assign materials to each of them. Such a task is executed by a dedicated part of the script (*"draw_motor_in_ansys.py"*). Maxwell numerates each area and line of the imported sketch from SyR-e; and it does so always with the same sequence and logic, then it was possible to automatically recognize the material area

by looking at the numbers assigned by Maxwell.
The numbering logic used in rows (*raw 74-89 and 178-199 of "draw_motor_in_ansys.py"*) is summarized below:

- Starting from 1_1it corresponds to the stator area
- From 1_2 there are alternatively a slot area and 3 lines of construction: two of them limit the air gap area under the slot and the third separates the two slot layers (ex.the second slot area is 1_6) .
- After the last slot, there are three lines related to the same slot and then the lines that separate the angular sectors of the stator (slot number plus one).
- From here begins the numbering of the flux semi-barriers and magnets, indexed starting from two. Thus, if the last slot was for example 1_22 the first semi-barrier would be 2_33, in a case with 6 slots (first barrier: 2_N where $N = 5 \cdot nslot + 3$).
- After the semi-barrier is detected, the upper ones are consecutively numerated. Last, the magnets inside the barriers are accordingly numerated.
- If the split barrier command is selected, it adds a certain number of area, belonging to the flow barriers, equal to twice the number of levels in which the split is applied.
- The last areas are all the magnets, the rotor plate and the shaft.

Having now identified all the parts of the machine, it's possible to work on the geometry that is not yet ready because some areas include parts that do not belong to them. Using mainly the commands *Imprint*, *Subtract* and *Detach* present in Ansys is possible to perform the actions shown in Fig. 33.

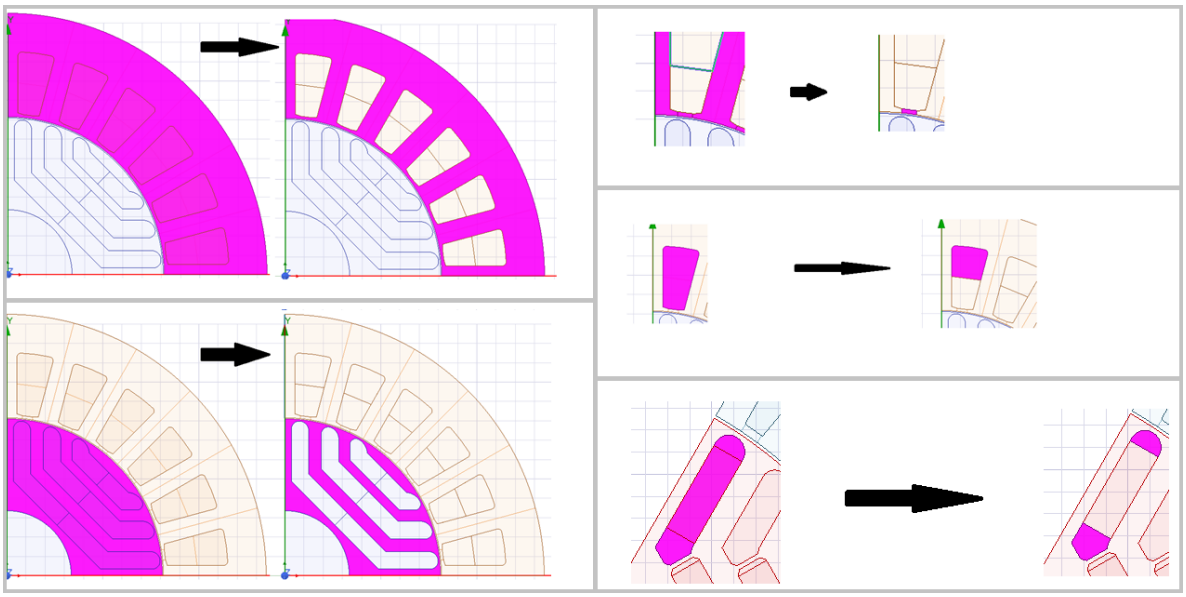


Fig. 33: Example of results of the dxf's post-processing on machines

Ansys assigns to the parts that are separated the name of the part from which they were attached plus the word *"Detach"*, in this case they are about the air gaps under the slot and the upper slot layer.

3.1.3 Material Assignment

Once the various parts that make up the machine have been identified, the next step is to assign to them the various materials, previously added to the library. This procedure is carried out by *draw_motor_in_ansys.py*

The command to assign the material also allows to select a color that has been chosen similar to the material itself for better understanding.

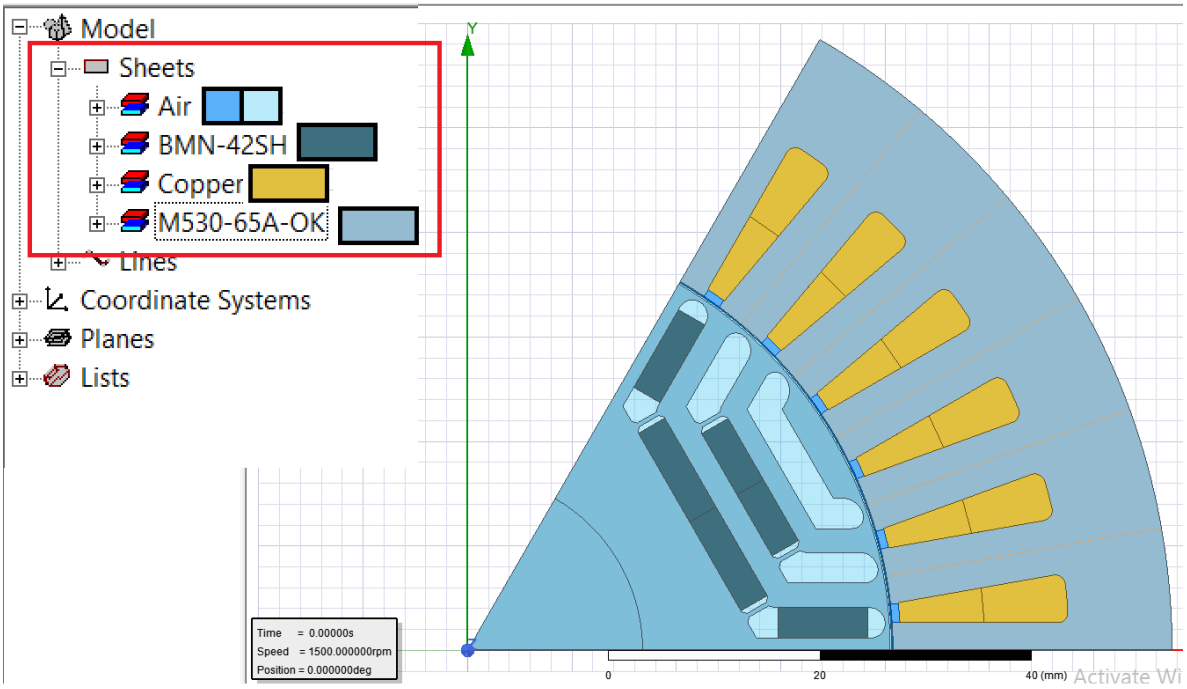


Fig. 34: Assigned material on machine *splittest.mat*

3.1.4 Geometry Setup Finalization

From the previous operations is understandable that some steps are missing to conclude the geometric export of the motor. First of all the air inside the air gap between rotor and stator must be defined and to do this it is built a circular sector that include all the motor area and then be sectioned with respect to the polar sector, this area has been called *"Region"*.

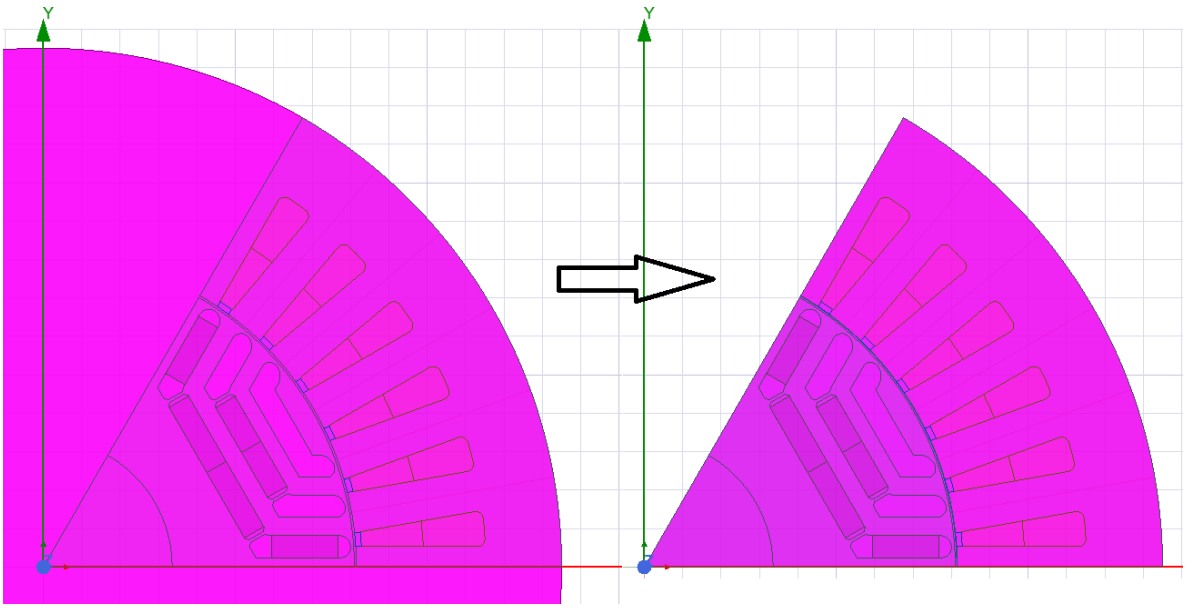


Fig. 35: Circular air sector creation that includes the motor (*splittest.mat*)

Similarly, two concentric sectors of smaller radius have been created, one in the middle of the **air gap** and the other always included in the gap but more external, useful for rotor handling settings. Having these areas of air that include the rotor, it was decided to do not assign the material "air" to the flow barriers, thus leaving the corresponding areas empty. This was done because in addition to being redundant redefining the material, it would be a further step that would reduce the robustness of the process. The two smaller regions are called: *Rotating_band_mid* and *Rotating_band_out*.

Finally, it is necessary to define within the design, the **model depth** and the periodicity of the problem (**symmetry multiplier**) that coincides with the number of poles. An illustration is shown in Fig.36 for clarity:

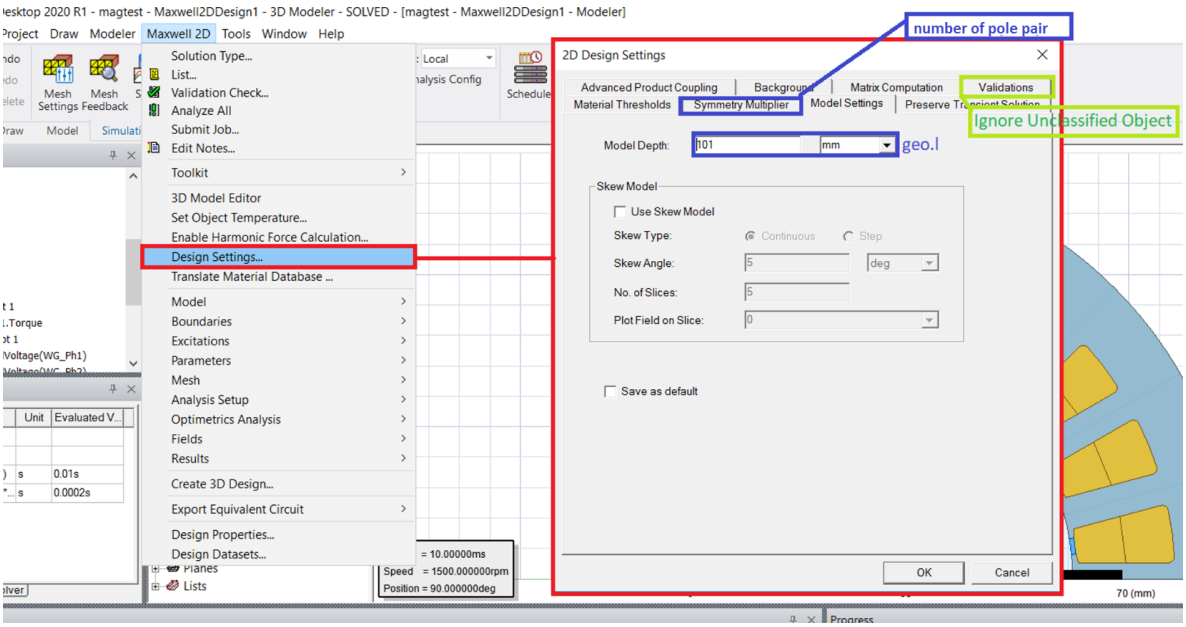


Fig. 36: Design settings assignation

3.2 Permanent Magnets and Core Properties Export

In a permanent magnet assisted synchronous reluctance machine (**PMASR**) the permanent magnets and the core are fundamentals to operation and it is therefore important to accurately characterize them. For this purpose, the two functions ***losscoefficients.m*** and ***MagHc_temp_fitting.m*** were used to calculate the loss coefficients and the linear temperature dependence coefficient of the coercive field.

3.2.1 Calculation of core material loss coefficients

SyR-e already has the information about the loss coefficients within the material library, but these are related to a different model than the one used in Ansys, so a process of fitting the loss curves was necessary. SyR-e uses the loss model in Eq.11:

$$Loss_{syre} = K_h \cdot f^{\alpha} \cdot B^{\beta} + K_e \cdot (f \cdot B)^2 \tag{11}$$

The second term is attributable to eddy currents while the first term should be separated into two terms:

- Hysteresis losses
- Additional/anomalous losses

Regarding Ansys, the losses are estimated according to the model below, which is discussed in detail in Chapter 2.3:

$$Loss_{ansys} = K_h \cdot f \cdot B^2 + K_e \cdot (f \cdot B)^{1.5} + K_c \cdot (f \cdot B)^2 \tag{12}$$

Attention must be paid in this case to the nomenclature because the subscript "e" is used to represent the additional losses, while "c" indicates the eddy currents. On the other hand, the first term always indicates the hysteresis losses but without including the additional ones.
Ansys allows to insert also a fourth term, as can be seen in Fig. 12,indeed the core loss

computation is based on the traditional three core loss coefficients **Kh**, **Kc** and **Ke**, plus the optional **Kdc**. In order to properly consider the impact of DC-biased hysteresis loss, a general rule to automatically compute Kdc is applied to improve the core loss from the classic Steinmetz equation. If the user leaves the default value of Kdc as zero, this indicates that Kdc will be determined automatically by the software; otherwise, the software will use the user-input Kdc value to consider the impact of DC-bias.
Since this loss term is not present in SyR-e it was not considered and it was left at its default value.

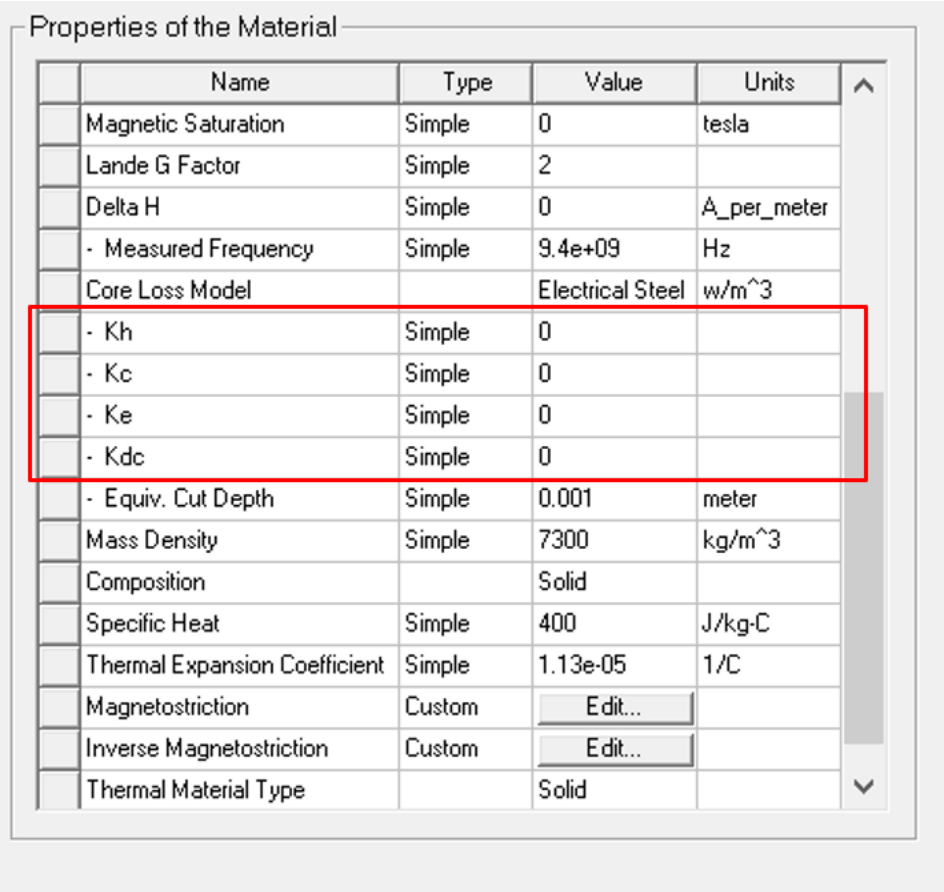


Fig. 37: Loss coefficients fields on Ansys materials library of the core material M530-65A

Comparing the two models, the terms related to eddy currents is equal and therefore there is no need to recalculate it, instead the first term of SyR-e must be decomposed in the two remaining of Ansys.

The Matlab script ***losscoefficients.m*** creates a series of loss curves at different frequencies as a function of magnetic induction with respect to the starting model and through the command **fittype** we interpolate them with respect to the new model.

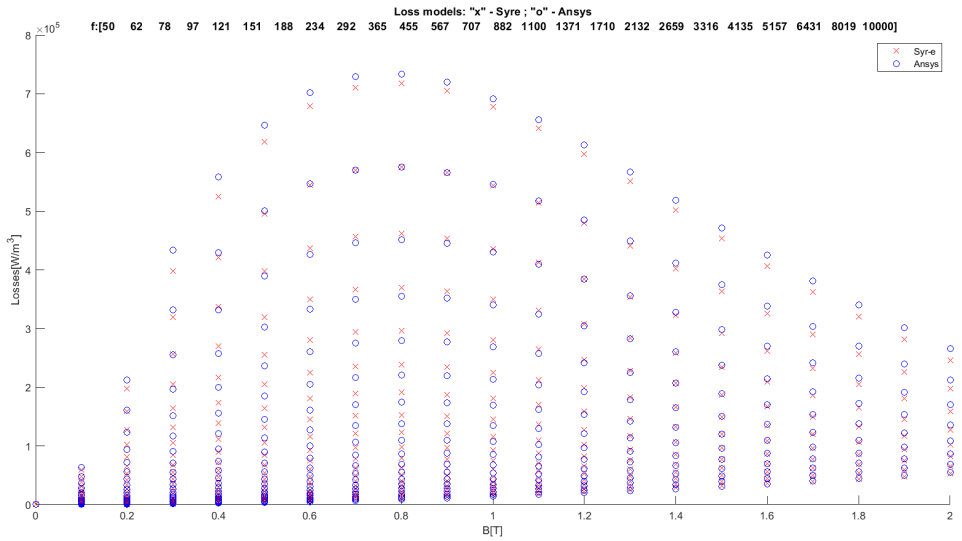


Fig. 38: Hysteresis and additional losses interpolation for core material M530-65A

The shape of the graph is due to the number of points that decrease as much the frequency increases. This choice was made in such a way as to reproduce actual loss tests.

If, on the other hand, the loss curves are known, it is always possible to manually insert the coefficients to obtain more reliable results.

3.2.2 Temperature dependence of the PMs coercive field

In SyR-e, inside the motname.mat file, as shown in example in the Fig. 39 there are the different curves that represent the dependence of the magnet to the temperature variation.

BMN-42SH

Category:	Layer
Mass density [kg/m^3]	7550
Relative permeability	1.05
Br [T]	1.33
Hc [A/m]	1007981.3062
Conductivity [S/m]	666666.6667

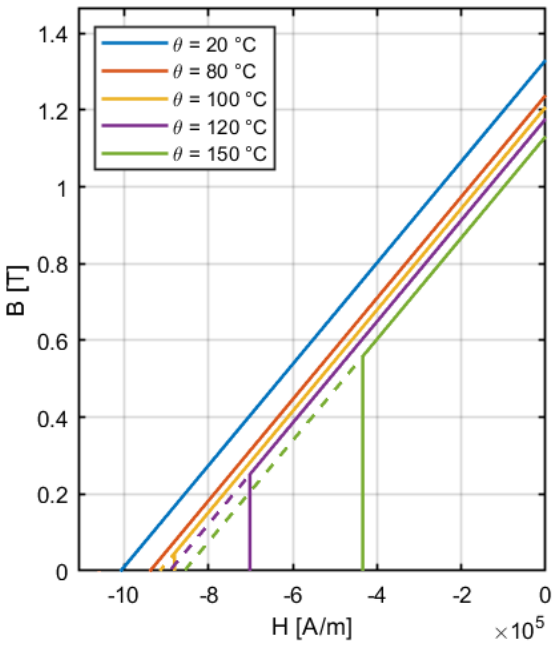


Fig. 39: Temperature dependence of material BMN-42SH

All these curves are parallel and this means that the dependence is linear and it is just necessary to obtain only one coefficient. To define the magnetization curve Ansys

requires as input the inclination of the curve, represented by the relative permeability, and the value that is encountered intersecting the x-axis, which is the coercive field H_C. The variable that must be related to temperature to define the parallel curves is therefore the coercive field.
In Ansys it is possible to relate a variable to temperature by selecting the field *"Thermal Modifier"* in the material library. The software provides a two coefficient quadratic model, also the possibility of entering any equation that links the two variables. In this case the quadratic was chosen, but in which the second coefficient was set to zero obtaining a linear relationship reported in Eq. 13:

$$H_C = H_{Cref} \cdot (1 + c_1 \cdot (T - T_{ref})) \tag{13}$$

The coefficient **c1** is calculated by means of the function ***"MagHc_temp_fitting.m"*** with respect to the reference temperature of 20°C. The implementation of the equation can be seen in Fig.31 which shows the material library of the permanent magnet.

3.2.3 Permanent Magnet Orientation

The magnetic field is a vector field characterized by a direction and an intensity; and it is therefore necessary to impose different direction of the magnetic field that the magnets exert. Taking as only example a single magnet and repeating the procedure with a for loop as first thing we need to create a relative coordinate system.
A coordinate system requires defining the position of the center and the direction of the axes, and in a two-dimensional system means having four variables for each magnet. This information was passed into *draw_motor_in_syre.m* through the pickle file and it is within the structure called geo shown in the Fig.40.

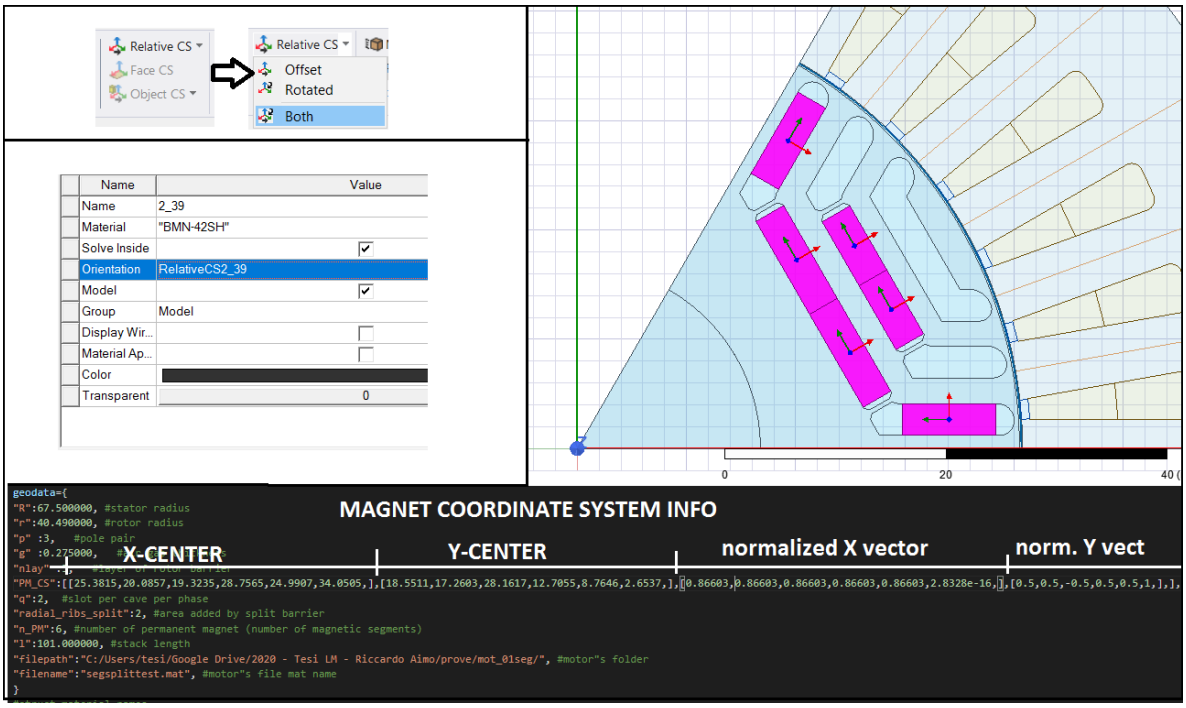


Fig. 40: Illustration of the implementation of the coordinate systems of the PMs

Once the coordinate system has been created and named with a name reminiscent of the relative permanent magnet, it is associated with the PM through the properties settings of the object. Since the properties of the selected area also include the magnetic material, the program knows already which is the direction of the field generated by each magnet.

3.3 Boundary Condition

Ansys Maxwell provides different types of boundary conditions based on the type of the simulation selected. The export is capable to run a ***"Transient"*** simulation. For such a simulation type it is possible to choose between the boundary conditions types listed in Fig. 41.

Boundary Type	H-Field Behavior	Used to model...
Default Boundary Conditions (Natural and Neumann)	Field behaves as follows: • Natural boundaries — Tangential **H** and normal **B** are continuous across surfaces without current density distribution. Tangential **H** has a jump if the surface has current density distribution. • Neumann boundaries — Magnetic field is normal to the boundary.	Initially, object interfaces have natural boundaries; outer boundaries, and excluded objects have Neumann boundaries.
Vector Potential	Sets the magnetic vector potential A_Z, or rA_ϕ, on the boundary. The behavior of H depends on whether A_Z or rA_ϕ is constant or functional.	Outer boundaries at specific vector potentials; externally applied magnetic fields.
Symmetry	Field behaves as follows: • Odd Symmetry (Flux Tangential) — **Magnetic Field** is tangential to the boundary; its normal components are zero. • Even Symmetry (Flux Normal) — **Magnetic Field** is normal to the boundary; its tangential components are zero.	Planes of geometric and magnetic symmetry.
Balloon	Models the case where the structure is "infinitely" far away from other magnetic fields, permanent magnets, or current sources.	Magnetically isolated structures.
Master and Slave (Matching)	The **Magnetic Field** on the slave boundary is forced to match the magnitude and direction (or the negative of the direction) of the **Magnetic Field** on the master boundary.	Planes of symmetry in periodic structures where **Magnetic Field**is oblique to the boundary.

Fig. 41: Transient boundaries types - Maxwell Help - Cap.11 Page 44 [2]

From the column *"Used model to..."* it can be understood that the optimal boundaries condition type for the transient simulation is the **Master and Slave** (Matching) one. Matching boundaries allow to model planes of periodicity. The magnetic field H on one surface must exactly matches the H field on another by forcing the magnetic field at each location on one surface (the "slave" boundary) to match the magnetic field at the corresponding location on the other surface (the "master" boundary). Matching boundaries are used in periodic structures and decrease the resources used in the computational process.

For matching boundaries, it is necessary to set up both the master and the slave boundary. Unlike symmetry boundaries on master and slave, the **H field** does not need to be either tangential or normal to these boundaries. However, the H field on the two boundaries must have the same magnitude and direction (or the same magnitude and opposite direction) at each time step. The variation in time of the fields at corresponding locations is the same on matching (master and slave) boundaries. [2]

Thanks to this boundary condition and to the periodicity of the machine, it is possible to study the whole machine representing even only one pole of it, thus reducing the computational burden of the analysis as well as the time required by the automation to build the geometry.

Master and Slave boundaries are assigned to the two extremes of the polar section, subsequently the outer radius of the stator is defined by means of the *"Vector Potential"* functionality set as null. The behavior of the magnetic field on a Vector Potential boundary depends on whether you define a constant or functional potential on the boundary. The magnetic vector potential $\mathbf{A}$ satisfies the equation:

$$\Delta \times \mathbf{A} = \mathbf{B} \tag{14}$$

Since the magnetostatic field solver assumes that A has only a z-component only and B lies in the xy-plane, their relationship is represented by the Eq.15:

$$\mathbf{B} = \frac{\delta \mathbf{A}_z}{\delta y}\mu_x - \frac{\delta \mathbf{A}_z}{\delta x}\mu_y \tag{15}$$

So by imposing a null potential vector $\mathbf{A}_z$ flux on that particular surface will be zero according to the Eq. 16:

$$\Phi = \int_S \mathbf{B} \cdot d\mathbf{S} = \oint_{\delta S} \mathbf{A} \cdot d\mathbf{l} \tag{16}$$

3.3.1 Boundary Implementation

The *draw_motor_in_ansys.m* script implements the three boundary conditions (Master, Slave and Vector Potential). First, it creates two straight lines and an arc to enclose the motor. Then, the three extremes are associated with the corresponding conditions.

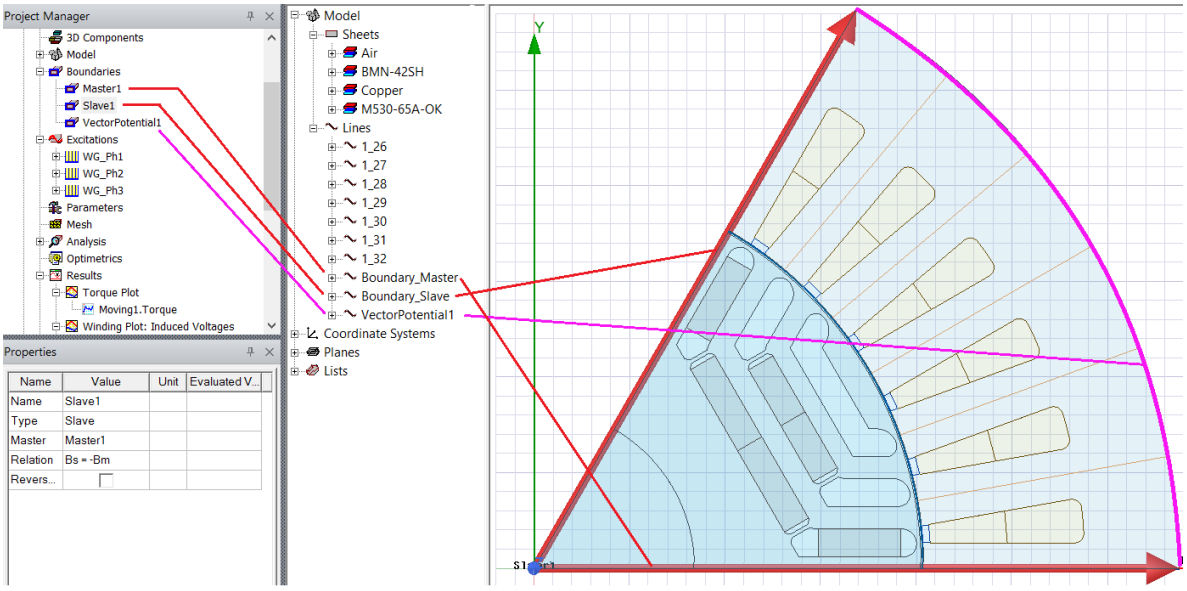


Fig. 42: Boundaries implementation in Ansys

4 Simulation Parameters and their Implementation

Starting from the quantities that can be inserted inside the 'Simulation' window in SyR-e (Fig.43), it is intended pass all the needed **parameters** to Ansys Maxwell through the SyR-e interface in order to perform the analysis of the machine.

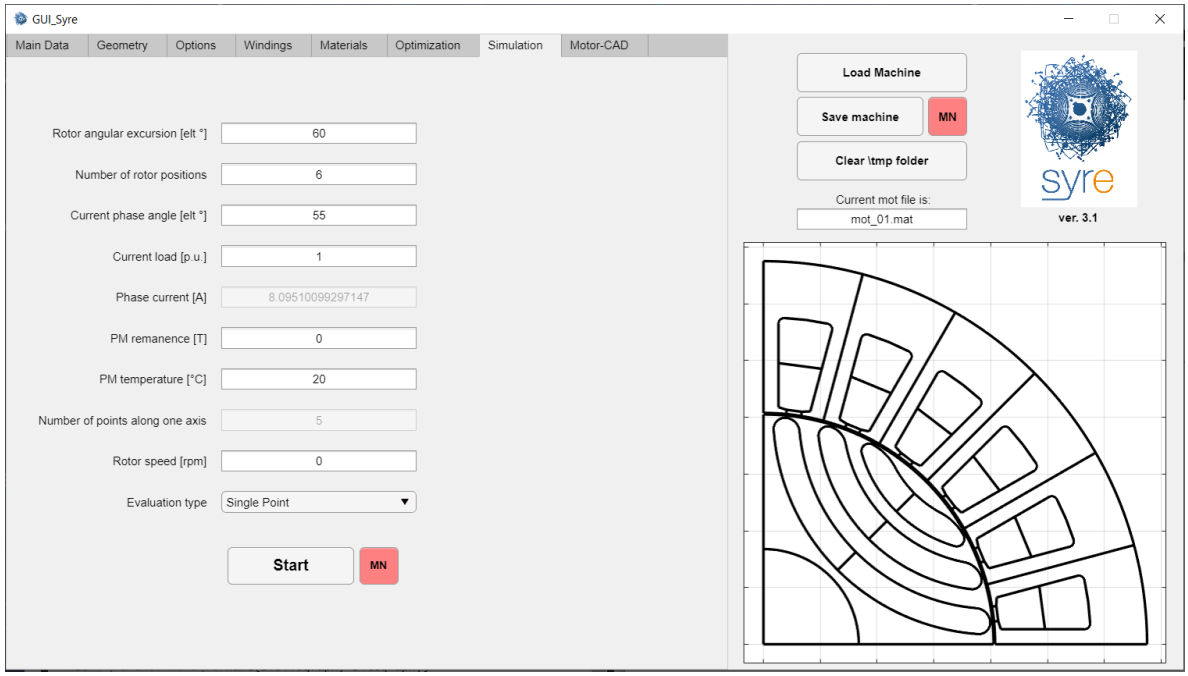


Fig. 43: Simulation Window in SyR-e GUI

Using this data as input, two codes called *simulate_xdeg_AM.m* and *sim_xdeg_AM.py* were created. According to the procedure explained previously, the first code, through the fprintf command, transmits the information coming from Matlab to Python. The second one takes care of the operative part, assigning in the Maxwell model the SyR-e parameters.

4.1 Windings and Excitations

In Ansys the excitation of the motor is realized by grouping the **coils** of the slots inside **windings** correlated with the respective number of conductors and current sign (*raw 27-78 of "sim_xdeg_AM.py*). The windings are actually the phases of the motor. The

arrangement of the coils in the phases has been taken from a matrix present in the .mat file of the SyR-e model.
The matrix mentioned above consists of two rows, corresponding to the layers of conductor groups in the quarry, and a number of columns equal to the number of slots. The slots are split in two layers because a pitch shortening factor can be applied to the three-phase windings. In Fig.44 an example of a windings matrix:

	Slot n° 1	Slot n° 2	Slot n° 3	Slot n° 4	Slot n° 5	Slot n° 6
Layer 1	1	1	2	-1	-1	2
Layer 2	-3	2	-3	-3	2	3

Fig. 44: Example of the windings matrix of a 6 slots per pole motor and a shortening pitch factor of 0.5

Then it is necessary to define the three windings or phases with their excitation **currents**. As a "Transient" simulation, the currents are described in a time-dependent form or with the classic sinusoidal notation:

$$I_{ph}(t) = I_{ph,pk} \cdot cos(p \cdot \omega_n \cdot t + \Phi_{ph} + \Phi_{init}) \tag{17}$$

$Where$:
$I_{ph,pk}$: Peak Phase Current
Φ_{ph} : Three-phase symmetrical shifts [0°,240°,120°]
Φ_{init} : Initial shift dependent on the machine
p : pole pair
ω_n : natural frequency

Once the windings have been created the corresponding coils are associated as shown in Fig.45:

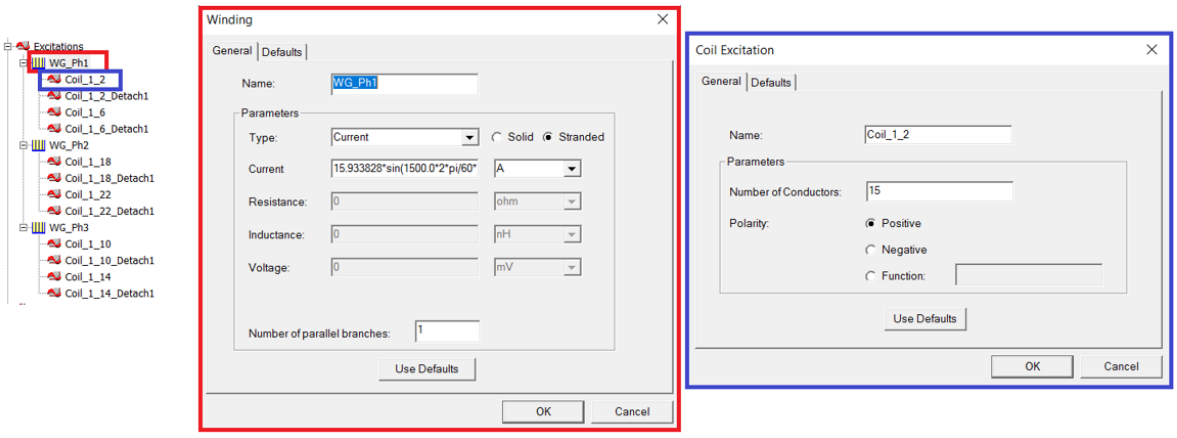


Fig. 45: Example winding and coils implementations

4.2 Motion and Solve Setup

Ansys Maxwell requires some settings that depend on the machine and that can be found under the fields **model motion setup** and **analysis setup**.

The first field is associated with the region containing the rotor, that is the circular sector mentioned in section 3.1.4 which includes all the moving parts. Therefore, the speed, the axis of rotation and the initial angle must be specified. Within the Analysis setup is possible to specify the time step and the end of the simulation, according to the input simulation data they can be expressed as:

$$t_{stop} = \frac{60}{p \cdot n_{rpm}} \cdot \frac{sim_{xdeg}}{360} \tag{18a}$$

$$t_{step} = \frac{t_{stop}}{n_{sim}} \tag{18b}$$

$Where$:

sim_{xdeg} : Simulated angle in electrical degree

n_{rpm} : Rotor velocity in rpm

n_{sim} : Number of simulation points

Note that from the GUI it is possible to start two types of simulation:

- **Single Point** - simulation of a single point of current on *dq* axis, of just 60 electrical degree of the machine, by exploiting its symmetry.

- **Core Loss** - simulation similar to the previous one but needs a 420°elt of excursion to the completion of the loss calculation (better expleined in Chap.4.4)

Furthermore, it is also possible to activate the **"Save Fields"** option in order to obtain plot Field overlays that are representations of basic or derived field quantities on the 2D machine sketch. However, such a option is disabled in the core loss simulation for the sake of the computation speed. All this settings are performed in *"sim_xdeg_AM.py" from 178-232*.

4.3 Last simulation parameters

As mentioned in Chapter 3.2.2, the magnetic materials have been defined according to a temperature dependent curve and it is therefore necessary to specify the temperature at which they operate (*raw 116-140 of "sim_xdeg_AM.py"*). In addition, to perform a Core Loss simulation, it is necessary to define in which parts the different losses will be calculated, which corresponds to where the coefficients have been defined, so the ferromagnetic parts(*raw 142-147*).

4.4 Plots

Now the software has received all the information needed to simulate the electrical machine, so the automation takes care of generating the **plots**, with the relative units of measurement, which will contain the quantities of interest.
Both single point and core loss simulations provide the following plots:

- Position

- Induced Voltage

- Input Current

- Flux Linkage
- Torque

If a simulation "Core Loss" is performed then the following **losses plots** would be added:

- Total Core
- Hysteresis
- Eddy Current
- Excess

All this data is saved in .csv files by *sim_xdeg_AM.py* (raw *414-434*) and then exported to Matlab for a future processing done by *eval_operatingPointAM.m*.
The post-processed quantities are currents, magnetic fluxes and losses. It is important that the first two are converted using the **Clarke and Park** transforms in order to move from the three-phase time domain (abc) into an orthogonal rotating reference frame (dq). Once the magnitudes on the dq references are obtained, the **IPF** can be obtained as:

$$IPF = \sin(\arctan(i_q/i_d) - \arctan(\lambda_q/\lambda_d)) \tag{19}$$

The IPF as mentioned before is approximately equal to the classic power factor and it is therefore important to know it in order to have an indication of the percentage of reactive power absorbed by the machine.
Therefore, the first type of simulation (single point) returns four Matlab plots is shown in Fig.46 .

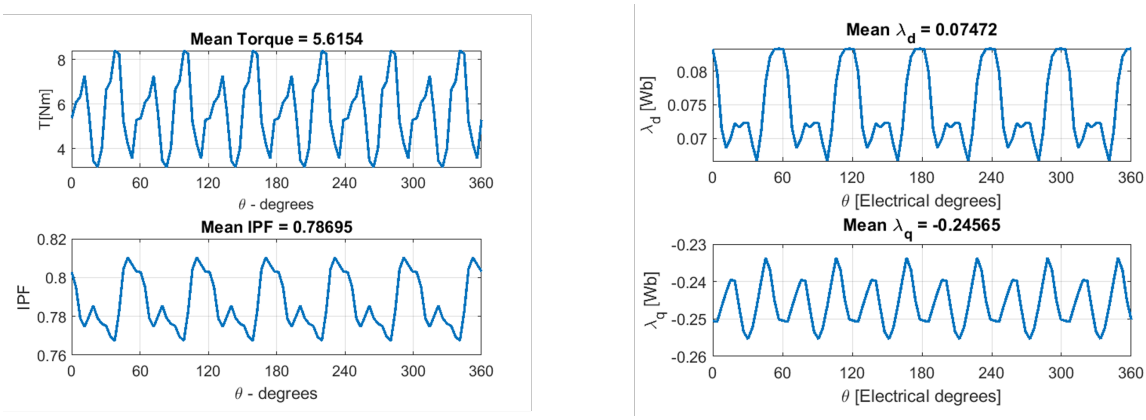


Fig. 46: Example Matlab plots of a single point simulation

Concerning the core loss simulation,, Ansys Maxwell returns us loss trends dependent on the time. Obviously these trends do not represent any transient physical phenomenon, but rather they are the results of the calculation method used by the software. For this reason, it is necessary that a simulation of 420 degrees is carried out in such a way that the calculation transients are completed by making an entire turn and after which the following 60 degrees are used to derive an average value.

To demonstrate that there is a 60° periodicity after completing a full turn, an example of total core loss is shown in Fig.47. This plot is performed on 720 electrical degrees for a 4 poles machine and as can be seen after the period of 10ms there is a repetition of the waveform every sixth period.

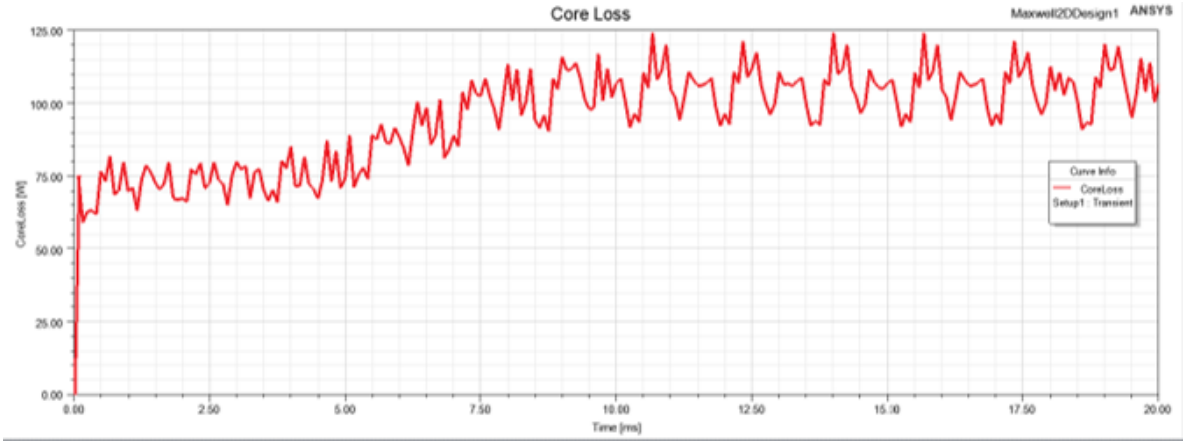


Fig. 47: Example Ansys Maxwell core loss plot

Knowing this the averages on the last 60 of the 420 electrical degrees simulated are then performed. Using these values a histogram is created showing the power core losses and the different components of the model used by Ansys. An example is given in Fig.48.

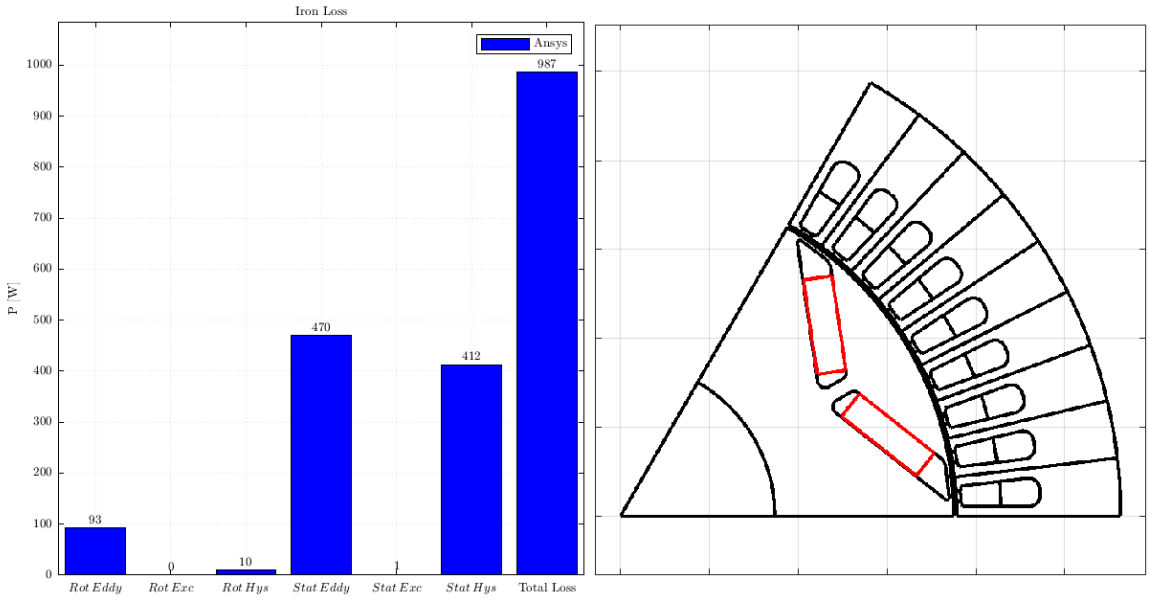


Fig. 48: Histogram core loss example: Tesla Model 3 machine - $\gamma = 55$ - $n = 5krpm$ - $T_{PM} = 80°C$

5 Partner Company Prototype Study through the New Integrated Tool

5.1 Automotive e-Machine Prototype

Having implemented and finished the new export tool, it is then possible to reproduce any electrical machine on SyR-e and then study it on Ansys Maxwell.
In this regard, the partner company provides us the model of one of their prototypes, in order to validate the built procedure.
The prototype is a traction IPM (Interior Permanent Magnet) motor which is reproduced in SyR-e.

Final 2d representation of the machine is omitted because part of the information protected by Non-Disclosure Agreement (NDA)

The procedure is validated with two critical operating points: maximum torque at the base speed and at maximum speed. The model is built in the SyR-e GUI and the he flux maps are calculated and in the **MMM** GUI (Magnetic Model Manipulation) the requested operating point specifications are found. Tab. shows the motor data required by SyR-MMM:

Table 1: e-Machine prototype data

I_{Rated}	$[Apk]$
I_{MAX}	$[Apk]$
$V_{DC-link}$	$[V]$
n_{max}	$[rpm]$
T_{PM}	$[^{\circ}C]$
$T_{windings}$	$[^{\circ}C]$

All the geometric quantities and simulation parameters are included in the SyR-e model files. (visible in Fig. 3).

In order to compute the current trajectories and the torque speed limit, the flux maps are manipulated by the second GUI and the results are displayed in Fig.49-50.

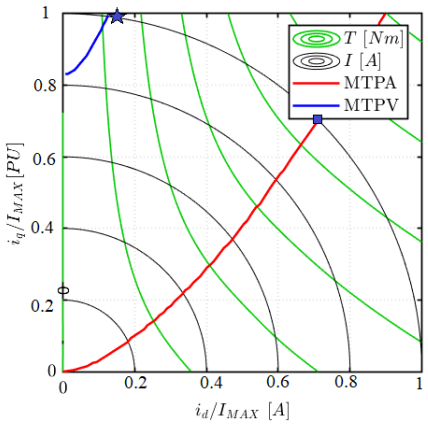


Fig. 49: Current vector trajectories for maximum power.

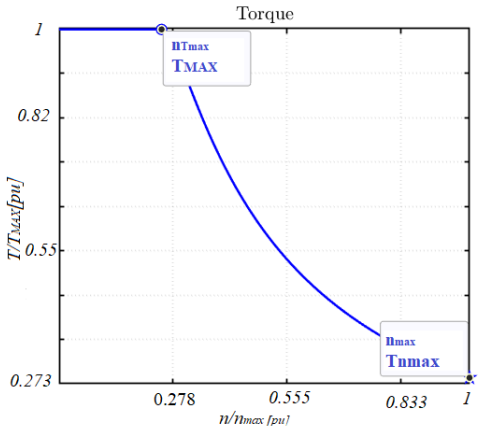


Fig. 50: Torque operating limit versus speed.

The two points of interest marked on the graphs are equivalent to the maximum torque at the base speed and at the maximum speed given the DC voltage limit.

In order to simulate the two points of interest, it is therefore necessary to obtain the corresponding current angles using simply the (Eq.20):

$$\gamma = \arctan(\frac{i_q}{i_d}) \tag{20}$$

Therefore:

Table 2: Current Angle for the selected working points

	$i_d[pu]$	$i_q[pu]$	$\gamma[^\circ]$
$T_{MAX} - n_{base}$	0.712	0.703	44.64
$T_{MAX} - n_{MAX}$	0.137	0.991	82.12

Owning all the necessary data, it is possible to execute the two simulations inserting the data above.

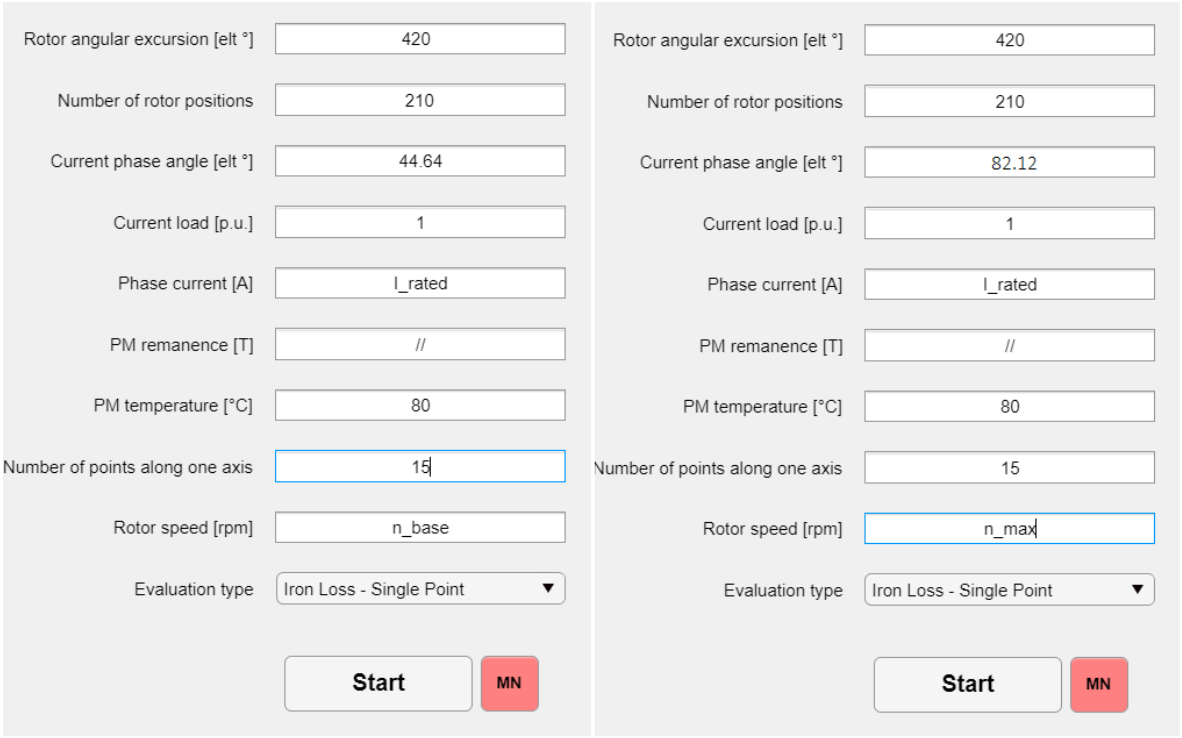


Fig. 51: Input data for a Core Loss simulation for two different current angle and speed of the e-Machine under investigation

5.2 Core Loss Results for the Automotive e-Machine

The machine was studied through core loss simulations in the two operating points found in the previous Chapter.
The results obtained with Ansys Maxwell were then compared with the software already linked to SyR-e: **FEMM, Motor-CAD and MagNet**.

5.2.1 Results of CoreLoss Simlation

This section shows the graphs obtained by simulating the speed corresponding to the maximum torque and the maximum current and the relative current angle. Losses were calculated with respect to coefficients derived directly from the loss curves of the

ferromagnetic material, using curve fitting of the curves.

Table 3: Torque and Speed Simulated Points

	Company		Script	
	n_{base}	n_{max}	n_{base}	n_{max}
$Torque[pu]$	0.957	0.257	1	0.285

The data are reported mainly in PU to obey the NDA.

As can be seen from Tab.3 the values obtained from the company differ by a few percentage points from those obtained with the export, this difference is caused by the adaptation of the electrical machine to the SyR-e software.

The losses are all normalized to the maximum total loss value obtained in Maxwell (corresponding to the maximum velocity),instead the fluxes are normalize on the d-axis mean value obtained at the base speed.

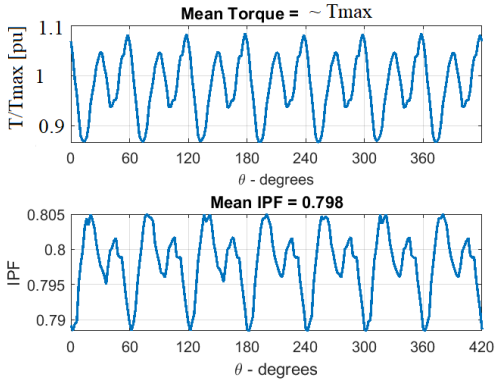


Fig. 52: Torque vs electrical position - $MeanT = 1pu$ $\gamma = 44.64$ and n_{Tmax}

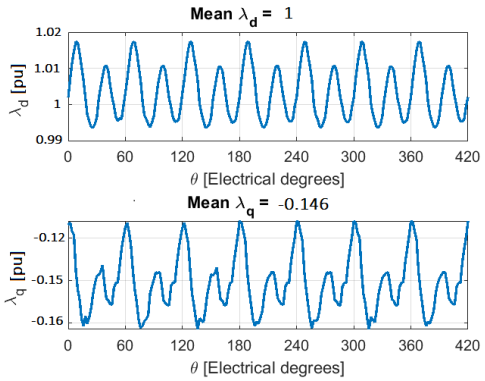
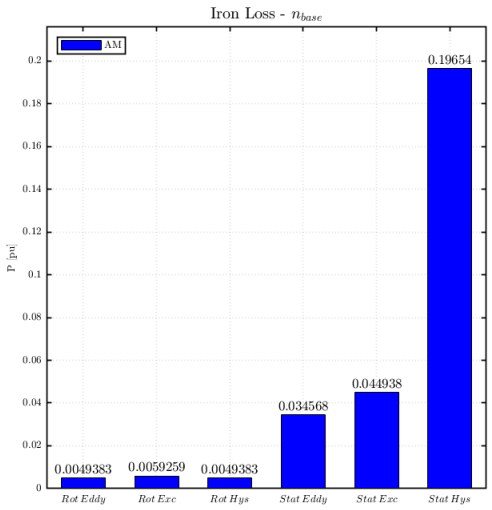
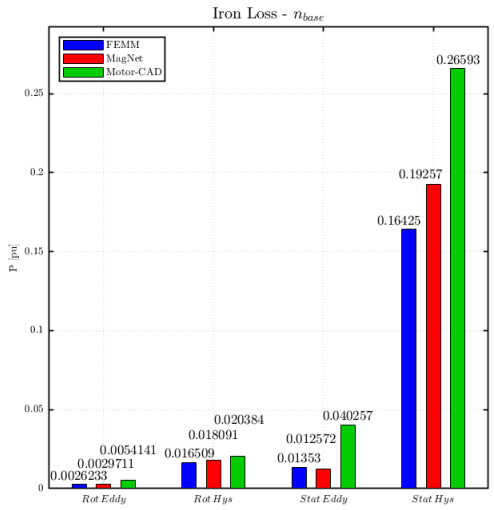


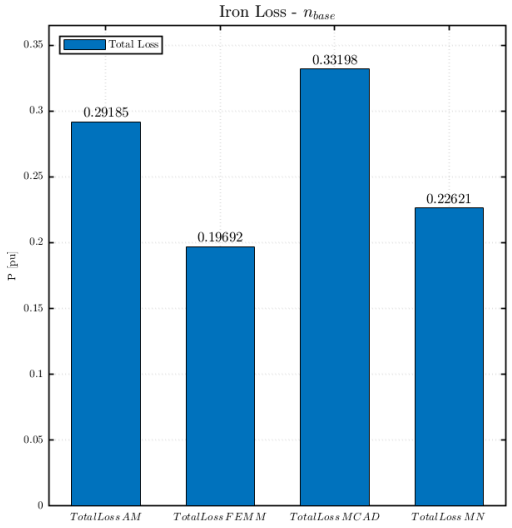
Fig. 53: dq fluxes vs electrical position - $\gamma = 44.64$ and n_{Tmax}



(a) Partial core losses calculated by Ansys Maxwell



(b) Partial core losses calculated by comparison software



(c) Total Losses Comparison

Fig. 54: Core losses comparison $\gamma = 44.64$ and n_{base}

In Fig. 55-56-57 are showed the results of the second operating point, including the core loss obtained with the others FEA software. As before the data are reported mainly in per unit to obey the NDA.

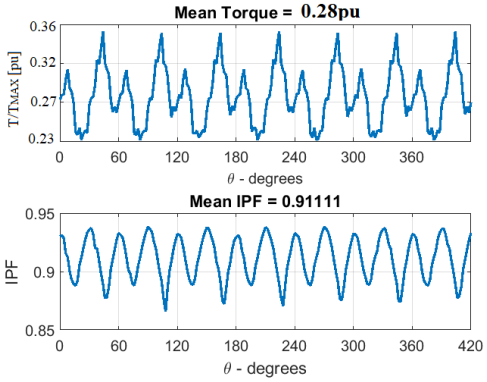


Fig. 55: Torque vs electrical position - $\gamma = 82.12$ and n_{max}

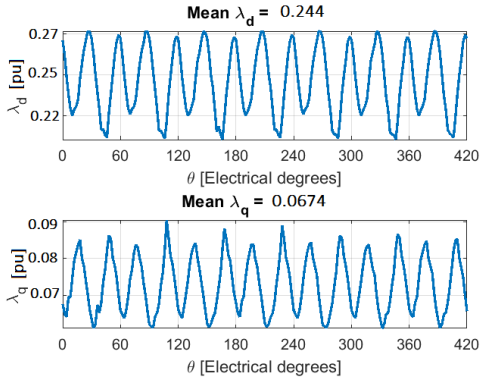
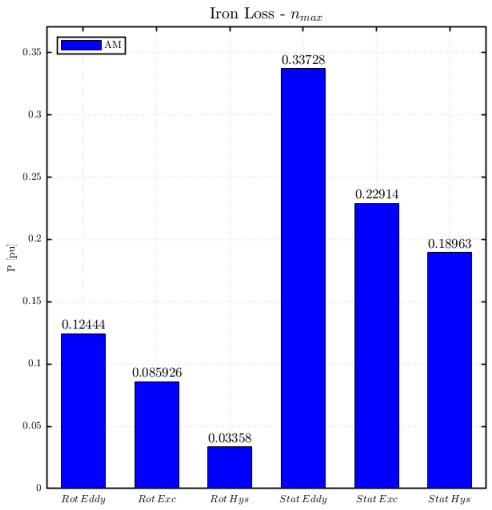


Fig. 56: dq fluxes vs electrical position - $\gamma = 82.12$ and n_{max}

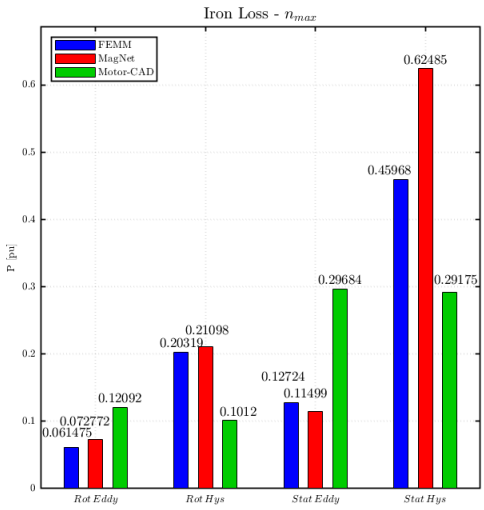
The company shared loss data obtained through two simulations also performed on Ansys for the same working points. The main difference between the tests performed with SyR-e and those performed by the company is that the machine was readjusted and parameterized to the SyR-e software, so there is a small difference between the two geometries. Tab. 4 shows the comparison between the different results:

Table 4: Iron loss comparison with company results

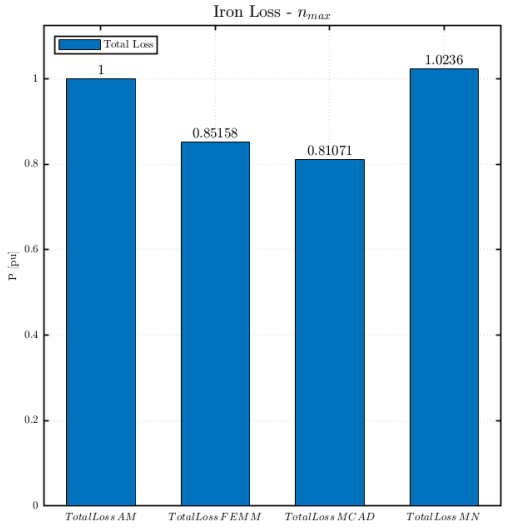
	Company		Script	
$IronLosses$	n_{base}	n_{max}	n_{base}	n_{max}
$Statorhysteresis$	0.234	0.329	0.197	0.190
$Statorexcess$	0.0200	0.103	0.0449	0.229
$Statoreddycurrents$	0.0339	0.342	0.0346	0.337
$Rotorhysteresis$	0.0150	0.0747	0.00493	0.0336
$Rotorexcess$	0.00346	0.0435	0.00593	0.0859
$Rotoreddycurrents$	0.00649	0.147	0.00494	0.124
$Total$	0.313	1.04	0.292	1



(a) Partial core losses calculated by Ansys Maxwell



(b) Partial core losses calculated by comparison software



(c) Total losses comparison

Fig. 57: Core losses comparison $\gamma = 44.64$ and n_{base}

It is clear from Tab.4 how the results of the partial losses are slightly different, but this is due to a different method used to calculate the coefficients. Instead it is easy to see that the total losses differ by a few percentage units and knowing that the loss

calculation is not precise from the point of view of FEA the results can be considered satisfactory.

6 Conclusion

This report analyzes the procedure that allowed to implement export and simulation of SyR motors from SyR-e platform to Ansys Maxwell software. The key contributions of the Thesis are:

- The new coupling: export automatically the machine geometry and model materials from SyR-e to Ansys Maxwell through codes written in Python and Matlab.
- Single point simulations: launch the simulations in Maxwell directly and automatically from the SyR-e interface. The results were validated with FEMM.
- Iron loss simulations:launch the simulations in Maxwell directly and automatically from the SyR-e interface.
- Validate the procedure by simulating an automotive high performance electric machine designed by partner company and comparing the core loss results with the ones that can be obtained in SyR-e.(FEMM, MAGNET and Motor-CAD)
- Add a dedicated button on the SyR-e interface in order to start the presented procedures in Ansys Maxwell.

Potential future developments:

- Obtain 3D geometry in Ansys Maxwell from exported 2D geometry.
- Multi-physics Analysis

Bibliography-Sitography

[1] Maxwell Scripting Guide - Release 2020 R1 - January 2020 ANSYS, Inc. - Southpointe - 2600 ANSYS Drive - Canonsburg, PA 15317

[2] Maxwell Help (Manual) - Release 2020 R1 - January 2020 ANSYS, Inc. - Southpointe - 2600 ANSYS Drive - Canonsburg, PA 15317

[3] Ansys Maxwell Website and Description:
www.ansys.com/it-it/products/electronics/ansys-maxwell

[4] Motor-CAD Website and Description:
www.motor-design.com/motor-cad/

[5] FEMM Website and Description:
www.femm.info/wiki/HomePage

[6] Simcenter Magnet Website and Description:
www.plm.automation.siemens.com/global/it/products/simcenter/magnet.html

[7] The software SyR-e and related documentation can be found for free at the following link:
github.com/SyR-e/syre_public

[8] B. Ban, S. Stipetić and M. Klanac, "Synchronous Reluctance Machines: Theory, Design and the Potential Use in Traction Applications," 2019 International Conference on Electrical Drives and Power Electronics (EDPE), 2019, pp. 177-188

[9] Performance of IPM-PMASR Motors with Ferrite Injection for Home Appliance Washing Machine - Eric Armando, Paolo Guglielmi, Michele Pastorelli, Gianmario Pellegrino, Alfredo Vagati

[10] Progetto di Motori Sincroni a Riluttanza in SyR-e e Motor-CAD - Antonio Mandarano, Prof. Gianmario Pellegrino, Ing. Simone Ferrari

[11] Appunti delle Lezioni di Macchine Elettriche 2 parte di Dinamica delle Macchine Elettriche - Prof. Andrea Cavagnino - Anno Accademico 2018-2019

[12] Appunti delle Lezioni di Macchine Elettriche 2 parte di Costruzione Elettromeccaniche - Prof. Andrea Cavagnino - Anno Accademico 2018-2019

[13] F. Cupertino, G. Pellegrino and C. Gerada, "Design of Synchronous Reluctance Motors With Multiobjective Optimization Algorithms," in IEEE Transactions on Industry Applications, vol. 50, no. 6, pp. 3617-3627, Nov.-Dec. 2014

[14] S. Ferrari, P. Ragazzo, G. Dilevrano and G. Pellegrino, "Flux-Map Based FEA Evaluation of Synchronous Machine Efficiency Maps", 2021 IEEE Workshop on Electrical Machine Design, Control and Diagnosis (WEMDCD), Modena (Italy)

[15] P. Ragazzo, S. Ferrari, N. Rivière, M. Popescu and G. Pellegrino, "Efficient Multiphysics Design Workflow of Synchronous Reluctance Motors", 2020 XIV International Conference on Electrical Machines (ICEM), Goteborg, 2020

[16] A. Krings and C. Monissen, "Review and Trends in Electric Traction Motors for Battery Electric and Hybrid Vehicles," 2020 International Conference on Electrical Machines (ICEM), 2020, pp. 1807-1813

[17] A. Varatharajan, D. Brunelli, S. Ferrari, P. Pescetto and G. Pellegrino, "syreDrive: Automated Sensorless Control Code Generation for Synchronous Reluctance Motor Drives," 2021 IEEE Workshop on Electrical Machines Design, Control and Diagnosis (WEMDCD), 2021, pp. 192-197

[18] V-Shape IPM Rotors Design Consideration and Concerns - 2020 Windings, Inc www.windings.com/wp-content/uploads/ipm-rotor-considerations-guide.pdf

A Matlab Codes

draw_motor_in_ansys.m

```
close all
clc
clear all
% path motor
[filename, filepath] = uigetfile('\*.mat');
addpath(filepath);
load(filename)

filepath = dataSet.currentpathname;
filename = dataSet.currentfilename;
load([filepath filename]);

syreToDxfansys(geo.stator,geo.rotor,filepath,filename) %create dxf of selected motor

filepath      = strrep(filepath,'\','/');
currentFolder = pwd;
currentFolder = strrep(currentFolder,'\','/');
%iron python inside ansys path
%ipypath='"C:\Program Files\AnsysEM\AnsysEM20.1\Win64\common\IronPython\ipy64.exe" ';
defipypath          = 'C:\Program Files\AnsysEM\AnsysEM20.1\Win64\common\IronPython\ipy64.exe';
[ipy64exe, ipypath] = uigetfile(defipypath,'Select ipy64.exe (Ansys) Directory');
ipypath=strcat('"',ipypath,ipy64exe,'" "');

```

draw_motor_in_ansys.m

```
%% Savings Variabiles
n_pontR           = nnz(geo.pontR);                                    % number of radial gap (split ribs considered after)
R                 = geo.R;                                             %stator radius
r                 = geo.r;                                             %rotor radius
p                 = geo.p;                                             %pole pair
g                 = geo.g;                                             %air gap thickness
nlay              = geo.nlay;                                          %layaer of rotor barriers
q                 = geo.q;                                             %slot per cave per phase
l                 = geo.l;                                             %stack length
radial_ribs_split = nnz(geo.radial_ribs_split.*geo.pontR); %half added barrier area using split

Rotor_MatName    = mat.Rotor.MatName;    %rotor material
Stator_MatName   = mat.Stator.MatName;   %stator material
Shaft_MatName    = mat.Shaft.MatName;    %shaft material
SlotCond_MatName = mat.SlotCond.MatName; %slot conductor material
LayerMag_MatName = mat.LayerMag.MatName; %magnet material
LayerAir_MatName = mat.LayerAir.MatName; %air material

LayerMag_Hc      = mat.LayerMag.Hc;      %magnet Hc
```

draw_motor_in_ansys.m

```
43 LayerMag_mu      = mat.LayerMag.mu;      %magnet permeability
44 LayerMag_sigmaPM = mat.LayerMag.sigmaPM; %magnet conductivity
45 LayerMag_kgm3    = mat.LayerMag.kgm3;    %magnet density
46
47
48 SlotCond_sigma = mat.SlotCond.sigma; %cond conductivity
49 SlotCond_alpha = mat.SlotCond.alpha; %cond thermal coefficient
50 SlotCond_kgm3  = mat.SlotCond.kgm3;  %cond density
51
52 Shaft_alpha = mat.Shaft.alpha; %alfa loss coefficient shaft
53 Shaft_beta  = mat.Shaft.beta;  %beta loss coefficient shaft
54 Shaft_kh    = mat.Shaft.kh;    %hysteresis loss constant
55 Shaft_ke    = mat.Shaft.ke;    %eddy current loss constant
56 Shaft_BH    = mat.Shaft.BH;    %BH curve shaft
57 Shaft_kgm3  = mat.Shaft.kgm3;  %Shaft density;
58
59 Rotor_alpha = mat.Rotor.alpha; %alfa loss coefficient rotor
60 Rotor_beta  = mat.Rotor.beta;  %beta loss coefficient rotor
61 Rotor_kh    = mat.Rotor.kh;    %hysteresis loss constant
62 Rotor_ke    = mat.Rotor.ke;    %eddy current loss constant
63 Rotor_BH    = mat.Rotor.BH;    %BH curve Rotor
64 Rotor_kgm3  = mat.Rotor.kgm3;  %Rotor density;
65
66 Stator_alpha = mat.Stator.alpha; %alfa loss coefficient rotor
67 Stator_beta  = mat.Stator.beta;  %beta loss coefficient rotor
68 Stator_kh    = mat.Stator.kh;    %hysteresis loss constant
69 Stator_ke    = mat.Stator.ke;    %eddy current loss constant
70 Stator_BH    = mat.Stator.BH;    %BH curve Stator
```

draw_motor_in_ansys.m

```
71 Stator_kgm3  = mat.Stator.kgm3;  %Stator density;
72
73 BLKLABELS_rotore_xy = geo.BLKLABELS.rotore.xy; %coordinate materiali
   ↪  rotore
74
75
76 %% Export Variables
77
78 fclose('all');
79 pyfile = fopen( 'setup_draw_motor_in_ansys.txt', 'wt' );
80
81 % coordinate system permanent magnets
82 [k,]     = find(BLKLABELS_rotore_xy(:,3)==6);
83 geo.n_PM = length(k);   %number Permanent Magnet
84 n_PM     = geo.n_PM;
85 PM_CS    = '"PM_CS":['; %init. coordinate system PMs
86
87
88 for ii = [1,2,6,7]
89     PM_CS = strcat(PM_CS,'[');
90     for jj = 1:length(k)
91         PM_CS =
           ↪  strcat(PM_CS,num2str(BLKLABELS_rotore_xy(jj,ii)),',');
92     end
93     PM_CS = strcat(PM_CS,'],');
94 end
95 PM_CS = strcat(PM_CS,'],');
96
```

draw_motor_in_ansys.m

```
97  %radial_split_ribs
98  export={
99      'import sys,os'
100     'import pickle'
101
102     'geodata={'
103     '"R":%f, #stator radius'
104     '"r":%f, #rotor radius'
105     '"p" :%d,   #pole pair'
106     '"g" :%f,   #air gap thickness'
107     '"nlay" :%d,   #layer of rotor barrier'
108     '%s #Coordinate system magnet: 0-x 1-y 2-xvect 3-yvect'
109     '"q":%d,  #slot per cave per phase'
110     '"radial_ribs_split":%d, #area added by split barrier'
111     '"n_PM":%d, #number of permanent magnet (number of magnetic
        ↪  segments)'
112     '"l":%f, #stack length'
113     '"filepath":"%s", #motor"s folder'
114     '"filename":"%s", #motor"s file mat name'
115     '}'
116     '#struct material names'
117     'material={'
118     '"rotor":"%s",'
119     '"stator":"%s",'
120     '"shaft":"%s",'
121     '"slotcond":"%s",'
122     '"magnet":"%s",'
123     '}'
```

draw_motor_in_ansys.m

```
124
125     "with open('%s/temp/temp.pkl', 'wb') as export:"
126     '    pickle.dump([geodata,material],export,protocol=2)'
127 };
128
129 str = sprintf('%s\n',export{:});
130 fprintf(pyfile,str,R,r,p,g,nlay,PM_CS,q,radial_ribs_split,...
131     n_PM,l,filepath,filename,Rotor_MatName,Stator_MatName,...
132     Shaft_MatName,SlotCond_MatName,LayerMag_MatName,currentFolder);
133 clear export
134
135
136 %% Programm Start
137
138 start={
139     'sys.path.append(r"C:/Program Files/AnsysEM/AnsysEM20.1/Win64")'
140     'sys.path.append(r"C:/Program
    ↪  Files/AnsysEM/AnsysEM20.1/Win64/PythonFiles/DesktopPlugin")'
141     'import ScriptEnv'
142     ''
143     'ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")'
144     'oDesktop.RestoreWindow()'
145     'oProject = oDesktop.NewProject()'
146     'oProject.InsertDesign("Maxwell 2D", "Maxwell2DDesign1",
    ↪  "Transient", "")'
147     'oDesign = oProject.SetActiveDesign("Maxwell2DDesign1")'
148     'oEditor = oDesign.SetActiveEditor("3D Modeler")'
149     'oProject.SaveAs("%s%s.aedt", True)'
```

draw_motor_in_ansys.m

```
    'oDesktop.ClearMessages("", "",3)'};
str=sprintf('%s\n',start{:});
fprintf(pyfile,str,filepath,erase(filename,".mat"));
clear start

%%%%%%%%%%%%%%%%Air%%%%%%%%%%%%%%%

air={
    '#########Add materials to library'
    '#########Air###########'
    'oDefinitionManager = oProject.GetDefinitionManager()'
    'oDefinitionManager.AddMaterial('
    '  ['
    '    "NAME:%s ",'
    '    "CoordinateSystemType:=", "Cartesian",'
    '    "BulkOrSurfaceType:="  , 1,'
    '    ['
    '      "NAME:PhysicsTypes",'
    '      "set:="      , ["Electromagnetic"]'
    '    ]'
    '  ])'};
str=sprintf('%s\n',air{:});
fprintf(pyfile,str,LayerAir_MatName);
clear air

%%%%%%%%%%%%%%%%Magnet%%%%%%%%%%%%%%%%%%%%%%

if LayerMag_Hc == 0
```

draw_motor_in_ansys.m

```
178 mag={''};
179 else
180     if isfield(mat.LayerMag,'temp')
181         [LayerMag_Tcoeff,LayerMag_Href] = MagHc_temp_fitting(mat);
182     else
183         LayerMag_Tcoeff=0;
184         LayerMag_Href=LayerMag_Hc;
185     end
186 mag={
187     '############Magnet############'
188     'oDefinitionManager.AddMaterial('
189     '  ['
190   '  "NAME:%s",'
191   '  "CoordinateSystemType:=", "Cartesian",'
192   '  "BulkOrSurfaceType:="  , 1,'
193   '  ['
194   '    "NAME:PhysicsTypes",'
195   '    "set:="      , ["Electromagnetic"]'
196   '  ],'
197   '  ['
198   '    "NAME:ModifierData",'
199   '    ['
200   '      "NAME:ThermalModifierData",'
201   '      "modifier_data:="  , "thermal_modifier_data",'
202   '      ['
203   '        "NAME:all_thermal_modifiers",'
204   '        ['
205   '          "NAME:one_thermal_modifier",'
```

draw_motor_in_ansys.m

```
206   '          "Property::="    , "magnetic_coercivity",'
207   '          "Index::="    , 0,'
208   '          "prop_modifier:="  , "thermal_modifier",'
209   '          "use_free_form:="  , False,'
210   '          "Tref:="    , "20cel",'
211   '          "C1:="      , "%f",'
212   '          "C2:="      , "0",'
213   '          "TL:="      , "-273.15cel",'
214   '          "TU:="      , "1000cel",'
215   '          "auto_calculation:="  , True'
216   '        ]'
217   '      ]'
218   '    ]'
219   '  ],'
220   '  "permeability:="  , "%f",'
221   '  "conductivity:="  , "%f",'
222   '  ['
223   '    "NAME:magnetic_coercivity",'
224   '    "property_type:="  , "VectorProperty",'
225   '    "Magnitude:="    , "-%fA_per_meter",'
226   '    "DirComp1:="    , "1",'
227   '    "DirComp2:="    , "0",'
228   '    "DirComp3:="    , "0"'
229   '  ],'
230   '  "mass_density:="  , "%f"'
231   '])'
232     };
233
```

draw_motor_in_ansys.m

```
234
235 str=sprintf('%s\n',mag{:});
236
237 fprintf(pyfile,str,LayerMag_MatName,LayerMag_Tcoeff,LayerMag_mu,...
238 LayerMag_sigmaPM,LayerMag_Href,LayerMag_kgm3);
239
240 clear mag
241 end
242 %%%%%%%%%%%%%%%%%%%% Slot Conductor %%%%%%%%%%%%%%%%%%%%%%%%%
243
244 slotcond={
245     '########## Slot conductor ##########'
246     'oDefinitionManager.AddMaterial('
247   '['
248   '  "NAME:%s",'
249   '  "CoordinateSystemType:=", "Cartesian",'
250   '  "BulkOrSurfaceType:="  , 1,'
251   '  ['
252   '    "NAME:PhysicsTypes",'
253   '    "set:="      , ["Electromagnetic","Thermal"]'
254   '  ],'
255   '  "conductivity:="  , "%f",'
256   '  "thermal_conductivity:=", "%f",'
257   '  "mass_density:="  , "%f"'
258   '])'
259     };
260 str=sprintf('%s\n',slotcond{:});
261 fprintf(pyfile,str,SlotCond_MatName,SlotCond_sigma,...
```

```
262         SlotCond_alpha*1e5,SlotCond_kgm3);
263 clear slotcond
264
265 %%%%%%%%%%%%%%%%%%%% Shaft %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
266 if strcmp(Shaft_MatName,'ShaftAir')
267     shaft={
268     '##########Add materials to library'
269     '##########ShaftAir###########'
270     'oDefinitionManager = oProject.GetDefinitionManager()'
271     'oDefinitionManager.AddMaterial('
272     '  ['
273     '    "NAME:%s ",'
274     '    "CoordinateSystemType:=", "Cartesian",'
275     '    "BulkOrSurfaceType:="  , 1,'
276     '    ['
277     '      "NAME:PhysicsTypes",'
278     '      "set:="      , ["Electromagnetic"]'
279     '    ]'
280     '  ])'};
281 str=sprintf('%s\n',shaft{:});
282 fprintf(pyfile,str,Shaft_MatName);
283 clear shaft
284
285 else
286 [Kh,Kadd,Ke]=losscoefficients(Shaft_alpha,Shaft_beta,...
287                         Shaft_kh,Shaft_ke,Shaft_kgm3);
288 shaft={
289     '############ Shaft plates ###########'
```

draw_motor_in_ansys.m

```
290     'oDefinitionManager.AddMaterial('
291   '['
292   '  "NAME:%s",'
293   '  "CoordinateSystemType:=", "Cartesian",'
294   '  "BulkOrSurfaceType:="  , 1,'
295   '  ['
296   '    "NAME:PhysicsTypes",'
297   '    "set:="      , ["Electromagnetic"]'
298   '  ],'
299   '  ['
300   '    "NAME:permeability",'
301   '    "property_type:="  , "nonlinear",'
302   '    "BTypeForSingleCurve:="  , "normal",'
303   '    "HUnit:="    , "A_per_meter",'
304   '    "BUnit:="    , "tesla",'
305   '    "IsTemperatureDependent:=", False,'
306   '    ['
307   '      "NAME:BHCoordinates",'
308   '      ['
309   '        "NAME:DimUnits", '
310   '        "", '
311   '        ""'
312   '      ],'
313     };
314 str=sprintf('%s\n',shaft{:});
315 fprintf(pyfile,str,Shaft_MatName);
316 clear shaft
317
```

draw_motor_in_ansys.m

```
318 for ii=1:length(Shaft_BH(:,1))
319 shaft={
320   '      ["NAME:Coordinate",["NAME:CoordPoint",%f,%f]],'
321     };
322 str=sprintf('%s\n',shaft{:});
323 fprintf(pyfile,str,Shaft_BH(ii,2),Shaft_BH(ii,1));
324 clear shaft
325 end
326     clear shaft
327     shaft={
328   '    ],'
329   '    ['
330   '      "NAME:Temperatures"'
331   '    ]'
332   '  ],'
333   '  ['
334   '    "NAME:magnetic_coercivity",'
335   '    "property_type:="  , "VectorProperty",'
336   '    "Magnitude:="    , "0A_per_meter",'
337   '    "DirComp1:="    , "1",'
338   '    "DirComp2:="    , "0",'
339   '    "DirComp3:="    , "0"'
340   '  ],'
341   '  ['
342   '    "NAME:core_loss_type",'
343   '    "property_type:="  , "ChoiceProperty",'
344   '    "Choice:="    , "Electrical Steel"'
345   '  ],'
```

draw_motor_in_ansys.m

```
  '  "core_loss_kh:="  , "%f",'
  '  "core_loss_kc:="  , "%f",'
  '  "core_loss_ke:="  , "%f",'
  '  "core_loss_kdc:="  , "0",'
  '  "mass_density:="  , "%f",'
  '  "core_loss_equiv_cut_depth:=", "0.001meter"'
  '])'

};
str=sprintf('%s\n',shaft{:});
fprintf(pyfile,str,Kh,Ke,Kadd,Shaft_kgm3);
clear shaft
end

%%%%%%%%%%%%%%%%%%% Rotor's plates %%%%%%%%%%%%%%%%%%%
if strcmp(Shaft_MatName,Rotor_MatName)==0
[Kh,Kadd,Ke]=losscoefficients(Rotor_alpha,Rotor_beta,...
                        Rotor_kh,Rotor_ke,Rotor_kgm3);
rotor={
    '############ Rotor"s plates ###########'
    'oDefinitionManager.AddMaterial('
  '['
  '  "NAME:%s",'
  '  "CoordinateSystemType:=", "Cartesian",'
  '  "BulkOrSurfaceType:="  , 1,'
  '  ['
  '    "NAME:PhysicsTypes",'
  '    "set:="      , ["Electromagnetic"]'
```

draw_motor_in_ansys.m

```
  '  ],'
  '  ['
  '    "NAME:permeability",'
  '    "property_type:="  , "nonlinear",'
  '    "BTypeForSingleCurve:="  , "normal",'
  '    "HUnit:="    , "A_per_meter",'
  '    "BUnit:="    , "tesla",'
  '    "IsTemperatureDependent:=", False,'
  '    ['
  '      "NAME:BHCoordinates",'
  '      ['
  '        "NAME:DimUnits", '
  '        "", '
  '        ""'
  '      ],'
    };
str=sprintf('%s\n',rotor{:});
fprintf(pyfile,str,Rotor_MatName);
clear rotor
for ii=1:length(Rotor_BH(:,1))
rotor={
  '      ["NAME:Coordinate",["NAME:CoordPoint",%f,%f]],'
    };
str=sprintf('%s\n',rotor{:});
fprintf(pyfile,str,Rotor_BH(ii,2),Rotor_BH(ii,1));
clear rotor
end
    clear rotor
```

draw_motor_in_ansys.m

```
    rotor={
  '    ],'
  '    ['
  '      "NAME:Temperatures"'
  '    ]'
  '  ],'
  '  ['
  '    "NAME:magnetic_coercivity",'
  '    "property_type:="  , "VectorProperty",'
  '    "Magnitude:="    , "0A_per_meter",'
  '    "DirComp1:="    , "1",'
  '    "DirComp2:="    , "0",'
  '    "DirComp3:="    , "0"'
  '  ],'
  '  ['
  '    "NAME:core_loss_type",'
  '    "property_type:="  , "ChoiceProperty",'
  '    "Choice:="    , "Electrical Steel"'
  '  ],'
  '  "core_loss_kh:="  , "%f",'
  '  "core_loss_kc:="  , "%f",'
  '  "core_loss_ke:="  , "%f",'
  '  "core_loss_kdc:="  , "0",'
  '  "mass_density:="  , "%f",'
  '  "core_loss_equiv_cut_depth:=", "0.001meter"'
  '])'

};
```

draw_motor_in_ansys.m

```
430 str=sprintf('%s\n',rotor{:});
431 fprintf(pyfile,str,Kh,Ke,Kadd,Rotor_kgm3);
432 clear rotor
433 end
434 %%%%%%%%%%%%%%%%%%% Stator's plates %%%%%%%%%%%%%%
435 if strcmp(Stator_MatName,Rotor_MatName)==0 &&...
436         strcmp(Stator_MatName,Shaft_MatName)==0
437 [Kh,Kadd,Ke]=losscoefficients(Stator_alpha,Stator_beta,...
438                         Stator_kh,Stator_ke,Stator_kgm3);
439 stator={
440     '############ Stator"s plates ############'
441     'oDefinitionManager.AddMaterial('
442   '['
443   '  "NAME:%s",'
444   '  "CoordinateSystemType:=", "Cartesian",'
445   '  "BulkOrSurfaceType:="  , 1,'
446   '  ['
447   '    "NAME:PhysicsTypes",'
448   '    "set:="      , ["Electromagnetic"]'
449   '  ],'
450   '  ['
451   '    "NAME:permeability",'
452   '    "property_type:="  , "nonlinear",'
453   '    "BTypeForSingleCurve:="  , "normal",'
454   '    "HUnit:="    , "A_per_meter",'
455   '    "BUnit:="    , "tesla",'
456   '    "IsTemperatureDependent:=", False,'
457   '    ['
```

draw_motor_in_ansys.m

```
458   '      "NAME:BHCoordinates",'
459   '      ['
460   '        "NAME:DimUnits", '
461   '        "", '
462   '        ""'
463   '      ],'
464     };
465 str=sprintf('%s\n',stator{:});
466 fprintf(pyfile,str,Stator_MatName);
467 clear stator
468 for ii=1:length(Stator_BH(:,1))
469 stator={
470   '      ["NAME:Coordinate",["NAME:CoordPoint",%f,%f]],'
471     };
472 str=sprintf('%s\n',stator{:});
473 fprintf(pyfile,str,Stator_BH(ii,2),Stator_BH(ii,1));
474 clear stator
475 end
476     clear stator
477     stator={
478   '    ],'
479   '    ['
480   '      "NAME:Temperatures"'
481   '    ]'
482   '  ],'
483   '  ['
484   '    "NAME:magnetic_coercivity",'
485   '    "property_type:="   , "VectorProperty",'
```

draw_motor_in_ansys.m

```
    '    "Magnitude:="    , "0A_per_meter",'
    '    "DirComp1:="     , "1",'
    '    "DirComp2:="     , "0",'
    '    "DirComp3:="     , "0"'
    '  ],'
    '  ['
    '    "NAME:core_loss_type",'
    '    "property_type:="  , "ChoiceProperty",'
    '    "Choice:="     , "Electrical Steel"'
    '  ],'
    '  "core_loss_kh:="  , "%f",'
    '  "core_loss_kc:="  , "%f",'
    '  "core_loss_ke:="  , "%f",'
    '  "core_loss_kdc:="  , "0",'
    '  "mass_density:="  , "%f",'
    '  "core_loss_equiv_cut_depth:=", "0.001meter"'
    '])'

};

str=sprintf('%s\n',stator{:});
fprintf(pyfile,str,Kh,Ke,Kadd,Stator_kgm3);
clear stator
end


runpy={'sys.exit()'};
str=sprintf('\n%s\n',runpy{:});
```

draw_motor_in_ansys.m

```
514 fprintf(pyfile,str,currentFolder);
515 clear runpy
516
517 fclose(pyfile);
518 clear pyfile
519
520 %from txt to py
521 movefile('setup_draw_motor_in_ansys.txt',...
522         'setup_draw_motor_in_ansys.py','f');
523
524 %% Create closing project script
525 fclose("all");
526
527 pyfile= fopen('close_projectAM.txt', 'wt');
528
529 closer = {
530     'import sys'
531     'sys.path.append(r"C:/Program Files/AnsysEM/AnsysEM20.1/Win64")'
532     'sys.path.append(r"C:/Program
    ↪  Files/AnsysEM/AnsysEM20.1/Win64/PythonFiles/DesktopPlugin")'
533     'import ScriptEnv'
534     'ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")'
535     'oDesktop.RestoreWindow()'
536     'oDesktop.CloseProject("%s")'
537     };
538
539 str=sprintf('%s\n',closer{:});
540 fprintf(pyfile,str,erase(filename,".mat"));
```

draw_motor_in_ansys.m

```
541 clear closer
542
543 fclose(pyfile);
544
545 movefile('close_projectAM.txt','close_projectAM.py','f');
546
547 %Ansys start e py run
548 command=strcat(ipypath,strrep(currentFolder,'/','\'),...
549                     '\setup_draw_motor_in_ansys.py"');
550
551 [status,cmdout] = dos(command);
552
553 if status ~= 0
554     disp('an error occurred - program aborted - \n %s',cmdout)
555 end
556 command=strcat(ipypath,strrep(currentFolder,'/','\'),...
557                         '\draw_motor_in_ansys.py"');
558 [status,cmdout] = dos(command);
559
560 % if status ~= 0
561 %     disp('an error occurred - program aborted - \n %s',cmdout)
562 % end
```

eval_operatingPointAM.m

```
function eval_operatingPointAM(geo,per,mat,filepath,filename,ipypath) %dataIn from GUI

%Manual input Simulation Data
eval_type = 'singt'; %-singt  -coreloss
per.gamma = 55; %idq angle  %dataSet.GammaPP;
per.CurrLoPP = 1; %Current [pu] %dataSet.CurrLoPP;
per.EvalSpeed = 1800; %Evaluation Speed [rpm]
per.nsim_singt = 10;%Simulation Points %dataSet.NumOfRotPosPP
per.delta_sim_singt = 60;%Angular Excursion %dataSet.AngularSpanPP
per.tempPP = 80; %permanent magnets temperature
per.corelossflag = strcmp(eval_type,'coreloss');
per.save_fields = 0; %activate and save fields every # step (0 deactiveted) must be < nsim_singt


[~,~,out,~] = AMfitness(geo,per,mat,eval_type,filepath,filename,ipypath);

%output data
%Theta=out.th;
T = out.T;
fd = out.fd;
fq = out.fq;
id = out.id;
iq = out.iq;
IPF = out.IPF;
```

eval_operatingPointAM.m

```
25 xdeg = out.xdeg;
26
27 if per.corelossflag==1
28    CoreLoss=out.CoreLoss;
29    CoreLossAvg = out.CoreLossAvg;
30    HysteresisLossAvg = out.HysteresisLossAvg;
31    EddyLossAvg = out.EddyLossAvg;
32    ExcessLossAvg = out.ExcessLossAvg;
33    per.savefield = 0;
34 end
35
36 %repeat the 60° results for an entire round of 360°
37 if xdeg < 360
38     nrep=360/xdeg;
39 else
40     nrep=1;
41 end
42
43 T=repmat(T,nrep,1);
44 fd=repmat(fd',nrep,1);
45 fq=repmat(fq',nrep,1);
46 id=repmat(id',nrep,1);
47 iq=repmat(iq',nrep,1);
48 IPF=repmat(IPF',nrep,1);
49 Theta=linspace(0,nrep*xdeg,length(T));
50
51
52 % plots
```

eval_operatingPointAM.m

```
53 FontSize = 12;
54 FontName = 'TimesNewRoman';
55
56 %Fluxes
57 hdq=figure();
58 subplot(2,1,1);
59 hd1=plot(Theta,fd);
60 title(['Mean \lambda_d = ' num2str(mean(fd))]);
61 grid on
62 set(hd1,'LineWidth',2);
63 xl_hd=xlabel('\theta [degrees]');
64 set(xl_hd,'Rotation',0,'FontName',FontName,'Fontsize',FontSize);
   ↪ %,'FontWeight','Bold');
65 yl_hd=ylabel('\lambda_d [Wb]');
66 set(yl_hd,'Rotation',90,'FontName',FontName,'Fontsize',FontSize);
   ↪ %,'FontWeight','Bold');
67 xlim([0 Theta(end)]);
68 set(gca,'FontSize',FontSize), %,'FontWeight','Bold');
69 ti = 0:60:xdeg*nrep;  set(gca,'XTick',ti);
70
71 hq = subplot(2,1,2);
72 hq1 = plot(Theta,fq);
73 title(['Mean \lambda_q = ' num2str(mean(fq))]);
74 grid on
75 set(hq1,'LineWidth',2);
76 xl_hq=xlabel('\theta [degrees]');
77 set(xl_hq,'Rotation',0,'FontName',FontName,'Fontsize',FontSize);
   ↪ %'FontWeight','Bold');
```

eval_operatingPointAM.m

```
78  yl_hq=ylabel('\lambda_q [Wb]');
79  set(yl_hq,'Rotation',90,'FontName',FontName,'Fontsize',FontSize);
    ↪  %,'FontWeight','Bold');
80  xlim([0 Theta(end)]);
81  set(gca,'FontSize',FontSize); %,'FontWeight','Bold');
82  ti = 0:60:xdeg*nrep;  set(gca,'XTick',ti);
83
84  %Torque
85  figure()
86  subplot(2,1,1)
87  pl = plot(Theta,abs(T)); grid on
88  title(['Mean Torque = ' num2str(mean(T))]);
89  set(pl,'LineWidth',[2]);
90  xlim([0 Theta(end)]), %ylim([ymin ymax]),
91  set(gca,'FontName',FontName,'FontSize',FontSize);
92  ti = 0:60:360; set(gca,'XTick',ti);
93  xl = xlabel('\theta - degrees');
    ↪  set(xl,'Rotation',[0],'Fontsize',FontSize);
94  yl = ylabel('T[Nm]');
95  set(yl,'Rotation',[90],'FontName',FontName,'Fontsize',FontSize);
96
97  %IPF
98  subplot(2,1,2)
99  pl = plot(Theta,IPF);
100 title(['Mean IPF = ' num2str(mean(IPF))]);
101 grid on
102 set(pl,'LineWidth',2);
```

eval_operatingPointAM.m

```
103 xl = xlabel('\theta - [degrees]');
    ↪  set(xl,'Rotation',0,'FontName',FontName,'Fontsize',FontSize);
104 yl=ylabel('IPF');
105 set(yl,'Rotation',90,'FontName',FontName,'Fontsize',FontSize);
106 xlim([0 Theta(end)]);
107 set(gca,'FontName',FontName,'FontSize',FontSize);
108 ti = 0:60:xdeg*nrep;  set(gca,'XTick',ti);
109
110
111 %CoreLoss
112 if per.corelossflag==1
113     figure()
114     pl = plot(Theta,CoreLoss); grid on
115     title(['Mean Core Loss at S.S.= '
        ↪  num2str(mean(CoreLoss(ceil(6*end/7)+1:end)))]);
116     set(pl,'LineWidth',[2]);
117     xlim([0 Theta(end)]), %ylim([ymin ymax]),
118     set(gca,'FontName',FontName,'FontSize',FontSize);
119     ti = 0:60:xdeg*nrep; set(gca,'XTick',ti);
120     xl = xlabel('\theta - [degrees]');
        ↪  set(xl,'Rotation',[0],'Fontsize',FontSize);
121     yl = ylabel('Core Loss [W]');
122     set(yl,'Rotation',[90],'FontName',FontName,'Fontsize',FontSize);
123
124    Pfe_tot = CoreLossAvg(1); Pfes_tot = CoreLossAvg(2); Pfer_tot =
       ↪  CoreLossAvg(3);
125    Pfe_h = HysteresisLossAvg(1); Pfes_h = HysteresisLossAvg(2);
       ↪  Pfer_h = HysteresisLossAvg(3);
```

eval_operatingPointAM.m

```
126     Pfe_c = EddyLossAvg(1); Pfes_c = EddyLossAvg(2); Pfer_c =
        ↪ EddyLossAvg(3);
127     Pfe_ex = ExcessLossAvg(1); Pfes_ex = ExcessLossAvg(2); Pfer_ex =
        ↪ ExcessLossAvg(3);
128
129     c = categorical({'$Stat\,Hys$','$Stat\,Eddy$','$Stat\,Exc$',...
130     '$Rot\,Hys$','$Rot\,Eddy$','$Rot\,Exc$','Total
        ↪ Loss'},'Ordinal',true);
131     Pfe = [Pfes_h,Pfes_c,Pfes_ex,Pfer_h,Pfer_c,Pfer_ex,Pfe_tot];
132
133      % Plot
134      figure()
135      figSetting(15,15,8)
136      set(gca,'YLim',[0 Pfe_tot*1.1])
137      b = bar(c,Pfe,0.8);
138      xtips1 = (b(1).XEndPoints);
139      ytips1 = (b(1).YEndPoints);
140      labels1 = string(round(b(1).YData));
141      text(xtips1,ytips1,labels1,'HorizontalAlignment','center',...
142      'VerticalAlignment','bottom')
143      legend ('Ansys')
144      title('Iron Loss')
145      ylabel('P [W]')
146
147
148 end
149
150 end
```

AMfitness.m

```
function [geo,mat,out,filepath] =
↪ AMfitness(geo,per,mat,eval_type,filepath,filename,ipypath)

[SOL]=simulate_xdeg_AM(geo,per,mat,eval_type,filepath,filename,ipypath);

th0=geo.th0;
%from cls files we obtain column vectors
T=SOL.T{:,2}; %[t(ms),T(Nm)]
F=SOL.F{:,2:end}; %[t(ms),f1(Wb),f2(Wb),f3(Wb)]
Theta=geo.p*SOL.Theta{:,2}; %[t(ms),Theta(deg_e)]

I=SOL.I{:,2:end}; %[t(ms),i1(A),i2(A),i3(A)]
V=SOL.V{:,2:end}; %[t(ms),v1(V),v2(V),v3(V)]

[id,iq] = abc2dq_AM(I(:,1)',I(:,2)',I(:,3)',Theta',th0);
[fd,fq] = abc2dq_AM(F(:,1)',F(:,2)',F(:,3)',Theta',th0);
IPF = sin(atan(iq./id)-atan(fq./fd));

%output function's value
out.th=Theta;
out.T=T;
out.fd=fd;
out.fq=fq;
out.id=id;
out.iq=iq;
out.IPF=IPF;
out.xdeg=SOL.xdeg;

if per.corelossflag==1
```

AMfitness.m

```
29   CoreLoss=SOL.CoreLoss{:,2}; %[t(ms),CoreLoss(W)]
30   out.CoreLoss=CoreLoss;
31
32   out.CoreLossAvg = SOL.CoreLossAvg;
33   out.HysteresisLossAvg = SOL.HysteresisLossAvg ;
34   out.EddyLossAvg = SOL.EddyLossAvg;
35   out.ExcessLossAvg = SOL.ExcessLossAvg ;
36  end
37
38  end
```

simulate_xdeg_AM.m

```
function [SOL] =
↪ simulate_xdeg_AM(geo,per,mat,eval_type,filepath,filename,ipypath)

%Solution Type
switch eval_type
    case 'singt'
        nsim = per.nsim_singt; %Simulation Points
        xdeg = per.delta_sim_singt; %Angular Excursion
        gamma = per.gamma;  %idq angle
    case 'coreloss'
        nsim = per.nsim_singt; %Simulation Points
        xdeg = 420; %Angular Excursion
        gamma = per.gamma;  %idq angle
    otherwise
        error('Ansys Maxwell Simulations not available for the
        ↪ selected evaluation type!');
end

th0 = geo.th0(1); %initial angle between CS dq and abc
p   = geo.p;
corelossflag = per.corelossflag;
% ns  = geo.ns;
N_parallel = 1;
N_cond = geo.win.Nbob; %Ncond per slot
↪ geo.win.Ns/geo.p/(geo.q)/size(geo.win.avv,1);
q = geo.q;
gamma_ = gamma*pi/180; %current angle dq [rad]
win_avv = geo.win.avv; %layer win
```

simulate_xdeg_AM.m

```
26 nlay = geo.nlay; %layaer of rotor barriers
27 radial_ribs_split = nnz(geo.radial_ribs_split*geo.pontR'); %half
   ↪ added barrier area using split
28 n_PM = geo.n_PM;%number of PM's
29 tempPP = per.tempPP; %temperature of PMs
30 CurrLoPP = per.CurrLoPP;  %Current [pu]
31
32 if per.EvalSpeed==0
33     prompt={'The Evaluation Speed is 0! Please insert a different
       ↪ value in [rpm]:'};
34     data=inputdlg(prompt,'EvalSpeed',[1 35],{'1500'});
35     EvalSpeed=str2double(data{1});
36 else
37     EvalSpeed=per.EvalSpeed;
38 end
39
40
41 % evaluation of the phase current values for all positions to be
   ↪ simulated
42 iAmp = CurrLoPP*per.i0; %Peak Current
43
44 id = iAmp * cos(gamma_);
45 iq = iAmp * sin(gamma_);
46 %Imod = abs(id + 1i* iq);        % Modulo corrente (A)
47
48 phi_init=th0*pi/180+gamma_; %initial current phase angle
49
50 %export simulation data on python
```

simulate_xdeg_AM.m

```
51
52 pyfile = fopen( strcat(pwd,'\temp\simdata.txt'), 'wt' );
53 export={
54 'import pickle'
55 'import sys'
56 'simdata={'
57     '"filename":"%s", #motfilename'
58     '"filepath":"%s", #motfilepath'
59     '"iAmp":%f, #Currents Amplitude'
60     '%s #matrix with windings index'
61     '"N_parallel" :%d,   #'
62     '"N_cond" :%d,   #N cond per slot'
63     '"EvalSpeed" :%f,   #layer of rotor barrier'
64     '"phi_init":%f, #initial current phase'
65     '"p":%d,#pole pair'
66     '"q":%d,#slot/(phase*p)'
67     '"radial_ribs_split":%d, #area added by split barrier'
68     '"n_PM":%d, #number of permanent magnet (number of magnetic
       ↪  segments)'
69     '"nlay" :%d,   #layer of rotor barrier'
70     '"tempPP" :%f, #PMs temp'
71     '"gamma":%f,#current angle dq [rad]'
72     '"nsim":%d,#simulation points'
73     '"xdeg":%f,#angular excursion [deg]'
74     '"corelossflag":%d,#flag that activate core loss'
75     '"save_fields":%d,#save fields every # steps'
76     '}'
77     'with open("%s/temp/temp.pkl", "wb") as export:'
```

simulate_xdeg_AM.m

```
    '    pickle.dump(simdata,export,protocol=2)'
    '###'
    'sys.exit()'
    };
str=sprintf('%s\n',export{:});

win_avvpy='"win_avv":[';
for ii=1:size(win_avv,1)
    win_avvpy=strcat(win_avvpy,'[');
    for jj=1:size(win_avv,2)
        win_avvpy=strcat(win_avvpy,num2str(win_avv(ii,jj)),',');
    end
    win_avvpy=strcat(win_avvpy,'],');
end
win_avvpy=strcat(win_avvpy,'],');

fprintf(pyfile,str,filename,filepath,iAmp,win_avvpy,N_parallel,...
        N_cond,EvalSpeed,phi_init,p,q,radial_ribs_split,n_PM,...
        nlay,tempPP,gamma_,nsim,xdeg,per.corelossflag,...
        per.save_fields,strrep(pwd,'\','/'));

fclose(pyfile);



%from txt to py

movefile(strcat(pwd,'\temp\simdata.txt'),...
```

simulate_xdeg_AM.m

```
106 strcat(pwd,'\temp\simdata.py'),'f');
107
108 command=strcat(ipypath,pwd,'\temp\simdata.py"');
109
110 [status,cmdout] = dos(command);
111
112 if status ~= 0
113     disp('an error occurred - programm aborted - \n %s',cmdout)
114 end
115
116 [status,cmdout] = dos(command);
117
118 if status ~= 0
119     disp('an error occurred - programm aborted - \n %s',cmdout)
120 end
121
122 outdatafolder=strcat(filepath,filename(1:end-4),'_results\');
123
124 addpath(outdatafolder);
125 opts = detectImportOptions('FluxesPlotData.csv','NumHeaderLines',1);
126 opts.VariableNamesLine=1;
127 opts.PreserveVariableNames=1;
128
129 T = readtable('TorquePlotData.csv',opts); %[t(ms),T(Nm)]
130 F = readtable('FluxesPlotData.csv',opts);
    ↪ %[t(ms),f1(Wb),f2(Wb),f3(Wb)]
131 Theta = readtable('PositionData.csv',opts); %[t(ms),Theta(deg)]
```

simulate_xdeg_AM.m

```
132 I = readtable('CurrentsPlotData.csv',opts);
    ↪ %[t(ms),i1(A),i2(A),i3(A)]
133 V = readtable('VoltagesPlotData',opts); %[t(ms),v1(V),v2(V),v3(V)]
134
135
136
137 SOL.T=T;
138 SOL.F=F;
139 SOL.Theta=Theta;
140 SOL.I=I;
141 SOL.V=V;
142 SOL.xdeg=xdeg;
143
144
145 if per.corelossflag==1
146    CoreLoss =  readtable('CoreLossData.csv',opts);
       ↪ %[t(ms),CoreLoss(W)]
147    SOL.CoreLoss=CoreLoss;
148    opts = detectImportOptions('CoreLossAvg.csv','NumHeaderLines',1);
149    opts.VariableNamesLine=1;
150    opts.PreserveVariableNames=1;
151
152    CoreLossAvg = xlsread('CoreLossAvg.csv'); %[Tot;Stat;Rot]
153    HysteresisLossAvg = xlsread('HysteresisLossAvg.csv');
       ↪ %[Tot;Stat;Rot]
154    EddyLossAvg = xlsread('EddyCurrentsLossAvg.csv');
       ↪ %[Tot;Stat;Rot]
155    ExcessLossAvg = xlsread('ExcessLossAvg.csv'); %[Tot;Stat;Rot]
```

simulate_xdeg_AM.m

```
156
157     SOL.CoreLossAvg = CoreLossAvg;
158     SOL.HysteresisLossAvg = HysteresisLossAvg;
159     SOL.EddyLossAvg = EddyLossAvg;
160     SOL.ExcessLossAvg = ExcessLossAvg;
161
162 end
163 end
```

losscoefficients.m

```
function [Kh,Kadd,Ke]=losscoefficients(alpha,beta,kh,ke,density)
warning off;
%[W/kg]->%[m/m^3]
kh=kh*density;
Ke=ke*density;
B = [0:0.1:2];
f=round(logspace(log10(50),4,25));

%perdite isteresi calcolate con modello Syr-e
LossH= kh*f'.^alpha.*B.^beta;

for i=4:length(LossH(:,1))
    LossH(i,end-(i-9):end)=NaN(1,i-8);
end


[xData, yData, zData] = prepareSurfaceData( f,B, LossH);
ft = fittype( 'kadd*(x*y)^1.5+kh*x*y^2', 'independent', {'x', 'y'},
 ↪  'dependent', 'z' );

opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';opts.Lower = [0 0];opts.Robust =
 ↪  'LAR';opts.StartPoint = [0,kh];

fitresult = fit( [xData, yData], zData, ft, opts );

coeff = coeffvalues(fitresult);
Kadd=coeff(1);
Kh=coeff(2);
```

losscoefficients.m

```
28
29 % %PLOT RESULT
30 % figure
31 % hold on
32 % for i=1:length(f)
33 %     LossHSyre= kh*f(i).^alpha*B.^beta;
34 %     LossHAnsys=Kh*f(i)*B.^2+Kadd*(f(i)*B).^1.5;
35 %     if i>3
36 %         LossHAnsys(end-(i-9):end)=NaN(1,i-8);
37 %         LossHSyre(end-(i-9):end)=NaN(1,i-8);
38 %     end
39 %     plot(B,LossHSyre,'rx',B,LossHAnsys,'bo')
40 % end
41 % xlabel('B[T]');
42 % ylabel('Losses[W/m^3]');
43 % tit=sprintf('Loss models: "x" - Syre ; "o" - Ansys\n
   ↪  f:[%s]',num2str(f));
44 % title(tit);
45 % legend('Syr-e','Ansys')
46 % hold off
47
48 warning on;
49 end
```

MagHc_temp_fitting.m

```
function [c1,Href] = MagHc_temp_fitting(mat)
temp=mat.LayerMag.temp.temp';
BrT=mat.LayerMag.temp.Br';
mur=mat.LayerMag.mu;
HcT=BrT/(mur*4*pi*10^-7);
k=temp==20;
Br20=BrT(k);
Href = HcT(k);
c1=(BrT(end)/Br20-1)/(temp(end)-20);


% figure()
% plot(temp,HcT,'rx',temp,Href*(1+c1*(temp-20)))
end

% ft = fittype(@(c1,temp)
↪  (Href*(1+c1*(temp-20)+c2(temp-20)^2)),'independent',
↪  {'temp'});
% fo = fitoptions( 'Method','NonlinearLeastSquares',
↪  'StartPoint',[0 0]);
% fitresult=fit(temp,HcT,ft,fo);
%
% coeff = coeffvalues(fitresult);
% c1=coeff(1);
% c2=coeff(2);
```

B Python Codes

Example of "draw_motor_in_ansys.py" generated by fprintf

```
import sys
import pickle
import os
currentFolder = os.path.dirname(os.path.realpath(__file__))
sys.path.append(r"C:\\Program Files\\AnsysEM\\AnsysEM20.1\\Win64")
sys.path.append(r"C:/Program Files/AnsysEM/AnsysEM20.1/Win64/PythonFiles/DesktopPlugin")
import math

################ export data from mat file (through pickle) #################
filepath = r'%s/temp/temp.pkl'%(currentFolder)
with open(filepath,'rb') as exportdata:
    geo,material = pickle.load(exportdata)

import ScriptEnv
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.SetActiveProject("%s"%(geo["filename"][:-4]))
oDesign = oProject.SetActiveDesign("Maxwell2DDesign1")
oEditor = oDesign.SetActiveEditor("3D Modeler")



################ import dxf by file directory #############
oDesktop.AddMessage ("%s"%(geo["filename"][:-4]), "Maxwell2DDesign1", 0,
"%s%s"%(geo["filepath"],geo["filename"][:-4]), "")
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
25 oEditor.ImportDXF(
26   [
27     "NAME:options",
28     "FileName:="     , "%s%s.dxf"
       ↪  %(geo["filepath"],geo["filename"][:-4]),
29     "Scale:="    , 0.001,
30     "AutoDetectClosed:="   , True,
31     "SelfStitch:="     , True,
32     "DefeatureGeometry:=" , True,
33     "DefeatureDistance:=" , 0.0001,
34     "RoundCoordinates:="   , False,
35     "RoundNumDigits:="   , 4,
36     "SelfStitchTolerance:=" , 0.0001,
37     "WritePolyWithWidthAsFilledPoly:=", True,
38     "ImportMethod:="   , 1,
39     "2DSheetBodies:=" , True,
40     [
41       "NAME:LayerInfo",
42       [
43         "NAME:0",
44         "source:="     , "0",
45         "display_source:="   , "0",
46         "import:="     , True,
47         "dest:="     , "0",
48         "dest_selected:=" , False,
49         "layer_type:="     , "signal"
50       ],
51       [
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
52          "NAME:1",
53          "source:="      , "1",
54          "display_source:="  , "1",
55          "import:="      , True,
56          "dest:="        , "1",
57          "dest_selected:=" , False,
58          "layer_type:="      , "signal"
59        ],
60        [
61          "NAME:2",
62          "source:="      , "2",
63          "display_source:="  , "2",
64          "import:="      , True,
65          "dest:="        , "2",
66          "dest_selected:=" , False,
67          "layer_type:="      , "signal"
68        ]
69      ]
70    ])
71
72  oEditor.FitAll()
73  ######################## detach slot  ################
74  ii=0
75  slotconds=""
76  lineslotairgaps=""
77  slotairgaps=""
78  statplate="1_1"
79
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
for ii in range(geo["q"]*3):

  slotindex=2+ii*4
  lineindex=slotindex+1
  slot="1_%d" %(slotindex)
  line="1_%d"%(lineindex)
  lineairgap="1_%d,1_%d"%(lineindex+1,lineindex+2)
  slotconds = slotconds + "," + slot + "," + slot + "_Detach1"
  ↪  ##list slot/cond, useful for detachment
  lineslotairgaps=lineslotairgaps+","+lineairgap
  slotairgaps=slotairgaps+","+statplate+"_Detach"+"%d"%(ii+1)


  ########## Slot
  oEditor.Imprint(
    [
      "NAME:Selections",
      "Blank Parts:="    , slot,
      "Tool Parts:="     , line
    ],
    [
      "NAME:ImprintParameters",
      "KeepOriginals:=" , False
    ])

  facesIDs=oEditor.GetFaceIDs(slot)

  faceID=int(facesIDs[1])
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
107   oEditor.DetachFaces(
108     [
109       "NAME:Selections",
110       "Selections:="    , slot,
111       "NewPartsModelFlag:=" , "Model"
112     ],
113     [
114       "NAME:Parameters",
115       [
116         "NAME:DetachFacesToParameters",
117         "FacesToDetach:=" , [faceID]
118       ]
119     ])
120
121 lineslotairgaps=lineslotairgaps[1:]
122
123 slotconds=slotconds[1:] #removing initial comma of the string
124 slotconds_array=slotconds.split(',') #vector containig the elements
    ↪  of the string separated by the comma
125
126 slotairgaps=slotairgaps[1:]
127 slotairgaps_array=slotairgaps.split(',')
128
129 ############ detach slot from stator's iron ###############
130 oEditor.Subtract(
131   [
132     "NAME:Selections",
133     "Blank Parts:="   , statplate,
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
134     "Tool Parts:="    , slotconds
135   ],
136   [
137     "NAME:SubtractParameters",
138     "KeepOriginals:=" , True
139   ])
140 ############ detach slot airgap from stator's iron
141
142
143 oEditor.Imprint(
144   [
145     "NAME:Selections",
146     "Blank Parts:="   , statplate,
147     "Tool Parts:="    , lineslotairgaps
148   ],
149   [
150     "NAME:ImprintParameters",
151     "KeepOriginals:=" , False
152   ])
153
154 facesIDs=oEditor.GetFaceIDs(statplate)
155 facesIDs=facesIDs[:-1] #remove last term
156 facesIDs=map(int,facesIDs) #array of string -> integer array
157
158 oEditor.DetachFaces(
159   [
160     "NAME:Selections",
161     "Selections:="    , statplate,
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
162       "NewPartsModelFlag:=" , "Model"
163     ],
164     [
165       "NAME:Parameters",
166       [
167         "NAME:DetachFacesToParameters",
168         "FacesToDetach:=" , facesIDs
169       ]
170     ])
171
172
173
174     ############ Subract barrier from rotor's iron ###############
175   barriers=""
176   PMs=""
177   temp=0
178   for jj in range (2):
179     for ii in range(geo["nlay"]+geo["radial_ribs_split"]):
180       rotorindex=(3+5*geo["q"]*3)*(1-jj)+ii+temp*jj
181       barrier="2_%d" %(rotorindex)
182       barriers=barriers+","+barrier
183
184
185
186     for kk in range(int(geo["n_PM"]/2)):
187       rotorindex=rotorindex+1
188       PM="2_%d" %(rotorindex)
189       PMs=PMs+","+PM
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
190
191   temp=rotorindex+1
192 #oDesktop.AddMessage ("%s"%(geo["filename"][:-4]),
    ↪  "Maxwell2DDesign1", 0, "%s"%(barriers), "")
193 barriers=barriers[1:]
194 barriers_array = barriers.split(",")
195 PMs=PMs[1:]
196 PMs_array=PMs.split(',')
197
198 rotorplate="2_%d" %(temp)
199 shaft="2_%d" %(temp+1)
200
201 oEditor.Subtract(
202   [
203     "NAME:Selections",
204     "Blank Parts:="    , rotorplate,
205     "Tool Parts:="     , barriers
206   ],
207   [
208     "NAME:SubtractParameters",
209     "KeepOriginals:=" , False
210   ])
211
212 if geo["n_PM"]!=0:
213   oEditor.Subtract(
214     [
215       "NAME:Selections",
216       "Blank Parts:="    , rotorplate,
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
217       "Tool Parts:="     , PMs
218     ],
219     [
220       "NAME:SubtractParameters",
221       "KeepOriginals:=" , True
222     ])
223
224 ################### Material Assignment ####################
225 ###Rotor
226 oEditor.ChangeProperty(
227   [
228     "NAME:AllTabs",
229     [
230       "NAME:Geometry3DAttributeTab",
231       [
232         "NAME:PropServers",rotorplate
233       ],
234       [
235         "NAME:ChangedProps",
236         [
237           "NAME:Solve Inside","Value:="    , True
238         ],
239         [
240           "NAME:Color","R:=", 140,"G:=", 160, "B:=", 175
241         ],
242         [
243           "NAME:Material",
244           "Value:="    , "\"%s\"" %(material["rotor"])
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
245         ]
246       ]
247     ]
248   ])
249
250
251
252 ###Stator
253 oEditor.ChangeProperty(
254   [
255     "NAME:AllTabs",
256     [
257       "NAME:Geometry3DAttributeTab",
258       [
259         "NAME:PropServers",statplate
260       ],
261       [
262         "NAME:ChangedProps",
263         [
264           "NAME:Solve Inside","Value:="    , True
265         ],
266         [
267           "NAME:Color","R:=", 140,"G:=", 160, "B:=", 175
268         ],
269         [
270           "NAME:Material",
271           "Value:="    , "\"%s\"" %(material["stator"])
272         ]
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
273         ]
274       ]
275     ])
276
277   ###Shaft
278   oEditor.ChangeProperty(
279     [
280       "NAME:AllTabs",
281       [
282         "NAME:Geometry3DAttributeTab",
283         [
284           "NAME:PropServers",shaft
285         ],
286         [
287           "NAME:ChangedProps",
288           [
289             "NAME:Solve Inside","Value:="    , True
290           ],
291           [
292             "NAME:Color","R:=", 140,"G:=", 160, "B:=", 175
293           ],
294           [
295             "NAME:Material",
296             "Value:="    , "\"%s\"" %(material["shaft"])
297           ]
298         ]
299       ]
300     ])
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
301
302 oEditor.SetWCS(["NAME:SetWCS Parameter","Working Coordinate
    ↪  System:=", "Global","RegionDepCSOk:=", False])
303 ###Magnet
304 if geo["n_PM"]!=0:
305
306   for ii in range(len(PMs_array)):
307     oEditor.CreateRelativeCS(
308     [
309       "NAME:RelativeCSParameters",
310       "Mode:="    , "Axis/Position",
311       "OriginX:="   , "%smm"%(geo["PM_CS"][0][ii]),
312       "OriginY:="   , "%smm"%(geo["PM_CS"][1][ii]),
313       "OriginZ:="   , "0mm",
314       "XAxisXvec:="   , "%smm"%(geo["PM_CS"][2][ii]),
315       "XAxisYvec:="   , "%smm"%(geo["PM_CS"][3][ii]),
316       "XAxisZvec:="   , "0mm",
317       "YAxisXvec:="   , "%smm"%(-geo["PM_CS"][3][ii]),
318       "YAxisYvec:="   , "%smm"%(geo["PM_CS"][2][ii]),
319       "YAxisZvec:="   , "0mm"
320     ],
321     [
322       "NAME:Attributes",
323       "Name:="    , "RelativeCS%s"%(PMs_array[ii])
324     ])
325
326     oEditor.ChangeProperty(
327       [
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
328         "NAME:AllTabs",
329         [
330           "NAME:Geometry3DAttributeTab",
331           [
332             "NAME:PropServers",PMs_array[ii]
333           ],
334           [
335             "NAME:ChangedProps",
336             [
337               "NAME:Solve Inside","Value:="    , True
338             ],
339             [
340               "NAME:Color","R:=", 50,"G:=", 50, "B:=", 50
341             ],
342             [
343               "NAME:Material",
344               "Value:="    , "\"%s\"" %(material["magnet"])
345             ],
346             [
347               "NAME:Orientation",
348               "Value:="    , "RelativeCS%s"%(PMs_array[ii])
349             ]
350           ]
351         ]
352       ])
353     oEditor.SetWCS(["NAME:SetWCS Parameter","Working Coordinate
      ↪  System:=", "Global","RegionDepCSOk:=", False])
354 ###Conductor
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
for ii in range(len(slotconds_array)): ##3 fasi
  oEditor.ChangeProperty(
    [
      "NAME:AllTabs",
      [
        "NAME:Geometry3DAttributeTab",
        [
          "NAME:PropServers",slotconds_array[ii]
        ],
        [
          "NAME:ChangedProps",
          [
            "NAME:Solve Inside","Value:="    , True
          ],
          [
            "NAME:Color","R:=", 220,"G:=", 165, "B:=", 30
          ],
          [
            "NAME:Material",
            "Value:="    , "\"%s\"" %(material["slotcond"])
          ]
        ]
      ]
  ])

# barrier removed because air background, programm more stable
# ###Aria: Barriere di flusso
# for ii in range(len(barriers_array)):
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
#    oEditor.ChangeProperty(
#      [
#        "NAME:AllTabs",
#        [
#          "NAME:Geometry3DAttributeTab",
#          [
#            "NAME:PropServers",barriers_array[ii]
#          ],
#          [
#            "NAME:ChangedProps",
#            [
#              "NAME:Solve Inside","Value:="    , True
#            ],
#            [
#              "NAME:Color","R:=", 80,"G:=", 150,  "B:=", 250
#            ],
#            [
#              "NAME:Material",
#              "Value:="    , "\"Air\""
#            ]
#          ]
#        ]
#      ])

###Aria: Traferro di cava
for ii in range(len(slotairgaps_array)):
  oEditor.ChangeProperty(
    [
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
411       "NAME:AllTabs",
412       [
413         "NAME:Geometry3DAttributeTab",
414         [
415           "NAME:PropServers",slotairgaps_array[ii]
416         ],
417         [
418           "NAME:ChangedProps",
419           [
420             "NAME:Solve Inside","Value:="    , True
421           ],
422           [
423             "NAME:Color","R:=", 80,"G:=", 150,  "B:=", 250
424           ],
425           [
426             "NAME:Material",
427             "Value:="    , "\"Air\""
428           ]
429         ]
430       ]
431     ])
432
433 ##clen warning due to solve inside
434 oDesktop.ClearMessages("", "",2)
435
436 ############################## Boundaries Geometry
    ↪  ##############################
437 #Circular Region Conteining Motor
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
438 oEditor.CreateCircle(
439   [
440     "NAME:CircleParameters","IsCovered:=", True,"XCenter:=",
      ↪  "0mm","YCenter:=", "0mm","ZCenter:=", "0mm","Radius:=",
      ↪  "%smm"%(geo["R"]),"WhichAxis:=", "Z","NumSegments:=", "0"
441   ],
442   [
443     "NAME:Attributes","Name:=", "Region","Flags:=", "", "Color:=",
      ↪  "(70 180 250)","Transparency:="   ,
      ↪  0.85,"PartCoordinateSystem:=","Global","UDMId:=",
      ↪  "","MaterialValue:=", "\"Air\"",
444     "SurfaceMaterialValue:=", "\"\"","SolveInside:=",
      ↪  True, "IsMaterialEditable:=",
      ↪  True,"UseMaterialAppearance:=", False,"IsLightweight:=" ,
      ↪  False
445   ])
446 #Master Boundary (lower segment)
447 oEditor.CreatePolyline(
448   ["NAME:PolylineParameters","IsPolylineCovered:=",
    ↪  True,"IsPolylineClosed:=" , False,
449     ["NAME:PolylinePoints",["NAME:PLPoint","X:=", "0mm","Y:=",
      ↪  "0mm","Z:=", "0mm"],
450       ["NAME:PLPoint","X:=", "%smm"%(geo["R"]),"Y:=", "0mm","Z:=",
        ↪  "0mm"]
451     ],
452     ["NAME:PolylineSegments",["NAME:PLSegment","SegmentType:=",
      ↪  "Line","StartIndex:=", 0,"NoOfPoints:=",2]],
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
453     ["NAME:PolylineXSection","XSectionType:=",
        ↪  "None","XSectionOrient:=", "Auto","XSectionWidth:=",
        ↪  "0mm","XSectionTopWidth:=", "0mm","XSectionHeight:=",
        ↪  "0mm","XSectionNumSegments:=" , "0","XSectionBendType:="   ,
        ↪  "Corner"]
454   ],
455   [
456     "NAME:Attributes","Name:=", "Boundary_Master","Flags:=",
        ↪  "","Color:=","(143 175 143)","Transparency:=",
        ↪  0,"PartCoordinateSystem:=", "Global", "UDMId:=",
        ↪  "","MaterialValue:=", "\"vacuum\"",
457     "SurfaceMaterialValue:=", "\"\"","SolveInside:=",
        ↪  True, "IsMaterialEditable:=",
        ↪  True,"UseMaterialAppearance:=", False,"IsLightweight:=" ,
        ↪  False
458   ])
459
460 #Slave Boundary (vertical segment)
461
462 motorangle=math.pi/geo["p"]
463 oEditor.CreatePolyline(
464   ["NAME:PolylineParameters","IsPolylineCovered:=",
    ↪  True,"IsPolylineClosed:=" , False,
465     ["NAME:PolylinePoints",["NAME:PLPoint","X:=", "0mm","Y:=",
      ↪  "0mm","Z:=", "0mm"],
466       ["NAME:PLPoint","X:=",
        ↪  "%smm"%(geo["R"]*math.cos(motorangle)),"Y:=",
        ↪  "%smm"%(geo["R"]*math.sin(motorangle)),"Z:=", "0mm"]
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
467       ],
468       ["NAME:PolylineSegments",["NAME:PLSegment","SegmentType:=",
          ↪  "Line","StartIndex:=", 0,"NoOfPoints:=",2]],
469       ["NAME:PolylineXSection","XSectionType:=",
          ↪  "None","XSectionOrient:=", "Auto","XSectionWidth:=",
          ↪  "0mm","XSectionTopWidth:=", "0mm","XSectionHeight:=",
          ↪  "0mm","XSectionNumSegments:=" , "0","XSectionBendType:="   ,
          ↪  "Corner"]
470     ],
471     [
472       "NAME:Attributes","Name:=", "Boundary_Slave","Flags:=",
          ↪  "","Color:=","(143 175 143)","Transparency:=",
          ↪  0,"PartCoordinateSystem:=", "Global",  "UDMId:=",
          ↪  "","MaterialValue:=", "\"vacuum\"",
473       "SurfaceMaterialValue:=", "\"\"","SolveInside:=",
          ↪  True, "IsMaterialEditable:=",
          ↪  True,"UseMaterialAppearance:=", False,"IsLightweight:=" ,
          ↪  False
474     ])
475   #Outer Stator Arc, 0 vector potenzial
476   oEditor.CreatePolyline(
477     ["NAME:PolylineParameters","IsPolylineCovered:=",True,"IsPolylineClosed:=",False,[
478         ["NAME:PLPoint","X:=","%smm"%(geo["R"]),"Y:=","0mm","Z:=",
            ↪  "0mm"], #primo pt arco
479         ["NAME:PLPoint","X:=",
            ↪  "%smm"%(geo["R"]*math.cos(motorangle/2)),"Y:=",
            ↪  "%smm"%(geo["R"]*math.sin(motorangle/2)),"Z:=", "0mm"],
            ↪  #pt centale arco
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
480         ["NAME:PLPoint","X:=","%smm"%(geo["R"]*math.cos(motorangle)),"Y:=",
            ↪  "%smm"%(geo["R"]*math.sin(motorangle)),"Z:=","0mm"]#ultimo
            ↪  pt arco
481       ],
482       ["NAME:PolylineSegments",["NAME:PLSegment","SegmentType:=",
          ↪  "AngularArc","StartIndex:=", 0, "NoOfPoints:=",
          ↪  3,"NoOfSegments:=", "0",
483           "ArcAngle:=", "%srad"%(motorangle),"ArcCenterX:=",
              ↪  "0mm","ArcCenterY:=","0mm","ArcCenterZ:=",
              ↪  "0mm","ArcPlane:=","XY"]],
484       ["NAME:PolylineXSection","XSectionType:=" ,
          ↪  "None","XSectionOrient:=" ,
          ↪  "Auto","XSectionWidth:=","0mm","XSectionTopWidth:=","0mm",
485         "XSectionHeight:=","0mm","XSectionNumSegments:=","0","XSectionBendType:=","Cor
486     ],
487     ["NAME:Attributes","Name:=","VectorPotential1","Flags:=","","Color:=","(143
        ↪  175 143)","Transparency:=", 0,"PartCoordinateSystem:=",
        ↪  "Global",
488       "UDMId:=","","MaterialValue:=",
          ↪  "\"vacuum\"","SurfaceMaterialValue:=",
          ↪  "\"\"","SolveInside:=", True,"IsMaterialEditable:=", True,
489       "UseMaterialAppearance:=", False,"IsLightweight:=", False
490     ])
491
492 ##Region(sector) conteining moving parts (rotor)
493 r_mov_mid=geo["r"]+geo["g"]/2 #raggio regione mid
494 r_mov_out=geo["r"]+geo["g"]*3/4 #raggio regione out
495
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
496 #circuilar region outer rotor
497 oEditor.CreateCircle(
498   [
499     "NAME:CircleParameters","IsCovered:=", True,"XCenter:=",
      ↪  "0mm","YCenter:=", "0mm","ZCenter:=", "0mm","Radius:=",
      ↪  "%smm"%(r_mov_out),"WhichAxis:=", "Z","NumSegments:=", "0"
500   ],
501   [
502     "NAME:Attributes","Name:=", "Rotating_band_out","Flags:=",
      ↪  "",  "Color:=", "(70 180 250)","Transparency:="   ,
      ↪  0.85,"PartCoordinateSystem:=","Global","UDMId:=",
      ↪  "","MaterialValue:=", "\"Air\"",
503     "SurfaceMaterialValue:=", "\"\"","SolveInside:=",
      ↪  True, "IsMaterialEditable:=",
      ↪  True,"UseMaterialAppearance:=", False,"IsLightweight:=" ,
      ↪  False
504   ])
505
506 #cut circular region
507 oEditor.Imprint(["NAME:Selections","Blank
    ↪  Parts:=","Region,Rotating_band_out","Tool Parts:=",
    ↪  "Boundary_Master,Boundary_Slave"  ],["NAME:ImprintParameters","KeepOriginals:=",'
508
509 facesIDs=oEditor.GetFaceIDs("Region")
510 faceID=int(facesIDs[1])
511 oEditor.DetachFaces(["NAME:Selections", "Selections:=","Region","NewPartsModelFlag:=
    ↪  [faceID]]])
512 oEditor.Delete(["NAME:Selections","Selections:=","Region_Detach1"])
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
513
514 facesIDs=oEditor.GetFaceIDs("Rotating_band_out")
515 faceID=int(facesIDs[1])
516 oEditor.DetachFaces(["NAME:Selections", "Selections:=","Rotating_band_out","NewParts
    ↪  [faceID]]])
517 oEditor.Delete(["NAME:Selections","Selections:=","Rotating_band_out_Detach1"])
518
519 #circuilar region mid airgap
520 oEditor.CreateCircle(
521   [
522     "NAME:CircleParameters","IsCovered:=", True,"XCenter:=",
      ↪  "0mm","YCenter:=", "0mm","ZCenter:=", "0mm","Radius:=",
      ↪  "%smm"%(r_mov_mid),"WhichAxis:=", "Z","NumSegments:=", "0"
523   ],
524   [
525     "NAME:Attributes","Name:=", "Rotating_band_mid","Flags:=",
      ↪  "",  "Color:=", "(70 180 250)","Transparency:="   ,
      ↪  0.85,"PartCoordinateSystem:=","Global","UDMId:=",
      ↪  "","MaterialValue:=", "\"Air\"",
526     "SurfaceMaterialValue:=", "\"\"","SolveInside:=",
      ↪  True, "IsMaterialEditable:=",
      ↪  True,"UseMaterialAppearance:=", False,"IsLightweight:=" ,
      ↪  False
527   ])
528
529 #cut circular region
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
530 oEditor.Imprint(["NAME:Selections","Blank
    ↪ Parts:=","Rotating_band_mid","Tool Parts:=",
    ↪ "Boundary_Master,Boundary_Slave" ],["NAME:ImprintParameters","KeepOriginals:=",T
531
532 facesIDs=oEditor.GetFaceIDs("Rotating_band_mid")
533 faceID=int(facesIDs[1])
534 oEditor.DetachFaces(["NAME:Selections", "Selections:=","Rotating_band_mid","NewPartsl
    ↪ [faceID]]])
535 oEditor.Delete(["NAME:Selections","Selections:=","Rotating_band_mid_Detach1"])
536
537
538 #boundaries assignment
539 oModule = oDesign.GetModule("BoundarySetup")
540 oModule.AssignMaster(["NAME:Master1","Objects:=",
    ↪ ["Boundary_Master"],"ReverseV:=",False])
541 oModule.AssignSlave(["NAME:Slave1","Objects:=",["Boundary_Slave"],"ReverseU:=",
    ↪ False,"Master:=","Master1","SameAsMaster:=",False])
542 oModule.AssignVectorPotential(["NAME:VectorPotential1","Objects:=",
    ↪ ["VectorPotential1"],"Value:=","0","CoordinateSystem:=", ""])
543
544 oDesktop.AddMessage ("%s"%(geo["filename"][:-4]),
    ↪ "Maxwell2DDesign1", 0, "end", "")
545
546 #Design Settings
547
548 oDesign.SetDesignSettings(
549   [
550     "NAME:Design Settings Data",
```

Example of "draw_motor_in_ansys.py" generated by fprintf

```
551     "Perform Minimal validation:=", False,
552     "EnabledObjects:="  , [],
553     "PreserveTranSolnAfterDatasetEdit:=", False,
554     "ComputeTransientInductance:=", False,
555     "ComputeIncrementalMatrix:=", False,
556     "PerfectConductorThreshold:=", 1E+30,
557     "InsulatorThreshold:="  , 1,
558     "ModelDepth:="    , "%smm"%(geo["l"]),
559     "UseSkewModel:="  , False,
560     "EnableTranTranLinkWithSimplorer:=", False,
561     "BackgroundMaterialName:=", "vacuum",
562     "SolveFraction:=" , False,
563     "Multiplier:="    , "%d"%(geo["p"]*2)
564   ],
565   [
566     "NAME:Model Validation Settings",
567     "EntityCheckLevel:="  , "Strict",
568     "IgnoreUnclassifiedObjects:=", True,
569     "SkipIntersectionChecks:=", False
570   ])
571
572
573
574
575
576 #oProject.Save()
577 sys.exit()
```

Example of "setup_draw_motor_in_ansys.py" generated by fprintf

```
import sys,os
import pickle
geodata={
"R":67.500000, #stator radius
"r":40.490000, #rotor radius
"p" :3,   #pole pair
"g" :0.275000,   #air gap thickness
"nlay" :3,   #layer of rotor barrier
"PM_CS":[[25.3815,20.0857,19.3235,28.7565,24.9907,34.0505,],\
  [18.5511,17.2603,28.1617,12.7055,8.7646,2.6537,],\
    [0.86603,0.86603,0.86603,0.86603,0.86603,2.8328e-16,],\
      [0.5,0.5,-0.5,0.5,0.5,1,],], #Coordinate system magnet: 0-x
      ↪ 1-y 2-xvect 3-yvect
"q":2,  #slot per cave per phase
"radial_ribs_split":2, #area added by split barrier
"n_PM":6, #number of permanent magnet (number of magnetic segments)
"l":101.000000, #stack length
"filepath":"C:/Users/tesi/Google Drive/2020 - Tesi LM - Riccardo
↪ Aimo/prove/mot_01seg/", #motor"s folder
"filename":"segsplittest.mat", #motor"s file mat name
}
#struct material names
material={
"rotor":"M530-65A-OK",
"stator":"M530-65A-OK",
"shaft":"M530-65A-OK",
"slotcond":"Copper",
"magnet":"BMN-42SH",
```

Example of "setup_draw_motor_in_ansys.py" generated by fprintf

```
}
with open('C:/Users/tesi/Google Drive/2020 - Tesi LM - Riccardo Aimo/syre - \
  r489/syreExport/syre_AnsysMaxwell/temp/temp.pkl', 'wb') as export:
    pickle.dump([geodata,material],export,protocol=2)
sys.path.append(r"C:/Program Files/AnsysEM/AnsysEM20.1/Win64")
sys.path.append(r"C:/Program Files/AnsysEM/AnsysEM20.1/Win64/PythonFiles/DesktopPlugin")
import ScriptEnv

ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject = oDesktop.NewProject()
oProject.InsertDesign("Maxwell 2D", "Maxwell2DDesign1", "Transient", "")
oDesign = oProject.SetActiveDesign("Maxwell2DDesign1")
oEditor = oDesign.SetActiveEditor("3D Modeler")
oProject.SaveAs("C:/Users/tesi/Google Drive/2020 - Tesi LM - Riccardo Aimo/prove/mot_01seg/segsplittest.aedt", True)
oDesktop.ClearMessages("", "",3)
##########Add materials to library
##########Air###########
oDefinitionManager = oProject.GetDefinitionManager()
oDefinitionManager.AddMaterial(
  [
    "NAME:Air ",
    "CoordinateSystemType:=", "Cartesian",
    "BulkOrSurfaceType:=" , 1,
```

Example of "setup_draw_motor_in_ansys.py" generated by fprintf

```
    [
      "NAME:PhysicsTypes",
      "set:="     , ["Electromagnetic"]
    ]
  ])
############Magnet############
oDefinitionManager.AddMaterial(
  [
  "NAME:BMN-42SH",
  "CoordinateSystemType:=", "Cartesian",
  "BulkOrSurfaceType:=" , 1,
  [
    "NAME:PhysicsTypes",
    "set:="     , ["Electromagnetic"]
  ],
  [
    "NAME:ModifierData",
    [
      "NAME:ThermalModifierData",
      "modifier_data:=" , "thermal_modifier_data",
      [
        "NAME:all_thermal_modifiers",
        [
          "NAME:one_thermal_modifier",
          "Property::="   , "magnetic_coercivity",
          "Index::="    , 0,
          "prop_modifier:=" , "thermal_modifier",
          "use_free_form:=" , False,
```

Example of "setup_draw_motor_in_ansys.py" generated by fprintf

```
            "Tref:="    , "20cel",
            "C1:="      , "-0.001150",
            "C2:="      , "0",
            "TL:="      , "-273.15cel",
            "TU:="      , "1000cel",
            "auto_calculation:="  , True
          ]
        ]
      ]
    ],
    "permeability:="  , "1.050000",
    "conductivity:="  , "666666.666667",
    [
      "NAME:magnetic_coercivity",
      "property_type:=" , "VectorProperty",
      "Magnitude:="   , "-1007981.306249A_per_meter",
      "DirComp1:="    , "1",
      "DirComp2:="    , "0",
      "DirComp3:="    , "0"
    ],
    "mass_density:="  , "7550.000000"
])
########## Slot conductor ##########
oDefinitionManager.AddMaterial(
[
    "NAME:Copper",
    "CoordinateSystemType:=", "Cartesian",
    "BulkOrSurfaceType:=" , 1,
```

Example of "setup_draw_motor_in_ansys.py" generated by fprintf

```
  [
    "NAME:PhysicsTypes",
    "set:="     , ["Electromagnetic","Thermal"]
  ],
  "conductivity:="  , "58000000.000000",
  "thermal_conductivity:=", "400.000000",
  "mass_density:="  , "8940.000000"
])
############ Shaft plates ############
oDefinitionManager.AddMaterial(
[
  "NAME:M530-65A-OK",
  "CoordinateSystemType:=", "Cartesian",
  "BulkOrSurfaceType:=" , 1,
  [
    "NAME:PhysicsTypes",
    "set:="     , ["Electromagnetic"]
  ],
  [
    "NAME:permeability",
    "property_type:=" , "nonlinear",
    "BTypeForSingleCurve:=" , "normal",
    "HUnit:="   , "A_per_meter",
    "BUnit:="   , "tesla",
    "IsTemperatureDependent:=", False,
    [
      "NAME:BHCoordinates",
      [
```

Example of "setup_draw_motor_in_ansys.py" generated by fprintf

```
        "NAME:DimUnits",
        "",
        ""
      ],
      ["NAME:Coordinate",["NAME:CoordPoint",0.000000,0.000000]],
      ["NAME:Coordinate",["NAME:CoordPoint",12.326973,0.018848]],
      ["NAME:Coordinate",["NAME:CoordPoint",22.163343,0.038574]],
      ["NAME:Coordinate",["NAME:CoordPoint",30.264441,0.059313]],
      ["NAME:Coordinate",["NAME:CoordPoint",37.121027,0.081241]],
      ["NAME:Coordinate",["NAME:CoordPoint",43.070056,0.104587]],
      ["NAME:Coordinate",["NAME:CoordPoint",48.356479,0.129670]],
      ["NAME:Coordinate",["NAME:CoordPoint",53.171473,0.156946]],
      ["NAME:Coordinate",["NAME:CoordPoint",57.680775,0.187116]],
      ["NAME:Coordinate",["NAME:CoordPoint",62.054787,0.221361]],
      ["NAME:Coordinate",["NAME:CoordPoint",66.524258,0.261981]],
      ["NAME:Coordinate",["NAME:CoordPoint",71.574238,0.314919]],
      ["NAME:Coordinate",["NAME:CoordPoint",82.040378,0.442722]],
      ["NAME:Coordinate",["NAME:CoordPoint",92.228953,0.571656]],
      ["NAME:Coordinate",["NAME:CoordPoint",96.637904,0.625062]],
      ["NAME:Coordinate",["NAME:CoordPoint",100.157055,0.666042]],
      ["NAME:Coordinate",["NAME:CoordPoint",103.243889,0.700590]],
      ["NAME:Coordinate",["NAME:CoordPoint",106.074730,0.731027]],
      ["NAME:Coordinate",["NAME:CoordPoint",108.740062,0.758545]],
      ["NAME:Coordinate",["NAME:CoordPoint",111.294023,0.783850]],
      ["NAME:Coordinate",["NAME:CoordPoint",113.772411,0.807403]],
      ["NAME:Coordinate",["NAME:CoordPoint",116.200692,0.829524]],
      ["NAME:Coordinate",["NAME:CoordPoint",118.598054,0.850448]],
      ["NAME:Coordinate",["NAME:CoordPoint",120.979661,0.870348]],
```

Example of "setup_draw_motor_in_ansys.py" generated by fprintf

```
        ["NAME:Coordinate",["NAME:CoordPoint",123.358001,0.889363]],
        ["NAME:Coordinate",["NAME:CoordPoint",125.743736,0.907601]],
        ["NAME:Coordinate",["NAME:CoordPoint",128.146277,0.925149]],
        ["NAME:Coordinate",["NAME:CoordPoint",130.574173,0.942082]],
        ["NAME:Coordinate",["NAME:CoordPoint",133.035398,0.958459]],
        ["NAME:Coordinate",["NAME:CoordPoint",135.537568,0.974331]],
        ["NAME:Coordinate",["NAME:CoordPoint",138.088109,0.989743]],
        ["NAME:Coordinate",["NAME:CoordPoint",140.694397,1.004733]],
        ["NAME:Coordinate",["NAME:CoordPoint",143.363871,1.019333]],
        ["NAME:Coordinate",["NAME:CoordPoint",146.104146,1.033572]],
        ["NAME:Coordinate",["NAME:CoordPoint",148.923102,1.047477]],
        ["NAME:Coordinate",["NAME:CoordPoint",151.828985,1.061068]],
        ["NAME:Coordinate",["NAME:CoordPoint",154.830499,1.074368]],
        ["NAME:Coordinate",["NAME:CoordPoint",157.936901,1.087393]],
        ["NAME:Coordinate",["NAME:CoordPoint",161.158107,1.100160]],
        ["NAME:Coordinate",["NAME:CoordPoint",164.504800,1.112683]],
        ["NAME:Coordinate",["NAME:CoordPoint",167.988555,1.124977]],
        ["NAME:Coordinate",["NAME:CoordPoint",171.621969,1.137054]],
        ["NAME:Coordinate",["NAME:CoordPoint",175.418819,1.148923]],
        ["NAME:Coordinate",["NAME:CoordPoint",179.394229,1.160597]],
        ["NAME:Coordinate",["NAME:CoordPoint",183.564868,1.172084]],
        ["NAME:Coordinate",["NAME:CoordPoint",187.949177,1.183392]],
        ["NAME:Coordinate",["NAME:CoordPoint",192.567630,1.194531]],
        ["NAME:Coordinate",["NAME:CoordPoint",197.443032,1.205507]],
        ["NAME:Coordinate",["NAME:CoordPoint",202.600876,1.216327]],
        ["NAME:Coordinate",["NAME:CoordPoint",208.069751,1.226998]],
        ["NAME:Coordinate",["NAME:CoordPoint",213.881823,1.237525]],
        ["NAME:Coordinate",["NAME:CoordPoint",220.073407,1.247915]],
```

Example of "setup_draw_motor_in_ansys.py" generated by fprintf

```
        ["NAME:Coordinate",["NAME:CoordPoint",226.685632,1.258173]],
        ["NAME:Coordinate",["NAME:CoordPoint",233.765228,1.268303]],
        ["NAME:Coordinate",["NAME:CoordPoint",241.365470,1.278311]],
        ["NAME:Coordinate",["NAME:CoordPoint",249.547281,1.288200]],
        ["NAME:Coordinate",["NAME:CoordPoint",258.380550,1.297974]],
        ["NAME:Coordinate",["NAME:CoordPoint",267.945695,1.307639]],
        ["NAME:Coordinate",["NAME:CoordPoint",278.335510,1.317196]],
        ["NAME:Coordinate",["NAME:CoordPoint",289.657352,1.326650]],
        ["NAME:Coordinate",["NAME:CoordPoint",302.035700,1.336004]],
        ["NAME:Coordinate",["NAME:CoordPoint",315.615139,1.345261]],
        ["NAME:Coordinate",["NAME:CoordPoint",330.563787,1.354424]],
        ["NAME:Coordinate",["NAME:CoordPoint",347.077151,1.363496]],
        ["NAME:Coordinate",["NAME:CoordPoint",365.382334,1.372479]],
        ["NAME:Coordinate",["NAME:CoordPoint",385.742406,1.381377]],
        ["NAME:Coordinate",["NAME:CoordPoint",408.460588,1.390191]],
        ["NAME:Coordinate",["NAME:CoordPoint",433.883625,1.398923]],
        ["NAME:Coordinate",["NAME:CoordPoint",462.403418,1.407577]],
        ["NAME:Coordinate",["NAME:CoordPoint",494.455569,1.416153]],
        ["NAME:Coordinate",["NAME:CoordPoint",530.513134,1.424655]],
        ["NAME:Coordinate",["NAME:CoordPoint",571.073764,1.433084]],
        ["NAME:Coordinate",["NAME:CoordPoint",616.638773,1.441442]],
        ["NAME:Coordinate",["NAME:CoordPoint",667.683943,1.449730]],
        ["NAME:Coordinate",["NAME:CoordPoint",724.624100,1.457951]],
        ["NAME:Coordinate",["NAME:CoordPoint",787.776350,1.466105]],
        ["NAME:Coordinate",["NAME:CoordPoint",857.329160,1.474195]],
        ["NAME:Coordinate",["NAME:CoordPoint",933.324786,1.482222]],
        ["NAME:Coordinate",["NAME:CoordPoint",1015.660001,1.490188]],
        ["NAME:Coordinate",["NAME:CoordPoint",1104.105207,1.498094]],
```

Example of "setup_draw_motor_in_ansys.py" generated by fprintf

```
219         ["NAME:Coordinate",["NAME:CoordPoint",1198.337168,1.505940]],
220         ["NAME:Coordinate",["NAME:CoordPoint",1297.977718,1.513730]],
221         ["NAME:Coordinate",["NAME:CoordPoint",1402.631005,1.521463]],
222         ["NAME:Coordinate",["NAME:CoordPoint",1511.914103,1.529140]],
223         ["NAME:Coordinate",["NAME:CoordPoint",1625.478829,1.536765]],
224         ["NAME:Coordinate",["NAME:CoordPoint",1743.024982,1.544336]],
225         ["NAME:Coordinate",["NAME:CoordPoint",1864.306583,1.551856]],
226         ["NAME:Coordinate",["NAME:CoordPoint",1989.133134,1.559325]],
227         ["NAME:Coordinate",["NAME:CoordPoint",2117.367739,1.566744]],
228         ["NAME:Coordinate",["NAME:CoordPoint",2248.923537,1.574115]],
229         ["NAME:Coordinate",["NAME:CoordPoint",2383.759484,1.581438]],
230         ["NAME:Coordinate",["NAME:CoordPoint",2521.876100,1.588714]],
231         ["NAME:Coordinate",["NAME:CoordPoint",2663.311587,1.595944]],
232         ["NAME:Coordinate",["NAME:CoordPoint",2808.138478,1.603129]],
233         ["NAME:Coordinate",["NAME:CoordPoint",2956.460916,1.610270]],
234         ["NAME:Coordinate",["NAME:CoordPoint",3108.412545,1.617368]],
235         ["NAME:Coordinate",["NAME:CoordPoint",3264.155013,1.624423]],
236         ["NAME:Coordinate",["NAME:CoordPoint",3423.877028,1.631436]],
237         ["NAME:Coordinate",["NAME:CoordPoint",3587.793930,1.638408]],
238         ["NAME:Coordinate",["NAME:CoordPoint",3756.147740,1.645340]],
239         ["NAME:Coordinate",["NAME:CoordPoint",3929.207663,1.652232]],
240         ["NAME:Coordinate",["NAME:CoordPoint",4107.271004,1.659084]],
241         ["NAME:Coordinate",["NAME:CoordPoint",4290.664509,1.665899]],
242         ["NAME:Coordinate",["NAME:CoordPoint",4479.746100,1.672675]],
243         ["NAME:Coordinate",["NAME:CoordPoint",4674.907038,1.679415]],
244         ["NAME:Coordinate",["NAME:CoordPoint",4876.574501,1.686118]],
245         ["NAME:Coordinate",["NAME:CoordPoint",5085.214626,1.692785]],
246         ["NAME:Coordinate",["NAME:CoordPoint",5301.336026,1.699417]],
```

Example of "setup_draw_motor_in_ansys.py" generated by fprintf

```
247         ["NAME:Coordinate",["NAME:CoordPoint",5525.493837,1.706014]],
248         ["NAME:Coordinate",["NAME:CoordPoint",5758.294337,1.712576]],
249         ["NAME:Coordinate",["NAME:CoordPoint",6000.400205,1.719105]],
250         ["NAME:Coordinate",["NAME:CoordPoint",6252.536483,1.725601]],
251         ["NAME:Coordinate",["NAME:CoordPoint",6515.497322,1.732064]],
252         ["NAME:Coordinate",["NAME:CoordPoint",6790.153620,1.738494]],
253         ["NAME:Coordinate",["NAME:CoordPoint",7077.461626,1.744893]],
254         ["NAME:Coordinate",["NAME:CoordPoint",7378.472654,1.751261]],
255         ["NAME:Coordinate",["NAME:CoordPoint",7694.344022,1.757598]],
256         ["NAME:Coordinate",["NAME:CoordPoint",8026.351347,1.763904]],
257         ["NAME:Coordinate",["NAME:CoordPoint",8375.902346,1.770180]],
258         ["NAME:Coordinate",["NAME:CoordPoint",8744.552291,1.776427]],
259         ["NAME:Coordinate",["NAME:CoordPoint",9134.021251,1.782645]],
260         ["NAME:Coordinate",["NAME:CoordPoint",9546.213258,1.788834]],
261         ["NAME:Coordinate",["NAME:CoordPoint",9983.237478,1.794995]],
262         ["NAME:Coordinate",["NAME:CoordPoint",10447.431450,1.801128]],
263         ["NAME:Coordinate",["NAME:CoordPoint",10941.386290,1.807233]],
264         ["NAME:Coordinate",["NAME:CoordPoint",11467.973820,1.813311]],
265         ["NAME:Coordinate",["NAME:CoordPoint",12030.374990,1.819362]],
266         ["NAME:Coordinate",["NAME:CoordPoint",12632.109290,1.825387]],
267         ["NAME:Coordinate",["NAME:CoordPoint",13277.063830,1.831385]],
268         ["NAME:Coordinate",["NAME:CoordPoint",13969.520790,1.837358]],
269         ["NAME:Coordinate",["NAME:CoordPoint",14714.181050,1.843305]],
270         ["NAME:Coordinate",["NAME:CoordPoint",15516.181070,1.849228]],
271         ["NAME:Coordinate",["NAME:CoordPoint",16381.099350,1.855125]],
272         ["NAME:Coordinate",["NAME:CoordPoint",17314.947640,1.860998]],
273         ["NAME:Coordinate",["NAME:CoordPoint",18324.141340,1.866846]],
274         ["NAME:Coordinate",["NAME:CoordPoint",19415.442930,1.872671]],
```

Example of "setup_draw_motor_in_ansys.py" generated by fprintf

```
275         ["NAME:Coordinate",["NAME:CoordPoint",20595.871870,1.878472]],
276         ["NAME:Coordinate",["NAME:CoordPoint",21872.575840,1.884250]],
277         ["NAME:Coordinate",["NAME:CoordPoint",23252.659780,1.890004]],
278         ["NAME:Coordinate",["NAME:CoordPoint",24742.973530,1.895736]],
279         ["NAME:Coordinate",["NAME:CoordPoint",26349.864180,1.901445]],
280         ["NAME:Coordinate",["NAME:CoordPoint",28078.906020,1.907133]],
281         ["NAME:Coordinate",["NAME:CoordPoint",29934.628090,1.912798]],
282         ["NAME:Coordinate",["NAME:CoordPoint",31920.264540,1.918441]],
283         ["NAME:Coordinate",["NAME:CoordPoint",34037.555240,1.924063]],
284         ["NAME:Coordinate",["NAME:CoordPoint",36286.621250,1.929663]],
285         ["NAME:Coordinate",["NAME:CoordPoint",38665.931930,1.935243]],
286         ["NAME:Coordinate",["NAME:CoordPoint",41172.369030,1.940802]],
287         ["NAME:Coordinate",["NAME:CoordPoint",43801.379810,1.946340]],
288         ["NAME:Coordinate",["NAME:CoordPoint",46547.200390,1.951858]],
289         ["NAME:Coordinate",["NAME:CoordPoint",49403.123290,1.957355]],
290         ["NAME:Coordinate",["NAME:CoordPoint",52361.781180,1.962833]],
291         ["NAME:Coordinate",["NAME:CoordPoint",55415.422000,1.968292]],
292         ["NAME:Coordinate",["NAME:CoordPoint",58556.156180,1.973730]],
293         ["NAME:Coordinate",["NAME:CoordPoint",61776.164400,1.979150]],
294         ["NAME:Coordinate",["NAME:CoordPoint",65067.860920,1.984550]],
295         ["NAME:Coordinate",["NAME:CoordPoint",68424.013020,1.989932]],
296         ["NAME:Coordinate",["NAME:CoordPoint",71837.820730,1.995295]],
297         ["NAME:Coordinate",["NAME:CoordPoint",75302.963180,2.000639]],
298         ["NAME:Coordinate",["NAME:CoordPoint",78813.618240,2.005966]],
299         ["NAME:Coordinate",["NAME:CoordPoint",82364.462130,2.011274]],
300         ["NAME:Coordinate",["NAME:CoordPoint",85950.654730,2.016564]],
301         ["NAME:Coordinate",["NAME:CoordPoint",89567.815290,2.021836]],
302         ["NAME:Coordinate",["NAME:CoordPoint",93211.992480,2.027091]],
```

Example of "setup_draw_motor_in_ansys.py" generated by fprintf

```
303         ["NAME:Coordinate",["NAME:CoordPoint",96879.631450,2.032329]],
304         ["NAME:Coordinate",["NAME:CoordPoint",100567.540000,2.037549]],
305         ["NAME:Coordinate",["NAME:CoordPoint",104272.855100,2.042753]],
306         ["NAME:Coordinate",["NAME:CoordPoint",107993.011500,2.047939]],
307         ["NAME:Coordinate",["NAME:CoordPoint",111725.711100,2.053109]],
308         ["NAME:Coordinate",["NAME:CoordPoint",115468.895800,2.058262]],
309         ["NAME:Coordinate",["NAME:CoordPoint",119220.722400,2.063399]],
310         ["NAME:Coordinate",["NAME:CoordPoint",122979.539100,2.068520]],
311         ["NAME:Coordinate",["NAME:CoordPoint",126743.865500,2.073624]],
312         ["NAME:Coordinate",["NAME:CoordPoint",130512.374000,2.078713]],
313         ["NAME:Coordinate",["NAME:CoordPoint",134283.873100,2.083786]],
314         ["NAME:Coordinate",["NAME:CoordPoint",138057.293000,2.088843]],
315         ["NAME:Coordinate",["NAME:CoordPoint",141831.672400,2.093885]],
316         ["NAME:Coordinate",["NAME:CoordPoint",145606.147000,2.098911]],
317         ["NAME:Coordinate",["NAME:CoordPoint",149379.938900,2.103922]],
318         ["NAME:Coordinate",["NAME:CoordPoint",153152.348300,2.108919]],
319         ["NAME:Coordinate",["NAME:CoordPoint",156922.744200,2.113900]],
320         ["NAME:Coordinate",["NAME:CoordPoint",160690.558400,2.118866]],
321         ["NAME:Coordinate",["NAME:CoordPoint",164455.278200,2.123818]],
322         ["NAME:Coordinate",["NAME:CoordPoint",168216.441400,2.128755]],
323         ["NAME:Coordinate",["NAME:CoordPoint",171973.630800,2.133678]],
324         ["NAME:Coordinate",["NAME:CoordPoint",175726.469800,2.138586]],
325         ["NAME:Coordinate",["NAME:CoordPoint",179474.618800,2.143480]],
326         ["NAME:Coordinate",["NAME:CoordPoint",183217.771000,2.148361]],
327         ["NAME:Coordinate",["NAME:CoordPoint",186955.649700,2.153227]],
328         ["NAME:Coordinate",["NAME:CoordPoint",190688.005200,2.158079]],
329         ["NAME:Coordinate",["NAME:CoordPoint",194414.612500,2.162918]],
330         ["NAME:Coordinate",["NAME:CoordPoint",198135.268600,2.167743]],
```

Example of "setup_draw_motor_in_ansys.py" generated by fprintf

```
331         ["NAME:Coordinate",["NAME:CoordPoint",201849.790800,2.172555]],
332         ["NAME:Coordinate",["NAME:CoordPoint",205558.014800,2.177354]],
333         ["NAME:Coordinate",["NAME:CoordPoint",209259.793100,2.182139]],
334         ["NAME:Coordinate",["NAME:CoordPoint",212954.993100,2.186911]],
335         ["NAME:Coordinate",["NAME:CoordPoint",216643.496300,2.191670]],
336         ["NAME:Coordinate",["NAME:CoordPoint",220325.196700,2.196416]],
337         ["NAME:Coordinate",["NAME:CoordPoint",224000.000000,2.201149]],
338       ],
339       [
340         "NAME:Temperatures"
341       ]
342     ],
343     [
344       "NAME:magnetic_coercivity",
345       "property_type:=" , "VectorProperty",
346       "Magnitude:="   , "0A_per_meter",
347       "DirComp1:="    , "1",
348       "DirComp2:="    , "0",
349       "DirComp3:="    , "0"
350     ],
351     [
352       "NAME:core_loss_type",
353       "property_type:=" , "ChoiceProperty",
354       "Choice:="    , "Electrical Steel"
355     ],
356     "core_loss_kh:="  , "0.033707",
357     "core_loss_kc:="  , "0.000000",
358     "core_loss_ke:="  , "0.000047",
```

Example of "setup_draw_motor_in_ansys.py" generated by fprintf

```
359   "core_loss_kdc:=" , "0",
360   "mass_density:="  , "7700.000000",
361   "core_loss_equiv_cut_depth:=", "0.001meter"
362 ])
363
364 sys.exit()
```

sim_xdeg_AM.py

```
import pickle
import sys
import os
import os.path
import math

currentFolder = os.path.dirname(os.path.realpath(__file__))

filepath = r'%s/temp/temp.pkl'%(currentFolder)
with open(filepath,'rb') as exportdata:
    simdata = pickle.load(exportdata)

sys.path.append(r"C:/Program Files/AnsysEM/AnsysEM20.1/Win64")
sys.path.append(r"C:/Program
↪ Files/AnsysEM/AnsysEM20.1/Win64/PythonFiles/DesktopPlugin")
import ScriptEnv

ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")
oDesktop.RestoreWindow()
oProject =
↪ oDesktop.SetActiveProject("%s"%(simdata["filename"][:-4]))
oDesign = oProject.SetActiveDesign("Maxwell2DDesign1")
oEditor = oDesign.SetActiveEditor("3D Modeler")

#oDesktop.AddMessage ("%s"%(geo["filename"][:-4]),
↪ "Maxwell2DDesign1", 0, "%s"%(simdata["iAmp"]), "")

############### Current assignment excitation per phases
```

sim_xdeg_AM.py

```
26
27 oModule = oDesign.GetModule("BoundarySetup")
28 oModule.DeleteAllExcitations()
29 oModule.AssignWindingGroup(
30   [
31     "NAME:WG_Ph1",
32     "Type:="     , "Current",
33     "IsSolid:="    , False,
34     "Current:="    , "%s*cos(%s*2*pi/60*%s*time+(%s))A"%\
35       (simdata["iAmp"],simdata["EvalSpeed"],simdata["p"],\
36         simdata["phi_init"]),
37     "Resistance:="     , "0ohm",
38     "Inductance:="     , "0nH",
39     "Voltage:="    , "0mV",
40     "ParallelBranchesNum:=" , "1"
41   ])
42 oModule.AssignWindingGroup(
43   [
44     "NAME:WG_Ph2",
45     "Type:="     , "Current",
46     "IsSolid:="    , False,
47     "Current:="    , "%s*cos(%s*2*pi/60*%s*time+(240)*pi/180+(%s))A"\
48       %(simdata["iAmp"],simdata["EvalSpeed"],simdata["p"],\
49         simdata["phi_init"]),
50     "Resistance:="     , "0ohm",
51     "Inductance:="     , "0nH",
52     "Voltage:="    , "0mV",
53     "ParallelBranchesNum:=" , "1"
```

sim_xdeg_AM.py

```
54    ])
55  oModule.AssignWindingGroup(
56    [
57      "NAME:WG_Ph3",
58      "Type:="     , "Current",
59      "IsSolid:="    , False,
60      "Current:="    , "%s*cos(%s*2*pi/60*%s*time+(120)*pi/180+(%s))A"\
61        %(simdata["iAmp"],simdata["EvalSpeed"],simdata["p"],\
62          simdata["phi_init"]),
63      "Resistance:="     , "0ohm",
64      "Inductance:="     , "0nH",
65      "Voltage:="    , "0mV",
66      "ParallelBranchesNum:=" , "1"
67    ])
68
69  ############ lists name slots
70  ii=0
71  slotconds=""
72  lineslotairgaps=""
73  slotairgaps=""
74  statplate="1_1"
75
76  for ii in range(simdata["q"]*3):
77
78    slotindex=2+ii*4
79    lineindex=slotindex+1
80    slot="1_%d" %(slotindex)
81    line="1_%d"%(lineindex)
```

sim_xdeg_AM.py

```
82    lineairgap="1_%d,1_%d"%(lineindex+1,lineindex+2)
83    slotconds = slotconds + "," + slot + "," + slot + "_Detach1"
      ↪  ##list slot/cond, useful for detachment
84    lineslotairgaps=lineslotairgaps+","+lineairgap
85    slotairgaps=slotairgaps+","+statplate+"_Detach"+"%d"%(ii+1)
86
87  slotconds=slotconds[1:] #removing initial comma of the string
88  slotconds_array=slotconds.split(',') #vector containig the elements
    ↪  of the string separated by the comma
89
90
91  for jj in range(len(simdata["win_avv"][0])): #n slot per layer
92
93    for ii in range(len(simdata["win_avv"])): # n layer slot
94
95      if simdata["win_avv"][ii][jj]>0:
96        oModule.AssignCoil(
97          [
98            "NAME:
              ↪  Coil_%s"%(slotconds_array[len(simdata["win_avv"])*jj+ii]),
99            "Objects:="    ,
              ↪  [slotconds_array[len(simdata["win_avv"])*jj+ii]],
100           "Conductor number:="   , "%s"%(simdata["N_cond"]),
101           "PolarityType:="   , "Positive"
102         ])
103     else:
104       oModule.AssignCoil(
105         [
```

sim_xdeg_AM.py

```
106             "NAME:
                ↪  Coil_%s"%(slotconds_array[len(simdata["win_avv"])*jj+ii]),
107             "Objects:="    ,
                ↪  [slotconds_array[len(simdata["win_avv"])*jj+ii]],
108             "Conductor number:="   , "%s"%(simdata["N_cond"]),
109             "PolarityType:="   , "Negative"
110           ])
111
112       phasenum=abs(int(simdata["win_avv"][ii][jj]))
113       oModule.AddWindingCoils("WG_Ph%d"%(phasenum),
          ↪  ["Coil_%s"%(slotconds_array\
114         [len(simdata["win_avv"])*jj+ii])])
115
116   ####### Magnets Temperature ############
117   if simdata["n_PM"]!=0:
118     PMs_temp=""
119     temp=0
120     for jj in range (2):
121       for ii in range(simdata["nlay"]+simdata["radial_ribs_split"]):
122         rotorindex=(3+5*simdata["q"]*3)*(1-jj)+ii+temp*jj
123
124       for kk in range(int(simdata["n_PM"]/2)):
125         rotorindex=rotorindex+1
126         PM="2_%d" %(rotorindex)
127         PMs_temp=PMs_temp + "," + PM + "," +
          ↪  "%fcel"%(simdata["tempPP"])
128
129       temp=rotorindex+1
```

sim_xdeg_AM.py

```
130   PMs_temp=PMs_temp[1:]
131   PMs_temp_array=PMs_temp.split(',')
132
133
134   oDesign.SetObjectTemperature(
135     [
136       "NAME:TemperatureSettings",
137       "IncludeTemperatureDependence:=", True,
138       "EnableFeedback:="  , False,
139       "Temperatures:="  , PMs_temp_array
140     ])
141
142 ####### Setup Core Loss Part ######
143 if simdata['corelossflag']==1:
144   rotorindex=3+5*simdata["q"]*3+2*(simdata["nlay"]+\
145     simdata["radial_ribs_split"])+simdata["n_PM"]
146   rotorplate="2_%d" %(rotorindex)
147   oModule.SetCoreLoss([statplate,rotorplate], False)
148   #if shaft!="ShaftAir":
149   #   oModule.SetCoreLoss([shaft], False)
150
151 ####### Setup Moving part ##########
152
153 oModule = oDesign.GetModule("ModelSetup")
154 MotionSetupName = oModule.GetMotionSetupNames ()
155
156 if MotionSetupName == ["MotionSetup1"]:
157   oModule = oDesign.GetModule("ReportSetup")
```

sim_xdeg_AM.py

```
158   oModule.DeleteAllReports()
159   oModule = oDesign.GetModule("ModelSetup")
160   oModule.DeleteMotionSetup(["MotionSetup1"])
161
162 oModule.AssignBand(
163   [
164     "NAME:Data",
165     "Move Type:="    , "Rotate",
166     "Coordinate System:=" , "Global",
167     "Axis:="     , "Z",
168     "Is Positive:="    , True,
169     "InitPos:="    , "0deg",    ## Initial Rotor Position
170     "HasRotateLimit:="   , False,
171     "NonCylindrical:="   , False,
172     "Consider Mechanical Transient:=", False,
173     "Angular Velocity:="   , "%srpm"%(simdata["EvalSpeed"]),   ##
      ↪  Rotor Speed
174     "Objects:="    , ["Rotating_Band_out"]
175   ])
176
177
178 ####### Analisys Setup ##########
179
180 if simdata["save_fields"]==0:
181   savefieldstype= "None"
182 else:
183   savefieldstype= "Every N Steps"
184
```

sim_xdeg_AM.py

```
185
186 oModule = oDesign.GetModule("AnalysisSetup")
187 oModule.ResetAllToTimeZero()
188
189 SetupName = oModule.GetSetups()
190 #oDesktop.AddMessage ("%s"%(simdata["filename"][:-4]),
    ↪  "Maxwell2DDesign1", 0, "%s"%(SetupName), "")
191 if SetupName == ["Setup1"]:
192   oModule.DeleteSetups(["Setup1"])
193   oDesktop.ClearMessages("", "",2)
194
195 oModule.InsertSetup("Transient",
196   [
197     "NAME:Setup1",
198     "Enabled:="    , True,
199     [
200       "NAME:MeshLink",
201       "ImportMesh:="     , False
202     ],
203     "NonlinearSolverResidual:=", "0.001",
204     "TimeIntegrationMethod:=", "BackwardEuler",
205     "SmoothBHCurve:=" , False,
206     "StopTime:="     , "60/(%s*%s*360/%s)
      ↪  s"%(simdata['EvalSpeed'],simdata['p'],simdata["xdeg"]),
207     "TimeStep:="     ,  "60/(%s*%s*360/%s*%s)
      ↪  s"%(simdata['EvalSpeed'],\
208       simdata['p'],simdata["xdeg"],simdata["nsim"]),
209     "OutputError:="    , False,
```

sim_xdeg_AM.py

```
210     "UseControlProgram:=" , False,
211     "ControlProgramName:="  , " ",
212     "ControlProgramArg:=" , " ",
213     "CallCtrlProgAfterLastStep:=", False,
214     "FastReachSteadyState:=", False,
215     "AutoDetectSteadyState:=", False,
216     "IsGeneralTransient:="  , True,
217     "IsHalfPeriodicTransient:=", False,
218     "SaveFieldsType:="  , "%s"%(savefieldstype),
219     "N Steps:="   , "%s"%(simdata["save_fields"]),
220     "Steps From:="    , "0s",
221     "Steps To:="    , "60/(%s*%s*360/%s)
        ↪  s"%(simdata['EvalSpeed'],simdata['p'],simdata["xdeg"]),
222     "CacheSaveKind:=" , "Count",
223     "NumberSolveSteps:="  , 1,
224     "RangeStart:="    , "0s",
225     "RangeEnd:="    , "0.1s",
226     "UseAdaptiveTimeStep:=" , False,
227     "InitialTimeStep:=" , "0.002s",
228     "MinTimeStep:="   , "0.001s",
229     "MaxTimeStep:="   , "0.003s",
230     "TimeStepErrTolerance:=", 0.0001
231   ])
232
233 ############## Solutions Plot ##############
234
235 #Torque
236 oModule = oDesign.GetModule("ReportSetup")
```

sim_xdeg_AM.py

```
237
238 oModule.CreateReport("Torque Plot", "Transient", "Rectangular Plot",
    ↪ "Setup1 : Transient",
239   [
240     "Domain:="    , "Sweep"
241   ],
242   [
243     "Time:="    , ["All"]
244   ],
245   [
246     "X Component:="    , "Time",
247     "Y Component:="    , ["Moving1.Torque"]
248   ])
249 oModule.ChangeProperty(["NAME:AllTabs",["NAME:Scaling",\
250   ["NAME:PropServers","Torque Plot:AxisY1"],["NAME:ChangedProps",\
251     ["NAME:Auto Units", "Value:=", False],["NAME:Units","Value:=",
      ↪ "NewtonMeter"]]]])
252
253 #Induced Voltages
254 oModule.CreateReport("Induced Voltages", "Transient", "Rectangular
    ↪ Plot", "Setup1 : Transient",
255   [
256     "Domain:="    , "Sweep"
257   ],
258   [
259     "Time:="    , ["All"]
260   ],
261   [
```

sim_xdeg_AM.py

```
262     "X Component:="    , "Time",
263     "Y Component:="    , ["InducedVoltage(WG_Ph1)",\
264       "InducedVoltage(WG_Ph2)","InducedVoltage(WG_Ph3)"]
265   ])
266 oModule.ChangeProperty(["NAME:AllTabs",["NAME:Scaling",\
267   ["NAME:PropServers","Induced
    ↪  Voltages:AxisY1"],["NAME:ChangedProps"\
268     ,["NAME:Auto Units",  "Value:=", False],["NAME:Units","Value:=",
      ↪  "V"]]]])
269
270 #Input Currents
271 oModule.CreateReport("Input Currents", "Transient", "Rectangular
  ↪  Plot", "Setup1 : Transient",
272   [
273     "Domain:="     , "Sweep"
274   ],
275   [
276     "Time:="     , ["All"]
277   ],
278   [
279     "X Component:="    , "Time",
280     "Y Component:="    , ["InputCurrent(WG_Ph1)",\
281       "InputCurrent(WG_Ph2)","InputCurrent(WG_Ph3)"]
282   ])
283 oModule.ChangeProperty(["NAME:AllTabs",["NAME:Scaling",\
284   ["NAME:PropServers","Input
    ↪  Currents:AxisY1"],["NAME:ChangedProps",\
```

sim_xdeg_AM.py

```
285     ["NAME:Auto Units", "Value:=", False],["NAME:Units","Value:=",
        ↪  "A"]]]])
286
287 #Flux
288 oModule.CreateReport("Fluxes Linkages", "Transient", "Rectangular
    ↪  Plot", "Setup1 : Transient",
289   [
290     "Domain:="    , "Sweep"
291   ],
292   [
293     "Time:="    , ["All"]
294   ],
295   [
296     "X Component:="    , "Time",
297     "Y Component:="    , ["FluxLinkage(WG_Ph1)",\
298       "FluxLinkage(WG_Ph2)","FluxLinkage(WG_Ph3)"]
299   ])
300 oModule.ChangeProperty(["NAME:AllTabs",["NAME:Scaling",\
301   ["NAME:PropServers","Fluxes
    ↪  Linkages:AxisY1"],["NAME:ChangedProps",\
302     ["NAME:Auto Units", "Value:=", False],["NAME:Units","Value:=",
      ↪  "Wb"]]]])
303
304 #Mechanical Position Theta
305 oModule.CreateReport("Mechanical Position", "Transient",
    ↪  "Rectangular Plot", "Setup1 : Transient",
306   [
307     "Domain:="    , "Sweep"
```

```
308   ],
309   [
310     "Time:="    , ["All"]
311   ],
312   [
313     "X Component:="   , "Time",
314     "Y Component:="   , ["Moving1.Position"]
315   ])
316 oModule.ChangeProperty(["NAME:AllTabs",["NAME:Scaling",\
317   ["NAME:PropServers","Mechanical
      ↪  Position:AxisY1"],["NAME:ChangedProps",\
318     ["NAME:Auto Units", "Value:=", False],["NAME:Units","Value:=",
        ↪  "deg"]]]])
319
320 ####### Core Loss
321 if simdata['corelossflag']==1:
322
323   oModule.CreateReport("Core Loss", "Transient", "Rectangular Plot",
      ↪  "Setup1 : Transient",
324     [
325       "Domain:="    , "Sweep"
326     ],
327     [
328       "Time:="    , ["All"]
329     ],
330     [
331       "X Component:="   , "Time",
332       "Y Component:="   , ["CoreLoss","CoreLoss(%s)"%(statplate),\
```

sim_xdeg_AM.py

```
333         "CoreLoss(%s)"%(rotorplate)]
334       ])
335     oModule.AddTraceCharacteristics("Core Loss", "avg", [],
        ↪  ["Specified",
        ↪  "60/(%s*%s)s"%(simdata['EvalSpeed'],simdata['p']),
        ↪  "60/(%s*%s*360/%s)
        ↪  s"%(simdata['EvalSpeed'],simdata['p'],simdata["xdeg"])])
336     oModule.ChangeProperty(["NAME:AllTabs",["NAME:Scaling",\
337       ["NAME:PropServers","Core Loss:AxisY1"],["NAME:ChangedProps",\
338         ["NAME:Auto Units", "Value:=", False],["NAME:Units","Value:=",
          ↪  "W"]]]])
339
340     oModule.CreateReport("Hysteresis Loss", "Transient", "Rectangular
        ↪  Plot", "Setup1 : Transient",
341       [
342         "Domain:="    , "Sweep"
343       ],
344       [
345         "Time:="    , ["All"]
346       ],
347       [
348         "X Component:="    , "Time",
349         "Y Component:="    , ["HysteresisLoss","HysteresisLoss(%s)"\
350           %(statplate),"HysteresisLoss(%s)"%(rotorplate)]
351       ])
```

sim_xdeg_AM.py

```
352 oModule.AddTraceCharacteristics("Hysteresis Loss", "avg", [],
    ↪ ["Specified",
    ↪ "60/(%s*%s)s"%(simdata['EvalSpeed'],simdata['p']),
    ↪ "60/(%s*%s*360/%s)
    ↪ s"%(simdata['EvalSpeed'],simdata['p'],simdata["xdeg"])])
353 oModule.ChangeProperty(["NAME:AllTabs",["NAME:Scaling",\
354   ["NAME:PropServers","Hysteresis
      ↪ Loss:AxisY1"],["NAME:ChangedProps",\
355     ["NAME:Auto Units", "Value:=", False],["NAME:Units","Value:=",
        ↪ "W"]]]])
356
357 oModule.CreateReport("Eddy Currents Loss", "Transient",
    ↪ "Rectangular Plot", "Setup1 : Transient",
358   [
359     "Domain:="    , "Sweep"
360   ],
361   [
362     "Time:="    , ["All"]
363   ],
364   [
365     "X Component:="    , "Time",
366     "Y Component:="    , ["EddyCurrentLoss","EddyCurrentLoss(%s)"\
367       %(statplate),"EddyCurrentLoss(%s)"%(rotorplate)]
368   ])
```

sim_xdeg_AM.py

```
369   oModule.AddTraceCharacteristics("Eddy Currents Loss", "avg", [],
      ↪  ["Specified",
      ↪  "60/(%s*%s)s"%(simdata['EvalSpeed'],simdata['p']),
      ↪  "60/(%s*%s*360/%s)
      ↪  s"%(simdata['EvalSpeed'],simdata['p'],simdata["xdeg"])])
370   oModule.ChangeProperty(["NAME:AllTabs",["NAME:Scaling",\
371     ["NAME:PropServers","Eddy Currents
        ↪  Loss:AxisY1"],["NAME:ChangedProps",\
372       ["NAME:Auto Units", "Value:=", False],["NAME:Units","Value:=",
          ↪  "W"]]]])
373
374   oModule.CreateReport("Excess Loss", "Transient", "Rectangular
      ↪  Plot", "Setup1 : Transient",
375     [
376       "Domain:="    , "Sweep"
377     ],
378     [
379       "Time:="    , ["All"]
380     ],
381     [
382       "X Component:="    , "Time",
383       "Y Component:="    , ["ExcessLoss","ExcessLoss(%s)"\
384         %(statplate),"ExcessLoss(%s)"%(rotorplate)]
385     ])
```

sim_xdeg_AM.py

```
386   oModule.AddTraceCharacteristics("Excess Loss", "avg", [],
      ↪  ["Specified",
      ↪  "60/(%s*%s)s"%(simdata['EvalSpeed'],simdata['p']),
      ↪  "60/(%s*%s*360/%s)
      ↪  s"%(simdata['EvalSpeed'],simdata['p'],simdata["xdeg"])])
387   oModule.ChangeProperty(["NAME:AllTabs",["NAME:Scaling",\
388     ["NAME:PropServers","Excess Loss:AxisY1"],["NAME:ChangedProps",\
389       ["NAME:Auto Units", "Value:=", False],["NAME:Units","Value:=",
          ↪  "W"]]]])
390
391
392
393 ############ Initial Mesh Settings #################
394
395 oModule = oDesign.GetModule("MeshSetup")
396 oModule.InitialMeshSettings(
397   [
398     "NAME:MeshSettings",
399     [
400       "NAME:GlobalSurfApproximation",
401       "CurvedSurfaceApproxChoice:=", "UseSlider",
402       "SliderMeshSettings:="   , 5
403     ],
404     [
405       "NAME:GlobalModelRes",
406       "UseAutoLength:=" , True
407     ],
408     "MeshMethod:="     , "AnsoftClassic"
```

sim_xdeg_AM.py

```
409   ])
410 ############ Launch Analisys ##############
411
412 oDesign.AnalyzeAll()
413
414 ############ Export Plots #################
415 resultspath="%s%s_results"%(simdata["filepath"],\
416   simdata["filename"][:-4])
417 if not os.path.exists(resultspath):
418   os.makedirs(resultspath)
419
420 oModule = oDesign.GetModule("ReportSetup")
421 oModule.ExportToFile("Torque Plot",
    ↪  "%s/TorquePlotData.csv"%(resultspath), False)
422 oModule.ExportToFile("Induced Voltages",
    ↪  "%s/VoltagesPlotData.csv"%(resultspath), False)
423 oModule.ExportToFile("Input Currents",
    ↪  "%s/CurrentsPlotData.csv"%(resultspath), False)
424 oModule.ExportToFile("Fluxes Linkages",
    ↪  "%s/FluxesPlotData.csv"%(resultspath), False)
425 oModule.ExportToFile("Mechanical Position",
    ↪  "%s/PositionData.csv"%(resultspath), False)
426
427 if simdata['corelossflag']==1:
428   #first graph value
429   oModule.ExportToFile("Core Loss",
      ↪  "%s/CoreLossData.csv"%(resultspath), False)
430   #avg value
```

sim_xdeg_AM.py

```
431   oModule.ExportTableToFile("Core Loss",
      ↪  "%s/CoreLossAvg.csv"%(resultspath), "Legend")
432   oModule.ExportTableToFile("Hysteresis Loss",
      ↪  "%s/HysteresisLossAvg.csv"%(resultspath), "Legend")
433   oModule.ExportTableToFile("Eddy Currents Loss",
      ↪  "%s/EddyCurrentsLossAvg.csv"%(resultspath), "Legend")
434   oModule.ExportTableToFile("Excess Loss",
      ↪  "%s/ExcessLossAvg.csv"%(resultspath), "Legend")
435
436
437
438 sys.exit()
```